Pre-Proceedings of

The 8th Asian Conference on Membrane Computing (ACMC2019)

Xiamen University
Xiamen, China
November 14-17, 2019

Gexiang Zhang
Linqiang Pan
Xiangrong Liu

Editors

The list of authors and some participants as well as their e-mail address are given below, with the aim of facilitating the further communication and interaction:

Henry N. Adorna, Algorithms & Complexity Lab Department of Computer Science University of the Philippines Diliman Diliman 1101 Quezon City, Philippines
E-mail: hnadorna@up.edu.ph

Celine Anne A. Moredo, Algorithms & Complexity, Dept. of Computer Science, University of the Philippines Diliman Diliman 1101 Quezon City, Philippines.

Rodica Ceterchi, University of Bucharest Faculty of Mathematics and Computer Science 14 Academiei St, 010014 Bucharest, Romania

Francis George C. Cabarle, Algorithms and Complexity Laboratory Department of Computer Science University of the Philippines Diliman, Quezon City, Philippines; School of Information Science and Technology Xiamen University
E-mail: fccabarle@up.edu.ph

Wei Cen, School of Electrical and Electronic Engineering, Wuhan Polytechnic University, Wuhan 430023, Hubei, China.

Dionne Peter Cailipan, Algorithms & Complexity, Dept. of Computer Science, University of the Philippines Diliman Diliman 1101 Quezon City, Philippines.

Yunhui Chen, State Grid Sichuan Electric Power Company, Chengdu 610094, China

Ying Chen, State Grid Sichuan Electric Power Company, Chengdu 610094, China

Hong Chen, School of Electrical Engeneering, Southwest Jiaotong University, Chengdu, China

Matteo Cavaliere, Faculty of Science and Engineering, Manchester Metropolitan University, Manchester, Britain

Lovely Joy Casauay, Algorithms & Complexity Dept. of Computer Science, University of the Philippines Diliman Diliman 1101 Quezon City, Philippines.

Yingying Duan, School of Electrical and Engineering, Southwest Jiaotong University, Chengdu 610031, China
E-mail: 208130907@163.com

Jianping Dong, School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 61003, China

Fang Deng, School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 61003

Haocheng Fang, School of Math and Computer, Development Strategy Institute of reserve of food and material, Wuhan Polytechnic University, Wuhan 430023, China

Xiantai Gou, School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 61003, China

Ivan Cedric H. Macababayao, Algorithms & Complexity Dept. of Computer Science, University of the Philippines Diliman Diliman 1101 Quezon City, Philippines.

Zhixin He, School of Math and Computer, Development Strategy Institute of reserve of food and material, Wuhan Polytechnic University, Wuhan 430023, China

Qiyao Huang, School of Economics and Management, Wuhan Polytechnic University, Wuhan 430023, Hubei, China

Ivan Cedric H. Macababayao, Algorithms & Complexity Dept. of Computer Science, University of the Philippines Diliman Diliman 1101 Quezon City, Philippines.

YuLei Huang, School of Electrical Engineering and Electronic Information, Sichuan Province Key Laboratory of Power Electronics Energy-saving Technologies& Equipment, Key Laboratory of Fluid and Power Machinery , Ministry of Education, Xihua University,Chengdu 610039, P. R. China.

S James Immanuel, Department of Mathematics, Madras Christian College, Tambaram, Chennai ---600 059, India
E-mail: James_imch@yahoo.co.in

Meng Hu, School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 61003

Falin Jiang . School of Math and Computer, Development Strategy Institute of reserve of food and material, Wuhan Polytechnic University, Wuhan 430023, China
E-mail: jiang93214750@gmail.com

Ryan Chester J. Supelana, Algorithms & Complexity, Dept. of Computer Science, University of the Philippines Diliman Diliman 1101 Quezon City, Philippines.

S Jayasankar, Department of Mathematics, Ramakrishna Mission Vivekananda College, Chennai --- 600 004, India

E-mail: fksjayjay@gmail.com

Deting Kong, School of Management Science and Engineering, Shandong Normal University, Jinan,250014, China.

Shuo Liu, School of Economics and Management, Wuhan Polytechnic University, Wuhan 430023, Hubei, China.
E-mail: liushuo1979@hotmail.com

Xiangrong Liu, School of Math and Computer, Development Strategy Institute of reserve of food and material, Wuhan Polytechnic University, Wuhan 430023, China

Qifen Liu, School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 61003

Yezhou Liu, The University of Auckland, Auckland, New Zealand

Wanying Liang, School of Math and Computer, Wuhan Polytechnic University, Wuhan 430023, Hubei, China

Xiyu Liu, School of Management Science and Engineering, Shandong Normal University, Jinan,250014, China.
E-mail: sdxyliu@163.com

Ren Tristan A. de la Cruz, Algorithms and Complexity Laboratory Department of Computer Science University of the Philippines Diliman, Quezon City, Philippines;
E-mail: rentristandelacruz@gmail.com

Miguel Ańgel Martínez-del-Amor, Research Group on Natural Computing Department of Computer Science and Articial Intelligence University of Seville
E-mail: mdelamor@us.es

Atulya K Nagar, Department of Mathematics and Computer Science, Liverpool Hope University, Liverpool, United Kingdom
E-mail: nagara@hope.ac.ukg

Radu Nicolescu, Department of Computer Science, University of Auckland, Private Bag 92019, Auckland, New Zealand.
E-mail: r.nicolescu@auckland.ac.nz

Yunyun Niu, School of Information Engineering China University of Geosciences in Beijing, Beijing 100083, China
E-mail: niuyunyun1003@163.com

Ferrante Neri, COL Laboratory, School of Computer Science, University of Nottingham, Nottingham, United Kingdom

Hong Peng, School of Computer and Software Engineering, Xihua University, Chengdu, 610039,Sichuan, China

Ignacio Pérez-Hurtado, Research Group on Natural Computing, Dpt. Computer Science and Artificial Intelligence, School of Computer Engineering, Universidad de Sevilla, Seville, Spain

Linqiang Pan, School of Articial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, 430074, China

Mario J. Pérez-Jiménez, Research Group on Natural Computing, Dpt. Computer Science and Artificial Intelligence, School of Computer Engineering, Universidad de Sevilla, Seville, Spain

Pirthwineel Paul, School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 61003, China

Meenakshi Paramasivan, Institut für Informatik, Universität Leipzig, D-04009 Leipzig, Germany
E-mail: meena_maths@yahoo.com

Huaqing Qi, School of Economics and Management, Wuhan Polytechnic University, Wuhan 430023, China
E-mail: qihuaqing@sohu.com

Dunwu Qi, Chengdu Research Base of Giant Panda Breeding, Chengdu, Sichuan, China

Haina Rong, School of Electrical Engeneering, Southwest Jiaotong University, Chengdu, 610031, China
E-mail: ronghaina@126.com

Zeyi Shang, School of Electrical Engeneering, Southwest Jiaotong University, Chengdu, Sichuan, China
Laboratoire d'Algorithmique, Complexit´ e et Logique, Universit´ e Paris Est Cr´ eteil, Cr´ eteil, France

Bosheng Song, College of Information Science and Engineering, Hunan University, Changsha, China
E-mail: boshengsong@hust.edu.cn

Michael Stachowicz, Faculty of Science and Engineering, Manchester Metropolitan University, Manchester, Britain

K.G. Subramanian, Department of Mathematics, Madras Christian College, Tambaram, Chennai 600059, India
E-mail: kgsmani1948@gmail.com

Jianchi Sun, School of Math and Computer, Wuhan Polytechnic University, Wuhan 430023, Hubei, China
E-mail: 2233417652@qq.com

Jing Sun, The University of Auckland, Auckland, New Zealand

Hang Shu, School of Math and Computer, Development Strategy Institute of reserve of food and material, Wuhan Polytechnic University, Wuhan 430023, China

D G Thomas, Department of Mathematics, Madras Christian College, Tambaram, Chennai ---600 059, India
E-mail: dgthomasmcc@yahoo.com

T.Robinson, Department of Mathematics, Madras Christian College, Tambaram, Chennai ---600 059, India
E-mail: robin.mcc@gmail.com

Sergey Verlan, Laboratoire d'Algorithmique, Complexit´e et Logique, D´epartement Informatique, Universit´e Paris Est
E-mail: verlan@u-pec.fr

Luis Valencia-Cabrera, Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Sevilla, Spain
E-mail: lvalencia@us.es

Yuan Wang, School of Management Science and Engineering, Shandong Normal University, Jinan,250014, China.

Di Wang, School of Management Science and Engineering, Shandong Normal University, Jinan,250014, China.

Jun Wang, School of Electrical Engineering and Electronic Information, Sichuan Province Key Laboratory of Power Electronics Energy-saving Technologies& Equipment, Key Laboratory of Fluid and Power Machinery , Ministry of Education, Xihua University,Chengdu 610039, P. R. China.

Tao Wang, School of Electrical Engineering and Electronic Information, Sichuan

Province Key Laboratory of Power Electronics Energy-saving Technologies& Equipment, Key Laboratory of Fluid and Power Machinery , Ministry of Education, Xihua University,Chengdu 610039, P. R. China.

Tianbao Wu, State Grid Sichuan Electric Power Company, Chengdu 610094, China

Jianhua Xiao, School of Information Engineering China University of Geosciences in Beijing, Beijing 100083, China
E-mail: jhxiao@nankai.edu.cn

Jie Xue, School of Management Science and Engineering, Shandong Normal University, Jinan,250014, China.

Jian Xie, China Grain Wuhan Scientfic Reserch & Design Institute, Co.ltd, Wuhan 430023, China

Hua Yang, School of Math and Computer, Development Strategy Institute of reserve of food and material, Wuhan Polytechnic University, Wuhan 430023, China
E-mail: Huay20@163.com

Zehua Yang, School of Information Engineering China University of Geosciences in Beijing, Beijing 100083, China

Zhao Yao, School of Math and Computer, Development Strategy Institute of reserve of food and material, Wuhan Polytechnic University, Wuhan 430023, China

Zhibing Yu, School of Electrical Engeneering, Southwest Jiaotong University, Chengdu, 610031, China

Xiangxiang Zeng, Department of Computer Science, Xiamen University, Xiamen 361005, Fujian, China

Gexiang Zhang, School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 61003, China
E-mail: zhgxdylan@126.com

Xihai Zhang, School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 61003, China

Zhe Zhang, Business School, Shandong Normal University, Jinan, China

Luping Zhang, School of Articial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, 430074, China

Kang Zhou, School of Math and Computer, Development Strategy Institute of reserve of food and material, Wuhan Polytechnic University, Wuhan 430023, China
E-mail: zhoukang_wh@163.com

Jian Zhou, college of Food Science and Engineering, Wuhan Polytechnic University, Wuhan 430023, China

Committees

Steering Committee:

Henry Adorna (Quezon City, Philippines)
Artiom Alhazov (Chisinau, Moldova)
Bogdan Aman (Iasi, Romania)
Matteo Cavaliere (Manchester, UK)
Erzsébet Csuhaj-Varjú (Budapest, Hungary)
Giuditta Franco (Verona, Italy)
Rudolf Freund (Wien, Austria)
Marian Gheorghe (Bradford, UK) – Honorary member
Thomas Hinze (Cottbus, Germany)
Florentin Ipate (Bucharest, Romania)
Shankara N. Krishna (Bombay, India)
Alberto Leporati (Milan, Italy)
Taishin Y. Nishida (Toyama, Japan)
Linqiang Pan (Wuhan, China) – Co-Chair
Gheorghe Păun (Bucharest, Romania) – Honorary member
Mario J. Pérez-Jiménez (Sevilla, Spain)
Agustín Riscos-Núñez (Sevilla, Spain)
José M. Sempere (Valencia, Spain)
Petr Sosík (Opava, Czech Republic)
Kumbakonam Govindarajan Subramanian (Penang, Malaysia)
György Vaszil (Debrecen, Hungary)
Sergey Verlan (Paris, France)
Claudio Zandron (Milan, Italy)
Gexiang Zhang (Chengdu, China) – Co-Chair

Program Committee:

Henry Adorna (Quezon City, Philippines)
Cătălin Buiu (Bucharest, Romania)

Jun Wang (Chengdu, China)

Tingfang Wu (Suzhou, China)

Jianhua Xiao (Tianjing, China)

Jie Xue (Jinan, China)

Hsu-Chun Yen (Taiwai, China)

Jianying Yuan (Chengdu, China)

Claudio Zandron (Milan, Italy)

Xiangxiang Zeng (Xiamen University)

Gexiang Zhang (Chengdu, China) – Co-Chair

Xingyi Zhang (Anhui, China)

Xue Zhang (Boston, USA)

Xuncai Zhang (Zhengzhou, China)

Ming Zhu (Chengdu, China)

Organizing Committee:

Xiangrong Liu (Xiamen University, China)– Chair
Xiangxiang Zeng (Xiamen University, China)
Minli Tang (Xiamen University, China)
Lianmin Zhao (Xiamen University, China)

Contents

Part 1

Invited talks

Challenges for hardware implementations of P Systems

Sergey Verlan
University of Paris Paris Est, France

Abstract: A P system is a computational model that features a massive parallelism. In order to be able to use this feature in applications it is important to have a truly massively parallel implementation of P systems. At this moment, the only possibility to practically realize this is a digital circuit or FPGA (reconfigurable circuit) implementation. In this talk we explain the challenges raised by such an implementation. As example we discuss recent FPGA implementations of numerical P systems (NPS) that is a variant of P systems allowing to easily model physical processes based on differential equations. NPS allows many applications, the most prominent ones being in the area of robotic control. We will show how to obtain a hardware FPGA implementation of NPS allowing to achieve extreme speeds (~10^8 computational steps per second) and featuring massive parallelism (with more than 5000 parallel computational units). This opens new perspectives for the design of on-chip fast robotic controllers based on P systems. Another important consequence is the ability to use some variants of P systems as a programming language for algorithm description that can be efficiently translated in FPGA hardware by non-specialists.

Short bio: Dr. hab. Sergey Verlan is an associated professor at the University of Paris Est Creteil, France. He obtained his PhD in Computer Science in 2004 at the University of Metz, France and his habilitation in 2010 at the University of Paris Est. Dr Verlan's main research focus is the area of theoretical computer science and natural computing. He has expertise in the area of formal language theory, DNA computing, membrane computing, modeling of biological systems and hardware design.

Epistasis in Optimisation Problems

Ferrante Neri
University of Nottingham, Nottingham

Abstract: In biological systems, epistasis is the correlation between pairs of genes. By loosely interpreting this concept, in optimisation epistasis refers to the correlation between the variables with respect to an objective function. A low epistatic problem can be decomposed in multiple simpler problems, each of them with being with a reduced dimensionality with respect to the original problem.This talk presents the topic of correlation among variables and analyses the related topics of separability and rotation invariance. An overview of the literature on techniques to handle epistasis with reference to Differential Evolution is offered. Furthermore, this talk explores the relation between dimensionality and correlation among the variables and demonstrates that experimental conditions are an inherent part of the optimisation problem.

Short bio: Ferrante Neri received a Master's degree and a PhD in Electrical Engineering from the Technical University of Bari, Italy, in 2002 and 2007 respectively. In 2007, he also received a PhD in Scientific Computing and Optimization from University of Jyväskylä, Finland. From the latter institution, he received the DSc degree (Habilitation) in Computational Intelligence in 2010. He was Research Fellow with Academy of Finland in the period 2009-2014. Dr Neri moved to De Montfort University, United Kingdom in 2012, where he was appointed Reader in Computational Intelligence and in 2013, promoted to Full Professor of Computational Intelligence Optimisation. Since 2019 Ferrante Neri moved to the School of Computer Science, University of Nottingham, United Kingdom where he was appointed Associate Professor. Ferrante Neri's teaching expertise lies in mathematics for computer science, especially linear and abstract algebra. He is HEA Senior Fellow and author of the recently published book "Linear Algebra for Computational Sciences and Engineering". His research interests include algorithmics, hybrid heuristic-exact optimisation, scalability in optimisation and large scale problems. Dr Neri published over 150 publications in these topics. Dr Neri is an IEEE Senior Member and is Associate Editor and member of the Editorial Board of numerous journals including Information Sciences, Integrated Computer-Aided Engineering, and Memetic Computing. He organised over twenty symposia, special sessions, and workshops in International Conferences. Dr Neri is on the panel of funding bodies in seven countries. He has been in the Programme Committee of over 100 conferences.

Some Variants of P Systems

Bosheng Song
Hunan University, Changsha, China

Abstract: Membrane computing is an unconventional computing area that aims to abstract computing ideas (e.g., computing models, data structures, data operations) from the structure and functioning of living cells, as well as from more complex biological entities, like tissues, organs and populations of cells. The computational models that are part of this paradigm are generically called P systems, which are distributed and parallel computing devices. In this talk, inspired by different biological facts, some variants of P systems are introduced; besides, some new results of these P systems and open problems are presented.

Short bio: Bosheng Song received the Ph.D. degree in control science and engineering from Huazhong University of Science and Technology, Wuhan, China, in 2015. He spent eighteen months working in the Research Group on Natural Computing, University of Seville, Seville, Spain, from November, 2013 to May, 2015. He was worked as a post-doctoral researcher with the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, from March, 2016 to February, 2019. He is currently an Associate Professor with the College of Information Science and Engineering, Hunan University, Changsha, China. His current research interests include membrane computing and formal language theory.

Space Filling Curves: Representations and Generation

Rodica Ceterchi
University of Bucharest

Abstract: Space Filling curves have been of interest for mathematicians since the end of the 19th century, when Peano and Hilbert discovered the first examples. More recently, they turned out to be useful tools in Computer Science, with many applications, especially to scientific computing. Their representations as strings (chain-code words) have been introduced, and different generations with formal languages tools have been studied. We present here several results concerned with the generation of finite approximations of space-filling curves in string representation, with parallel rewriting rules, controlled by P systems. We also introduce a novel representation of them as 2D-arrays, their generation with array rewriting rules, and the connections between the string representation and the array representation.

Short bio: Prof. Dr. Habil. Rodica Ceterchi is Professor at the Faculty of Mathematics and Computer Science, Department of Computer Science, University of Bucharest, Romania. She obtained her PhD in 1991, and her Habilitation in 1997 from the same University. Her research interests include algebras for many-valued logics, formal models for semantics, unconventional models of computation inspired by nature, membrane computing. She was in 2003 a recipient of the “Grigore C. Moisil” award of the Romanian Academy for her contributions in this area.

Bibliometric Analysis of Membrane Computing

Gexiang Zhang
Chengdu University of Technology

Abstract: This talk will discuss the development process of membrane computing and the statistical data with respect to the publications appearing in international journals, conferences and workshops, the researchers in geographical distributions and the funds. The bibliometric analysis results of membrane computing will be reported in detail. Some future suggestions will be provided.

Short bio: Gexiang Zhang is currently a full professor of College of Information Science & Technology at Chengdu University of Technology, Chengdu, China. He worked at The University of Sheffield, UK, at the University of Seville, Spain, and at the New York University, USA. His areas contain artificial intelligence (membrane computing, machine learning, etc.), robotics and smart grids. He is the principle investigator of five projects funded by National Natural Science Foundation of China. He has published three books, over 200 refereed book chapters, journal and conference papers, and authored more than 40 Chinese patents. He won 4 best paper awards of international conferences, IMCS Prize and 3 provincial science and technology advancement awards. He supervised 5 PhD students and more than 70 master students.
He is the founding President of International Membrane Computing Society (IMCS), IET Fellow, IET Fellow Assessor, IEEE Senior member, and Managing Editor of Journal of Membrane Computing (Springer). He serviced ACMC2013 and ACMC2017 as conference chairs, PC member of tens of international conferences, and also services ACMC as Steering Committee member and co-chair of PC, and hundreds of international journals as a reviewer.

Part 2

Regular Papers

Evaluation and analysis of loss and waste in the processing of rapeseed oil

Falin Jiang[1], Hua Yang[1] *, Kang Zhou[1], Huaqing Qi[2], and Jian Zhou[3]

[1] School of Math and Computer,
Development Strategy Institute of reserve of food and material,
Wuhan Polytechnic University, Wuhan 430023, China
[2] School of Economics and Management,
Wuhan Polytechnic University, Wuhan 430023, China
[3] college of Food Science and Engineering,
Wuhan Polytechnic University, Wuhan 430023, China
{jiang93214750@gmail.com,Huay20@163.com}

Abstract. In order to meet people's requirements for vision and taste and increase the market share of rapeseed oil, rapeseed oil processing enterprises often over-process or under-process rapeseed, resulting in nutrient loss and unnecessary waste. From the perspective of loss and waste in rapeseed oil processing, an evaluation method for evaluating rapeseed oil processing industry or enterprise processing by data analysis was proposed. The specific content of the method is based on the production data of rapeseed oil of different scales, equipments and processes of various enterprises in the country, and the fitting simulation model of loss and waste in the processing of rapeseed oil is established, thereby establishing a rapeseed oil processing evaluation system; Based on the evaluation of rapeseed oil processing industry and enterprise production data, a two-level multi-objective optimization model was constructed. The simulation model of processing loss and waste produced by rapeseed oil processing industry was improved, and the evaluation result was finally obtained. This evaluation method for the food industry provides practical guidance for the rapeseed oil processing process in the processing of rapeseed oil.

Keywords: Processing of rapeseed oil, Loss and waste, Fitting simulation model, Two-layer multi-objective optimization model, Evaluation method

1 Introduction

As one of the most oil-producing oil crops, rapeseed plays an important role in the supply of edible oil in China. According to the industry data released by the country, the rapeseed oil consumption of traditional edible oil in China is 6.856 million tons, accounting for 28.5% of the total edible vegetable oil. It is

* Corresponding author.

the second largest consumer of oil in China, and it is the edible oil market in China. It has a pivotal position [1–3].

The processing of rapeseed oil is in the middle of the rapeseed industry chain, which is connected with the rapeseed production link upstream of the rapeseed industry chain, and then connected with the rapeseed oil market consumption link downstream of the rapeseed industry chain[4]. Therefore, the rise and fall of the rapeseed oil processing industry is related to the rise and fall of the rapeseed industry to a certain extent. For a long time, consumers have chosen the appearance, taste and nutritional value of rapeseed oil, but at the time of purchase, attention has been focused more on the appearance and taste of rapeseed oil. For this reason, in order to satisfy people's visual and taste requirements, and also to increase the market share of rapeseed oil, rapeseed oil processing enterprises often over- or under-process rapeseed, resulting in the lack unnecessaryof nutrition and waste[5, 6].

In fact, the excessive or insufficient processing of rapeseed will not only result in a low yield, but also a large loss of nutrients in the rapeseed oil[7, 8]. Therefore, processing enterprises are particularly critical to the proper processing of rapeseed oil, and how to know whether the processing of rapeseed oil processing in processing enterprises is moderate processing. Therefore, this paper proposes an evaluation method for loss and waste in processing of rapeseed oil. Firstly, the evaluation system and fitting simulation model of rapeseed oil processing link loss and waste are constructed[9–11], and then based on the production data of rapeseed oil processing industry and enterprises two-layer multi-objective optimization model. The combination of the two results in the evaluation results and optimization schemes of the rapeseed oil processing enterprises to be evaluated, which provides a theoretical basis for the moderate processing of rapeseed oil in the country[12–14].

2 Related concepts and definitions

2.1 Basic concept

Whether the processing of rapeseed oil is excessive or insufficient can be reflected by the total production rate and total loss rate of processed rapeseed oil in rapeseed oil processing enterprises. When the total production rate of a processing enterprise is much lower than the social average, then this processing enterprises have the possibility of over-processing. There are many factors that can affect the total production rate and total loss rate of processing enterprises. These influencing factors are mainly divided into rapeseed raw materials, processing links, processing equipment and processing scale and others. Therefore, this paper intends to establish a rapeseed oil processing loss and waste evaluation system to explore whether the rapeseed oil processing enterprises have undergone moderate processing.

The production rate and loss rate of rapeseed oil is an important factor reflecting the production efficiency of rapeseed oil processing enterprises. And the

loss rate of edible materials in the residual materials during processing is a direct reflection of whether the rapeseed oil processing enterprises are over-processed. Therefore, the production rate of each link of rapeseed oil processing enterprises, the loss rate of each link and the loss rate of residual edible materials in each link are selected as important indicators for constructing the evaluation system of rapeseed oil processing enterprises.The rapeseed oil processing loss assessment system is a system for evaluating whether the rapeseed oil processing enterprise products are properly processed. The evaluation system for the loss of processing of rapeseed oil consists of two parts: evaluation index and evaluation standard value. The evaluation indicators are divided into three types of indicators: link production rate, loss rate and loss rate of edible substance. The evaluation standard value is the social average production level of the index produced by the rapeseed oil processing enterprise of the same scale and the same equipment.

2.2 Related definition

The production rate and loss rate of rapeseed oil processing enterprises producing rapeseed oil at a certain time and the loss rate of leftover edible substance during processing are defined as follows: Let W be the weight of the total input of rapeseed processed by the processing enterprise in a certain period of time, M denotes the total mass of rapeseed oil produced during this period, and R denotes the weight of the edible substance remaining in the process during the processing period.p indicates the production rate of rapeseed oil processing enterprises during this period, q indicates the loss rate of rapeseed oil processing enterprises during this period, and $\overline{p}$ indicates the loss rate of edible materials in the residual materials of rapeseed oil processing enterprises during this period.

The production rate of rapeseed oil processing enterprises processing rapeseed oil at a certain time is as follows:

$$p = \frac{M}{W} * 100\% \tag{1}$$

The loss rate of rapeseed oil processing enterprises processing rapeseed oil at a certain time is as follows:

$$q = 1 - p \tag{2}$$

The loss rate of edible materials processed by rapeseed oil processing enterprises during the processing of rapeseed oil at a certain time is as follows:

$$\overline{p} = \frac{R}{W} * 100\% \tag{3}$$

The production rate is divided into the total production rate and the production rate of each link, the loss rate is divided into the total loss rate and the loss rate of each link, and the loss rate of the remaining edible materials is divided into the total loss rate of edible materials and the loss rate of edible materials in each link.

Let j be the j-th process of rapeseed oil processing, that is, the production rate of the rapeseed oil processing enterprise in the j-th process of a certain period of time is as follows:

$$p_j = \frac{M_{j+1}}{M_j} * 100\% \tag{4}$$

The loss rate of the j-th process of processing rapeseed oil in a certain period of time by rapeseed oil processing enterprises is as follows:

$$q_j = 1 - p_j \tag{5}$$

The loss rate of the remaining edible materials in the j-th process of processing rapeseed oil produced by rapeseed oil processing enterprises at a certain time is as follows:

$$\overline{p}_j = \frac{R_{j+1}}{M_j} \tag{6}$$

The leftover edible substance refers to the substance that can be directly consumed by humans in the residue of the processing.

3 Analysis of rapeseed oil processing loss and waste evaluation model

When studying the loss and waste evaluation methods in the processing of rapeseed oil, there are three problems in the research, data preprocessing in the database, design of the fitting simulation model for evaluating the standard values, and algorithm design for loss and waste in rapeseed oil processing.The following will be introduced separately.

3.1 Sample data set and its preprocessing

This dataset comes from questionnaire survey data of rapeseed oil processing industry in the whole society, which inevitably causes the problems of low quality, high dimension, noisy and inconsistency of this dataset. Therefore, data preprocessing of this dataset is an important task. The data preprocessing used in this paper mainly includes three methods: data cleaning, data integration and data reduction. The order of the three processing operations is not sequential, and a certain preprocessing method may be performed multiple times use. The following is a brief introduction:

(1) Data cleaning is mainly to deal with missing and dirty data in the data set;
(2) Data integration mainly integrates and processes data in multiple data sources and solves the problem that semantic ambiguity is integrated into a consistent data set with the same attribute and consistent name, or the correlation between attributes is detected and deleted;
(3) Data reduction is mainly to identify the data set that needs to be mined, delete the irrelevant dimension, reduce the amount of data, and reduce the processing range.

3.2 Simulation model for loss and waste in processing of rapeseed oil

Through the production data of the whole society rapeseed oil processing enterprises, the simulation model of loss and waste in the processing of rapeseed oil was calculated, and the average rapeseed production rate, loss rate and loss rate of residual edible materials in different provinces, scales and equipment were obtained.

Therefore, firstly, the production data of rapeseed oil processing enterprises in the whole society are classified according to provinces, enterprise scales and processing equipment. We divide the scale of the enterprise into three categories: large, medium and small. The processing scale is small-scale with a daily processing capacity of less than 200 tons, and is larger than 400 tons. The middle part is medium-scale. Processing equipment is divided into domestic equipment, imported equipment and mixing equipment. The processing steps of rapeseed oil are: feeding, screening, stone removal, cleaning, broken, shelling, hydration degumming (alkali refining) and decolorization.

The overall idea of constructing a fitted simulation model for evaluating the standard value is to treat the enterprise production data under the same province, scale and equipment as a whole, based on the idea of linear least squares fitting, the data and needs of each sample point. The principle of calculating the total production rate, loss rate and loss rate of residual edible substances is calculated by the principle of minimizing the total difference (sum of distance) between the index points of the simulation. The following is a fitting simulation model for the evaluation standard value of the link production rate, the link loss rate and the loss rate of the edible substances in each link.

The fitted simulation model of production rate, loss rate and loss rate of residual edible substances is as follows:

$$\min \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{s} (p_j - p_{ij})^2} \tag{7}$$

The constraint is:

$$p_j > 0 \tag{8}$$

Where n is the number of samples, the production rate of each process link of a company is taken as a sample; s is the sum of the number of processes for processing rapeseed oil; p_{ij} is the production rate of the j-th process of the i-th sample; p_j is the standard value of each process link that needs to be calculated when all the samples are combined. The fitting simulation model of the loss rate and the loss rate of the remaining edible material replaces the production rate p in the formula 7 with the corresponding loss rate q or the residual edible material loss rate $\overline{p}$.

3.3 Optimization model for loss and waste in processing of rapeseed oil

The optimization model for loss and waste in the processing of rapeseed oil is to determine whether the processing of rapeseed processing enterprises is over-processed or whether the loss of processed materials is too large. A dual-objective optimization model is established for the maximum and minimum loss rates of different scales and different production equipments, so as to find the appropriate optimization scheme and optimize the rapeseed processing process.

The first layer, an optimized model for all links of rapeseed processing. For all links of rapeseed processing, calculate the loss rate of each link based on the maximum loss rate and the maximum variance model to determine which part of the process is over-processed; the second layer, rapeseed processing scale and processing equipment optimization model . Starting from different dimensions, the model is built using the principle of maximum loss rate and maximum variance, and the loss rate of each processing scale and different production equipment is calculated. According to the law of loss rate during processing, the process of maximizing the excessive processing loss of rapeseed oil is finally found, so as to optimize the processing of rapeseed.

Let a_s denote the k -th $(k = 1, 2, \cdots s)$ sample collected, then the mathematical form of a_s is as follows:

$$a_k = (a_{k1}, a_{k2}, a_{k3}, a_{k4}, l_{kj}) \tag{9}$$

Where a_{k1} indicates the type of the k -th sample, a_{k2} indicates the province of the k-th sample, a_{k3} indicates the size of the production enterprise of the k-th sample, a_{k4} indicates the type of production equipment of the k-th sample, and l_{kj} indicates the k -th sample of the j-th sample The loss rate of the process, l_j represents the average loss rate of the j-th process of all samples.

Let p_j denote the variance of the loss rate of the j-th process of all samples, and $\lambda_i \, (i = 1, 2)$ denote the weight of the loss rate and the variance. The optimization model of the loss of waste in the processing of rapeseed oil is as follows:

$$\max\{\lambda_1 \frac{l_j}{\sum\limits_{j=1}^{s} l_j} + \lambda_2 \frac{p_j}{\sum\limits_{j=1}^{s} p_j}\} \tag{10}$$

The constraint is:

$$l_j = \frac{1}{t} \sum_{k=1}^{t} l_{kj} \tag{11}$$

$$p_j = \frac{1}{t} \sum_{k=1}^{t} (l_{kj} - l_j)^2 \tag{12}$$

$$\sum_{i=1}^{2} \lambda_i = 1 \tag{13}$$

The output variable is the most suitable processing procedure for rapeseed oil, and the input variable is the pre-processed sample data $a_k = (a_{k1}, a_{k1}, a_{k1}, a_{k1}, l_{kj})$.

4 Experimental results and analysis

4.1 Experimental results and analysis

The data is derived from a sample survey data set using three-stage stratification across provinces, scales, and processing equipment across the country. The collected data includes 103 data from 23 provinces. and the data is pre-processed and stored in a database. After that, the sample data in the database is classified according to the process conditions. The data of each province is divided into three categories according to the equipment: imported equipment, domestic equipment and domestic imported hybrid equipment, and then divided into large, medium and small by scale.

4.2 Experimental results of loss and waste in the processing of rapeseed oil in the country

After classifying and data processing the data of the national rapeseed oil processing enterprises, the standard loss of each link of the national rapeseed oil under various scales was calculated by using the fitting simulation model (formulas 7 and 8) of the loss of waste in the processing of rapeseed oil. The rate and standard yield results are shown in Table 1.

Table 1. Standard production rate and standard loss rate of all stages of national rapeseed oil under various scales

scale		equipment	Feeding	Screening	Remove stone	Clean up broken	Shelling	Alkali refining	Decolorization
large	Production rate	Domestic	100	99	98.79	98.97	100	100	35.16	99.09
		import	100	99.01	98.58	98.97	100	100	35.32	98.91
		mixing	100	99	99	98.97	100	100	35	99.26
	Loss rate	Domestic	0	1	1.21	1.03	0	0	64.84	0.91
		import	0	0.99	1.42	1.03	0	0	64.68	1.09
		mixing	0	1	1	1.03	0	0	65	0.74
middle	Production rate	Domestic	100	99	98.8	98.97	100	100	34.94	99.15
		import	100	99	99	98.97	100	100	34.74	99.3
		mixing	100	99	98.59	98.97	100	100	35.15	99
	Loss rate	Domestic	0	1	1.2	1.03	0	0	65.06	0.85
		import	0	1	1	1.03	0	0	65.26	0.7
		mixing	0	1	1.41	1.03	0	0	64.85	1
small	Production rate	Domestic	100	99	98.99	98.97	100	100	34.92	99.27
		import	100	98.92	99	98.97	100	100	35.12	99.41
		mixing	100	98.83	99.01	98.98	100	100	35.33	99.55
	Loss rate	Domestic	0	1	1.01	1.03	0	0	65.08	0.73
		import	0	1.08	1	.03	0	0	64.88	0.59
		mixing	0	1.17	0.99	1 1.02	0	0	64.67	0.45

From the above data, it can be seen that the decolorization, screening and hydration degumming (alkali refining) links in the processing of rapeseed oil in the country are relatively large, and the loss rate of edible materials in the remaining materials is the most serious in the alkali refining and decolorization. The calculation data of the second layer according to the two-layer multi-objective optimization model (formulas (9)-(13)) is shown in Table 2.

Table 2. The second layer of data in the national double-objective multi-objective optimization model of rapeseed oil

classification	Feeding	Screening	Remove stone	Clean up broken	Shelling	Alkali refining	Decolorization
large-scale	0	0.06	0.02	0.02	0.03	0	0.44	0.43
Medium scale	0	0.16	0.01	0.01	0	0	0.56	0.26
Small scale	0	0.01	0.01	0.01	0.01	0	0.46	0.5
Imported equipment	0	0.01	0.03	0	0	0	0.58	0.36
Domestic equipment	0	0.04	0.12	0.05	0	0	0.78	0.02
Mixing equipment	0	0.03	0.04	0.03	0	0	0.83	0.07

As can be seen from the results of the second layer of data, the loss rate of the decolorizing link is the largest regardless of which device is used. For enterprises using imported equipment, the loss rate of alkali refining is the second; enterprises that choose domestic equipment, followed by screening and alkali refining; enterprises use mixing equipment, hydration degumming indicates discoloration In the process of sieving, alkali refining, the waste loss is serious. The waste in the decolorization process is in the alkali refining, and the loss rate of imported equipment is the largest; in the management process, the loss of domestic equipment The rate is the biggest, which indicates that the loss of these two links is serious. When using domestic or imported equipment, the production of hybrid equipment is preferred in equipment selection.

In summary, it can be concluded that in terms of processing scale, it is recommended to develop medium-sized processing enterprises, encourage small-scale enterprises to integrate, and large enterprises to optimize distributed processing; when selecting equipment, try to use mixed equipment to process and produce rapeseed oil. Encourage the use of blended products to optimize the suitability of the equipment; in all processing steps, the control of the screening and hydration degumming, especially the decolorization, is optimized throughout the entire process.

5 Conclusion and Outlook

From the perspective of loss and waste in the processing of rapeseed oil, this paper proposes an evaluation model for the processing of rapeseed oil by means of data analysis, and establishes an evaluation system for the loss and waste in the processing of rapeseed oil. By evaluating the model analysis and experimental results, it can be seen that the evaluation model has the following advantages:

(1) The loss and waste assessment model of rapeseed oil processing can truly and effectively reflect whether the rapeseed oil processing enterprise is over-processed;
(2) The rapeseed oil processing link loss and waste assessment model can quickly and effectively evaluate the rapeseed oil processing industry;
(3) The loss and waste assessment model of rapeseed oil processing can provide a theoretical basis for the moderate processing of rapeseed oil.

In recent years, the processing technology of rapeseed oil has been improved and the equipment has been updated. The supervision of various processes has been in place, but many processing enterprises in the rapeseed oil processing industry have over- or under-processed rapeseed,Lead to nutrient loss and unnecessary waste. Therefore, it is urgent to summarize historical experience. In the future, we will further improve the evaluation model by collecting processing data of more varieties of rapeseed oil processing enterprises, such as scale, rapeseed model, processing equipment, region, processing technology, rapeseed. Relevant information on rapeseed oil processing enterprises with more dimensions such as origin and quality, continue to expand and optimize this evaluation system, take the progress of science and technology as the driving force, and promote the new leap of rapeseed industry as the goal, to maintain the safety of national edible oil supply. Make new contributions.

Acknowledgments. This work was supported by the Science and Technology Research Project of Hubei Provincial Department of Education (Grant No. D20191604) and subproject of the National Key Research and Development Program of China(Grant No. 2017YFD0401102-02).

References

1. Rahman A N F, Genisa J, Dirpan A, et al Modification of dry grain processing for rice nutrition produced. IOP Conference Series Earth and Environmental Science. 2018: 157.
2. Wu Y, Zhai X, Xu F, et al: Research on the Process of Rapeseed Refining in China. Grain Science and Technology. 2018, 43(12): 61-64.
3. Chen Z, Deng Y, Zhang S, Hu L, Wang X: Research on the current situation and countermeasures of loss and waste in grain processing in China. Grain Science and Technology. 2018, 43(05): 96-99..
4. Xie H, Tan T, Luo Q, Yang L, Chen G: Current Status and Opportunities of Rape Industry Development.Crop Research. 2018,32(05):431-436.
5. Liu C, Hunag J, Leng B, Feng Z, Li J: Current status, development dilemma and suggestions of rapeseed industry in China. Journal of China Agricultural University.2017, 22(12): 203-210.
6. Guo Y, Yang Y, Sun J: The Status Quo and Countermeasures of the Development of Rape Industry in China. Agricultural Economy. 2016(07):44-46.
7. Qu W, Jiang Y, Ren C: Thoughts and Suggestions on the Development of Rape Production in Henan Province. Henan Agriculture. 2019 (13): 13-14.
8. Wang H, Li Y: Zoomlion's overall solution for building grain processing intelligent engineering. Agricultural Machinery Quality and Supervision. 2018 (10): 41
9. Chen Y: Cost Management Problems and Countermeasures of Grain Processing Enterprises. Finance and Accounting, 2018 (18): 123-125.
10. Mei X, Feng Z, Zheng Y: Analysis of the "differentiation" development model of rapeseed oil processing industry. China Oils and Fats, 2015, 40 (06): 1-6.
11. Peng S, Wu W: Quantitative and qualitative analysis of tea seed oil adulteration based on regression equation. Journal of the Chinese Cereals and Oils Association, 2019: 1-7.

12. QiangANG, Agricultural products processing characteristics and quality evaluation, Journal of Integrative Agriculture, Volume 17, Issue 5, 2018, Pages 975-976.
13. Ying Xu, Biao Zhang, Lingxian Zhang, A technical efficiency evaluation system for vegetable production in China, Information Processing in Agriculture, Volume 5, Issue 3, 2018, Pages 345-353.
14. Wuhan University of Light Industry. Data analysis system and method for grain processing loss: China, CN201810408482.2[P].2018-09-25.

A Learning Spiking Neural P System and Its Applications to Classification Problems

Xihai Zhang, Haina Rong*, Gexiang Zhang, Pirthwineel Paul, Zhibing Yu, and Xiantai Gou

School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 61003, China
E-mail: ronghaina@126.com

Abstract. Spiking neural P systems (SN P systems), as an important branch of the third generation neural network models, has been applied in many real-life areas. Exploration of solutions for pattern recognition problems by using the SN P systems is a challenging and ongoing topic. This paper proposes learning SN P systems (LSN P systems) to solve supervised classification problems. This is the first attempt to construct LSN P systems to show great potential for handling real-life classification problems. LSN P systems are designed by using weighted fuzzy SN P systems, a multi-layer network structure with adaptive weights adjustment rule and a selection method of output neurons, and ascending dimension techniques for non-linear classification problems. Experiments conducted on the problems in the University of California, Irvine (UCI) machine learning repository show the feasibility and effectiveness of LSN P systems.

Keywords: Spiking neural P systems · Learning spiking neural P systems · Supervised learning · Adaptive weight adjustment rule.

1 Introduction

Artificial neural networks, as one of the most important tools in artificial intelligence (AI), have been developed for more than sixty years since the introduction of perceptron linear algorithm [1]. With the development of neural networks, many network models have been proposed to achieve pattern recognition problems, such as the first generation neural networks represented by McCulloch - Pitts neurons [2] and Hopfield Neural Network (HNN) [3], the second generation neural networks represented by Back Propagation Neural Network (BPNN) [4] and Extreme Learning Machine (ELM) [5], the third generation neural networks represented by Spiking Neural Networks (SNNs) [6] and Pulse Coupled Neural Network (PCNN) [7]. It has been widely used in various fields, such as image identification [8], voice recognition [9], autonomous vehicles [10], medical diagnosis [11]. However, the first generation and the second generation neural networks also have some disadvantages. The output of the first generation neural networks must be digital. This weakness limits the application of these networks only to

boolean functions [12–14]. One of the most outstanding characteristics of the second generation neural networks is that it can process the continuous input and output values, but the second generation neural networks adopt frequency coding which is not suitable in some biological neurons [12–14]. The third generation neural networks represented by SNNs in which the coding scheme is a kind of time coding. In another word, the information of input data is encoded by generating spikes at different time, which is similar to human neural activity. However, there is only one type of neuron in SNNs, i.e., spiking neurons [6]. In real neurobiology research, not only there exist spiking neurons in our brain but also have other types of neurons, like astrocytes [15], oligodendrocyte [16], etc.

Membrane computing models (usually called P systems) are nature-inspired models, proposed by Gh. Păun [17]. These models are abstracted from the structure and the functioning of the biological cells, organs, and colonies. In the last few decades, many variants of P systems have been introduced, such as cell-like P systems [17], tissue-like P systems [18], neural-like P systems [19]. These models also have been used to solve problems in theory and applications [20–23]. In recent years, the SN P systems have gained popularity because of its similarities with SNNs. The SN P systems can be regarded as a combination of SNNs and P systems. A great deal of work has been done on the theoretical aspects of SN P systems and plenty of other variants with different biological phenomena, such as SN P systems with astrocytes [24], SN P systems with anti-spikes [25], SN P systems with structural plasticity [26] have been proposed. The real-life applications of SN P systems have been in skeletonizing images [27], combinatorial optimization [28], fault diagnosis [29, 30], decoder design [31], etc. Also, some researchers have investigated the feasibility of pattern recognition problems by using the SN P systems in [32, 33]. Although they have explored the models of pattern recognition by using SN P systems in different aspects, these models have some drawbacks such as the structure of the network is not universal and the accuracy of the model also needs some improvements.

In the field of membrane computing, the attempt to extend P systems to achieve pattern recognition is a challenging and ongoing research topic. However, the papers about SN P systems with learning ability are very rare. The learning ability in SN P systems is a very important aspect to expand the real-life application. Under these circumstances, this paper proposes a design strategy of neural systems capable of solving pattern recognition problems. The proposed learning model is unlike the models in [32, 33], it archives the pattern recognition problems in an explanatory network structure and can be used to recognize some real-life problems. The learning SN P systems (LSN P systems) are organized in a multi-layer network structure in which the properties of the neurons in Weighted Fuzzy Spiking Neural P Systems (WFSN P systems) and Taylor theorem are considered comprehensively. In order to make the model fit better with the streaming inputs, the Widrow-Hoff learning law is employed to adjust weights during the iterative learning process. Moreover, the Taylor series expansion is used to generate higher-dimensional information and these higher dimensional informations transform non-linearly separable data into linearly separable, and a

network in the framework of WFSN P systems with Widrow-Hoff learning algorithm has been introduced for identification of the pattern recognition problem. In this manner, the pattern recognition problem can be solved perfectly in LSN P systems. Furthermore, it is the first known attempt to construct a supervised learning algorithm for solving pattern recognition problems using the WFSN P systems, and the experimental results in the UCI machine learning repository illustrate that the model is powerful and better than BPNNs and SNNs in the performance of classification. In this paper, we not only construct a new supervised learning algorithm for multilayer networks in the framework of membrane computing but also expand the scope of membrane computing models in solving real-world problems.

The remainder of this paper is organized in the following way. Section 2 recalls the WFSN P systems briefly. Section 3 presents the proposed LSN P systems in detail and use several simple cases to illustrate how to achieve the function of learning in the LSN P systems. Experimental results are described in Section 4. Concluding remarks are given in Section 5.

2 WFSN P Systems

In this section, we introduce the WFSN P systems and its advantages in construcing the LSN P systems.

Definition 1. *The WFSN P systems of degree* $m(\geq 1)$ *is a construct of the form [34]:*

$$\Pi = (O, N_p, N_r, syn, IN, OUT)$$

where

(1) $O = \{a\}$ is the singleton alphabet and a is called spike.
(2) $N_p = \{\sigma_{p1}, \sigma_{p2}, ..., \sigma_{pm}\}$ is *proposition neurons set*. The fuzzy proposition within the fuzzy knowledge base is associated with each proposition neuron σ_{pi},$1 \leq i \leq m$. Every proposition neuron σ_{pi} has the form $\sigma_{pi} = (\alpha_i, \overrightarrow{\omega_i}, \lambda_i, \mathbf{r}_i)$ where
   a) $\alpha_i \in [0, 1]$ is the potential value of pulse in the proposition neuron σ_{pi}, α_i is the fuzzy truth value of the proposition for each proposition neuron σ_{pi}.
   b) $\overrightarrow{\omega_i} = (\omega_{i1}, \omega_{i2}, ..., \omega_{is_j})$ is the output weight vector for each proposition neuron σ_{pi}, where each $\omega_{ij} \in [0, 1](1 \leq j \leq s_j)$ is the weight from σ_{pi} to σ_{rj} and s_j is the number of rule neurons in the next layer.
   c) $\mathbf{r}_i$ is a finite set of firing/spiking rules of the form $E/a^\alpha \rightarrow a^\alpha; d$, where $\alpha \in (0, 1]$ and $d \in \mathbb{N}$. E is the firing condition where $E = \{\alpha \geq \lambda_i\}$. It means that if $\alpha \geq \lambda_i$, then the firing rules will be used. $\lambda_i \in [0, 1)$ is called the threshold of firing rule.
(3) $N_r = \{\sigma_{r1}, \sigma_{r2}, ..., \sigma_{rn}\}$ is *rule neurons set*, the weighted fuzzy production rule in fuzzy knowledge base is associated with each rule neuron σ_{rj}, $1 \leq j \leq n$. For every rule neuron σ_{rj} has the form $\sigma_{rj} = (\alpha_j, \gamma_j, \overrightarrow{v_j}, \tau_j, \mathbf{r}_j)$ where

a) $\alpha_j \in [0,1]$ is the potential value of pulse in the rule neuron σ_{rj}.
b) $\gamma_j \in [0,1]$ is the certain factor. It represents the strength of belief in the weighted fuzzy production rule with rule neuron σ_{rj}.
c) $\overrightarrow{v_j} = (v_{j1}, v_{j2}, ..., v_{jt_i})$ is the output weight vector for each rule neuron σ_{rj}, where each $v_{ji} \in [0,1](1 \leq i \leq t_i)$ is the weight from σ_{rj} to σ_{pi} and t_i is the number of proposition neurons in the next layer.
d) $\mathbf{r}_j$ is a finite set of firing/spiking rules of the form $E/a^\alpha \rightarrow a^\beta; d$, where $\alpha \in (0,1]$, $\beta \in (0,1]$, $d \in \mathbb{N}$ and $E = \{\alpha \geq \tau_j\}$ is the firing condition. It means that if $\alpha \geq \tau_j$, then the firing rule can be used, where $\tau_j \in [0,1)$ is called the threshold of firing rule.

(4) $syn \subseteq (N_p \times N_r) \cup (N_r \times N_p)$ represents the synaptic connections between proposition neurons and rule neurons. The connection between proposition neurons and proposition neurons(or rule neurons and rule neurons) is not allowed in the WFSN P systems.
(5) $IN, OUT \subseteq N_p$, i.e., the *input neurons* and *output neurons* are all *proposition neurons*.

There are two types of neurons in the WFSN P systems, one is proposition neurons and the other is rule neurons. For different neurons, the output value is totally different. In the following, the rules of two types of neurons are discussed.

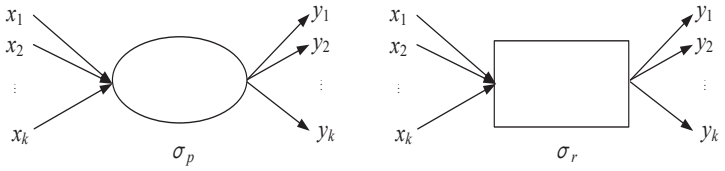
For proposition neurons, when the firing condition $E = \{\alpha \geq \lambda_i\}$ is satisfied, then the firing rule $E/a^\alpha \rightarrow a^\alpha; d$ can be applied. There are two situations in the proposition neurons:

(1) If the spike arrives in proposition neurons at different time, then the proposition neurons will judge whether the firing condition is satisfied separately. If it is satisfied, then the firing rule will be applied and the output value of potential is $\alpha \otimes \omega$, where $\otimes$ is multiplication operator of fuzzy truth values and ω is output weight of proposition neuron, otherwise the firing rule will not be used.
(2) If the spike arrives at the same time, then the proposition neurons will use a logical operator to judge whether the firing condition is satisfied. The logical "OR" operator ($\vee$) will be used to estimate the potential values of spikes received from the predecessor neurons. For example, as shown in Fig.1(a), if a proposition neuron receives a series of potential values of spikes at the same time, then the potential value will be $\alpha_{in} = x_1 \vee x_2 \vee ... \vee x_k$. So the environment will receive the potential $\alpha_{out} = (x_1 \vee x_2 \vee ... \vee x_k) \otimes \omega$.

For rule neurons, when firing condition $E = \{\alpha \geq \tau_j\}$ is satisfied, then the firing rule $E/a^\alpha \rightarrow a^\beta; d$ can be applied. There are also two situations in the rule neurons:

(1) If the spikes arrive in rule neuron at different time, then the rule neurons will judge whether the firing condition is satisfied separately. If it is satisfied, then the firing rule will be used and the output value of the potential is $(\alpha \oslash v) \otimes \gamma$, where $\oslash$ is division operator of fuzzy truth values, v is output weight of rule neuron and γ is certain factor, otherwise the firing rule will not be used.

(2) If the spikes arrive at the same time, then the rule neurons use logical operator to judge whether the firing condition is satisfied. The addition operator "$\oplus$" is used to calculate the potential values of spikes received from the predecessor neurons. For example, as in Fig.1(b) , if a rule neuron receives a series of potential values of spikes at the same time, then the potential value will be $\alpha_{in} = x_1 \oplus x_2 \oplus ... \oplus x_k$. So the environment receives the value $\beta_{out} = ((x_1 \oplus x_2 \oplus ... \oplus x_k) \oslash v) \otimes \gamma$.



(a) proposition neurons (b) rule neurons

Fig. 1. Two types of neurons

From the above introduction we know that the WFSN P systems integrate the advantages of SN P systems and fuzzy logic, i.e., the model not only can process information in a distributed and parallel manner but also can perform operations on real numbers by introducing fuzzy logic [35]. These properties greatly broaden the application of the WFSN P systems and make it feasible to solve problems of pattern recognition. One such model has been discussed in [33] where the adjustment of the weights in the network is done by using the Widrow-Hoff learning law. Although the model in [33] only analyzes the method of changing weights and is not used to solve real-life problems. Moreover, its achievements greatly expand the perspective of researchers and provide a feasible principle of weight adjustment.

Inspired by the excellent performance of WFSN P systems, this paper proposes a novel network to solve classification problems in the framework of WFSN P systems. Unlike past studies which only consider the activity of individual neurons [37, 38], this model takes into account the activity of the single neuron in the prefrontal cortex (PFC), which is the higher-order brain structures and has an impact in planning complex cognitive behavior, decision making and moderating social behavior [39]. The experiment in [40] shows that the PFC can mix selectivity in complex cognitive tasks. It means that the neurons in PFC can obtain high-dimensional information by non-linear mixed encoding and the high-dimensional information can help us to make a decision. The LSN P systems are proposed based on the above considerations mentioned above.

3 LSN P Systems

In this section, we introduce the structure of LSN P systems. The learning process and two examples (OR-problem, XOR-problem) are introduced to explain the working of LSN P systems.

3.1 The structure of LSN P systems

The network architecture consists of a feedforward network of proposition neurons and rule neurons with different spiking rules and firing conditions, as shown in Fig.2. The neurons in the network generate a new potential value when the incoming potential crosses the firing condition. Depending on the special style of code found in the brain and the WFSN P systems with more biological characteristics, it is reasonable to combine these two points. The learning system based on Π WFSN P systems is of the following form:

$$\Pi = (A, \left\{\sigma_{p1}^1, ..., \sigma_{pk}^1, \sigma_{p1}^3, \sigma_{p1}^5\right\}, \left\{\sigma_{r1}^2, ..., \sigma_{rn}^2, \sigma_{r1}^4, ..., \sigma_{rn}^4, \right\}, syn, IN, OUT)$$

where:

(1) A=$\{a\}$ is the singleton alphabet(a is called spike).
(2) For each proposition neuron i have the form $\sigma_{pi}^h = \left\{0, w_{ij}^h, \lambda_i^h, r_i^h\right\}$, where h is the label of the layers in the network.
   a) 0 denotes that there is no potential value of pulse in all proposition neurons.
   b) The weights between the proposition neurons and rule neurons are of the form $w_{ij}^1(i = 1, ...k, j = 1, ..., n) = rand(0, 1)$ and $w_{1j}^3(j = 1, ..., n) = 1$. The process of learning is to adjust w_{ij}^1 to find a set of weights with the highest fitness.
   c) The set of rules r_i^h is firing/spiking rule of form $r_i^1 : E^1/a^{\alpha_i} \rightarrow a^{\alpha_i}(i = 1, \ldots, k-1)$, where $E^1 = \left\{\alpha_i \geq \lambda_i^1\right\}$ and $\lambda_i^1(i = 1, ..., k) = 0$. The last neuron is called bias neuron and have the form $r_k^1 : E^1/a \rightarrow a$. The function of bias neuron is discussed in the next subsection; $r_1^3 : E^3/a^o \rightarrow a^o$, where $E^3 = \left\{o \geq \lambda_1^3\right\}$, $\lambda_1^3 = 0$ and $o = \theta_1 \vee \theta_2 \vee ... \vee \theta_n$.
(3) Each rule neuron has the form $\sigma_{rj}^k = \left\{0, 1, 1, \tau_j^k, r_j^k\right\}$, where k is the label of the layers in the network.
   a) 0 denotes that there is no potential value of pulse in the rule neurons.
   b) 1 denotes that the certain factor in the rule neurons.
   c) The weights between the rule neurons and proposition neurons are always 1. In another word, the weights in rule neurons do not participate in the weight adjustment process.
   d) The spiking rules of $\sigma_{r1}^2, ..., \sigma_{rn}^2$ have the form $r_j^2(j = 1, 2...n) : E^2/a^{\theta_j} \rightarrow a^{\theta_j}$, where $E^2 = \{\theta_j \geq \tau_j^2\}$, $\tau_j^2(j = 1, 2, ..., n) = 0$ and $\theta_j = (w_{1j} \otimes \alpha_1) \oplus (w_{2j} \otimes \alpha_2) \oplus ... \oplus (w_{kj} \otimes \alpha_k)$; the neurons $\sigma_{r1}^4, \sigma_{r2}^4, ..., \sigma_{rn}^4$ have the spiking rules r_1^4:$E_j^4/a^{\theta_j} \rightarrow a; d_j$, where $E_j^4 = \{\theta_j \geq \tau_j^4\}$ and $\tau_j^4(j = 1, 2, ..., n) = o$. It means that only the neurons with the maximum potential value of pulse in the previous layer of rule neurons will be activated in this layer.

(4) $syn = \{(\sigma^1_{p1}, \sigma^2_{r1}), (\sigma^1_{p1}, \sigma^2_{r2}), ..., (\sigma^1_{pk}, \sigma^2_{rn}), (\sigma^2_{r1}, \sigma^3_{p1}), (\sigma^2_{r2}, \sigma^3_{p1}), ..., (\sigma^2_{rn}, \sigma^3_{p1}), (\sigma^3_{p1}, \sigma^4_{r1}), (\sigma^3_{p1}, \sigma^4_{r2}), ..., (\sigma^3_{p1}, \sigma^4_{rn}), (\sigma^4_{r1}, \sigma^5_{p1}), (\sigma^4_{r2}, \sigma^5_{p1}), ..., (\sigma^4_{rn}, \sigma^5_{p1})\}$
(5) $IN = \{\sigma^1_{p1}, \sigma^1_{p2}, ..., \sigma^1_{pk}\}$, $OUT = \{\sigma^5_{p1}\}$.

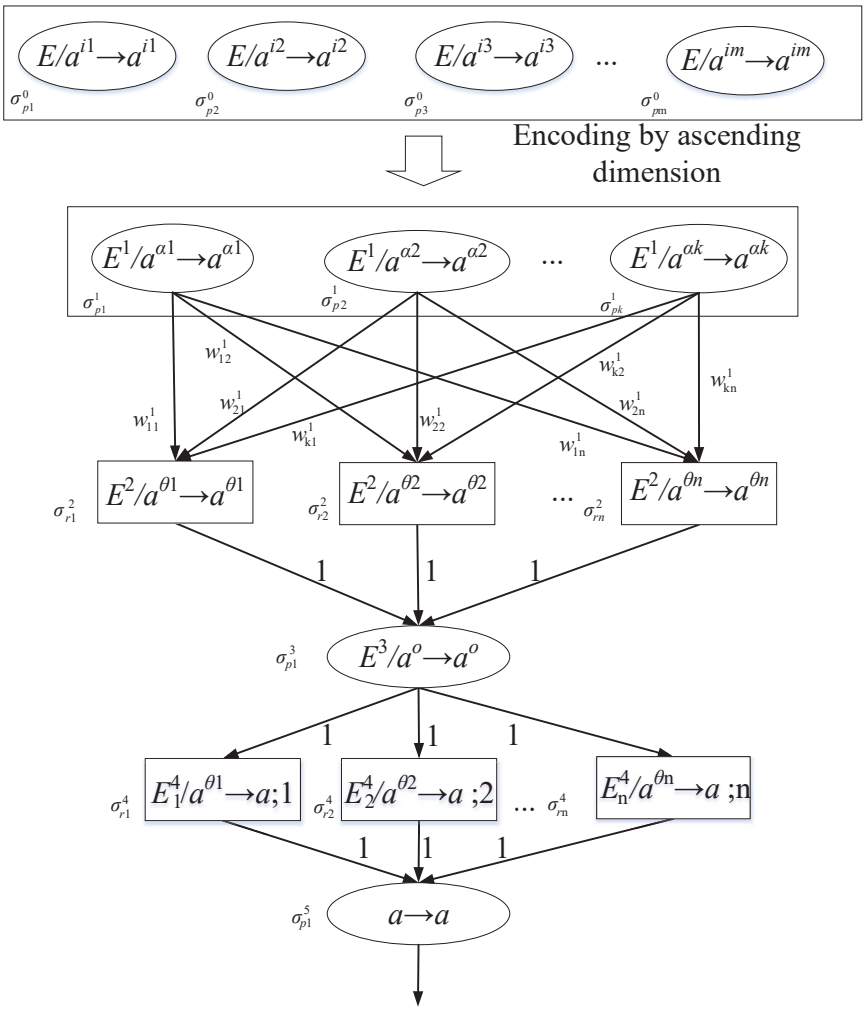


Fig. 2. The structure of learning model

3.2 The learning process of the LSN P systems

In order to process the pattern recognition problem by using the LSN P systems, we assume that:

(1) The LSN P system Π is employed to deal with the learning problem.
(2) The operation of normalization of input data has been processed before entering the system Π.
(3) The parameters of LSN P systems are known, like the weights, thresholds and certainty factor, etc., i.e., the LSN P systems have been initialized.

The LSN P system contains six layers of neurons, each layer in the system has different function. The first layer is the input layer which consists of the neurons $\sigma_{p1}^0, \sigma_{p2}^0, ..., \sigma_{pm}^0$. The input data is fed into the system through these neurons.

The neurons $\sigma_{p1}^1, \sigma_{p2}^1, ..., \sigma_{pk}^1$ $(k > m)$ compose the second layer. The function of this layer is to acquire higher dimensional information. From the biological phenomenons, several scientists have found that the decision making in our brain is through higher-dimensional information rather than the same dimensional information by sensory organs [40]. The most obvious advantage of higher-dimensional information is that it can transform non-linearly separable data into linear separable. However, the real encoding scheme is also not clear so far [40]. Since all pattern recognition problems can be deemed as a problem of finding a smooth curve that meets the requirement of classification [41] and any smooth curve can be represented by a Taylor series. In this case, the Taylor series can be used to achieve the method of ascending dimension.
The Taylor expansion can be described as follows:

$$\begin{aligned} f(x_1, ..., x_n) =& f(x_{1_0}, ..., x_{n_0}) + (h_1 \frac{\partial}{\partial x_1} + ... + h_n \frac{\partial}{\partial x_n}) f(x_{1_0}, ..., x_{n_0}) \\ &+ ... + \frac{1}{k!} (h_1 \frac{\partial}{\partial x_1} + ... + h_n \frac{\partial}{\partial x_n})^k f(x_{1_0}, ..., x_{n_0}) + R_{k+1} \end{aligned} \tag{1}$$

where:

(1) $\overrightarrow{x} = (x_{1_0}, \ldots, x_{n_0})$ is any point on the curve $f(x_1, \ldots, x_n)$;
(2) $h_i = x_i - x_{i_0} (1 \leq i \leq n)$;
(3) $R_{k+1} = \frac{1}{(k+1)!} (h_1 \frac{\partial}{\partial x_1} + \ldots + h_n \frac{\partial}{\partial x_n})^{k+1} f(x_{1_0} + \theta h_1, \ldots, x_{n_0} + \theta h_n)$, where $\theta \in (0, 1)$.

In the Equation.1, if the order of expansion is appropriate, then the R_{k+1} can be ignored because this term is very close to zero. The constant term $f(x_{1_0}, ..., x_{n_0})$ can be used by a neuron at the end of input data as a bias of the LSN P systems. The bias is very important for learning system. If the bias term does not exist in the learning system, this model will be useless because it can not approach that curve more accurately. At last, it is not difficult to see that the value of the output is a linear combination of high-dimension data. So for some complex problems (linear indivisible), the Taylor series is a powerful tool to solve these problems.

The third layer is a hidden layer, the synapses between the second and third layer have weights. The output spikes of last layer are multiplied by the weights and sum the spikes present in the neurons in third layer. The next layer is a

comparison layer, it can compare the potential of spikes which is sent from the last layer. The spikes with the highest potential value is considered as output. The fifth layer is a selection layer. The role of this layer is to select the neuron having the highest value. Furthermore, the neuron with the maximum potential value will fire according to the delay associated with the rules in the subsequent steps.

The last layer is an output layer, the neuron σ_{p1}^{5} will send a spike and the time of firing in the output layer is considered as the result of the system. Then the Widrow-Hoff learning law (Least Mean Square) [42] is used to change the weights between the second and third layer.
The Widrow-Hoff learning law is described in Equation.2:

$$W_{ij}(t+1) = W_{ij}(t) + \alpha(d_i - y_i)x_j(t) \tag{2}$$

where:

(1) W_{ij} is the weight from neuron j to neuron i;
(2) α represents the learning rate;
(3) d_i is the expected output of neuron i;
(4) y_i is the real output of neuron i;
(5) $x_j(t)$ represents the state of neuron j.

3.3 The LSN P systems to identify base classification problem

In this subsection, we use the learning model to classify base classification problems like OR-problem and XOR-problem. In order to let readers have a better understanding of how the network works, we use an example to elaborate on the solution of OR-problem.

The model to solve the OR-problem The OR-problem is a typical linear classification problem, so the technology of ascending dimension is unnecessary. These values are entered directly into the LSN P systems and the truth table can be seen in Table.1.

Table 1. The input and output value of OR-problem.

Input		Output
1	1	1
1	0	1
0	1	1
0	0	0

According to the truth table, the dimension of input data is two and the state of output is also two. So the structure of network can be confirmed and

the weight between σ_{pi}^1 and σ_{rj}^2 is randomly initialized. Suppose that $w_{11}^1 = 0.46, w_{21}^1 = 0.18, w_{31}^1 = 0.67, w_{12}^1 = 0.63, w_{22}^1 = 0.62, w_{32}^1 = 0.49$ and learning rate is set to 0.1. The w_{31}^1 and w_{32}^1 are the weights from the bias neuron.
1^{st}iteration:
$0.46 * 1 + 0.18 * 1 + 0.67 < 0.63 * 1 + 0.62 * 1 + 0.49$, which is inconsistent with real output. So use Widrow-Hoff learning rule to change weights.
After the change, the new weights are $w_{11}^1 = 0.56, w_{21}^1 = 0.28, w_{31}^1 = 0.77; w_{12}^1 = 0.53, w_{22}^1 = 0.52, w_{32}^1 = 0.39$.
$0.56 * 1 + 0.28 * 0 + 0.77 > 0.53 * 1 + 0.52 * 0 + 0.39$, which is consistent with real output, so do not change weights.
$0.56 * 0 + 0.28 * 1 + 0.77 > 0.53 * 0 + 0.52 * 1 + 0.39$, which is consistent with real output, so do not change weights.
$0.56 * 0 + 0.28 * 0 + 0.77 > 0.53 * 0 + 0.52 * 0 + 0.39$, which is inconsistent with real output. So use Widrow-Hoff learning rule to change weights.
After the change, the new weights are $w_{11}^1 = 0.56, w_{21}^1 = 0.28, w_{31}^1 = 0.67; w_{12}^1 = 0.53, w_{22}^1 = 0.52, w_{32}^1 = 0.49$. The accuracy rate is 50%.
2^{nd}iteration:
$0.56 * 1 + 0.28 * 1 + 0.67 < 0.53 * 1 + 0.52 * 1 + 0.49$, which is inconsistent with real output. So use Widrow-Hoff learning rule to change weights.
After the change, the new weights are $w_{11}^1 = 0.66, w_{21}^1 = 0.38, w_{31}^1 = 0.77; w_{12}^1 = 0.43, w_{22}^1 = 0.42, w_{32}^1 = 0.39$.
$0.66 * 1 + 0.38 * 0 + 0.77 > 0.43 * 1 + 0.42 * 0 + 0.39$, which is consistent with real output, so do not change weights.
$0.66 * 0 + 0.38 * 1 + 0.77 > 0.43 * 0 + 0.42 * 1 + 0.39$, which is consistent with real output, so do not change weights.
$0.66 * 0 + 0.38 * 0 + 0.77 > 0.43 * 0 + 0.42 * 0 + 0.39$, which is inconsistent with real output. So use Widrow-Hoff learning rule to change weights.
After the change, the new weights are $w_{11}^1 = 0.66, w_{21}^1 = 0.38, w_{31}^1 = 0.67; w_{12}^1 = 0.43, w_{22}^1 = 0.42, w_{32}^1 = 0.49$. The accuracy rate is 50%.
3^{rd} iteration:
$0.66 * 1 + 0.38 * 1 + 0.67 > 0.43 * 1 + 0.42 * 1 + 0.49$, which is consistent with real output, so do not change weights.
$0.66 * 0 + 0.38 * 1 + 0.67 > 0.43 * 0 + 0.42 * 1 + 0.49$, which is consistent with real output, so do not change weights.
$0.66 * 1 + 0.38 * 0 + 0.67 > 0.43 * 1 + 0.42 * 0 + 0.49$, which is consistent with real output, so do not change weights.
$0.66 * 0 + 0.38 * 0 + 0.67 > 0.43 * 0 + 0.42 * 0 + 0.49$, which is inconsistent with real output. So use Widrow-Hoff learning rule to change weights.
After the change, the new weights are $w_{11}^1 = 0.66, w_{21}^1 = 0.38, w_{31}^1 = 0.57; w_{12}^1 = 0.43, w_{22}^1 = 0.42 = 0.42, w_{32}^1 = 0.59$. The accuracy rate is 75%.
4^{th} iteration:
$0.66 * 1 + 0.38 * 1 + 0.57 > 0.43 * 1 + 0.42 * 1 + 0.59$, which is consistent with real output, so do not change weights.
$0.66 * 1 + 0.38 * 0 + 0.57 > 0.43 * 1 + 0.42 * 0 + 0.59$, which is consistent with real output, so do not change weights.

$0.66 * 0 + 0.38 * 1 + 0.57 < 0.43 * 0 + 0.42 * 1 + 0.59$, which is inconsistent with real output. So use Widrow-Hoff learning rule to change weights.
After the change, the weights are $w_{11}^1 = 0.66, w_{21}^1 = 0.48, w_{31}^1 = 0.67; w_{12}^1 = 0.43, w_{22}^1 = 0.32, w_{32}^1 = 0.49$.
$0.66 * 0 + 0.38 * 0 + 0.67 > 0.43 * 0 + 0.42 * 0 + 049$, which is inconsistent with real output. So use Widrow-Hoff learning rule to change weights.
After the change, the new weights are $w_{11}^1 = 0.66, w_{21}^1 = 0.48, w_{31}^1 = 0.57; w_{12}^1 = 0.43, w_{22}^1 = 0.32, w_{32}^1 = 0.59$. The accuracy rate is 50%.
5^{th} iteration:
$0.66 * 1 + 0.48 * 1 + 0.57 > 0.43 * 1 + 0.32 * 1 + 0.59$, which is consistent with real output, so do not change weights.
$0.66 * 1 + 0.48 * 0 + 0.57 > 0.43 * 1 + 0.32 * 0 + 0.59$, which is consistent with real output, so do not change weights.
$0.66 * 0 + 0.48 * 1 + 0.57 > 0.43 * 0 + 0.32 * 1 + 0.59$, which is consistent with real output, so do not change weights.
$0.66 * 0 + 0.48 * 0 + 0.57 < 0.43 * 0 + 0.32 * 0 + 0.59$, which is consistent with real output, so do not change weights. The accuracy rate is 100%.

Stop training and save this set of weights. The OR problem can be effectively identified by these set of weights.

The model to solve XOR-problem The XOR-problem is a classical non-linearity problem. The ability to solve XOR-problem can be very competitive and next we discuss how to solve the XOR-problem.

Since input data is non-linear, the idea of raising dimension is necessary to solve this problem. The formula for raising the dimension can be described in the following manner:

$$(x_1, x_2) \to (x_1, x_2, x_1^2, x_2^2, x_1 x_2) \tag{3}$$

From the Equation.3, it can be found that the order of Taylor expansion for this problem is two. The main reason for only choosing two is that the XOR problem a very simple non-linear model. For more complex non-linear data, the higher order Taylor expansion needs to be employed. So these higher-dimensional information are input to the model and the accuracy curve is shown in Fig.3.

4 Experimental Results

In this section, we discuss the learning model to solve some real-life application problems. The performance of binary classification problems and multiple classification problems are presented and compared with some other models in the existing literature.

4.1 The pattern recognition of binary classification problems

The binary classification problem is very general in real-life and it is also a relatively simple pattern recognition problems. Only if a model can solve binary

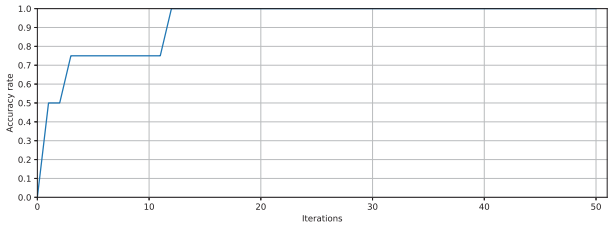


Fig. 3. The accuracy curve of XOR-problem

classification problem, it is able to solve multi-classification problem and have the possibility of generalization. The *Wisconsin breast cancer dataset* (WBCD) is a classic binary classification problem. This dataset consists of nine labels, such as age, tumor-size, inv-nodes, node-caps, deg-malign, irradiated, etc. The classes of this dataset are recurrence and no-recurrence. Since the input data consists of only real numbers, it is necessary to make input data normalized. Under the circumstances, the linear normalization is proposed to solve this problem. The formula can be constructed as follows:

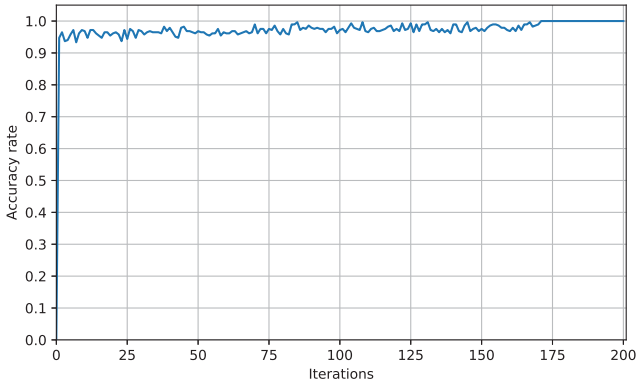
$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4}$$

After linear normalization, also adopt the coding scheme of rising dimension. The degree of Taylor expansion, in this case, is chosen as three in the coding scheme. The data is divided into two parts. One is training set, which accounts for 50% of the total, the other is testing set. The classification result can be seen in Fig.4(a), the accuracy rate of training set is 100 % at the end. After training the training set, the parameters of the model are saved and these parameters are used to test the ability of generalization. The result of testing set is 98.6%, as shown in Fig.4(b). The above simulation is executed many times to avoid accidental situations and the average accuracy is shown in Table.2 to compare with other algorithms.

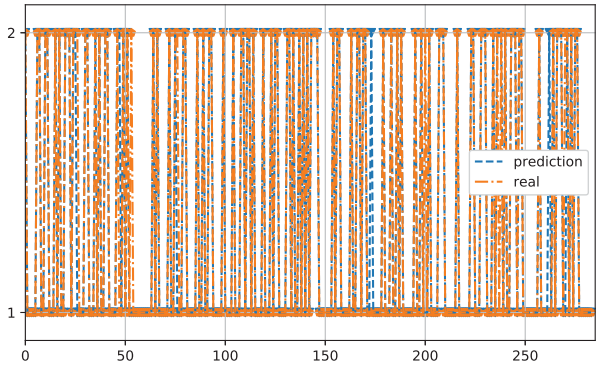
Table 2. Comparison with other methods on WBCD dataset.

	Max epochs	Training	Testing
BPNN [43]	$9.2 \cdot 10^6$	98.1%	96.3%
LM [43]	3500	97.7%	96.7%
SpikeProp[43]	1500	97.6%	97.0%
SRESNN[44]	306	97.7%	97.2%
MuSpNN[45]	100	98.2%	96.4%
Proposed	**200**	**99.5%**	**97.4%**

From Table.2 , we can see that the accuracy rate of proposed learning model in both training set and testing set are the best ones and in comparison with other models, the numbers of iteration steps are smaller. More specifically, the proposed learning model not only have a strong learning ability but also has a good generalization ability in binary classification problems when compared with other methods in [43–45].



(a) Training set



(b) Testing set

Fig. 4. The result on WBCD dataset

4.2 The pattern recognition of multiple classification problems

From the analysis of the last subsection, we can say that the model can solve binary classification problem commendably. In this section, the IRIS dataset is employed to test whether this model can solve multi-classification problem.

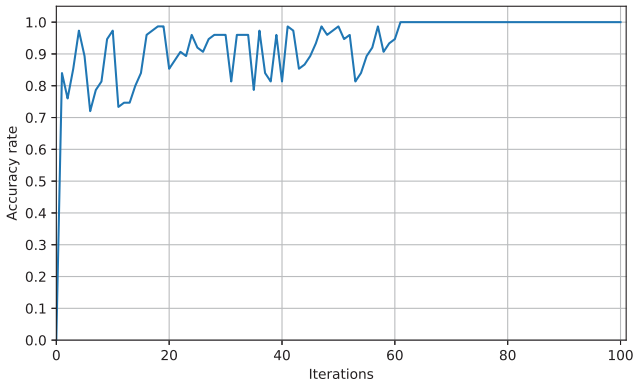
The IRIS dataset, also known as the iris flower dataset, is a collection of multivariate datasets. One hundred and fifty data are contained in the dataset which is divided into three classes evenly. It is possible to predict the species of the iris flower (Setosa, Versicolour, Virginia) by the length of sepal, the width of sepal, the length of petal, and the width of petal. The data set is divided into two parts, one part is training set which accounts for 50% of the total dataset. The other part is testing set. The learning parameters in *IRIS dataset* are same as *WBCD dataset* except the degree of Taylor expansion. The degree of Taylor expansion is chosen as four in this case. The classification result of training set is shown in Fig.5(a). After obtaining the parameters of the model, the testing set is used to test the ability of generalization. The result of testing set can be seen in Fig.5(b). In order to avoid the situation caused by randomized initial weights, repeat the above simulation many times. Table.3 shows that the average accuracy of this model and the comparisons in which some other algorithms also using the same dataset.

Table 3. Comparison with other methods on IRIS dataset.

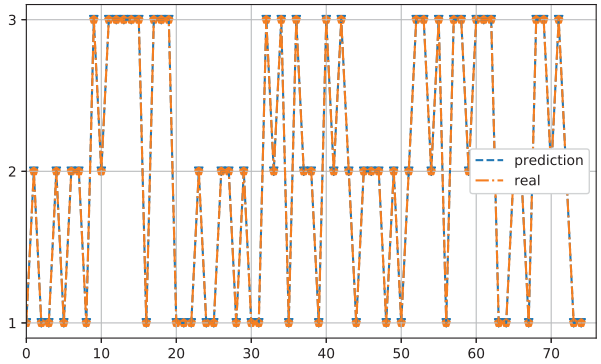
	Max epochs	Training	Testing
BPNN[43]	$2.6 \cdot 10^6$	98.2%	95.5%
LM[43]	3750	99%	95.7%
SpikeProp[43]	1000	97.4%	96.1%
SRESNN[44]	102	96.9%	97.3%
MuSpNN[45]	100	99.8%	95.7%
Proposed	**100**	**98.6%**	**97.5%**

From the result, we can infer that the proposed learning system can solve multiple classification problems easily. The experimental result says that although the accuracy rate in training set is not the highest one when compared with other algorithms in Table.3, the testing set is the best one. It says that the generalization ability of the proposed learning system is better than other algorithms.

From the result in Table.2 and Table.3, we can find that the proposed learning system is an excellent classifier. The implementation of LSN P systems can achieve pattern recognition problems easily. In addition, not only in the field of accuracy rate, but also the number of iterations has advantages. In short, the LSN P systems are a method that is worth popularizing and need further investigation.



(a) Training set



(b) Testing set

Fig. 5. The result in IRIS dataset

5 Concluding Remarks

This article proposes a model named LSN P systems to deal with pattern recognition problem. In this paper, a feasible way to use P systems to solve pattern recognition problem is proposed. In order to obtain the effectiveness of this model, the motivation, the description of algorithm and the experimental results are presented in the article. This work is inspired by several machine learning algorithms and the models of P systems. Although this work is not the first attempt to construct a learning model by using SN P systems and its variants, the results are gratifying and competitive when compared with other algorithms. Although more work needs to be done in order to be competitive

with existing machine learning algorithms. One of the biggest advantages of the proposed model is that it combines the phenomena in biology with some machine learning algorithms to make a model having human-level performance in solving problems.

Although this work is promising, two areas of this model need further improvement: One is for the multi-classification problem, in this model the output pattern can be described as *one against all* [46]. But when the number of training set increases, the complexity of problems increase rapidly, making the problem of computing speed more prominent. In this condition, whether we can find a way to solve the multi-classification problem efficiently is a very important scheme in this model. The second one is the way to regularize the network. As pointed out in [40] that nonlinear mixed selectivity is useful but also fragile. So the problem is to determine the degree of Taylor expansion or to add regularization items to the network.

6 Acknowledgment

This work was supported by the National Natural Science Foundation of China (61972324, 61672437, 61702428), the Sichuan Science and Technology Program (2018GZ0185, 2018GZ0086), Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06) and the postgraduate workstation in the State Grid Sichuan Electric Power Research Institute.

References

1. Rosenblatt, F.:The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review **65**(6), 386-408(1958).
2. McCulloch, W. S., Pitts, W.:A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, **5**(4), 115-133(1943).
3. Hopfield,J. J.:Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences,**79**(8), 2554-2558(1982).
4. Rumelhart, D. E., Hinton, G. E., Williams, R. J.:Learning representations by back-propagating errors. Cognitive Modeling, **5**(3), 1(1988).
5. Huang, G. B., Zhu, Q. Y., Siew, C. K.:Extreme learning machine: a new learning scheme of feedforward neural networks. Neural networks, **2**, 985-990(2004).
6. Ghoshdastidar, S., Adeli, H.:Spiking neural networks. International Journal of Neural Systems, **19**(4), 295-308(2009).
7. Johnson, J. L., Padgett, M. L.:PCNN models and applications. IEEE Transactions on Neural Networks, **10**(3), 480-498(1999).
8. ElSawy, A., Hazem, E. B., Loey, M.:CNN for handwritten arabic digits recognition based on LeNet-5. In: International Conference on Advanced Intelligent Systems and Informatics pp. 566-575. Springer, Cham(2016).
9. Wu, C.:Discriminant-function-based minimum recognition error rate pattern-recognition approach to speech recognition. Proceedings of the IEEE, **88**(8), 1201-1223(2000).

10. Prun, V.:Geometric filtration of classification-based object detectors in realtime road scene recognition systems.In: Eighth International Conference on Machine Vision pp.98750O. Barcelona(2015).
11. Sridar, K., Shanthi, D.:Web based medical diagnosis system using ann-arm for the diabetes mellitus. International Journal of Computers and Distributed Systems, **3**(3), 15-20(2013).
12. Bohte, S. M., Kok, J. N.:Applications of spiking neural networks. Information Processing Letters, **6**(95), 519-520(2005).
13. Maass, W.:Networks of spiking neurons: the third generation of neural network models. Neural Networks, **10**(9), 1659-1671(1997).
14. Siddique, N., McDaid, L., Kasabov, N., Widrow, B.:Special issue: Spiking neural networks introduction, International Journal of Neural Systems. **20**(6), v–vii(2010).
15. Ridet, J. L., Privat, A., Malhotra, S. K., Gage, F. H.:Reactive astrocytes: cellular and molecular cues to biological function. Trends in Neurosciences, **20**(12), 570-577(1997).
16. Bradl, M., Lassmann, H.:Oligodendrocytes: Biology and Pathology. Acta Neuropathologica, **119**(1), 37-53(2010).
17. Păun, Gh.:Computing with membranes, Journal of Computer and System Sciences, **61**(01), 108–143(2000).
18. Martín-Vide, C., Păun,Gh., Pazos, J., Rodríguez-Patón, A.:Tissue P systems. Theoretical Computer Science, **296**(2), 295-326(2003).
19. Ionescu, M., Păun,Gh., Yokomori, T.:Spiking neural P systems. Fundamenta Informaticae, **71**(2, 3), 279-308(2006).
20. Orellana-Martín, D., Valencia-Cabrera, L., Riscos-Núnez, A., Pérez-Jiménez, M. J., P systems with proteins: a new frontier when membrane division disappears. Journal of Membrane Computing, **1**(1), 29-39(2019).
21. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Peng, H.:Membrane computing and image processing: a short survey. Journal of Membrane Computing, **1**(1), 58-73(2019).
22. Orellana-Martín, D., Valencia-Cabrera, L., Riscos-Núnez, A., Pérez-Jiménez, M. J.:Minimal cooperation as a way to achieve the efficiency in cell-like membrane systems. Journal of Membrane Computing, **1**(2), 29-39(2019).
23. Nash, A., Kalvala, S.:A P system model of swarming and aggregation in a Myxobacterial colony, Journal of Membrane Computing. **1**(2), 103-111(2019).
24. Pan, L., Wang, J., Hoogeboom, H. J.:Spiking neural P systems with astrocytes. Neural Computation, **24**(3), 805-825(2012).
25. Pan, L., Păun, Gh.:Spiking neural P systems with anti-spikes. International Journal of Computers Communications & Control, **4**(3), 273-282(2009).
26. Cabarle, F. G. C., Adorna, H. N., Pérez-Jiménez, M. J., Song, T.:Spiking neural P systems with structural plasticity. Neural Computing and Applications, **26**(8), 1905-1917(2015).
27. Díaz-Pernil, D., Peña-Cantillana, F., Gutiérrez-Naranjo, M. A.:A parallel algorithm for skeletonizing images by using spiking neural P systems. Neurocomputing, **115**, 81-91(2013).
28. Zhang, G., Rong, H., Neri, F., Pérez-Jiménez, M. J.:An optimization spiking neural P system for approximately solving combinatorial optimization problems. International Journal of Neural Systems, **24**(05), 1440006(2014).
29. Wang, T., Zhang, G., Zhao, J., He, Z., Wang, J., Pérez-Jiménez, M. J.:Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems. IEEE Transactions on Power Systems, **30**(3), 1182-1194(2014).

30. Wang, T., Zhang, G., Pérez-Jiménez, M. J., Cheng, J.:Application of weighted fuzzy reasoning spiking neural P systems to fault diagnosis in traction power supply systems of high-speed railways. Journal of Computational and Theoretical Nanoscience, **12**(7), 1103-1114(2015).
31. Li, J., Huang, Y., Xu, J.:Decoder Design Based on Spiking Neural P Systems. IEEE Transactions on Nanobioscience, **15**(7), 639-644(2016).
32. Song, T., Pan, L., Wu, T., Zheng, P., Wong, M. D., Rodríguez-Patón, A.:Spiking neural P systems with learning functions. IEEE Transactions on Nanobioscience, **18**(2), 176-190(2019).
33. Wang, J., Peng, H.:Adaptive fuzzy spiking neural P systems for fuzzy inference and learning. International Journal of Computer Mathematics, **90**(4), 857-868(2013).
34. Wang, J., Shi, P., Peng, H., Pérez-Jiménez, M. J., Wang, T.:Weighted fuzzy spiking neural P systems. IEEE Transactions on Fuzzy Systems, **21**(2), 209-220(2012).
35. Zhang, G., Pérez-Jiménez, M. J., Gheorghe,M.:Real-life applications with membrane computing. Springer, Berlin(2017).
36. Abbott, L F.:Lapique's introduction of the integrate-and-fire model neuron. Brain Research Bulletin, **50**(5), 303-304(1999).
37. Gerstner W, Kistler W M.:Spiking neuron models: Single neurons, populations, plasticity. Cambridge University Press, New York(2002).
38. Abbott L F.:Lapique's introduction of the integrate-and-fire model neuron. Brain Research Bulletin, **50**(5), 303-304(1999).
39. Yang, Y., Raine, A.:Prefrontal structural and functional brain imaging findings in antisocial, violent, and psychopathic individuals: a meta-analysis. Psychiatry Research: Neuroimaging, **174**(2), 81-88(2009).
40. Rigotti, M., Barak, O., Warden, M. R., Wang, X. J., Daw, N. D., Miller, E. K., Fusi, S.:The importance of mixed selectivity in complex cognitive tasks. Nature, **497**(7451), 585(2013).
41. Bishop, C. M.:Pattern recognition and machine learning. Springer, Berlin(2006).
42. Wang, Z. Q., Manry, M. T., Schiano, J. L.:LMS learning algorithms: misconceptions and new results on converence. IEEE Transactions on Neural Networks, **11**(1), 47-56(2000).
43. Bohte, S. M., Kok, J. N., La Poutre, H.:Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing, **6**, 17-37(2002).
44. Dora, S., Subramanian, K., Suresh, S., Sundararajan, N.:Development of a self-regulating evolving spiking neural network for classification problem. Neurocomputing, **171**, 1216-1229(2016).
45. Taherkhani, A., Belatreche, A., Li, Y., Maguire, L. P.:A supervised learning algorithm for learning precise timing of multiple spikes in multilayer spiking neural networks. IEEE Transactions on Neural Networks and Learning Systems, **29**(11), 5394-5407(2018).
46. Hsu, C. W., Lin, C. J.:A comparison of methods for multiclass support vector machines. IEEE Transactions on Neural Networks, ,**13**(2), 415-425(2002).

Multi-stage stratified sampling for national grain processing loss and waste census

Zhao Yao[1], Jiangrong Liu[1] *, Kang Zhou[1], Huaqing Qi[2], Falin Jiang[1], and Jian Zhou[3]

[1] School of Math and Computer,
Development Strategy Institute of reserve of food and material,
Wuhan Polytechnic University, Wuhan 430023, China
[2] Development Strategy Institute of reserve of food and material,
Wuhan Polytechnic University, Wuhan 430023, China
[3] School of Food Science and Engineering,
Wuhan Polytechnic University, Wuhan 430023, China
{371663936@qq.com}

Abstract. In order to accurately estimate and evaluate the loss and waste of grain processing links in China, it is necessary to conduct in-depth research and obtain information from enterprises. However, there are nearly 10,000 enterprises in China, and it is impossible to conduct research one by one. Therefore, it is imperative to select representative samples reasonably. In order to complete the sampling and make the sample size more reasonable, on the basis of fully mining the existing historical statistical data, a three-stage stratified sampling is proposed and a sampling plan is formulated. First, preliminary sampling is conducted based on historical data. In the first stage, the output ratio of each province is calculated based on historical data. The national grain processing enterprises are divided into different regions (layers) according to the output ratio of each province. To calculate the sample size of each region. In the second stage, each region is further stratified by province. To calculate the sample size of different provinces within each layer. In the third stage, each province is further stratified by enterprises of different scales. To calculate the sample size of enterprises of different scales in each province. Then, based on the sample data of preliminary sampling, we calculate the loss rate of each province to replace the output ratio of each province in the preliminary sampling. So that the sample distribution of the second sampling is determined by the three-stage stratified sampling method again. This paper takes rice processing enterprises as an example to analyze and determine the number of enterprises of different scales in different provinces in China for the census.

Keywords: grain processing, loss and waste, sample size, multi-stage stratified sampling

* Corresponding author.

1 Introduction

Food security has always been a major strategic issue that affects China's economic development, social stability, and national independence. In 2013, the State Council's special plan, "Food Security Project Construction Plan (2013-2020)" explicitly proposed the overall strategic plan "to reduce post-production losses and waste and promote sustainable development of food security". There are many grain processing enterprises in China. At present, the phenomenon of loss and waste in grain processing links is very prominent. However, China is a country with extremely scarce resources. The relationship between grain supply and demand is tight for a long time. There is great potential and strategic significance to reduce the loss and waste of grain processing links. Therefore, the census of loss and waste in the grain processing links in China is a matter that needs to be solved urgently. The sampling of grain processing enterprises is the first problem to be faced. A good sampling can more accurately solve a series of practical problems in investigation and evaluation of losses in post-production processing of grain and oil crops such as rice, wheat, corn, soybean, peanut and rapeseed. The causes and influencing factors of the loss and waste of grain processing are investigated from the perspectives of technology, management and market. It is necessary to put forward concrete measures and action plans to reduce the loss and waste of grain processing links. Therefore, it is of great practical value to conduct census sampling on the processing links of grain processing enterprises.

Among the frequently used sampling organization methods, the stratified sampling has the smallest sampling variance (error), and the best sampling effect is widely used in the sampling inference of economic phenomena such as resources, population, and living standards of residents[1–4]. However, in the results of the existing research sampling methods, there is no sampling for the national grain processing link census data. Therefore, we can learn from the design ideas improved by the traditional sampling method, and design applicable and effective sampling methods based on the historical data of the national grain. Meanwhile, we propose multi-stage stratified sampling based on stratified sampling which has improved stratified sampling theoretically. It is a new and operational sampling method that provides a more effective solution for national grain processing census sampling.

In order to accurately calculate and evaluate the specific situation of the loss and waste of grain processing links in China, obtain detailed and accurate basic data, solve problem such as incomplete and inaccurate data of loss and waste in grain processing link and the impossibility of longitudinal tracking, comparison and monitoring, we need to obtain information through research. However, there are nearly 10000 processing enterprises in China, so it is particularly important to take representative samples reasonably. Therefore, this paper proposes a multi-stage stratified sampling method, which can provide scientific methods and theories for the sampling of grain processing links. At the same time, this sample survey system can also be used for survey research, program design, etc. in other related fields. Specifically, first of all, according to the historical sta-

tistical data of grain, the sample size of the preliminary sample is determined. By calculating the output ratio of various types of grain in each province, according to the characteristics of the regional distribution of grain processing of enterprises and the principle of proportional distribution of stratified sampling, we propose and design a three-stage stratified sampling method to achieve the sample distribution of preliminary sampling. Then, based on the sample data of preliminary sampling, the sample size of the second sampling is determined by the sample size determination method. The provincial loss rate of each province is calculated, and the three-stage stratified sampling method is used again to realize the sample distribution of the second sampling. The method assigns different numbers of samples to different regions according to the different grain output ratios of the regions, which makes the sampling process scientific and reasonable, and the sampling data is more representative.

2 Data source and analysis

"Statistical data of grain and oil processing industry" (2008 – 2013) [5] is the internal data of the Circulation and Technology Development Department of the State Grain Bureau. It has carried out the detailed statistics to the grain and oil processing enterprise's basic information that year, each kind of grain and oil product actual output, the production capacity and so on. In order to estimate the loss and waste of national grain processing links, we screened and sorted out the data, and sorted out the annual output table, annual processing capacity and the number of processing enterprises in the past years of rice, wheat, soybeans, peanuts, rapeseed and other grains, and established a database. For some data of missing years, it is searched and collected through the State Grain Bureau [6], the official website of the grain bureaus at all levels, and the official website of the National Bureau of Statistics [7]. However, for the data that is not really available, the gray prediction model is used for prediction.

Preliminary sampling based on historical data, preliminary samples taken nationwide, sample distribution should cover each province as much as possible, and more samples should be collected in major grain processing areas. After obtaining the sample data of the preliminary sampling, it can be used to calculate the provincial loss rate of each province. If there is missing data in a province, we can calculate the average loss rate of large, medium and small-scale enterprises in China, and then use the processing capacity of large, medium and small-scale enterprises in each province as the weight to calculate the provincial loss rate of the province. The loss rate of each province is used to replace the output ratio of each province in the historical data of the preliminary sampling, and the sample distribution of the second sampling is determined by the three-stage stratified sampling method of the preliminary sampling.

3 Sample size determination method

The determination of sample size is an important link in sampling investigation, which is directly related to the accuracy of sampling estimation and the cost and benefit of investigation. Let the random variable have a mean of μ and a variance of σ^2. Then the n samples have a mean of μ and a variance of σ^2/n. At a given confidence level $1-\alpha$, let the mean of the sample be $\bar{\mathrm{X}}$, and the error ε is given by the following formula: $\varepsilon = \left|\bar{X} - \mu\right|$. In most cases, σ^2 is unknown. In order to eliminate the influence of σ^2, some scholars have introduced a t-distribution. The above expression is a t-distribution with a degree of freedom of $n-1$, where S is the variance of the sample $t = \frac{\bar{X}-\mu}{S/\sqrt{n}}$, then $\varepsilon = \left|\bar{X} - \mu\right| = t_\alpha(n-1)\frac{S}{\sqrt{n}}$. The formula for obtaining n is:

$$n = \frac{t_\alpha^2(n-1)S^2}{\varepsilon^2}.$$

When the sample size is large and the confidence level $\alpha < 0.05$, the following formula can generally be approximated:

$$n = \frac{4S^2}{\varepsilon^2}. \text{ [8–11]}$$

In order to make the samples more representative, we should take samples from each province nationwide, and collect more samples from major grain production areas. In preliminary sampling, based on the above coverage principle, the number of preliminary samples collected is shown in Table 1.

Table 1. Sample size of preliminary sampling of various varieties of grain processing

Variety	rice	wheat	Soybean oil	Rapeseed oil	Peanut oil	corn
Number of samples	30	30	20	20	20	20

Based on the preliminary sampling results, the loss rate of each province and its sample variance of the preliminary sampling are calculated. We controlled the error at about 10% and substituted it into the sample size calculation formula to get the approximate number of the second sampling of each variety as shown in Table 2.

4 Design and implementation of three-stage stratified sampling method

4.1 Overall design of three-stage stratified sampling method

Taking into account the regional differences of each province and the differences in the number and scale of grain processing enterprises, separate sampling is

Table 2. Sample size of the second sampling of various varieties of grain processing

Variety	rice	wheat	Soybean oil	Rapeseed oil	Peanut oil	corn
Number of samples	150	100	50	50	50	50

conducted within each province. By obtaining the sample size of each province, the results can be predicted more accurately. Therefore, stratified sampling is adopted, and 28 provinces and autonomous regions represent different subgroups and levels. However, this is not enough. There are too many levels. The 28 provinces and autonomous regions need to be further stratified. Based on the principle that the intra-layer variance is the smallest and the variance between the layers is the largest, we divide the national grain processing enterprises into different regions according to the output ratio of each province, and then determine the sample size of each region.

First, preliminary sampling is conducted based on historical data. The sample size of the preliminary sample can be determined. In the first stage, the national grain processing enterprises are divided into different regions (layers) according to the output ratio of each province. For example, the rice is divided into 6 layers by province, that is, 6 regions. And calculating the sample size of each region. In the second stage, determining the sample size of each province in the 6-layer sampling area. In the third stage, each province is further stratified by enterprises of different scales. And to calculate the sample size of enterprises of different scales in each province. Different scales are large, medium and small scales. Enterprises with a daily production capacity of more than 400 tons are called Large-sized enterprises. Enterprises with a daily production capacity of 200-400 tons are called Medium-sized enterprises. Enterprises with a daily production capacity of less than 200 tons are called Small-sized enterprises.

Preliminary samples are taken nationwide through the above three-stage stratified sampling method. The sample distribution should cover each province as much as possible, and more samples should be collected in major grain processing areas. After obtaining the sample data of the preliminary samples, the sample capacity of the second sample is determined by the sample size determination method. The loss rate of each province can be calculated according to the sample data of the preliminary sampling. If there is missing data in a province, we can calculate the average loss rate of large, medium and small-scale enterprises in China, and then use the processing capacity of large, medium and small-scale enterprises in each province as the weight to calculate the provincial loss rate of the province. The loss rate of each province is used to replace the output ratio of each province in the historical data of the preliminary sampling, and the sample distribution of the second sampling is determined by the three-stage stratified sampling method of the preliminary sampling.

4.2 Implementation of three-stage stratified sampling method

To build an enterprise sample size allocation algorithm, the key is to determine the weight factor of the corresponding layers in each stage, so as to obtain the corresponding allocated sample size.

The three-stage stratified sampling method for preliminary sampling is designed as follows:

(1) **Stratify provinces**

Based on the principle that the intra-layer variance is the smallest and the variance between the layers is the largest, divide the national grain processing enterprises into different regions according to the output ratio of each province. The rice output ratio of each province is calculated as follows:

$$X_t = \frac{Q_{t0}}{Q_{t1}}, (t = 1, 2, \cdots, 31)$$

where X_t is the output ratio of province t , Q_{t0} is the rice output of the previous year (known data), and Q_{t1} is the rice processing capacity of the previous year (known data).

If the output ratio of the individual provinces is abnormal, it is listed as a single layer, using intra-layer random sampling. For example, in the sampling of rice processing enterprises, Guangdong and Shanxi are separately listed as two layers. In general, stratified sampling continues.

(2) **Under the condition that the total number of processing enterprises n is constant, the sample size n_i of the i-th layer (take 4-layer as an example, that is, $i = 1, 2, 3, 4$) is calculated as follows:**

$$n_i = n \cdot \omega_i(\beta) = n \cdot \frac{\varphi_i + \beta \cdot \phi_i}{1+\beta} (i = 1, 2, 3, 4) \qquad 15 \leq n_i \leq 35$$

$$\phi_i = \frac{\Delta X_i}{\sum_{i=1}^{4} \Delta X_i}, \Delta X_i = \sum_{j=1}^{n_i} \left(X_{ij} - \bar{X}_i\right)^2, \varphi_i = \frac{Q_{i1}}{\sum_{i=1}^{4} Q_{i1}}$$

where $\omega_i(\beta)$ is the sample size weighting factor of the i-th layer; φ_i is the variance ratio of the output ratio of the i-th layer, ΔX_i is the variance of the output ratio of the i-th layer, and X_{ij} is the output ratio of the j-th province of the i-th layer, $\bar{X}_i$ is the mean of output ratio of the i-th layer; ϕ_i is the ratio of processing amount of the i-th layer, Q_{i1} is the processing amount of the i-th layer; β is the adjustment parameter, and adjusts the proportional relationship between φ_i and ϕ_i , so that the final sample size distribution tends to be reasonable. For example, when β is taken as 1, it means that the two share the same proportion.

(3) **The formula for calculating the sample size n_{ij} of the j-th province of the i-th layer is as follows:**

$$n_{ij} = n_i \cdot \omega_{ij}(\lambda) = n_i \cdot \frac{Q_{ij}+\lambda\eta}{1+\lambda}$$
$$\eta = \frac{\sum_{k=1}^{3}(\alpha_{ijk}Z_{ijk})}{\sum_{m=1}^{j}(\sum_{k=1}^{3}\alpha_{imk}Z_{imk})}$$
$$(i = 1, 2, 3, 4; j = 1, 2, 3, \cdots n_i; m = 1, 2, 3, \cdots j; k = 1, 2, 3)$$

where $\omega_{ij}(\lambda)$ is the sample size weighting factor of the j-th province of the i-th layer; Q_{ij} is the ratio of processing amount of the j-th province of the i-th layer to the total processing amount of the i-th layer; α_{ijk} is the average processing capacity of the k-th scale of the j-th province of the i-th layer ($k = 1, 2, 3$ means large, medium and small scale); Z_{ij1} is the number of large-scale processing enterprises of the j-th province of the i-th layer, Z_{ij2} is the number of medium-scale processing enterprises of the j-th province of the i-th layer, Z_{ij3} is the number of small-scale processing enterprises of the j-th province of the i-th layer; λ is the adjustment parameter, is the proportion of adjustment processing amount and number of enterprises.

(4) **The formula for calculating the sample size n_{ijk} of processing enterprises of different scales of the j-th province of the i-th layer is as follows:**

$$n_{ijk} = n_{ij} \cdot \omega_{ijk} = n_{ij} \cdot \frac{\alpha_{ijk}Z_{ijk}}{\sum_{k=1}^{3}(\alpha_{ijk}Z_{ijk})}$$
$$(j = 1, 2, 3, \cdots n_i; i = 1, 2, 3, 4; k = 1, 2, 3)$$

where ω_{ijk} is the sample size weighting factor of processing enterprises of different scales of the j-th province of the i-th layer.

The sampling process of the second sampling is basically the same as the preliminary sampling process. Based on the sample data of preliminary sampling, the loss rate Y_t of each province can be calculated. The loss rate of each province is used to replace the output ratio of each province in the historical data of the preliminary sampling. The three-stage stratified sampling method of the preliminary sampling design is used again to calculate the sample size n_{ijk} of processing enterprises of different scales of the j-th province of the i-th layer. That a sample distribution for the second sample has been achieved.

Through multi-stage stratified sampling, we get the sample number of investigating rice processing enterprises of different scales in each province. This multi-stage stratified sampling method can also be applied to the sampling survey of processing enterprises for the loss and waste of wheat, corn, soybean and other grains. In order to make sampling more reasonable, we can continue to expand the sample size to further investigate the loss and waste data of grain processing in enterprises, and further decrease the loss rate of each province through the survey data, so as to obtain a more reasonable sampling of enterprises for the general survey of loss and waste of grain processing.

5 Sample distribution results of three stages stratified sampling

Taking rice as an example, we obtained the sample distribution results of the preliminary sampling and the second sampling by the three-stage stratified sampling method.

Based on historical data, we obtained the sample distribution results of the preliminary sampling of rice as shown in Table 3.

Table 3. The sample distribution results of the preliminary sampling of rice

output ratio	Rice processing area	Planting area (thousand hectares)	Yield ratio	Total number of samples in the region	Number of samples in each province	Number of samples of each scale		
						large-scale	medium-scale	small-scale
0.922034	Guangdong	1893.3	0.030186	2	2	1	1	0
0.764706	Hainan	312.2	0.001365	8	1	1	0	0
0.75	Beijing	0.2	0.004498		1	0	0	1
0.72	Shanghai	98.4	0.006024		1	1	0	0
0.702128	Zhejiang	824.2	0.018874		5	3	1	1
0.698113	Shanxi	123.4	0.004257	4	0	0	0	0
0.690476	Heibei	84.8	0.003373		0	0	0	0
0.689655	Xinjiang	75.1	0.002329		0	0	0	0
0.684579	Fujian	804.5	0.034375		1	1	0	0
0.681818	Tianjin	16.4	0.001767		0	0	0	0
0.677966	Liaoning	562.1	0.028431		0	0	0	0
0.677419	Shandong	122.4	0.00249		0	0	0	0
0.674058	Jiangsu	2271.7	0.072444		1	1	0	0
0.673945	Heilongjiang	3205.5	0.110353		2	1	1	0
0.666667	Yunnan	1144.7	0.004337	4	0	0	0	0
0.666667	Ningxia	78.1	0.007951		0	0	0	0
0.654321	Jilin	747.1	0.032528		1	1	0	0
0.652174	Chongqing	689.7	0.011083		0	0	0	0
0.643373	Henan	649.7	0.033331		1	1	0	0
0.638009	Guangxi	2026.2	0.01775		0	0	0	0
0.637213	Hunan	4120.7	0.094531		2	1	1	0
0.633731	Sichuan	1991.8	0.047145	11	1	1	0	0
0.633333	Guizhou	682	0.007228		0	0	0	0
0.630161	Anhui	2217.3	0.140069		3	1	1	1
0.605823	Hubei	2144	0.173801		4	2	1	1
0.6	Neimenggu	78.1	0.001606		0	0	0	0
0.588133	Jiangxi	3339.5	0.138061		3	2	1	0
0.333333	Shanxi	0.9	0.000267	1	1	1	0	0

Based on the sample data of preliminary sampling, we get the sample distribution results of the second sampling of rice as shown in Table 4.

Table 4. The sample distribution results of the second sampling of rice

Sampling area	Lose rate	Rice processing area	Planting area (thousand hectares)	Yield ratio	Total number of samples in the region	Number of samples in each province	Number of samples of each scale		
							large-scale	medium-scale	small-scale
1	17.79676913	Sichuan	1991.8	0.047145	45	28	16	7	5
	17.06668962	Shanxi	0.9	0.000267		17	17	0	0
2	16.4945402	Heibei	84.8	0.003373	51	1	1	0	0
	16.44633235	Neimenggu	78.1	0.001606		0	0	0	0
	16.43198014	Shanghai	98.4	0.006024		1	1	0	0
	16.40365849	Hubei	2144	0.173801		13	7	3	3
	16.34789773	Beijing	0.2	0.004498		1	0	0	1
	16.33666389	Jilin	747.1	0.032528		3	2	1	0
	16.3257105	Henan	649.7	0.033331		2	1	1	0
	16.29928436	Hunan	4120.7	0.094531		8	4	3	1
	16.29363861	Jiangsu	2271.7	0.072444		6	3	2	1
	16.27398525	Xinjiang	75.1	0.002329		0	0	0	0
	16.27320983	Fujian	804.5	0.034375		3	1	1	1
	16.25490384	Heilongjiang	3205.5	0.110353		13	7	4	2
3	16.19032566	Liaoning	562.1	0.028431	35	3	2	1	0
	16.18924528	Guangxi	2026.2	0.01775		2	1	1	0
	16.14685556	Jiangxi	3339.5	0.138061		12	8	2	2
	16.14059696	Guangdong	1893.3	0.030186		2	1	1	0
	16.13504913	Guizhou	682	0.007228		1	1	0	0
	16.10929886	Chongqing	689.7	0.011083		1	1	0	0
	16.0943753	Shanxi	123.4	0.004257		0	0	0	0
	16.05614766	Yunnan	1144.7	0.004337		1	1	0	0
	16.03178838	Ningxia	78.1	0.007951		1	1	0	0
	16.01289552	Hainan	312.2	0.001365		0	0	0	0
	15.94914763	Anhui	2217.3	0.140069		12	4	5	3
4	15.63171383	Tianjing	16.4	0.001767	19	1	1	0	0
	15.40249977	Zhejiang	824.2	0.018874		15	9	3	3
	14.98760527	Shandong	122.4	0.00249		3	2	0	1

6 Conclusions

In order to better evaluate the loss and waste of grain processing links in China, it is most important to extract reasonable samples. This paper firstly obtains the data of grain output, grain processing enterprises and grain processing amount of the whole country and all of the provinces through the existing historical statistical data of grain. If there is missing data, the gray model is used to predict it. Then we can determine the sample size of the preliminary sampling and calculate the output ratio of each province. Also we can use the three-stage stratified sampling method to determine the weighting factor of the corresponding layer in each stage, and calculate the number of enterprise samples of different scales in different provinces in China. Then, based on the sample data of preliminary sampling, the sample size of the second sampling is determined by the Sample size determination method. By using three-stage stratified sampling method again, we can obtain the sample distribution of the second sampling. The method assigns different numbers of samples to different regions according to the different grain output ratios of the regions, which makes the sampling process scientific and reasonable, and the sampling data is more representative. And the idea of the method can be applied to the sampling process in various fields, and can also be used in the statistical sampling of field rice yield loss prediction, light industry, heavy industry production survey, and can help various types of forecasting and measurement evaluation.

Acknowledgments. This work was supported by the subproject of the National Key Research and Development Program of China(Grant No. 2017YFD0401102-02).

References

1. Aiqin Liu,Yuxiang Wu.: Research on the Method of Allocating Sample Size in Stratified Sampling. Journal of Shandong University of Finance.4(2007).
2. Huifeng Xu.: Thoughts on Sample Size Allocation in Stratified Sampling. Corporate report.6(2011).
3. Yuying Luo.: Research on improving estimation accuracy under stratified sampling. New economy.8,(2016).
4. Xue Han.: Research on Railway Ticket Sales Data Extraction and Short-term Passenger Flow Forecasting. Beijing Jiaotong University.(2008).
5. “Statistical data of grain and oil processing industr”.: The Circulation and Technology Development Department of the State Grain Bureau.
6. State Grain Bureau.: http://www.chinagrain.gov.cn/.
7. National Bureau of Statistics of People’s Republic of China.: http://www.stats.gov.cn/.
8. Xianping Li,Chongsheng Shen,Ziyi Chen.: Probability Theory and Mathematical Statistics. Fudan University press, (2003).
9. Xiulin Geng.: Determination of sample size at mean estimation. Statistics and Decision, 10(2007).

10. Aiqin Liu.: Analysis of Factors Affecting Sample Size Determination in Random Sampling. Journal of Shandong University of Finance and Economics, 5, 60-64 (2006).
11. Keming Chen,Zhenlin Ning.: Determination of sample size in market research. China's statistical, 3(2005).

Simulating Tissue P systems with promoters through MeCoSim and P-Lingua

Luis Valencia-Cabrera[1] and Bosheng Song[2,⋆]

[1] Research Group on Natural Computing,
Department of Computer Science and Artificial Intelligence,
University of Sevilla, Spain
lvalencia@us.es

[2] College of Information Science and Engineering,
Hunan University, Changsha, China
boshengsong@hnu.edu.cn

Abstract. Tissue P systems constitute a well-known class of computing models in the research field of membrane computing. Inspired by the information exchange among cells and with the environment, they have attracted much attention over the last decade, with many interesting variants emerging along the years. One of such variants, the so-called tissue P systems with promoters, was proved to be Turing-universal even with a significantly restricted number of elements, and able to solve NP-complete problems. On the other hand, P-Lingua framework has provided useful tools to membrane computing community to model, debug and simulate different types of P systems, with a special relevance of P-Lingua language, pLinguaCore language and MeCoSim environment. This work presents new features introduced in the framework to cover the functionalities associated with tissue P systems with promoters, including extensions of the language, variants of tissue models and their corresponding simulators within P-Lingua version of MeCoSim. The new elements are described in detail, and the use of the tools is described through basic examples. Besides, a solution for `SAT` is experimentally validated using the developed software.

1 Introduction

Membrane computing [12] has been providing new computing models for the last two decades, inspired from different features observed in the structure and functioning of living cells, tissues or neurons. Many of these computational devices, generically called P systems, have proved their computational completeness (Turing-universal), and also a number of them have shown their ability to solve computationally hard problems as SAT [13, 20], HAM-CYCLE [18] or 3-COL [1], among others [11, 19].

⋆ Corresponding author

One of the groups of P systems that has attracted more attention over the years has been tissue P systems, inspired by the way cells organize and communicate in tissues [9]. Many variants exist where the features just mentioned were present, and among them in 2016 tissue P systems with promoters were proved to be both Turing-universal and able to solve SAT [20, 21], probably the best known computationally hard problem.

In any kind of P system, when defining a new computing model, it can possibly include interesting specific syntactic features, semantic restrictions, dynamic aspects or subtleties to consider. This implies that it might be worth studying different properties the new variants will offer. Thus, computability and complexity properties are usually explored for different variants of the models. Besides, for those providing solutions to hard problems, the designs proposed can be too complex and tedious to follow the evolution of the systems *without some external help*.

Therefore, this seems the perfect context to make use of helpful tools making our lives easier when debugging new models and simulating them under certain inputs or scenarios of interest, both for theoretical and real-life more practical applications [23]. However, it is a time-consuming task to develop such kind of tools for every model we can conceive [6, 22]. Besides, for every variant proposed new elements can be present in many aspects, so despite having rich frameworks covering many types of P systems, the compliance for future models cannot be ensured.

To overcome the highlighted limitation, significant efforts have been put in membrane computing community, trying to cover as many features as possible in a generic way [4, 5], with different approaches. Thus, P-Lingua framework includes many general features as a common language and some shared features for all the models, but cannot cover by default (in the last official version 4.0, nor in the last version included in MeCosim) new variants unless some further development is made.

On the other hand, UPSimulator provides a set of features of P systems that can be generically combined in a flexible way, thus allowing the use of new models not studied so far, by the combination of syntactic and semantic ingredients. This interesting software provides a significant contribution for the syntactic parsing and the simulation of P systems. However, it must pay a price for not restricting to the highly specific semantic peculiarities of each P system variant. This goal (*i.e.*, taking into account fine-grained subtle aspects of each variant) was present from the beginning in the former P-Lingua, in order to control that the proper specification of each variant is absolutely respected by the developer and the P system designer providing a solution for a given problem.

In order to provide general functionalities for computing models within the field of Membrane Computing, P-Lingua framework [4, 28] including MeCoSim environment [15, 27] serve the double purpose of: 1) a rich set of features for modeling, debugging, simulation, and final apps development, among others; and 2) the strict control, for the type and variant of P system used, of all the syntactic, semantic and dynamic aspects defined within the computing model.

We consider at the same level the flexibility of the tools and the fidelity to the rules of the game defined with any new computing model. Thus, the framework can control the use that any P system designer can make of the elements present in a solution given within a specific framework.

Thus, the present work extends the functionality provided by P-Lingua framework (specifically, the P-Lingua version - language and library - running inside MeCoSim environment) with all the features required to work with tissue P systems with promoters. These features must include, at least: 1) the specification in P-Lingua language of the ingredients trying to mimic the syntax of the promoters appearing in papers studying their computational completeness and complexity [20]; 2) the semantic restrictions imposed to different variants in terms of the types of rules allowed within the model; 3) the specific simulator following the dynamics described in the definition of the system [20]; and 4) the connection with all the pre-existing generic tools within the framework provided by P-Lingua and MeCoSim.

The paper is organized as follows: in Section 2, tissue P systems with promoters are defined; following, in Section 3 the main elements of the framework given by P-Lingua and MeCoSim are recalled; then, the new syntactic ingredients introduced in P-Lingua language to define tissue P systems with promoters is presented in Section 4, along with a brief description of the simulator adapted to work with promoters; finally, the work with tissue P systems in our platform is illustrated in Section 5, with simple examples plus the translation of the solution to SAT in P-Lingua, taken from [20].

2 Tissue-like P systems with promoters

This section introduces the computing model of tissue P systems with promoters, subject of the present paper and the simulator developed. More specifically, the type of devices covered is tissue P systems with promoters and cell division, thus incorporating the division rules to the basic tissue P systems with promoters. Similar variants using cell separation instead are out of the scope of our work.

2.1 Tissue P systems with promoters and cell division

In what follows the definition of our devices is recalled, as appeared in [20].

Definition 1. *A tissue P system with promoters and cell division, of degree $q \geq 1$, is a tuple*

$$\Pi = (\Gamma, \mathcal{E}, w_1, \ldots, w_q, \mathcal{R}, i_{out}),$$

where

- *Γ is an alphabet of objects;*
- *$\mathcal{E} \subseteq \Gamma$ is a set of objects initially located in the environment;*
- *w_i, $1 \leq i \leq q$, are finite multisets over Γ;*
- *$\mathcal{R}$ is a finite set of rules of the following forms:*

  - *Communication rules:*
    - *Symport rules:* $(pro \mid i, u/\lambda, j)$ *or* $(pro \mid i, \lambda/u, j)$*, where* $0 \leq i \neq j \leq q$*,* $pro, u \in M_f(\Gamma)$*,* $|u| > 0$*;*
    - *Antiport rules:* $(pro \mid i, u/v, j)$*, where* $0 \leq i \neq j \leq q$*,* $pro, u, v \in M_f(\Gamma)$*,* $|u| > 0$*,* $|v| > 0$*;*
  - *Division rules:*
    - $[\ a\ ]_i \rightarrow [\ b\ ]_i [\ c\ ]_i$*, where* $i \in \{1, \ldots, q\}$*,* $i \neq i_{out}$*,* $a, b, c \in \Gamma$*;*
- $i_{out} \in \{0, 1, \ldots, q\}$.

As also detailed in [20], tissue P system with promoters and cell division of degree $q \geq 1$ can be viewed as a set of q cells labeled by $1, \ldots, q$, providing the nodes of a directed graph; $w_1, \ldots, w_q$ represent the finite multisets of objects initially placed in these q cells; the important alphabet $\mathcal{E}$ contains the set of objects initially located in the environment of the system, which will be available in an arbitrary number of copies; $\mathcal{R}$ is the finite set of rules of the system, including symport/antiport rules operating on $pro, u, v \in M_f(\Gamma)$ (*i.e.*, multisets over the working alphabet), plus division rules (defined as usual); and i_{out} is a distinguished region that will hold the output of the system.

Thus, a *configuration* of this kind of system is described by the multisets of objects over Γ associated with all cells in the system, plus the multiset of objects over $\Gamma \setminus \mathcal{E}$ placed in the environment at that moment (as the objects from $\mathcal{E}$ are present in the environment in an arbitrarily large number of copies, the specific number of these objects is not relevant along the computation, considering they will be always enough to trigger the proper communication rules with the cells interacting with it). The *initial configuration* is, not surprisingly, $(w_1, \ldots, w_q; \emptyset)$.

Despite the fact that the applicability and effect produced by the different types of rules is properly described in [20], as it is crucial for the understanding of the semantics of the system, we will include such information in this preliminary section in order to avoid the reader the need of such additional access to the corresponding sources, thus making the reading of this document mostly self-contained. Therefore, in what follows we keep describing the semantic aspects of the system:

- A symport rule $(pro \mid i, u/\lambda, j) \in \mathcal{R}$ is applicable to a configuration if region i contains multiset u and the objects of the promoter pro are present in such i. Then, if such a rule is applied, u is sent from region i to j.
- An antiport rule $(pro \mid i, u/v, j) \in \mathcal{R}_i$ is applicable to a configuration if region i contains multiset u, region j contains multiset v, and the objects of the promoter pro are present in i. Thus, when it is applied, the objects of u are sent from i to region j, in exchange of objects of v, which are sent from j to i.
- A division rule $[\ a\ ]_i \rightarrow [\ b\ ]_i [\ c\ ]_i$ is applicable to a configuration if the cell i contains object a, and i is not the output cell. When such a rule is applied, i is divided into two cells (*with the same label*): in the first one, a is replaced by object b, and in the second one by object c; the rest of objects present in the original cell are all kept (replicated) in the two new cells.

Let us recall that, in a tissue P system with promoters and cell division, the presence of the promoter objects enables the use of the associated rule **as many times as possible**, without any restriction; consequently, a promoter can be used simultaneously by any number of rules associated with this promoter, which is not consumed nor blocked by the rule. Let us note that, if the rule does not require any promoter, the initial part of the rule is omitted, writing the rule as $(i, u/\lambda, j)$ (resp., $(i, u/v, j)$) instead of $(\emptyset \mid i, u/\lambda, j)$ (resp., $(\emptyset \mid i, u/v, j)$).

Regarding the dynamics of the system (*i.e.*, the execution strategy), the rules of these systems are applied in a maximally parallel manner: at each step, all cells which can evolve must evolve in a maximally parallel way (no further rule can be added being applicable, at any given instant). The only constraint to this statement is the following: when a cell is divided, the division rule is the only one to be applied involving such cell at that step, so the objects inside that cell not explicitly present in the syntax of the division rule do not evolve by means of communication rules.

The concepts of computation following the sequence of configurations, and the halting configuration are the usual ones in membrane systems. Concerning the result of the halting computations, it will be encoded by the number of objects present in the output region in the halting configuration.

3 P-Lingua framework for P systems

Along the last two decades, many types and variants of P systems have been defined, providing computing theoretical devices showing interesting properties. However, no physical implementation of such machines exists so far, and the solutions based on such systems are difficult to work with at certain level in a manual way.

In this context, having at disposal automatic tools to support the design, debug, analysis and simulation of these novel solutions may result crucial. In this sense, P-Lingua [4, 16] provides a uniform framework for the specification, debugging and simulation of this kind of computing models. Additionally, on top of P-Lingua, MeCoSim [15, 22] provides a user-friendly environment using the previous core to parse and simulate models (for technical users), along with a higher-level interface to handle models and deliver end-user applications based on this framework (for end users, not requiring knowledge about P systems).

This section outlines the main elements included in the framework provided by P-Lingua and MeCoSim.

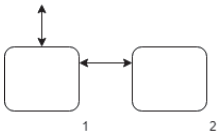
3.1 P-Lingua framework

The main components of *P–Lingua* framework[3] were initially a specification language to define P systems, and a Java [25] library called *pLinguaCore*, including parsers and simulators for different models within membrane computing. Later on, additional types of P systems have been covered, along with additional

features of the language. Along with these main elements, different command-line tools were provided from the beginning.

P-Lingua is intended to be a *standard* language for the definition of P systems through simple *text files*, easily processed by pLinguaCore directly or using some command-line tool or visual client (as MeCoSim, described in section 3.2). These files can specify P systems of families of them, depending on certain parameters included in the files (so that different values for the parameters instantiate different members of the family). The common syntax shared by many kinds of P systems including membrane structures, initial multisets, some types of rules, etc. decreases the learning curve for P systems designers learning different computing models within membrane computing.

Fig. 1 shows a small specification in P–Lingua language, for the definition of a simple P system. In this case, we are showing a tissue P system with promoters, using some new features introduced in this work. However, many elements (as the specification of the model, the definition of functions, the membrane structure or the initial multisets) take advantage of the general features provided by the language. This implies that someone familiar with P-Lingua and transition P systems, for instance, can understand most of the code, without prior experience with tissue P systems with promoters and cell division.



$r_1 : (1, a/p, 2)$

$r_2 : (p \mid 1, b/be, 0)$

$r_3 : (p \mid 1, c/ce, 0)$

$r_4 : (p \mid 1, bc/\lambda, 0)$

```
@model<tpdc>
def main()
{
  @mu = [[]'1 []'2]'0;
  @ms(0) = b,c,e;
  @ms(1) = a,b,c;
  @ms(2) = p;
  [a]'1 <--> [p]'2;
  [p | b]'1 <--> [b,e]'0;
  [p | c]'1 <--> [c,e]'0;
  [p | b,c]'1 <--> []'0;
}
```

Fig. 1. A simple P system specification in P–Lingua

Apart from the language, as mentioned above, P-Lingua framework includes the library `pLinguaCore`, providing *parsers* and *simulators* for the variants of P systems supported by the language, plus other useful tools. A detailed explanation can be found at [3, 4], and further information on a deeper level is given in [16] (this last one in Spanish).

3.2 MeCoSim (Membrane Computing Simulator)

MeCoSim provides visual tools to manage membrane computing models using P-Lingua. Thus, for P systems designers explore the models as white boxes to deepen in the study of the P systems themselves, while end users (possibly

unrelated to membrane computing) use them as black boxes to focus on the problems, abstracting from the internal details of the P systems actually solving such problems.

On the one hand, model designers find in MeCoSim a graphic tool to design, simulate, analyse and verify their models. On the other hand, end users receive MeCoSim-based applications (customized by the designers) adapted to their problems, where they simply enter the input data and check the results. As previously introduced, MeCoSim is built on top of P–Lingua: models are specified in P–Lingua language, processed by their parsers; then, the simulations are performed by executing simulation algorithms provided by pLinguaCore library, or through external simulators connected to MeCoSim.

In this visual environment, there exists a general application where any specific **single P system** written in P-Lingua can be processed, from the load and parsing to its simulation). However, if we want to have custom applications where the end user can introduced data to instantiate some member of a family of P systems and/or providing the external input to the system, the designer can configure a custom application through a spreadsheet file with `.xls` extension, adapting the inputs and outputs to the desired P system family, and indicating the way the input data by the end user will populate the parameters for the final P system to generate with each run of the system.

The overall process to customize apps and to use them, enriching the core parsing and simulation functionalities with further debugging and visualization tools, plus other add-ons using MeCoSim extension mechanisms, can be reviewed in detail in [22, 27].

4 Extensions of P-Lingua language

The previous section has outlined the main elements of our general framework provided by P-Lingua and MeCoSim.In the present work, the general tools of this framework are extended to handle tissue P systems with promoters and cell division. This includes genuine features added to the specific language for this kind of systems, including in their syntax some lexicon and grammar elements. Additionally, the existence of new ingredients (promoters) in the rules of these systems implies the need of new simulators capturing the dynamic behavior of these systems, following the description in [20].

In summary, we will take as a reference the existing P–Lingua syntax for P systems introduced in [16, 17], and will introduce the syntactic elements for tissue P systems with promoters and cell division along the following subsections. Then, a brief description of the simulator developed for this variant of P systems is given.

4.1 Model specification

Any P–Lingua file defining a tissue P system with cell division can be specified, in general, using `tpdc` as its model (similar to `TPDC`[20]):

```
@model<tpdc>
```

This use of `tpdc` will imply the allowance in (`t`)issue P systems of the use of (`p`)romoters (not present nor allowed in pre-existing models as `tissue_psystems` or `TSCS`), and cell (`d`)ivision, as well as the usual (`c`)ommunication rules.

Similarly, two additional restricted variants have been defined for this new type of P systems, both available to use instead of `tpdc`, depending on the type of constraints imposed:

```
@model<tpds>
@model<tpda>
```

As one might expect, the suffix `s` means that communication rules are restricted to (`s`)ymport rules only, while the `a` constraints the systems to allow (`a`)ntiport rules only instead.

The rest of the file will then define the main elements describing the P system, including the membrane structure containing the different cells, the initial multisets present in each one of them at the beginning of the computation and the set of rules. Let us note that some elements, as Γ or the underlying graph connecting the cells, are inferred from the initial multisets and the rules. On the other hand, $\mathcal{E}$ is also inferred from the initial multiset associated with the environment in the P-Lingua file.

The membrane structure, initial multisets, sets of rules and output region will be set, as explained in the following subsections.

4.2 Membrane structure and initial multisets

Tissue P systems with promoters are based on a graph structure as any other tissue–like P systems. As commonly used in P-Lingua models, in order to specify the initial membrane structure the reserve word `@mu` is used, defining the corresponding μ, as defined in Section 2. More specifically, if we have q cells, it would be defined as:

```
@mu = [ []'1 []'2 ... []'q ]'0;
```

This can be seen for a specific example in Fig. 1:

```
@mu = [ []'1 []'2 ]'0;
```

Regarding the objects in the initial configuration of the system, they can be defined by the reserved word `@ms`, assigning a certain multiset to the desired membrane. Thus, for the example in Fig. 1, the initial multisets are defined in P-Lingua as:

```
@ms(1) = a,b,c;
@ms(2) = p;
```

Similarly, the alphabet of the environment, establishing which objects will be present in the environment in an arbitrarily large number of copies along the computation, is defined with the same syntax, but assigning a set to label 0.

```
@ms(0) = b,c,e;
```

Let us recall that in the rest of regions (in the cells) we may have more than one copy of certain objects, and that P-Lingua expresses this by using operator *, followed by the multiplicity of the object, as in the following example:

```
@ms(1) = a,b*3,c*2;
```

Alternatively, the definition of these initial objects can be included directly in the definition of μ presented above, so that any membrane may include both child membranes and objects inside, as in the following code:

```
@mu = [ a*2 [ b ]'2 [ ]'3 [ c*5 ]'4 ]'1;
```

Besides the previous separate definitions of cells structure and initial multisets (plus definition of the alphabet of the environment), these elements might be combined to express in a more succinct way the same ideas. Thus, the corresponding lines into the example in Fig. 1 could be rewritten as:

```
@mu = [ b, c, e [ a, b, c ]'1 [ p ]'2 ]'0;
```

4.3 Definition of rules

As we recalled in Section 2, Tissue P systems with promoters and cell division introduce a new feature with respect to *classical* tissue P systems with cell division, in such a way that any symport/antiport rule can be conditioned by the presence of promoters. These elements were not present in P-Lingua so far, so they have been included in the language, and applied to the rules of the new models (`tpdc`, `tpds` and `tpda`). The example shown in Fig. 1 illustrate the use of the new feature:

```
[p | b]'1 <--> [b,e]'0;
[p | c]'1 <--> [c,e]'0;
[p | b,c]'1 <--> []'0;
```

As we can see in the example, the syntax `prom |` inside the left hand side of any of the rules is used, to express that the promoter object (`p` in the example) must be present in order to make the rule applicable.

4.4 Simulation of the new model with P-Lingua and MeCoSim

A new simulator has been developed within P-Lingua framework, making use of the general simulator available for tissue P systems, but including the semantic and dynamic aspects derived from the use of promoters in the rules of the new model.

Thus, once a P system in P-Lingua language is available in its text file, the simulator will perform a possible computation (let us recall our systems are non-deterministic) from the initial configuration given by the structure and multisets specified, producing the corresponding transitions until a halting configuration is reached.

Thus, the simulation algorithm follows the general schema present in most of the simulators included in the platform:

1. Initialization
2. For each computation step, while there are applicable rules:
   (a) Selection of rules
   (b) Execution of rules

The **initialization stage** sets initial structures used by the algorithm (technical details are not relevant here). Then, the main loop runs until a halting configuration is reached (*i.e.*, until no applicable rule is present).

The **selection phase** checks for the applicability of the rules for each cell. Then, for every membrane, if there are applicable rules, a maximal set is selected non-deterministically, following the constraints detailed in Section 2. The applicability of a rule is determined by the presence of at least the number of objects of each type present in each interacting region cell (let us recall that, if the environment is one of such regions, the objects of the alphabet of the environment will be present in an arbitrarily large number of copies). Additionally, if promoters are included in the rule, then their presence will also be required in order to make the rule applicable.

Finally, as a result of this selection phase, a set of rules have been selected. In this case, let us also recall that, while the multiplicity of each object restricts the choice of rules for our maximal set, in the case of promoters is different: with only one promoter object affecting many rules, all of them would be enabled in with respect to the promoter. In this sense, we say that rules do not compete for the promoters, they imply more a context condition that an object to consume (it is in fact not removed).

Then, the **execution phase** applies the change in the configuration, passing from C_t to C_{t+1}, removing the objects consumed by the rules selected, and adding the objects produced, with the semantics specified in Section 2.

4.5 Availability of the software tools developed

The tools described in the previous sections, concerning the language, parsing and simulation have made publicly available in the current version of MeCoSim, that can be downloaded from [27]. Once downloaded, whenever the software runs, if an Internet connection is active, it checks the presence of new versions of any of the files involved, thus guarantees it always provides the user with the last version of MeCoSim (that includes in its installation pLinguaCore).

4.6 Further technical considerations

The tools developed along this work aim to complement P-Lingua framework, in order to provide both P system experts/designers and end users of the P-system based applications with an environment where they can debug their designs, verify solutions to hard problems and run virtual experiments for a variant of P system not previously supported.

The main challenges faced during the development are derived from the definition of a sub-language for tissue P systems with promoters integrated in P-Lingua framework, plus the corresponding simulator handling this variant of P system.

On the one hand, the extension of the language includes new syntactic and grammatical elements for the parser, based on JavaCC [26], and the proper integration with the rest of P-Lingua language. On the other hand, the definition of the corresponding semantic constraints and the logic of the specific simulor, taking into account the semantic and dynamic aspects defined by tissue P systems with promoters and cell division [20], was developed in Java [25], and following the structure defined for other simulators and semantic decorations.

Let us recall that the aim of P-Lingua framework and MeCoSim environment is the precise definition of each variant of P system covered from a functional point of view. Other aspects as the development of the fastest possible solutions for these systems (*e.g.*, with parallel simulators using high performance computing - HPC) would be out of the scope of this work, and could improve the performance of the simulator developed. Those potential alternatives could still make use of the parsing tools developed with P-Lingua, and also be connected with MeCoSim environment acting as a client providing the interface and pre/post-processing tool for a server HPC-based simulator.

5 Case studies

In the previous section, the software tools developed for the design and simulation of tissue P systems with promoters and cell division have been described. In this last section, we will illustrate the use of the software, showing the corresponding P-Lingua files specifying the solutions in the input format for the computer tools, and some runs for simple and complex problems.

5.1 Basic example

In order to check the proper behavior of the new features included in our framework, we started with a very simple example, as included at the beginning of he paper:

```
@model<tpdc>
def main()
{
    @mu = [[]'1 []'2]'0;
    @ms(0) = b,c,e;
    @ms(1) = a,b,c;
    @ms(2) = p;
    [a]'1 <--> [p]'2;
    [p | b]'1 <--> [b,e]'0;
    [p | c]'1 <--> [c,e]'0;
    [p | b,c]'1 <--> []'0;
}
```

This simple example included both symport and antiport rules, so we made use of `tpdc` model, and started debugging in MeCoSim, as shown in Fig. 2.

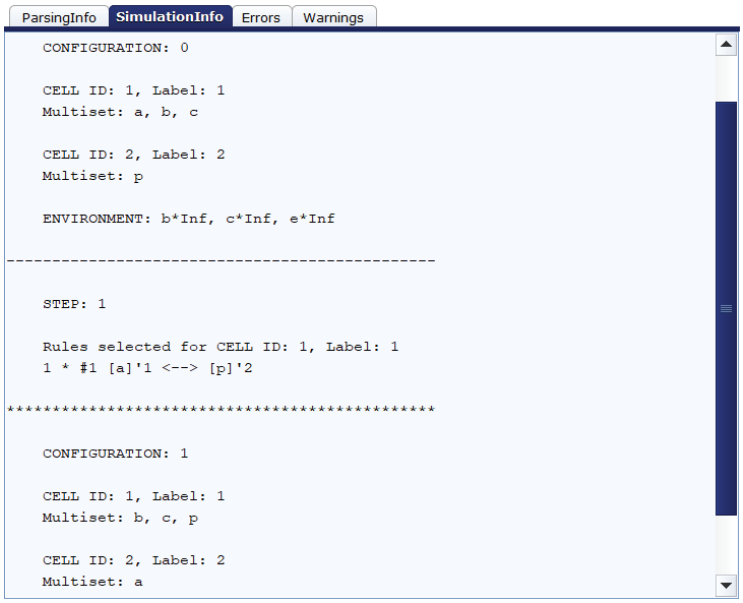


Fig. 2. Debugging a simple tissue P system with promoters in MeCoSim

If we had used a different model, as `tpda` or `tpds`, our environment would inform us about the error, thus disabling the possibility to run it until we correct this aspect, as we can see in Fig. 3.

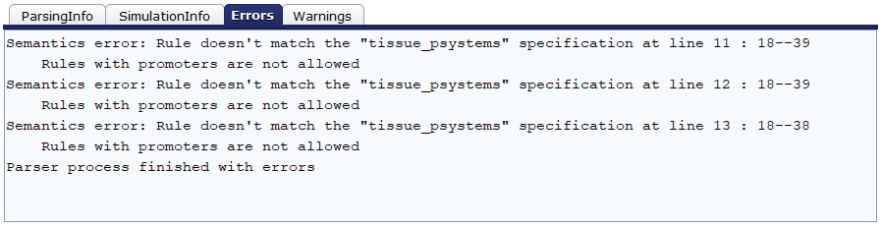


Fig. 3. Informing about semantic errors

5.2 Solving SAT by tissue P systems with promoters and cell division

In this case study, we will recall the solution for SAT given in [20], illustrating the behavior of these systems, and we will show the corresponding P-Lingua files specifying the solutions in the input format for the computer tools.

The SAT problem is a well known **NP**-complete problem [2], which is defined as follows: *given a Boolean formula in conjunctive normal form (CNF), determine whether or not there exists an assignment to its variables such that the formula is evaluated to be true.*

In what follows, we provide here the linear time solution to the SAT problem by a family of recognizer tissue P systems with promoters and cell division directly taken from [2]: $\mathbf{\Pi} = \{\Pi(t) \mid t \in \mathbb{N}\}$. As expressed there, each system $\Pi(t)$ processes all Boolean formulas φ in conjunctive normal form with n variables and m clauses, where $t = \langle n, m\rangle = ((n+m)(n+m+1)/2) + n$, assuming the corresponding input multiset $cod(\varphi)$ is supplied to the system.

For each $n, m \in \mathbb{N}$, we consider the recognizer tissue P system with promoters and cell division

$$\Pi(\langle n, m\rangle) = (\Gamma, \Sigma, \mathcal{E}, w_1, w_2, \mathcal{R}, i_{in}, i_{out}),$$

where

- $\Gamma = \Sigma \cup \{a_i, t_i, f_i \mid 1 \leq i \leq n\} \cup \{b_j, c_j \mid 1 \leq j \leq m\} \cup \{z_i \mid 0 \leq i \leq 2n+m+3\} \cup \{b_{m+1}, e, p, q, q', \texttt{yes}, \texttt{no}\}$,
- $\Sigma = \{x_{i,j}, \bar{x}_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$,
- $\mathcal{E} = \{a_i \mid 1 \leq i \leq n\} \cup \{b_j, c_j \mid 1 \leq j \leq m\} \cup \{z_i \mid 0 \leq i \leq 2n+m+3\} \cup \{b_{m+1}, e, q'\}$,
- $w_1 = \{a_1\}$, $w_2 = \{p, q, z_0, \texttt{yes}, \texttt{no}\}$,
- $i_{in} = 1$ is the input cell,
- $i_{out} = 0$ is the output region,
- The set $\mathcal{R}$ consists of the following rules:
  $r_{1,i} \equiv [\ a_i\ ]_1 \rightarrow [\ t_i\ ]_1 [\ f_i\ ]_1$, $1 \leq i \leq n$,
  $r_{2,i,j} \equiv (t_i \mid 1, x_{i,j}/c_j, 0)$, $1 \leq i \leq n, 1 \leq j \leq m$,
  $r_{3,i,j} \equiv (f_i \mid 1, \bar{x}_{i,j}/c_j, 0)$, $1 \leq i \leq n, 1 \leq j \leq m$,
  $r_{4,i} \equiv (1, t_i/a_{i+1}, 0)$, $1 \leq i \leq n$,
  $r_{5,i} \equiv (1, f_i/a_{i+1}, 0)$, $1 \leq i \leq n$,
  $r_6 \equiv (1, a_{n+1}/b_1, 0)$,
  $r_{7,j} \equiv (b_j \mid 1, c_j/b_{j+1}, 0)$, $1 \leq j \leq m$,
  $r_8 \equiv (1, b_{m+1}/p, 2)$,
  $r_9 \equiv (b_{m+1} \mid 2, \texttt{yes}/e, 0)$,
  $r_{10} \equiv (b_{m+1} \mid 2, q/e, 0)$,
  $r_{11,i} \equiv (2, z_i/z_{i+1}, 0)$, $0 \leq i \leq 2n+m+2$,
  $r_{12} \equiv (z_{2n+m+3} \mid 2, q/q', 0)$,
  $r_{13} \equiv (q' \mid 2, \texttt{no}/e, 0)$.

All the details about the behavior and verification of the model can be read in [2]. For our purposes, it is interesting to see the translation of this system to P-Lingua language, using the new model and elements introduced:

```
@model<tpda>

def main()
{
   call module_init_conf(n,m);
   call module_rules(n,m);
   call module_input();
}

def module_init_conf(n,m)
{
   @mu = [[]'1 []'2]'0;
   @ms(0) = b{m+1},e,qp;
   @ms(0) += a{i} : 1 <= i <= n+1;
   @ms(0) += b{j},c{j} : 1 <= j <= m;
   @ms(0) += z{i} : 0 <= i <= 2*n+m+3;
   @ms(1) = a{1};
   @ms(2) = p,q,z{0},yes,no;
}

def module_rules(n,m)
{
   /* r1{i} */   [a{i}]'1 --> [t{i}]'1 [f{i}]'1 : 1<=i<=n;
   /* r2{i,j} */ [t{i} | x{i,j}]'1 <--> [c{j}]'0: 1<=j<=m, 1<=i<=n;
   /* r3{i,j} */ [f{i} | nx{i,j}]'1 <--> [c{j}]'0: 1<=j<=m, 1<=i<=n;
   /* r4{i} */   [t{i}]'1 <--> [a{i+1}]'0 : 1<=i<=n;
   /* r5{i} */   [f{i}]'1 <--> [a{i+1}]'0 : 1<=i<=n;
   /* r6 */      [a{n+1}]'1 <--> [b{1}]'0;
   /* r7{j} */   [b{j} | c{j}]'1 <--> [b{j+1}]'0 : 1<=j<=m;
   /* r8 */      [b{m+1}]'1 <--> [p]'2;
   /* r9 */      [b{m+1} | yes]'2 <--> [e]'0;
   /* r10 */     [b{m+1} | q]'2 <--> [e]'0;
   /* r11{i} */  [z{i}]'2 <--> [z{i+1}]'0 : 0<=i<=2*n+m+2;
   /* r12 */     [z{2*n+m+3} | q]'2 <--> [qp]'0;
   /* r13 */     [qp | no]'2 <--> [e]'0;
}

def module_input()
{
   /* We define here the input for the P system */
   /* Let be m: number of clauses, and n: number of variables */

   @ms(1) += nx{variable{i},clause{i}}*valn{i}, x{variable{i},clause{i}}*
        val{i} : 1<=i<=nvals;
}
```

Figure 4 shows this last example loaded in MeCoSim, running a specific example, and visualizing both P-Lingua file editor and the multisets viewer.

Let us note that previous to getting this result a debugging process was conducted through the platform generic tools with the new features incorporated. Additionally, the formal verification of this solution given in [20] could be checked with the step-by-step run, thus checking that the statements made in the original paper are indeed satisfied for specific examples, what enriches the previous result with experimental validation.

Acknowledgments

The work of Luis Valencia-Cabrera was partially supported in part by the research project TIN2017-89842-P, cofinanced by Ministerio de Economa, Industria y Competitividad (MINECO) of Spain, through the Agencia Estatal de Investigacin (AEI), and by Fondo Europeo de Desarrollo Regional (FEDER) of the

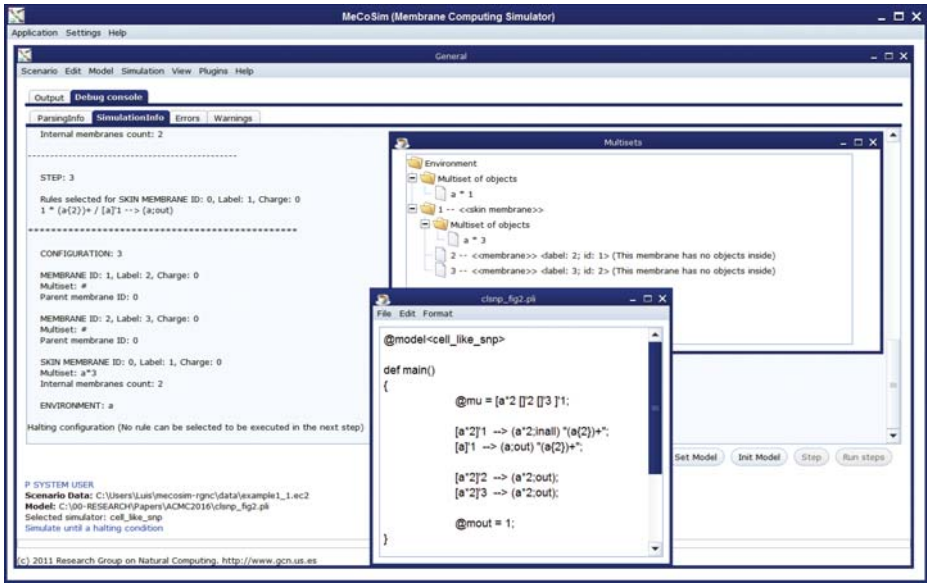


Fig. 4. SAT simulation in MeCoSim

European Union. The work of Bosheng Song was supported in part by National Natural Science Foundation of China (61972138, 61602192), and in part by the Fundamental Research Funds for the Central Universities (531118010355).

References

1. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J. Solving 3-COL with Tissue P Systems *Proc. Fourth Brainstorming Week on Membrane Computing*, Sevilla, 2006, 17–30.
2. M.R. Garey, D.J. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, San Francisco, 1979.
3. Díaz-Pernil, D., Pérez-Hurtado, I., Pérez-Jiménez, M.J., Riscos-Núñez, A. A P-Lingua programming environment for membrane computing. *Lecture Notes in Computer Science*, **5391** (2009), 187–203.
4. García-Quismondo, M., Gutiérrez-Escudero, R., Pérez-Hurtado, I., Pérez-Jiménez, M.J., Riscos-Núñez, A. An overview of P-Lingua 2.0. *Lecture Notes in Computer Science*, **5957** (2010), 264–288.
5. Guo, P., Quan, C., Ye, L. UPSimulator: A general P system simulator. *Knowledge-Based Systems*, **170** (2019), 20–25.
6. Lefticaru, R., Ipate, F., Valencia-Cabrera, L., Turcanu, A., Tudose, C., Gheorghe, M., Pérez-Jiménez, M.J., Niculescu, I.M., Dragomir, C. Towards an integrated approach for model simulation, property extraction and verification of P systems. *Proc. Tenth Brainstorming Week on Membrane Computing*, Sevilla, 2012, I, 291–318.

7. Macías-Ramos, L.F. Developing efficient simulators for cell machines. PhD thesis. University of Seville, 2016.
8. Macías-Ramos, L.F., Pérez-Hurtado, I., García-Quismondo, M., Valencia-Cabrera, L, Pérez-Jiménez, M.J., Riscos-Núñez, A. A P-Lingua based simulator for spiking neural P systems. *Lecture Notes in Computer Science*, **7184** (2012), 257–281.
9. Martín-Vide, C., Păun, Gh., Pazos, J., Rodríguez-Patón, A.: Tissue P systems, *Theoretical Computer Science*, **296**, 2 (2003), 295–326.
10. Pan, L., Alhazov, A.: Solving HPP and SAT by P Systems with Active Membranes and Separation Rules. Acta Informatica, **43**, 2 (2006), 131–145.
11. Pan, L., Song, B.: P systems with rule production and removal. *Fundamenta Informaticae*, 2020, **171**, (2020), 313–329.
12. Păun, Gh. Computing with membranes. *Journal of Computer and System Sciences*, **61**, 1 (2000), 108–143.
13. Păun, Gh. P Systems with Active Membranes: Attacking NP-Complete Problems. *Journal of Automata, Languages and Combinatorics*, **6**, 1 (2001), 75–90.
14. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) *The Oxford Handbook of Membrane Computing*. Oxford University Press, New York, 2010.
15. Pérez-Hurtado, I., Valencia-Cabrera, L., Pérez-Jiménez, M.J., Colomer, M. A., Riscos-Núñez, A. Mecosim: A general purpose software tool for simulating biological phenomena by means of p systems. *IEEE Fifth Intl. Conf. on Bio-inpired Computing: Theories and Applications (BIC-TA 2010)*, I (2010), 637–643.
16. Pérez-Hurtado, I. Desarrollo y aplicaciones de un entorno de programación para Computación Celular: P-Lingua. PhD thesis. University of Seville, 2010.
17. Pérez-Hurtado, I., Valencia-Cabrera, L., Chacón, J.M., Riscos-Núñez, A., Pérez-Jiménez, M.J. A P-Lingua based simulator for tissue P systems with cell separation, *Romanian Journal of Information Science and Technology*, **17** (2014), 89–102.
18. Porreca, A.E., Murphy, N., Pérez-Jiménez, M.J. An Optimal Frontier of the Efficiency of Tissue P Systems with Cell Division. In *Proc. Tenth Brainstorming Week on Membrane Computing*, Sevilla, 2012, II, 141–166.
19. Song, B., Kong, Y.: Solution to PSPACE-complete problem using P systems with active membranes with time-freeness. *Mathematical Problems in Engineering*, 5793234, (2019).
20. Song, B., Pan, L. The computational power of tissue-like P systems with promoters. *Theoretical Computer Science*, **641** (2016), 43–52.
21. Song, B., Zhang, C., Pan, L.: Tissue-like P systems with evolutional symport/antiport rules. *Information Sciences*, **378** (2017) 177–193.
22. Valencia Cabrera, L. An environment for virtual experimentation with computational models based on P systems. PhD thesis. University of Seville, 2015.
23. Zhang, G., Pérez-Jiménez, M.J., Gheorghe, M. *Real-life applications with Membrane Computing*. Springer Publishing Company, 2017.
24. GNU GPL Website, `http://www.gnu.org/copyleft/gpl.html`
25. Java tutorial Website,
`https://docs.oracle.com/javase/tutorial`
26. Javacc,
`https://javacc.org`
27. MeCoSim Website. `http://www.p-lingua.org/mecosim/`
28. P-Lingua Website. `http://www.p-lingua.org/`

The two-stage multi-objective optimization algorithm based on classified population

Hang Shu[1], Kang Zhou[1] *, Gexiang Zhang[2], and Zhixin He[1]

[1] School of Math and Computer,
Development Strategy Institute of reserve of food and material,
Wuhan Polytechnic University, Wuhan 430023, China
[2] School of Electrical and Engineering,
Southwest Jiaotong University, Chengdu 610031, China
{zhoukang65@whpu.edu.cn}

Abstract. The aim of the tri-objective vehicle routing optimization problem with time windows (TO-VRPTW) is to optimal the total travel cost, the number of vehicle and the travel cost of the farthest car. This paper presents the two-stage multi-objective optimization algorithm based on classified population (TSCOA). According to characteristics of discrete optimization problem, the concept of crowding degree is extended; According to characteristics of the number of vehicles, a classification-based population structure is proposed which can reduce dimensionality while ensuring the quality of the solutions found; According to the characteristics of population structure, the NSGA-II is improved. The framework of TSCOA develops a two-stage multi-objective evolutionary algorithm including initial population acquisition and multi-objective search. Stage I focuses on finding an initial population with optimal distribution structure by global search of single objective and bi-objective, while stage II uses elitist preservation strategy of "gradually optimizing main loop of intermediate generation". The experimental results based on the Solomon benchmark instances show that in term of the tri-objective, TSCOA outperforms other existing algorithms because of optimal solutions are closer to the Pareto front and their distribution and extensibility.

Keywords: TO-VRPTW , NSGA-II, TSCOA, Population Structure

1 Introduction

The vehicle routing problem with time windows (VRPTW) [1]is a combinatorial optimization. The problem can be described as using vehicles of same specifications service a set of geographically dispersed customers (Each vehicle starting from the dispatching center service customers, then return). The problem is also subject to the restriction that each customer must be visited exactly once and that the cumulative demands of the serviced customers must not exceed the capacity of the servicing vehicle.

* Corresponding author.

VRPTW is an extension of VRP [2, 3] with time constrained which has very important realistic meaning in logistics, grain scheduling, supply chain management, electronic commerce [4] and so on. However, there are many practical problems account for bi-objective or multi-objective optimization [5]. When optimizing the NP-hard[6] problem, many heuristic algorithms[7] and meta heuristic algorithms[8] have been well developed.

For now, the main study aiming at VRPTW is about single objective and bi-objective and has a great number of research achievements. The classical evolutionary algorithms include: Hong and Park[9] constructed a linear objective programming model for bi-objective VRPTW and proposed a heuristic algorithm to relieve a computational burden. The multi-objective genetic algorithm(MOGAs)[10, 11] maintain a population of candidate solutions to find Pareto solutions. Deb et al.[12] proposed improved NSGA[13] called NSGA-II. Tan et al.[14] proposed hybrid multi-objective evolutionary algorithm(HMOEA) to optimize the bi-objective VRPTW. Geiger et al.[15] presented an interactive multi-criteria approach for the resolution of rich vehicle routing problems. Gong[16] proposed a multi-objective PSO to solve the MO-VRPTW. With the increment of objectives, the number of Pareto solutions will increase. In this way, there are two problem caused by increasing objectives should be considered. The first one is the computational efficiency and the second one is whether the Pareto front is complete. In other words, it is the reason why some algorithms with excellent performance on single objective and bi-objective VRPTW show low performance while optimizing many-objective VRPTW.

In order to solve problems of low computational efficiency and accuracy caused by the many-objective optimization, researchers also proposed many excellent algorithms to solve many-objective optimization problem. Zhang et al.[17] presented a specially tailored evolutionary algorithm based on a decision variable clustering method to tackle such large-scale many-objective optimization problems. Afterwards, to optimize the two types of decision variables, a convergence optimization strategy and a diversity optimization strategy are adopted. In [18–20], MDVRPTW[21] with three objectives, four objectives and five objectives is optimized. Zhang et al.[22] proposed a knee point driven evolutionary algorithm to solve many-objective optimization problems.

Nevertheless, there are relatively few researches on the tri-objective VRPTW (TO-VRPTW) with only one distribution center. Therefore, it is necessary to study an optimal algorithm for TO-VRPTW. In this paper, for balancing the burden of the farthest car and making VRPTW be more realistic significance, we take the travel cost of the farthest car as the third objective. In the previous study of TO-VRPTW, Andreas Konstantinidis et al.[23] presented a multi-objective algorithm MOEA/D-aLS to optimize TO-VRPTW. MOEA/D-aLS has advantages in optimizing TO-VROTW, but its parameter setting and calculation process are complex. Therefor, its Pareto solutions and computational efficiency still have room to improve.

The contributions in this paper include:

1) construct an initial population based on the number of vehicles which makes the Pareto front have much more extensibility.

2) A two-stage evolution algorithm TSCOA based on improved NSGA-II which is used to optimize TO-VRPTW. TSCOA greatly improves computational efficiency by optimizing tri-objective to bi-objective while constructing a complete initial population.

The remaining chapters of this paper are mainly divided into following sections: section 2 gives the mathematical model and some basic knowledge about multi-objective optimization problems; section 3 introduces proposed TSCOA from computational framework, crowding degree, population structure, main loop and some other aspects; section 4 describes the experimental results by comparing the 56 Solomon[24] benchmark instances, NSGA-II and the algorithm in [23]. Among them, we use three indexes including approximation, distribution and extensibility of Pareto front to demonstrate the outperformance of TSCOA in optimizing TO-VRPTW and TSCOA also outperforms in optimizing bi-objective through experimental comparison with other excellent bi-objective optimization algorithm.

2 Basic Theory

2.1 Mathematical Model of TO-VRPTW

$M = \{1, 2, \cdots, m\}$, m is a set of vehicles. $N = \{1, 2 \cdots, n\}$ represents $n(n \geq m)$ customers and the dispatching center is denoted as 0. Each customer i must be served by only one vehicles in the time windows $[T_{si}, T_{ei}]$,where T_{si} and T_{ei} is the beginning and the end of the customer served. If the served vehicle arrives before T_{si}, it will wait for T_{si} to serve.

Before presenting the mathematical formulation, some following notations are defined:

d_{ij} is the distance between grain depot i and grain depot j,

r_i is the demand for goods in the grain depot i,

W is the maximum load capacity of the vehicle,

$p_k(p_k \subset N)$ is the grain storage point set for the number of vehicles k,

T_{ik} is the time point when the vehicle k arrives at the grain depot i

t_{ij} is travel time, K is a coefficient.

Decision variables:

$$x_{ijk} = \begin{cases} 1 \text{ if vehicle } k \text{ travels from grain depot } i \text{ to grain depot } j \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

The mathematical model of TO-VRPTW (optimization model I) is as follows:

$$\min f_1 = \sum_{k\in M}\sum_{j\in N} x_{0jk} \tag{1}$$

$$\min f_2 = \sum_{k\in M}\sum_{i\in N}\sum_{j\in N} d_{ij}x_{ijk} \tag{2}$$

$$\min f_3 = \max_{k\in M}\sum_{i\in p_k}\sum_{j\in p_k} d_{ij}x_{ijk} \tag{3}$$

$$s.t\begin{cases} \sum\limits_{k\in M}\sum\limits_{j\in N} x_{ijk} = 1, \forall i \in N \\ \sum\limits_{i\in N}(r_i \sum\limits_{j\in N} x_{ijk}) \le W, \forall k \in M \\ \sum\limits_{j\in N} x_{0jk} = \sum\limits_{i\in N} x_{i0k} \le 1, \forall k \in M \\ \sum\limits_{i\in N} x_{ihk} = \sum\limits_{j\in N} x_{hjk}, \forall h \in N, \forall k \in M \\ T_{ik} + t_{ij} - K(1 - x_{ijk}) \le T_{jk}, \forall i, j \in N, \forall k \in M \\ \sum\limits_{k\in M}\sum\limits_{i,j\in D} x_{ijk} \le |D| - 1, \forall D \subseteq N \\ T_{ei} \ge T_{ik} \ge 0 \end{cases} \tag{4}$$

In this model, Objective (1) is to minimum number of vehicles. Objective (2) is to shortest total distances. Objective (3) is to shortest distances traveled by the farthest vehicle. (4) is to show the mathematical model of all constraints.

2.2 Related Concepts of Multi-objective Optimization Problem

The Multi-objective Optimization Problem (MOP) is expressed as follows:

$$\begin{gathered} \min y = f(x) = (f_1(x).....f_m(x))^T \\ g_i(x) \le 0, i = 1, 2, \ldots, u \\ h_j(x) = 0, j = 1, 2, \ldots, v \end{gathered}$$

The basic concepts of the MOP solutions are described as follows:

Definition 1 Let $\Omega = \{x | \forall i\,(g_i(x) \le 0) \wedge \forall j\,(h_j(x) = 0) \wedge x \in R^n\}$ be feasible solutions of MOP, $V = \{y = (y_1, \ldots, y_m) | \, y_i = f_i(x) \wedge x \in \Omega\}$ be the m-dimensional target space.

1) For $x_1, x_2 \in \Omega$ if $\forall j\,(f_j(x_1) \le f_j(x_2)) \wedge \exists k\,(f_k(x_1) < f_k(x_2))$, x_1 dominates x_2,it can be recorded as $x_1 \succ x_2$.

2) For $\varphi \subset \Omega$, if $\forall x_1, x_2\,(x_1, x_2 \in \varphi \wedge \neg\,(x_1 \succ x_2 \vee x_2 \succ x_1))$, φ is a non-dominated set.

3) For $x\,(x \in \Omega)$, if $\forall x^{'}\left(x^{'} \in \Omega \wedge \neg\left(x^{'} \succ x\right)\right)$, x is denoted as a Pareto-optimal solution, while vertex set $\varphi \subset \Omega$, $x\,(x \in \varphi)$,if $\forall x^{'}\left(x^{'} \in \varphi \wedge \neg\left(x^{'} \succ x\right)\right)$,$x$ is denoted as a non-dominated solution of vertex set φ.

4)PS=$\left\{ x|\, x \in \Omega \wedge \forall x^{'} \left(x^{'} \in \Omega \wedge \neg \left(x^{'} \succ x \right) \right) \right\}$ is named as a set of Pareto solutions, while $\varphi \subset \Omega$,let $NS\left(\varphi\right) = \left\{ x|\, x \in \varphi \wedge \forall x^{'} \left(x^{'} \in \varphi \wedge \neg \left(x^{'} \succ x \right) \right) \right\}$ be a non-dominated solution of vertex set φ. Let PS^3 be a set of Pareto solutions of optimization model I.

5)PF=$\{ f\left(x\right)|\, x \in PS\}$is named as a Pareto front ,While $\varphi \subset \Omega$, let $NF\left(\varphi\right) = \{ f\left(x\right)|\, x \in NS\left(\varphi\right)\}$ be a non-dominated Pareto front of vertex set φ. Let PF^3 be a Pareto front of optimization model I.

The essence of NSGA-II is to find vertex set φ, in order to let the non-dominated front $NF\left(\varphi\right)$ approach to Pareto front PF^3 in an optimal way. Among them, whether the non-dominated front $NF\left(\varphi\right)$ is crowded is an important index to evaluate a set of non-dominated solutions$NS\left(\varphi\right)$, and the less crowded a set of non-dominated solutions $NS\left(\varphi\right)$ is, the more evenly the non-dominated front of $NF\left(\varphi\right)$ is distributed.

Definition 2(Crowding Degree) For a solution x_i of a non-dominated set φ=$\{x_1 x_2 ... x_n\}$.

1) For the j-th objective function, if $f_j(x_1) \leq f_j(x_2) \leq ... \leq f_j(x_n)$,$f_j(x_1) = f_j(x_n) = \infty$,

$$C_j(x_i) = \frac{[f_j(x_{i+1}) - f_j(x_{i-1})]}{(f_j^{\max}(x) - f_j^{\min}(x))}, (2 \leq i \leq n\text{-}1) \tag{5}$$

2) A crowding degree of x_i is a sum of crowding degree of all objective functions.

$$C(x_i) = \sum_{j=1}^{m} C_j(x_i) \tag{6}$$

3 The Two-Stage Multi-Objective Optimization Algorithm Based On Classified Population

In this section, the two-stage multi-objective optimization algorithm based on classified population (TSCOA) is proposed to optimize TO-VRPTW. In section 3.1, NSGA-II computing framework is improved. In section 3.2, crowding degree is extended. Section 3.3 introduces three parts separately: an improvement of population structure is explained in detail at the first part; the second part describes how to get an initial population; the third part introduces a distribution of fitness values.

3.1 Computing Framework of TSCOA

NSGA-II basic framework: Firstly, an initial parent population is randomly generated; then some genetic operations are used to produce an intermediate generation population, merging it with parent population, then intermediate generation population is deleted through a dominance relationship and a crowding degree relationship to obtain the offspring. This basic framework can be improved in two ways:

1) The quality of an initial population structure directly affects calculation accuracy and efficiency of TSCOA. Therefore, the algorithm of constructing an initial population is crucial.

2) Dominance relationship and crowding degree relationship have large computational costs. The size of an intermediate generation population is doubled first, and then it is shorten to a half through dominance relationship and crowding degree relationship. There is room in this process to improve due to excessive and invalid calculations.

In view of above two problems, the framework of NSGA-II is improved as follows:

1) Get an initial population. The optimization algorithm is designed to get an initial population with a more reasonable structure.

2) Gradually optimizing main loop of intermediate generation. In this main loop, when an intermediate generation which starts with parent population performs a genetic operation, it will be updated. Offspring is obtained until genetic operations are completed on the parent population.

The framework of TSCOA is shown in Fig.1.

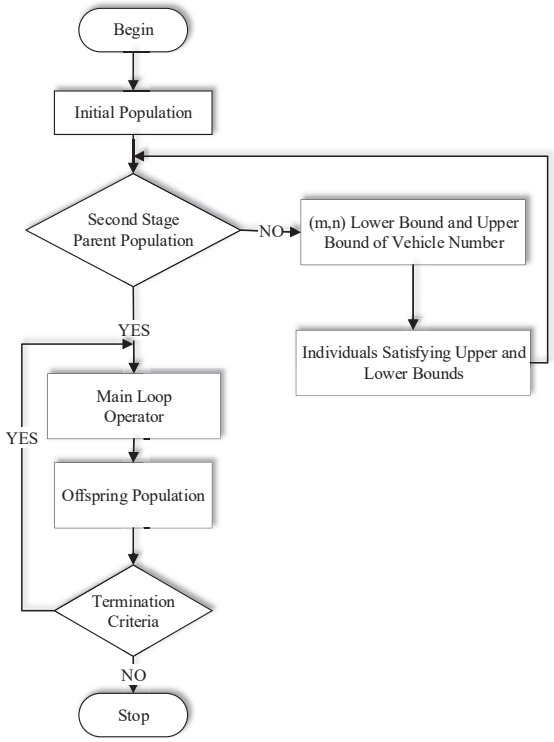


Fig. 1. the TSCOA framework

The framework of main loop operator shows in the section 3.4 fig.2.

3.2 Extension of Crowding Degree

Discrete optimization problems (especially VRPTW) have higher coincidence degree of objective function values among population individuals than continuous optimization problems. When calculating a crowding degree of VRPTW, it is important to process the coincidence degree of this objective function value, and extend a crowding degree of a discrete optimization problem. Next, we discuss the extension of crowding degree of a discrete optimization problem.

Let $S(x_i)$ be a solution set in all MOP which coincides with solutions in$f(x_i)$, $S(x_i) = \{x| \bigwedge_{k=1}^{m} f_k(x_i) = f_k(x)\}$.Obviously, there is $x_i \in S(x_i)$, the crowding degree of a solution which has a higher degree of polymerization is more smaller. If $|S(x_i)|= 2$, it means that there is a solution that coincides with x_i, we can set $C(x_i) = (1 - 2) = -1$. If $|S(x_i)|=3$, it means that there are two solutions that coincide with x_i,we can set $C(x_i) = (1 - 3) = -2$. So, while $|S(x_i)| \geq 2$,$C(x_i) = (1 - |S(x_i)|)$.

Definition 3(Extended Crowding Degree) For a solution x_i of a non-dominated set $\varphi= \{x_1x_2...x_n\}$,

$$C(x_i) = \begin{cases} \sum_{j=1}^{m} C_j(x_i) & |\mathrm{S}(x_i)|= 1 \\ (1 - |S(x_i)|) & |\mathrm{S}(x_i)| \geq 2 \end{cases} \tag{7}$$

There is a calculation process of the extended crowding degree in Algorithm 1.

Algorithm 1: crowding (Q)

```
1: For each q ∈ Q
2:   If (|S(q)| ≥ 2 )
3:       C_q = 1 − |S(q)|
4: End for
5: For j to end
6:    For each objective m
7:       sort(Q, m)
8:       C_1 = C_end = ∞
9:       C_i = C_i + (Q[j + 1].m − Q[j − 1].m)/(f_m^max − f_m^min)
10:   End for
11: End for
```

Where $Q[j].m$ represents m-th dimension objective function value of the solution j.

3.3 Improvement and Acquisition of Population Structure

3.3.1 Classification Population Structure

A criterion for evaluating optimal-solution set of Multi-objective intelligent optimization algorithm (**"Evaluating Optimal Solutions Criterion"**) is:

1) The non-dominated front $NF(\varphi)$ of optimal solutions is close enough to PF^3.

2) The non-dominated front $NF(\varphi)$ extensibility of optimal solutions has a wider range.

3) The non-dominated front $NF(\varphi)$ distribution of optimal solutions is more uniform.

PF of discrete multi-objective optimization problem is a finite vertex set, and the number of vehicles of TO-VRPTW is an integer value within a certain range. Therefore, when calculating about TOVRPTW, we firstly classify initial population according to the total number of vehicles in PF^3. PF^3 has been reached on objective f_1, meeting a criteria of the optimal model I.

Let total number of vehicles in PF^3 be in the range $[m, n]$. Bi-objective VRPTW aiming at the shortest distance and the farthest vehicle distance is composed by objective functions (2),(3) and a constraint condition (4). The mathematical model of bi-objective VRPTW is called optimization model II.

Theorem 1 Let PS^2 be a set of Pareto solutions of optimization model II then $n = \max\left\{f_1(x)|x \in R^2\right\}$ be a upper bound of the number of vehicles in PF^3.

Proof:For $\forall x^{'}\left(x^{'} \in \Omega \wedge f_1(x^{'}) > n\right)$,since PS^2 is a set of Pareto solutions of optimization model II , $\exists x\left(x \in PS^2\right)$,$f_2(x^{'}) \geq f_2(x) \wedge f_3(x^{'}) \geq f_3(x)$ and $f_1(x^{'}) > n \geq f_1(x)$.Therefore,$x \succ x^{'}$ is in a set PS^3, that n is a upper bound of the number of vehicles in PF^3.

According to theorem 1, in the TO-VRPTW, it is feasible to acquire total number of vehicles in PF^3 at the first time. therefore, we use the population structure classified according to the number of vehicles. The specific population structure is as follows:

$$D = \begin{pmatrix} m & x_{m,1} & ... & x_{m,k_m} \\ m+1 & x_{m+1,1} & ... & x_{m+1,k_{m+1}} \\ ... & ... & ... & ... \\ n & x_{n,1} & ... & x_{n,k_n} \end{pmatrix}$$

Where $x_{i,j}\,(m \leq i \leq n, 1 \leq j \leq k_i)$ is the j-th individual of class i, k_i is the number of individuals of class i, the number of population is $N = k_m + k_{m+1} + ... + k_n$.

3.3.2 Initial population acquisition

In order to obtain the required number of vehicles of all kinds, an idea of two-stage is used to design to acquire the initial population in our algorithm: In

the first stage, a single-objective optimization model consisted of the objective function (1) and the constraint condition (4) is calculated to obtain a lower bound . In the second stage, a double-layer multi-objective optimization model is designed, and this model is calculated to obtain an upper bound n.

The double-layer multi-objective optimization model includes constraint condition (4) ,P_1 and P_1.
P_1 and P_1 are shown as follows:

P_1 : $\min F_1 = (f_2, f_3)$

P_2 : $\max F_2 = f_1$

In the model, objective P_1 has a higher priority than objective P_2. Hence, an optimal solution calculated by the model is to maximize objective f_1 under the premise that objectives f_2 and f_3 are both the smallest.

In order to calculate an initial population, we just need to use an intelligent optimization algorithm to solve a single-objective optimization model at first; And then, a bi-objective NSGA-II of (f_2, f_3) is used to solve the double-layer multi-objective optimization model. Among them, dominance relationship of a bi-objective (f_2, f_3) is used to solve problems of fast non-dominated stratification, and ordering problem is solved according to the value of objective f_1 in each layer.

Let I be an initial population, Q be acquired initial population.

Algorithm 2: get-init-pop(I)

1: $t \leftarrow 0$
2: **Repeat**
3: $q_t \leftarrow$ car-generate(I_t) //f_1 calculates to get an optimal individual for each generation
4: $Q_1 \leftarrow q_t$
5: $m \leftarrow Min(car)$ //Get the minimum number of vehicles
6: $t \leftarrow t + 1$
7: **Until** end generation
8: $t \leftarrow 0$
9: **Repeat**
10: $Q_2 \leftarrow$ car-cardis-generate(I_t) //f_2, f_3 bi-objective calculates to get a set of Pareto solutions
11: **Until** termination condition fulfilled
12: $n \leftarrow Max(Q_2)$ // Get the maximum number of vehicles
13: $Q \leftarrow$merge (Q_1,Q_2) // Get an initial population

The algorithm specific process for calculating an initial population is shown in Algorithm 2. Since there are one and two objectives in the first and second stages, the population size of algorithm is reduced by at least one order of magnitude compared to tri-objective. Without calculation of crowding degree, complexity of this algorithm is further reduced. The selection of tournament makes the design of fitness value more simplified. Therefore, our algorithm designed for calculating initial population plays a role in reducing the dimension of objective functions.

3.3.3 Distribution of Fitness Values

To ensure that individuals' distribution between intra-population class and inter-population class are nearly uniform and the population approaches to PF^3 during iterative processes. We design a method for assigning population fitness values, which operates on each type of population D as follows:

1) Perform fast non-dominated stratification in the class according to dominance relationship of bi-objective (f_2, f_3),

2) In each layer, sort in ascending order of crowding degree according to the crowding degree relationship of bi-objective (f_2, f_3),

3) In the class, according to dominance relationship and crowding degree relationship, we number population D progressively from good to bad in order to get Matrix D^*:

$$\mathrm{D}^* = \begin{pmatrix} m & k_m & ... & 2 & 1 \\ m+1 & k_{m+1} & ... & 2 & 1 \\ ... & ... & ... & ..... & \\ n & k_n & ... & 2 & 1 \end{pmatrix}$$

4) Normalize within the class

$$p_{i,j} = \frac{j}{\sum_{j=1}^{k_i} j}, (m \le i \le n, 1 \le j \le k_i)$$

The fitness value matrix obtained after normalization is as follows:

$$P = \begin{pmatrix} m & p_{m,1} & p_{m,2} & ... & p_{m,j} \\ m+1 & p_{m+1,1} & p_{m+1,2} & ... & p_{m+1,j} \\ ... & ... & ... & ... & ... \\ n & p_{n,1} & p_{n,2} & ... & p_{n,j} \end{pmatrix}$$

The distribution of fitness values has following advantages compared to traditional NSGA-II:

1) Since this algorithm uses selection method of tournament, it only needs to be normalized in each class, thus reducing its computational costs.

2) Since the calculation of dominance relationship and crowding degree is carried out under bi-objective, compared with tri-objective, complexity of this algorithm is further reduced in layering and sorting, which plays a role in reducing the dimension of the objective functions.

3) The distribution of fitness value ensures that the number of individuals between classes is uniform on the premise that the population approaches PF^3.

3.4 Main Loop Operator of Intermediate Generation

The traditional NSGA-II population production process: at first, generate intermediate generation through genetic operators; secondly, we merge them with parent population; finally, we can obtain the offspring through a certain kind of deletion rule. The population which is merged by intermediate generation and

parent needs to obtain the basis of deletion by calculating fitness value. The main computational burden of this part is recalculation of fitness value. Therefore, how to reduce calculation of fitness value at this stage is the motivation for **" main loop operator of intermediate generation (middle-generation-update)"** proposed in this question. The idea of this main loop is: "taking parent as intermediate generation; the individual generated by each genetic operation is first inserted into intermediate generation, and then the intermediate generation is updated according to deletion rule until all genetic operations are completed; the intermediate generation is the offspring population." This main loop avoids double counting effectively.

The implementation of "gradually optimizing the main loop of intermediate generation" is dependent on two rules: the insertion rule and the deletion rule, which are described in detail below.

Insertion rule(add(q)): Let the individual to be inserted q belong to the i-th class; traversing each individual in the i-th class in turn; a sum of the number of individuals that dominate q is the dominance level of q, and the individual level dominated by q will increase one unit, and for the class which q is located re-arranging by fast non-domination stratification and re-sorting by crowding degree.

Deletion rule(del($Q, fit\_$ min)): Recalculate fitness value for the i-th class; select individuals with the smallest fitness value among populations to delete.

Let Q be an initial intermediate generation template, and operations of gradually optimizing the main loop of the intermediate generation as shown in Algorithm 3.

Algorithm 3: middle-generation-update(Q)

1: Q^* GA(Q) // Genetic manipulation results in intermediate generation to be inserted
2: For each $q \in Q^*$
3: $n_q = 0$
4: For each $p \in Q$
5: if($p \succ q$)
6: n_q++ //$n_q$is the dominant level of q
7: if($q \succ p$)
8: n_p++
9: fast-non-dominated-sort($Q[q]$) // Non-dominated ranking of the class in which q is located
10: crowding ($Q[q]$) //Calculate the crowding degree of q
11: add(q) //Insert q
12: del($Q, fit\_$ min) // Delete individuals with the lowest fitness value

In the middle-generation-update, the crowding degree calculation rule ensures that one class is more balanced, and the fitness value allocation strategy ensures a more balanced class. It achieves this purpose of PF^3 to be more uniform. The flow chart for gradually optimizing intermediate generation is as follows:

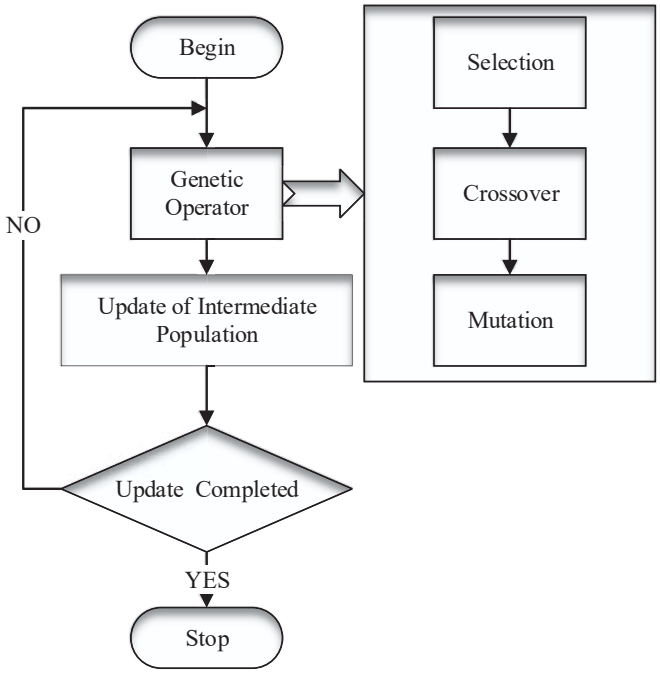


Fig. 2. the flow chart for gradually optimizing main loop of intermediate generation

4 Experimental Results and Analysis

In this paper, we verify feasibility and rationality of TSCOA through experiments. In the first part, we will introduce test instances, parameter tuning, performance metrics and experimental comparisons in sections 4.1 and 4.2. In the second part, we respectively compare TSCOA with other algorithms and the best known solutions in sections 4.3 and 4.6. In order to test performance of the proposed algorithm, we use a test set of Solomon's benchmarks for simulation experiments. The experimental environment for all experimental tests: experimental software:VS2013; programming language: C language; processor: Intel(R)Core(TM) i5-5300CPU 2.3GHz, Windows10,64 bits.

4.1 Test Case and Parameter Settings

This paper uses the well-known Solomon's benchmark test set to analyze computational performance of our algorithm, and compares solutions in the Solomon test set with solutions obtained by this algorithm, thus reflecting the reference significance of our algorithm in solving TO-VRPTW.

In order to ensure fairness of comparison experiments, the improved algorithm and the comparison algorithm are optimized under the premise of the program. We use traversal method to search for the best parameters, and control time and space complexity of two kinds of algorithm at the same level for comparison experiments. In the case of determining the number of iterations and the size of the population, we adjust the crossover probability Pc and the mutation probability Pm to get an optimal value with Pc=0.75, Pm=0.1. Similarly, after determining the probability parameters, we increase the number of iterations and the size of population while population size PN is 400 to 2000, an optimal initial population can be obtained in case of number of iterations of the three single objectives in the first stage (*N1, N2, N3*) is from 2000 to 10000, an optimal-solution set can be got in case of the number of iterations of multi-objective optimization DN in the second stage is from 400 to 800. When calculating performance metrics in Table 4, we set test population size and the number of iterations as shown in Table 1.

Table 1. setting test case parameter

Instance	*PN*	(N1,N2,N3)	*DN*
C101	1000	(2000, 5000, 1000)	400
C105	1000	(2000, 5000, 1000)	400
C201	1000	(2000, 5000, 1000)	400
R109	1000	(2000, 10000, 2000)	600
R112	1000	(2000, 10000, 2000)	600
R207	1000	(2000, 10000, 2000)	600
RC101	1000	(2000, 10000, 2000)	800
RC106	1000	(2000, 10000, 2000)	800
RC201	1000	(2000, 10000, 2000)	800

4.2 Performance Metric

In this paper, the experimental test of TSCOA calculation performance is compared with the three requirements of **"Evaluating Optimal Solutions Criterion"**. To this end, we design the evaluation index function from three aspects: the non-dominated front $NF(\varphi)$ of optimal solutions close to the extent of PF while its distribution and extensibility. we design the evaluation index function from three aspects: the non-dominated front $NF(\varphi)$ of optimal-solution set closes to the extent of PF, breadth of distribution, uniformity of distribution. We name them C index, S index and H index. Let A denote a solution set of NSGA-II and B denote a solution set of TSCOA.

1) The C index indicates degree of dominance between two solution sets, and degree of dominance of solution A to solution B is expressed as follows.

$$C(A,B) = \frac{|\{u \in NS(B) \,|\exists v \in NS(A) : v \succ u\}| \times \mu(A)}{|NS(B)|}$$

$NS(A)$ and $NS(B)$ are respectively a set of non-dominated solutions of solution set A and solution set B. When comparing the extent of solution set A and solution set B close to PF, we need to compare approximation efficiency of solution set, in addition to the comparison between non-dominated sets, that is, comparisons of densities of non-dominant solutions. $\mu(A)$ is the density of a set of non-dominated solutions in solution set A, $\mu(A) = \frac{|NS(A)|}{|A|}$.

2) The S index indicates the extensibility of the non-dominated solutions front, the extensibility of solution A is expressed as follows:

$$S(A) = \sum_{i=1}^{m} \frac{f_i^{\max}(NS(A)) - f_i^{\min}(NS(A))}{f_i^*}$$

$f_i^{\max}(NS(A))$ and $f_i^{\min}(NS(A))$ are respectively the upper and lower bounds of the non-dominated front of solution set A on the i-th objective dimension. f_i^* is an optimal solution value of the i-th objective function.

3) The H index indicates uniformity of the non-dominated front distribution of solution set, which can be described by variance of crowding degree of the non-dominated front distribution. It can be shown as follow:

$$H(A) = \frac{\sum_{j=1}^{m} \sum_{x \in \overline{NS}_j(A)} \left(C_j(x) - \bar{C}_j(NS(A))\right)^2}{|NS(A)| - 2}$$

Where $\bar{C}_j(NS(A))$ represents the average crowding degree of $\overline{NS}_j(A)$ which is a set of non-dominated solutions $NS(A)$ of A removing two endpoints from the j-th objective dimension.

4.3 Comparison and analysis of TSCOA and NSGA-II

To illustrate effectiveness of TSCOA, we compare optimization effects of TSCOA and NSAG-II according to TO-VRPTW. Table 2 shows the non-dominated front calculated by NSGA-II for each instance; Table 3 shows the non-dominated front calculated by TSCOA for each instance; Table 4 shows the performance metric data calculated by these two algorithms. In Tables 2 and 3, the set A represents optimal-solution set calculated by NSGA-II; the set B represents optimal-solution set calculated by TSCOA; data in these tables use the representation of $(f_1(x), f_2(x), f_3(x))$; the bolded part in Table 4 indicates that the performance metrics are dominant.

It is known from Table 2 and Table 3 that, in the case of same population size, except for case C201 and C105, the scale of vertexes obtained by TSCOA on the non-dominated front is no less than that obtained by NSAG-II. It can be seen that the elite acquisition ability of TSCOA is stronger than NSAG-II. It is known from Table 4 that TSCOA is comprehensively superior to NSGA-II in terms of C and S indexes. It shows that: 1) TSCOA is closer to than NSGA-II. TSCOA has the most outstanding performance on set R and its performance

Table 2. optimal-solution set calculated by NSGA-II

Instance	$NF(A) = \{(f_1(x), f_2(x), f_3(x)) \mid x \in NS(A)\}$
C101	(10,828.937,127.297), (11,940.352,120.813), (12,983.638,120.725), (14,1035.11,120.722) (21,1780.53,117.047)
C105	(10,828.937,127.297), (12,978.128,120.725), (12,1050.82,118.979), (14,1113.56,118.074) (14,1120.15,117.900), (17,1201.99,117.891), (18,1290.66,117.047)
C201	(4,641.026,232.075), (5,727.428,194.173), (6,819.874,165.409), (10,953.61,163.097) (11,1018.19,142.864), (13,1319.8,136.551), (15,1498.11,130.267), (16,1557.41,128.258), (18,1662.65,126.285), (19,1807.22,125.089), (21,1885.53,123.619), (24,1991.4,117.237)
R109	(14,1262.8,130.152), (15,1431.39,120.025), (16,1460.89,117.682), (17,1500.29,115.985) (18,1456.35,104.053), (19,1480.47,103.472), (20,1609.68,100.608), (21,1619.14,99.8599)
R112	(11,1010.96,128.553), (12,1092.5,117.581), (13,1134.72,115.874), (14,1199.89,114.859) (15,1301.89,104.125), (16,1364.85,104.055), (18,1490.35,103.179)
R207	(4,1308.17,392.373), (5,1120.87,265.386), (5,961.696,278.575), (6,1179.07,204.04) (6,919.885,299.814), (7,955.226,146.77), (7,919.885,149.757), (8,984.073,143.99) (9,1054.04,132.447), (9,957.117,147.2), (10,1108.58,117.772), (11,1153.14,116.164) (12,1204.07,115.795), (13,1264.32,113.245), (14,1347.79,106.824), (16,1453.88,103.828), (17,1476.13,103.72), (19,1578.44,103.719), (20,1610.21,103.704)
RC101	(18,1863.22,123.456), (18,1799.73, 127.465), (19,1903.2,123.387), (19, 1799.73, 127.465), (20,1931.28,123.387), (21,2036.82,119.965), (22,2088.46,119.443), (25,2227.39,118.537) (29,2393.87,117.047)
RC106	(14,1480.15,163.237), (15,1516,143.359), (16,1594.81,129.821), (18,1711.83,123.526), (19,1853.78,123.013), (21,1889.77,122.629), (20,2019.94,117.047), (21,2012.95,117.047)
RC201	(6,1497.630,488.211), (7,2117.52,477.698), (8,2431.52,371.876), (8,2108.03,352.881) (9,2247.15,270.33), (9,1870.1,304.614), (10,1456.67,219.41), (11,1497.65,203.187) (11,1394.56,203.956), (12,1494.31,150.599), (14,1559.28,142.428), (15,1620.96,132.586) (16,1705.59,127.096), (17,1823.42,124.018), (18,1827.32,121.885), (19,1850.19,119.789) (20,2056.1,117.494), (21,2057.82,117.237), (22,2149.36,117.047)

Table 3. optimal solutions calculated by TSCOA

Instance	ABC
C101	(10,828.937,127.297), (12,983.638,125.093), (14,1035.11,120.722), (21,1780.53,117.047) (21,1657.52,119.128)
C105	(10,828.937,127.297), (12,978.128,120.725), (12,1050.82,118.979), (14,1113.56,118.074) (17,1201.99,117.891), (18,1290.66,117.047)
C201	(3,591.557,238.208), (4,640.722,195.3), (5,697.187,160.815), (9,893.214,152.015) (10,934.786,152.015), (13,1238.73,145.075), (23,1761.83,139.052), (24,1853.61,139.052) (25,2019.31,117.047)
R109	(13,1230.16,120.985), (13,1237.270,116.244), (14, 1249.570, 113.626), (15,1318.420,106.823) (16,1393.860, 105.715), (17,1303.510,101.458), (19,1584.680,100.896), (20,1619.440,100.165) (21, 1664.630, 99.860)
R112	(10,953.630,177.426), (11,1010.96,128.553), (12, 1150.890, 117.365), (12, 1061.440,127.393) (13,1186.860, 108.578), (13,1090.470, 125.391), (14, 1279.790,104.479), (15, 1161.230,104.479) (16,1345.410, 103.179), (17,1363.620, 100.310)
R207	(3,870.330,504.677), (4,1479.620,492.628), (4,1598.850,480.433), (5,941.234,231.747) (5,890.669,244.677), (6,900.846,178.058), (6,932.096,177.943), (7,1031.840,168.448) (7,912.740,178.396), (7,1014.780,169.090), (8,1012.860,141.219), (9,1057.730,136.851) (10,1077.770,133.609), (11,1156.750,127.540), (12,1070.350,117.943), (13,1280.080,114.772) (13,1295.340,112.971), (14,1312.410,112.971), (15,1457.150,109.832), (16,1464.180,107.975)
RC101	(16,1749.170,158.033), (18,1830.870,145.488), (20,1977.310,122.963), (21,2095.120,118.078) (22,2107.15,123.214), (22,1970.67,131.992), (23,2174.99,121.465), (26,2284.03,118.077) (27,2376.1,117.047)
RC106	(13,1547.58,147.638), (13,1387.630,149.765), (14,1584.04,147.447), (15,1543.93,134.624) (16,1657.52,131.368), (17,1721.92,121.167), (18,1825.39,118.494), (18,1816.87,120.382) (19,1910.65,117.602), (20,2019.94,117.047), (21,2012.95,117.047)
RC201	(4, 1406.940, 447.663), (5,1279.650,441.433), (6,1397.630,401.211), (7,2368.1,373.728) (7,2365.87,381.191), (8,2118.16,350.171), (8,2108.03,352.881), (9,2247.15,270.33) (9,1863.6,334.448), (9,1664.18,343.694), (10,1456.67,219.41), (11,1497.65,203.187) (11,1394.56,203.956), (12,1494.31,150.599), (14,1559.28,142.428), (15,1620.96,132.586) (16,1705.59,127.096), (17,1823.42,124.018), (18,1827.32,121.885), (19,1850.19,119.789) (20,2056.1,117.494), (21,2057.82,117.237), (22,2149.36,117.047)

Table 4. calculation results of TSCOA and NSGA-II performance metrics

Instance	NSGA-II			TSCOA		
	A	B	C	D	E	F
C101	**0**	**3.124**	0.86	**0**	**3.124**	**0.814**
C105	**0**	**1.445**	0.177	**0**	**1.445**	**0.116**
C201	2	9.759	0.274	**2.25**	**10.782**	**0.11**
R109	0.875	1.078	0.35	**5.714**	**1.124**	**0.302**
R112	0.7	1.543	0.521	**4.28**	**1.893**	**0.047**
R207	2.85	7.96	0.047	**6.316**	**8.69**	**0.031**
RC101	**0**	1.041	0.098	**0**	**1.271**	**0.071**
RC106	0	1.289	0.256	**1.375**	**1.351**	**0.231**
RC201	1.056	7.012	0.047	**2.842**	**7.078**	**0.015**

on set RC is second. The C index difference of R109, R112 and R207 is 5.039, 3.58 and 3.526 respectively. Therefore TSCOA is superior to NSGA-II in deep optimization ability. In particular, the more complex the network structure is, the more prominent performance of our algorithm is. 2) The breadth distribution of TSCOA on the non-dominated front is better than that of NSGA-II. The average value of relative gap $^{S(A)-S(B)}/_{S(A)}$ of C index is 0.136. Instances above average are R112 and RC101. Especially, the performance of this algorithm is more outstanding for test instances of sets R1, C1 and RC1. From Table 4, the uniformity of TSCOA distribution on the non-dominant front is better than that of NSGA-II. In summary, the computing performance of TSCOA is better than that of NSGA-II, so the improvement of NSGA-II is effective.

4.4 Comparison and analysis of TSCOA and MOEA/D-aLS

In this section, TSCOA is compared with MOEA/D-aLS, where f_1 andf_2 represent the number of vehicle and total travel cost objective respectively. In order to compare with the algorithm MOEA/D-aLS proposed in reference [23], we will replace $f_3$with the goal of routes balancing [25] in this section. The results are shown in Table 5. It can be seen from the overall data that the computing performance of TSCOA is significantly better than that of MOEA/D-aLS. Among them, the calculation results of set C in f_1, f_2 and f_3 are reduced by 0.86, 142.158 and 1.23, respectively. In the set R, the average reduction was 1.50, 178.95 and 0.06, respectively. In the set RC, 1.50, 194.57 and 0.03 were reduced respectively. From these data, we can see that TSCOA has obvious advantages in calculating TO-VRPTW.

4.5 Comparison and Analysis of TSCOA with Other Algorithms

This section compares TSCOA with HMOEA, MOGA, MACS-IH, HSFLA, ILNSA, LGA, HACO and ACO-Tabu. The data used for comparison are bi-objective $(\bar{f_1}, \bar{f_2})$ mean values of six different sets of test instance. We can see

Table 5. experimental comparison between TSCOA and MOEA/D-aLS

Instances	TSCOA			MOEA/D-aLS		
	f1	f2	f3	f1	f2	f3
C101:	10	828.9	7.2	10	828.9	7.2
C105:	**10**	**828.9**	**5.6**	11	956.4	8.3
C107:	**10**	**828.9**	**3.6**	10	889.3	9.4
C109:	**10**	**928.9**	2.2	10	992.9	2.2
C201:	**3**	**591.6**	1.5	7	954	1.5
C205:	**3**	**597.4**	0.5	6	854.8	0.5
C208:	**3**	**588.5**	0.2	4	711.9	0.3
R101:	**20**	**11669.3**	8	22	1789	8
R102:	**17**	**1486.9**	3.9	19	1603.3	3.9
R105:	**15**	**1421.2**	3.5	17	1556.8	3.5
R109:	**13**	**1230.1**	**2.4**	14	1362.9	2.7
R112:	**10**	**953.6**	1.7	11	1199.8	1.7
R201:	**5**	**1214.2**	5	9	1538	5
R207:	3	**870.3**	**0.1**	3	1120.3	0.3
R211:	**4**	**827.6**	0.2	4	934.8	0.2
RC101:	**16**	**1749.2**	6.4	17	1879	6.4
RC106:	**13**	**1387.6**	2.5	14	1528.9	2.5
RC108:	**11**	**1161.5**	1.5	13	1363.1	1.5
RC201:	**4**	**1406.9**	2	9	1535	2
RC206:	**4**	**1153.5**	**0.3**	6	1530.9	0.5
RC207:	**4**	**1092.5**	0.2	4	1281.7	0.2

calculation results in Table 6. In Table 6, these eight algorithms can solve bi-objective $(\bar{f}_1, \bar{f}_2)$ optimization problem, while TSCOA can solve tri-objective $(\bar{f}_1, \bar{f}_2, \bar{f}_3)$ optimization problem. Therefore, we can only compare optimal solutions of bi-objective $(\bar{f}_1, \bar{f}_2)$ in tri-objectives$(\bar{f}_1, \bar{f}_2, \bar{f}_3)$ with those of the eight algorithms.

Table 6. calculation results of TSCOA and other algorithms

algorithm	C1	C2	R1	R2	RC1	RC2
HMOEA[14]	(10, 828.74)	(3, 590.688)	(12.916,1187.35)	(3.545,951.742)	(12.375,1355.365)	(4.250,1068.255)
MOGA[26]	(10, 828.48)	(3, 590.531)	(12.666,1212.58)	(3.090,956.734)	(12.375,1379.865)	(3.500,1148.661)
MACS-IH[27]	(10, 828.38)	(3, 589.858)	(11.916,1209.89)	(2.727,951.660)	(11.500,1384.164)	(3.25,1119.173)
HSFLA[28]	(10, 828.38)	(3, 589.858)	(11.916,1210.34)	(2.727,951.030)	(11.500,1384.165)	(3.25,1119.289)
ILNSA[29]	(10, 833.10)	(3, 591.313)	(12.250,1218.28)	(3.272,964.109)	(12.125,1369.566)	(3.75,1131.185)
LGA[30]	(10, 828.38)	(3, 590.393)	(13.500,1188.85)	(5.545,886.025)	(13.250,1360.961)	(6.875,1013.288)
HACO[31]	(10, 838.38)	(3, 644.876)	(12.5,1203.38)	(3.272,916.738)	(12.125,1363.366)	(3.75,1132.594)
ACO-Tabu[32]	(10, 829.01)	(3, 590.778)	(13, 1105.915)	(4.181,951.361)	(12.25,1380.554)	(4.75,1095.844)
TSCOA	(10, 828.48)	(3, 590.477)	(13, 1188.35)	(3.636,887.250)	(12.375,1355.365)	(3.75,1117.113)

We know that in order to achieve same calculation accuracy, the corresponding population size needs to be enlarged to a power order for each additional dimension of the objective function. For example, if population size of single-objective is 10^1, then the population size of bi-objective and tri-objective needs to reach 10^2 and 10^3 in order to achieve same calculation accuracy. The pressure of computational space and time makes the population size of TSCOA unable to meet such requirements. Table 6 reflects the results of tri-objective TSCOA compared with the other eight algorithms which are bi-objective in the absence of fairness. Therefore, if the results of TSCOA are not significantly different from the best results of other eight algorithms, it can be concluded that TSCOA has a certain degree of competition.

In order to quantify relative superiority of TSCOA and other algorithms, we calculate relative ratio of TSCOA results on the i-th dimension objective to optimal results $V_i = \frac{f_i^{CTSNSGA-II} - f_i^{\min}}{f_i^{\max} - f_i^{\min}}$, $f_i^{CTSNSGA-II}$ represents mean value of the i-th objective of TSCOA, respectively represent the upper and lower bounds of the i-th dimension objective. The smaller is, the closer TSCOA results are to the optimal algorithm. Fig.3 shows scatter plot of results of nine algorithms in Table 6. Analysis from Fig.3 is as follows:1) In C1, TSCOA basically coincides with optimal solutions of other algorithms, and $(V_1, V_2) = (0,\ 0.012)$. It indicates that TSCOA has obvious advantages in computing results in sets C1 and C2. 2) In R1, TSCOA is not dominant in scatter plot, and $(V_1, V_2) = (0.684,\ 0.734)$; In R2, TSCOA is located on the lower side of nine vertexes front in scatter plot. It indicates that calculation results of TSCOA in set R are not superior to that of set C, and $(V_1, V_2) = (0.321, 0.014)$. However, TSCOA is generally above medium level and still has certain advantages. 3) In RC1, TSCOA is located on the lower side of nine vertexes front in scatter plot, $(V_1, V_2) = (0.5,\ 0)$; in RC2,

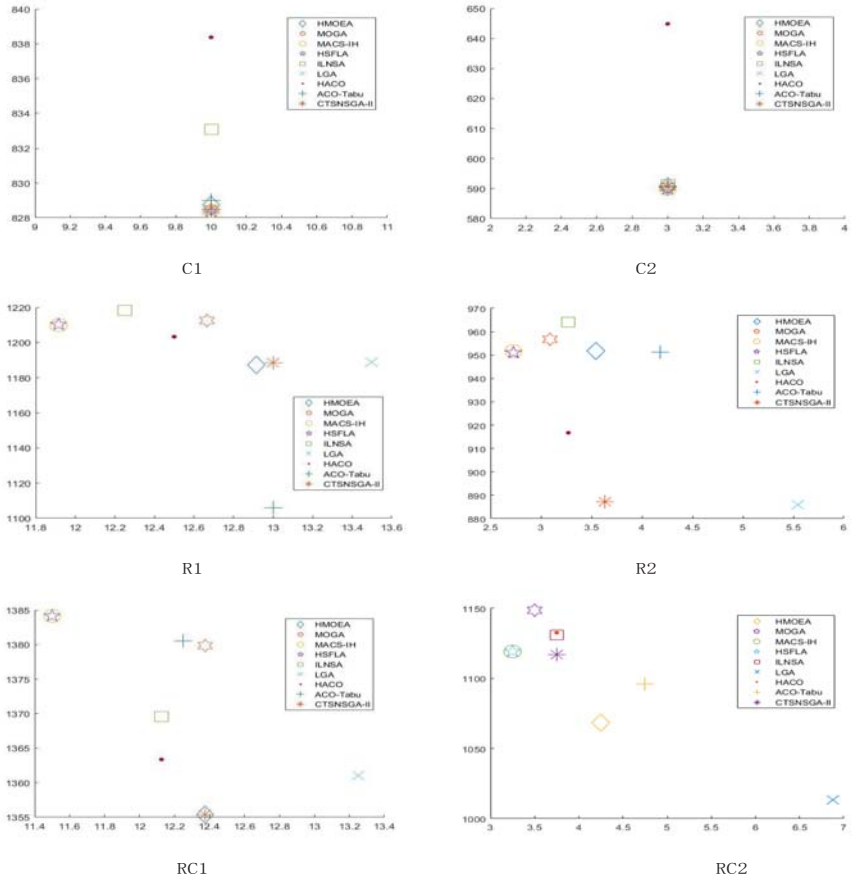


Fig. 3. Scatter plot in results of TSCOA and other algorithms

TSCOA is not dominant in scatter plot, $(V_1, V_2) = (0.138, 0.767)$. It indicates that calculation results of TSCOA in set RC are better than those in set R, but not as good as those in set C. Similarly, TSCOA is generally above medium level and still has certain advantages. In summary, compared with other algorithms, TSCOA is competitive.

4.6 Comparison and Analysis of TSCOA and Optimal Solutions

To evaluate superiority of TSCOA computing performance, we compare a solution of $NF(\varphi)$ with the best-known data. Among them. is the difference between a solution of $NF(\varphi)$ and the best data: $dif < 0$. indicates that the calculated $NF(\varphi)$of TSCOA is better than the best data; $dif > 0$ indicates that the calculated $NF(\varphi)$of TSCOA is no better than the best data; $dif = 0$ indicates that

the calculated $NF(\varphi)$of TSCOA is equal to the best data. Data in these tables use the representation of (f, f_2, f_3).

Table 7. comparisons with the best data

Instance	Best Known	TSCOA	*dif*
C101	(10, 828.937, 127.297)	(10, 828.937,1 27.297)	(0, 0, 0)
C105	(10,828.937, 127.297)	(10, 828.937, 127.297)	(0, 0, 0)
C107	(10, 828.9367,127.297)	(10, 828.937, 127.297)	(0, 0, 0)
C109	(10, 828.937, 127.297)	(10, 828.937, 127.297)	(0, 0, 0)
C201	(3, 591.556, 238.208)	(3, 591.556, 238.208)	(0, 0, 0)
C205	(3, 588.876, 235.528)	(3, 597.384, 238.706)	(0, 8.508, 0.178)
C208	(3, 588.324, 235.528)	(3, 588.493, 235.528)	(0, 0.169, 0)
R101	(19, 1650.799, 132.490)	(20, 1669.320, 118.417)	(1, 18.521, -14.073)
R102	(17,1486.859, 134.252)	(17, 1486.859, 134.252)	(0, 0, 0)
R105	(14, 1377.110, 126.362)	(15, 1421.210, 126.362)	(1, 44.1, 0)
R109	(11, 1194.734, 128.266)	(13, 1230.16, 120.985)	(2, 35.426, -7.281)
R112	(9, 982.139, 129.341)	(10, 953.630, 177.426)	(1, -28.509, 49.915)
R201	(4, 1252.370, 427.868)	(5, 1214.220, 409.877)	(1, -38.150, -17.991)
R207	(2, 890.608, 453.269)	(3, 870.330, 404.677)	(1, -20.278, -48.592)
R211	(2, 885.711, 469.241)	(4, 827.641, 630.573)	(2, -58.07, 161.332)
RC101	(14, 1696.949, 158.033)	(16, 1749.170, 158.033)	(2, 52.221, 0)
RC106	(11, 1424.733, 167.765)	(13, 1387.630, 149.765)	(2, -37.103, -18)
RC108	(10, 1139.821, 136.295)	(11, 1161.52, 129.728)	(1, 21.699, -6.567)
RC201	(4, 1406.940, 447.663)	(4, 1406.940, 447.663)	(0, 0, 0)
RC206	(3, 1146.317, 476.613)	(4, 1153.48, 416.86)	(1, 7.163, -59.753)
RC207	(3, 1061.144, 419.829)	(4, 1092.48, 407.919)	(1, 31.336, -11.91)

From Table 7, we can see the comparison between solutions of our algorithm and the best known in Solomon database. On set C, result of our algorithm is basically the same as that of the best known. Among them, the f_2 and f_3 objectives corresponding to C205 differ 1.44% and 1.35% from the best solution, while the f_1 objective corresponding to C208 differ only 0.169 from the best solution, while the other differences are all 0. On sets R and RC, the results of TSCOA show that the f_3 objective decreases by 12.35 on average compared with the best solution. Among them, the maximum gap of f_2 objective corresponding to R109 is only 2.95%, the maximum gap of f_1 objective is 2, and the reduction of f_3 objective corresponding to RC206 is 59.753. It can be seen that TSCOA has obvious advantages in solving TO-VRPTW.

5 Conclusion

In this paper, we consider a TO-VRPTW aiming to minimize number of vehicles, total distance and distance traveled by the farthest vehicle. Therefore, we propose the two-stage multi-objective optimization algorithm based on classified

population (TSCOA) for solving TO-VRPTW. Firstly, the initial population structure is classified, and then we optimize TO-VRPTW by using the idea of two-stage hierarchy of single-objective and multi-objective. Our framework of TSCOA divides the algorithm into two stages: initial population acquisition and multi-objective search. The first stage, an initial population with optimal individual distribution structure is obtained by global search of simple objective and bi-objective. The second stage, we use "gradually optimizing main loop of intermediate generation" of NSGA-II to perform global optimization. In this paper, we have improved the optimization process of the original NSGA-II, such as using elite strategy of parent generation as template while expanding crowding degree, reducing time complexity by staged dimensionality reduction. In this algorithm, both idea of population classification and elite strategy based on paternity template are used to make distribution of PF more uniform. Finally, through comparative experiment of Solomon, it is fully shown that TSCOA has better effect than the existing algorithms in optimization of TO-VRPTW. In the future research work, we will continue to improve TSCOA and apply it to other fields besides grain transportation, in order to further verify the practicability of this algorithm.

Acknowledgement

This work is partially supported by Key Project of Philosophy and Social Science Research Project of Hubei Provincial Department of Education in 2019(19D59)

References

1. M.M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, Oper. Res. 35 (2) (1987) 254–265.
2. B. Eksioglu, A. V. Vural, and A. Reisman, "The vehicle routing problem:A taxonomic review,"Comput. Ind. Eng., vol. 57, no. 4, pp. 1472–1483,2009.
3. R. Lahyani, M. Khemakhem, and F. Semet, "Rich vehicle routing problems:From a taxonomy to a definition," Eur. J. Oper. Res., vol. 241,no. 1, pp. 1–14, 2015.
4. K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," IEEE Trans. Syst., Man, Cybern.,Syst., vol. 47, no. 1, pp. 70–85, Jan. 2017.
5. N. Jozefowiez, F. Semet, E. Talbi, Multi-objective vehicle routing problems, Eur. J. Oper. Res. 189 (2) (2008) 293–309.
6. J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jiménez, and N. Herazo-Padilla, "A literature review on the vehicle routing problem with multiple depots," Comput. Ind. Eng., vol. 79, pp. 115–129, Jan. 2015.
7. R. Baldacci, A. Mingozzi, R. Roberti, Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints, European Journal of Operational Research 218 (1) (2012) 1–6.
8. N. A. El-Sherbeny, Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods, Journal of King Saud University (Science) (22) (2010) 123–131.

9. S. C. Hong, Y. B. Park, A heuristic for a bi-objective vehicle routing with time window constraints, International Journal of Production Economics 62 (3) (1999) 249–258.
10. G. Vaira, O. Kurasova, Genetic algorithm for VRP with constraints based on feasible insertion, Informatica 25 (1) (2014) 155–184.
11. D.M. Pierre, M.N. Zakaria, Partially optimized cyclic shift crossover for multi-objective genetic algorithms for the multi-objective vehicle routing problem with time-windows, in: 2014 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making, MCDM 2014, Orlando, FL, USA, December 9–12, 2014, 2014, pp. 106–115.
12. K. Deb, S. Agrawal, A. Pratap, et al, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II[C]//International conference on parallel problem solving from nature. Springer, Berlin, Heidelberg, 2000: 849-858.
13. N. Srinivas, K. Deb, Multi-objective optimization using nondominated sorting in genetic algorithms[J]. Evolutionary computation, 1994, 2(3): 221-248.
14. K. C. Tan, Y. H. Chew, L. H. Lee, A hybrid multi objective evolutionary algorithm for solving vehicle routing problem with time windows, Computational Optimization and Applications 34 (1) (2006) 115–151.
15. M. J. Geiger, W. Wenger, W. Habenicht, Interactive utility maximization in multi-objective vehicle routing problems: A "decision maker in the loop"-approach, in: Proceedings of the IEEE Symposium on Computational Intelligence in Multicriteria Decision Making, 2007, pp. 178–184.
16. Y. J. Gong, J. Zhang, O. Liu, Optimizing the vehicle routing problem with time windows: A discrete particle swarm optimization approach, IEEE Transactions on Systems, Man, and Cybernetics - Part C 42 (2) (2012) 254–267.
17. Xingyi Zhang, Ye Tian, Ran Cheng, Yaochu Jin. A Decision Variable Clustering-Based Evolutionary Algorithm for Large-scale Many-objective Optimization. IEEE Transactions on Evolutionary Computation,2018, 22(1): 97-112
18. X. Wang, C. Xu, and H. Shang, "Multi-depot vehicle routing problem with time windows and multi-type vehicle number limits and its genetic algorithm," in Proc. 4th IEEE Int. Conf. Wireless Commun. Netw. Mobile Comput., 2008, pp. 1–5.
19. U. Dharmapriya, S. Siyambalapitiya, and A. Kulatunga, "Artificial intelligence computational techniques to optimize a multi objective oriented distribution operations," in Proc. Int. Conf. Ind. Eng. Oper. Manag., 2010.
20. Wang J, Weng T, Zhang Q. A Two-Stage Multiobjective Evolutionary Algorithm for Multiobjective Multidepot Vehicle Routing Problem With Time Windows[J]. IEEE transactions on cybernetics, 2018, 49(7): 2467-2478.
21. J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jiménez, and N. Herazo-Padilla, "A literature review on the vehicle routing problem with multiple depots," Comput. Ind. Eng., vol. 79, pp. 115–129, Jan. 2015.
22. Zhang X, Tian Y, Jin Y. A knee point-driven evolutionary algorithm for many-objective optimization[J]. IEEE Transactions on Evolutionary Computation, 2014, 19(6): 761-776.
23. Konstantinidis A, Pericleous S, Charalambous C. Adaptive evolutionary algorithm for a multi-objective VRP[J]. International Journal on Engineering Intelligent Systems, 2014, 22(3/4).
24. M. M. Solomon, "Algorithms for the vehicle routing problem with time windows," Transportation Science, vol. 29, no. 2, pp. 156–166, 1995.

25. T.-R. Lee and J.-H. Ueng, “A study of vehicle routing problems with load-balancing,” International Journal of Physical Distribution & Logistics Management, vol. 29, no. 10, pp. 646–657, 1999.
26. B. Ombuki, B. Ross, F. Hanshar, Multi-objective genetic algorithms for vehicle routing problem with time windows[J]. Applied Intelligence, 2006, 24(1): 17-30.
27. S. R. Balseiro, I. Loiseau, J. Ramonet, An ant colony algorithm hybridized with insertion heuristics for the time dependent vehicle routing problem with time windows. Computers & Operations Research, 38(6):954–966, 2011.
28. J. Luo, X. Li, M.R. Chen, et al, A novel hybrid shuffled frog leaping algorithm for vehicle routing problem with time windows. Information Sciences, 316:266–292, 2015.
29. L. Hong. An improved lns algorithm for real-time vehicle routing problem with time windows. Computers & Operations Research, 39(2):151–163, 2012.
30. Z. Ursani, D. Essam, D. Cornforth, and R. Stocker. Localized genetic algorithm for vehicle routing problem with time windows. Applied Soft Computing, 11(8):5375–5390, 2011.
31. H.Z. Zhang, Q.W. Zhang, M.Liang, et al, A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows. Information Sciences,2019.
32. B. Yu, Z. Yang, B. Yao, A hybrid algorithm for vehicle routing problem with time windows[J]. Expert Systems with Applications, 2011, 38(1): 435-441.

Theory and Application of Three-dimensional Analysis about Propagation Data

Jianchi Sun[1], Shuo Liu[1*], Kang Zhou[1], Wei Cen[3], Qiyao Huang[2], and Wanying Liang[1]

[1] School of Math and Computer, Wuhan Polytechnic University, Wuhan 430023, Hubei, China 2233417652@qq.com
[2] School of Economics and Management, Wuhan Polytechnic University, Wuhan 430023, Hubei, China
[3] School of Electrical and Electronic Engineering, Wuhan Polytechnic University, Wuhan 430023, Hubei, China

Abstract. Most of the traditional grey system Verhulst models can only consider the propagation characteristics of time, but it can not consider the influence of spatial and other social factors. In order to solve this problem, a mathematical model of three-dimensional about data analysis is proposed in this paper. Firstly, the Verhulst model in grey system is established in each subspace, and the solution of general Verhulst model is obtained to solve the problem of prediction in time dimension. Secondly, in the spatial dimension, considering the mutual effect among subspaces, the spatial flow factor is added to the original Verhulst model to solve the propagation problem. Finally, in the dimension of social factors, it is necessary to solve problems led by the influence of many social factors on the response. In order to eliminate the natural factors of growth and highlight the changes of social growth factors to the response, this paper proposes a trend vector model. Each index with maximum and minimum subspace correlation is fitted with the two-dimensional data analysis model by the least square method, and the action factors of social factors are obtained, thus improving the three-dimensional data analysis of the spread data. In this paper, the eight-year drug use in five states of the United States and the seven-year index value of 149 social factors provided by the United States Bureau of Information Statistics are used as examples for simulation experiments. The accuracy of the model calculated by considering the spatial dimension is 10 times higher than that of the traditional grey system. Having taken the influence of social factors into consideration, the correlation of indicators and factors is believed to provide the direction for the government to make the policy to prevent the drug crisis. The model can be widely used in the field of big data analysis, which is beneficial to the prediction of data and the mining of data information, and provides help for human social life.

Keywords: Verhulst model· Multidimensional analysis · Trend vector · Least square method · Drug spread.

1 Introduction

In recent years, the problem of data dissemination has attracted the attention of social and scientific researchers, especially with the advent of the information age and the rapid development of computer science, the information presents explosive growth. Data analysis has become a hot topic at present. Two main purposes of data analysis are used to predict the future observations as accurately as possible, and the other is to scientifically understand the relationship between features (various information) and responses (research objects) [1].Under the current social background, the problem of data transmission is everywhere. At the beginning of this century, the most serious SARS (severe Acute Respiratory Syndrome) incident broke out in Shun De, Guangdong Province, China, and soon spread to the world. The SARS epidemic was gradually eliminated only in mid-2003. This is an unprecedented global epidemic of infectious diseases, causing social panic and leading to the death of a large area of patients, including medical personnel. It has caused great disaster to the world health and economy. With the continuous progress of medicine, SARS is basically conquered under the current medical conditions, but it is easy to spread such as epidemic cold, AIDS and so on. Infectious diseases have also been lurking around us, and population movements and controls, energy projections, and so on are all related data dissemination problems, especially the recent outbreak of classical swine fever in Africa and the drug crisis faced by countries around the world. This series of problems are closely related to nature and social sciences. The in-depth study of this kind of problem can be helpful to the development of the world society by making suggestions for the relevant departments and enterprises of the government.

At present, the common data analysis methods use mathematical probability statistics, differential equation and other related knowledge to establish mathematical models for analysis. Artificial intelligence and BP neural network [2] are also used to analyze the data in the field of machine learning. The same data source chooses different data analysis methods to obtain data analysis results, which may be completely different. The establishment of related mathematical models such as artificial intelligence requires a large amount of data to be trained, and it is not suitable for the method with fewer samples. The classical prediction data analysis methods include time series [3], grayscale prediction and fitting regression model, and so on. These models and methods play an important role in these models and methods. Important role, to provide guidance for the development of social-related industries [4].

Traditional data analysis models and methods have made remarkable achievements in the field of social science, establishing mathematical models of time series between the power sector and users to predict the unit price of energy for the next day. Provides vital information for producers and consumers [5] [6]. The discrete grey increment model and the new initial grey increment model are established to forecast the population in the future [7]. Especially in recent years, transmission of the virus seriously affects the quality of life of the people, through the establishment of differential equation model and parameter inver-

sion [8] method to predict the trend of the virus, and health sector advice. But traditional data of the analytical model only discusses the impact of time characteristics on its response without considering other characteristic factors, such as the spread of SARS virus and the spread of drugs, which are more affected by regional and macro factors. If only time characteristics are taken into account, their propagation characteristics cannot be fully described. Therefore, the traditional data analysis model cannot meet the needs of social science for the ability of multi-angle information mining. The main purpose of this paper is from three dimensions, that is, time, region, and so on. The social factors improved the traditional gray-scale prediction and put forward the three-dimensional analysis theory of the dissemination data. Finally, the application of the drug abuse problem recently faced by the international community as an example was carried out. The accuracy and practicability of the theory are verified.

2 The raising of the Problem of Propagation Data and the fundamental thoughts of the Model

The problem of the propagation of data is defined as the response value of each point in time in a certain area under other effects. This problem has a high degree of abstraction and a wide range of applications. For a non-negative data set D as the data source for the response of the study object, the set describes the short-term response values of different places and years, and the regions and years are uniform and adjacent, where $d_s^t \in D$, that represents the response value of the s region (subspace) in year t ,$t_0 \leq t \leq t_1$, $1 \leq s \leq m$.For a given set of social factors, L, where $l_{sk}^t \in L$, represents the value of the k-type characteristic index of the s region in the t year, $t_0 \leq t \leq t_1$, $1 \leq s \leq m$, $1 \leq k \leq K$. By establishing mathematical models to describe the diffusion patterns and characteristics between each other.

This problem is a typical problem of data prediction and data mining, which gives a large amount of information and abstract problems. It is difficult to break through the regional and social factors by using the traditional prediction model. This paper first carries on the simple time dimension simulation experiment through the traditional grayscale prediction model, through the simulation experiment result obtains the propagation law of the time dimension of a certain area, applies this law to consider the space dimension influence again. Optimize the original time dimension model. In the set of social factors, because there are many social factors given, it is not practical to consider the influence of each factor on its propagation one by one. In this paper, a trend vector model is established to extract the factors. In order to obtain the final three-dimensional data analysis law, the two-dimensional propagation model is further optimized for the social factors which have a great influence on the law of communication.

The first step is the gray-scale prediction of the time dimension of the model. In this paper, the Verhulst model in the grey system is established by using the data in the D set, and the rule of propagation in the time dimension is obtained [9].

The second step, the influence of spatial mobility on the original model. In the study of the response of one subspace and the spatial mobility factor, the average quantity of other subspace responses is calculated, and the spatial mobility factor is obtained by using the least square method into the time dimension model.

The third step is the influence of social factors on the two-dimensional analytical model. Since there may be a large number of social factors in the L-set, it is not realistic to take each factor into account, so first of all, it is necessary to streamline the social factors. The trend vector model is established for all the factors in the L-set and the predicted results of the two-dimensional model, and the most representative social two factors are obtained by the correlation analysis, and the factors are substituted into the two-dimensional analysis model. Then the results of three-dimensional data analysis are obtained.

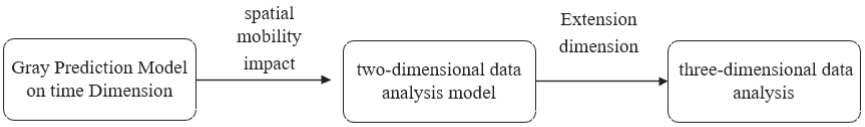


Fig. 1. Solution analysis.

3 Mathematical Model of three-dimensional data Analysis

3.1 Mathematical Model of time Dimension Analysis

When studying abstract systems such as social system, economic system and so on, random interference (noise) can be often encountered. In this paper, the number generation method is used to mine and find the regularity of the number by processing the data in the sequence to generate a new sequence of numbers. Common methods of generating numbers have cumulative generation [10](AGO). For non-negative sequences, cumulative generation can transform any random irregularity sequence into a non-decreasing, increasing sequence.

Assume that the original sequence is listed as

$$X = (x(1), x(2), \ldots, x(n)) \tag{1}$$

where $x\,(k) \geq 0$, we call it a non-negative sequence.

The cumulative generation has r accumulative generation, which is recorded as r-AGO, and its relationship is as follows.

$$x^{(r)}(k) = \sum_{i=1}^{k} x^{(r-1)}(i) \tag{2}$$

where $k = 1, 2, 3....n$, then the relation of the recursive primordial sequence is

$$x^{(r-1)}(k) = x^{(r)}(k) - x^{(r)}(k-1) \tag{3}$$

In grey system prediction, the influence of background value should be considered. The background value is usually generated by adjacent value, and its relation is as follows

$$z(k) = \alpha x(k-1) + (1-\alpha)x(k) \tag{4}$$

Grey model is a kind of differential equation model which is established by making use of discrete series to become more regular generating number. The common grey forecasting models are GM model and Verhulst model [11]. And Verhulst model is a generalization of the former. The Verhulst model is more suitable for the non-monotone oscillatory evolution sequence and the sequence similar to the S-type curves, and is more conducive to the study and description of its variation process [12].

The derivative of its $x^{(1)}$ is replaced by the difference equation[13]

$$d(k) = x^{(0)}(k) = x^{(1)}(k) - x^{(1)}(k-1) \tag{5}$$

Set the background value of grayscale as the formula (4), the difference equation model is defined as

$$d(k) + az^{(1)}(k) = b\left(z^{(1)}(k)\right)^2 \tag{6}$$

The discrete difference equation is continuous and its corresponding whitening equation is

$$\frac{dx^{(1)}}{dt} + ax^{(1)} = b\left(x^{(1)}\right)^2 \tag{7}$$

where $x^{(0)}(k)$ is grey derivative, a is developmental quotient, $z^{(1)}(k)$ is albino background value, a is ash action.

By using the Verhulst model, we can get the time-dependent response characteristics of the response in a subspace. Might as well set up $f(t)$ is the response function relation with time is calculated for the Verhulst model.

$$\begin{cases} x'(k) = \sum_{j=1}^{k} x(j), k = 1, 2, \ldots n \\ z'(k) = \alpha x'(k) + (1-\alpha)x'(k-1), k = 2, 3, \ldots n \\ x(k) + az'(k) = b\left(z'(k)\right)^2, k = 2, 3, \ldots n \\ x \geq 0, x \in X \end{cases} \tag{8}$$

where α is adjacent value generating coefficient in a given interval value$(0, 1)$. Usually α takes 0.5, that is, the variables we solve.

3.2 A Mathematical Model for Multidimensional Analysis

After the mathematical model of time dimension analysis has been established, the prediction sequence of time dimension can be set up as $f_1(t)$. Now considering the influence of its spatial propagation characteristics on the subspace response, let $g(t)$ function is the average response in other subspaces, as shown below

$$g(t) = \frac{1}{m-1} \sum_{s \in S} d_s^t \tag{9}$$

where S represents a collection that removes the subspace, d_s^t represents at t times the response of the s region, m represents the number of subspaces.

A two-dimensional data analysis and prediction model considering spatial propagation characteristics is established.λ is a spatial flow factor. Where F_2 is the prediction sequence of the two-dimensional analysis model,X is the actual response sequence.

$$\begin{aligned} &\min \sum (F_2 - X)^2 \\ &\begin{cases} F_1 = \left[f_1(1)\ f_1(2)\ ...\ f_1(n-1)\ f_1(n) \right]^T \\ G = \left[g(1)\ g(2)\ ...\ g(n-1)\ g(n) \right]^T \\ X = \left[x(1)\ x(2)\ ...\ x(n-1)\ x(n) \right]^T \\ F_2 = F_1 + \lambda G \end{cases} \end{aligned} \tag{10}$$

In order to obtain the relation between response and L set finding, we need to find some relationship between response and the quantity of each index. In order to describe this relationship, we put forward the concept of trend vector. Let F_2 be the predicted sequence of two-dimensional data analysis with geographical factors, and X be the real value sequence.

$$dert = X - F_2 \tag{11}$$

where, sequence *dert* indicates that the effect of promoting or suppressing the sequence change due to external factors, that is, the change trend is understood to be F_2 under the condition that there are no external environmental factors. It is because of the influence of a certain factor that this trend is caused the influence of *dert*, and turned *dert* into an X.

Let the order of an external factor over time be listed as

$$E_i = (e_i(1), e_i(2), ..., e_i(n)). \tag{12}$$

then

$$\begin{cases} E_{i\Delta}(1) = 0 \\ E_{i\Delta}(k) = e_i(k) - e_i(k-1) k = 2, 3, ..n \end{cases} \tag{13}$$

where $E_{i\Delta}$ is called factor change sequence.

For ease of comparison $E_{i\Delta}$ and *dert*, the relationship between the $E_{i\Delta}$ and *dert* needs to be standardized, and compared with *dert* similarity can tell the relationship between the two sequences.

The common similarity is distance similarity and cosine similarity, etc. In this paper, cosine similarity [14] is used to measure the similarity. In order to ensure dimensionless difference, two vectors substituted into cosine similarity formula are normalized. The similarity of two cosine is extended from cosine theorem to high dimensional space. The closer the cosine value is to 1, the closer the angle is to 0 degrees, that is, the more similar the two vectors are. When the cosine value is closer to-1, the angle is closer to 180 degrees, that is, the two vectors are inversely correlated, and when the cosine value is closer to 0, that is, the two vectors are vertical, indicating no similarity. The formula is as follows

$$sim = \frac{\boldsymbol{a} * \boldsymbol{b}}{|\boldsymbol{a}| * |\boldsymbol{b}|} \tag{14}$$

Based on the mathematical model of two-dimensional data analysis with spatial flow factor, let L be the set of social factors. Let the index i corresponding to the matrix l_i be. And the acting factors are p_i. In order to ensure that the initial value prediction is the same as the actual one, it is necessary to add a constant.

Through the above analysis, a three-dimensional data analysis model is established, in which, F_3 is the prediction sequence of a three-dimensional analytical model with a social factor, X for the actual response sequence, K for all social factors to be considered. $\overrightarrow{dert}, \overrightarrow{E_{i\Delta}}$ is the vector that in order to standardize each other. $E_{i\Delta}$ is a series of changes known as indicator number i. F_2 is the prediction sequence of the two-dimensional analysis model with spatial flow factor.

$$\begin{aligned}
&\min \sum \left(F_3 - X\right)^2 \\
&\begin{cases}
dert = X - F_2 \\
E_{i\Delta}\left(1\right) = 0 \\
E_{i\Delta}\left(k\right) = e_i\left(k\right) - e_i\left(k-1\right) k = 2, 3, ...n. \\
sim_i = \dfrac{\overrightarrow{dert} * \overrightarrow{E_{i\Delta}}}{\left|\overrightarrow{dert}\right| * \left|\overrightarrow{E_{i\Delta}}\right|} \\
F_3 = F_2 + p_i l_i + p_j l_j + b, i \in \max\left\{sim\right\} j \in \min\left\{sim\right\} \\
F_3(1) = X(1) \\
1 \le i \le K
\end{cases}
\end{aligned} \tag{15}$$

4 Three-dimensional data analysis model solving algorithm

For the mathematical model of the time dimension (8), the first-order linear partial differential equation is used for the calculation. Let Y be a partial data matrix and B be a parameter matrix. β be the parameter matrices, these are as follows:

$$Y = (x(2), x(3), \ldots, x(n))^T \tag{16}$$

$$B = \begin{bmatrix} z'(2)^2 & z'(3)^2 \ldots z'(n)^2 \\ -z'(2) & -z'(3) \ldots - z'(n) \end{bmatrix}^T \tag{17}$$

$$\beta = (a, b)^T \tag{18}$$

Then the mathematical model representing the time dimension is the matrix equation:

$$Y = B\beta \tag{19}$$

It can be obtained by least squares method:

$$\begin{cases} \beta = (a, b)^T = \left(B^T B\right)^{-1} B^T Y \\ x_0' = x(1) \end{cases} \tag{20}$$

By solving the first-order nonlinear differential equation solution, we can get:

$$x'(t) = \frac{ax_0'}{bx_0' + (a - bx_0')\, e^{at}} \tag{21}$$

Then the discrete time prediction is

$$x'(k+1) = \frac{ax_0'}{bx_0' + (a - bx_0')\, e^{ak}} \tag{22}$$

Restore sequence is

$$f_1(k+1) = x(k+1) - x(k) \tag{23}$$

where $f_1\,(k+1)$ is the predicted value of the model, and $F_1 = [f_1\,(1)\,, f_1\,(2)\,, ..., f_1\,(n)]$ is the prediction vector.

For the two-dimensional data analysis model (10) with spatial mobility factor, the least squares method can be used to obtain:

$$\lambda = \left(G^T G\right)^{-1} G^T (X - F) \tag{24}$$

In the three-dimensional data analysis model, the least squares method is also used to obtain values in the model (14).

$$[p_i, p_j] = \left([l_i, l_j]^T * [l_i, l_j]\right)^{-1} * [l_i, l_j] * (X - F_2) \tag{25}$$

In order to ensure that the predicted value of the first year is the same as the actual value, the constant term is solved by the undetermined coefficient.

$$b = x(1) - (F_2 + p_i l_i + p_j l_j) \tag{26}$$

In summary, the final data analysis equation is

$$\begin{cases} F_3(k+1) = x'(k+1) - x'(k) + \lambda g(k+1) + p l_i(k+1) + p_j l_j(k+1) + b \\ x(k+1) = \frac{\alpha x_0'}{b x_0' + \left(a - b x_0'\right) e^{\alpha^k}} \\ g(t) = \frac{1}{m-1} \sum_{i=5}^{t} d_s^t \\ x_0' = x(1) \end{cases} \tag{27}$$

of which,S represents a collection that removes the location, d_s^t indicates the response at the moment t and the position S , m indicates the number of subspaces.

5 Simulation application of three-dimensional analysis model

5.1 Simulation experiment of Gray Prediction Model on time Dimension

According to the theory of the three-dimensional analysis model of transmission data, this paper takes drug abuse data analysis as an example to apply the model. The simulation object data source is DEA/National Forensic Laboratory Information System (NFLIS), which provides annual reports of drug testing results and related data of drug cases analyzed by five states (Ohio, Kentucky, West Virginia, Virginia and Pennsylvania) and their counties, local forensic laboratories from 2010 to 2017. Secondly, the data sources of social factors are the social and economic factors of five states and counties from 2010 to 2016, such as education level, age ratio, family structure and other related characteristics. These simulation data sources can be downloaded on the 2019 MCM/ICM Official Website. Drug use is only eight years old in time series, so it is inappropriate to use AI correlation analysis methods. Traditional time series models are not enough to calculate relatively good results because of few years. At the same time, the data of a certain type of drug use are often interrelated with other factors (such as the use of drugs in the previous year, the amount of other subspace drugs used, etc.). The relationship between them can only be predicted by a small number of sample data, which is consistent with the characteristics of grey system [11]. The traditional gray prediction model can only predict and analyze in the time dimension, but not in the multi-dimensional feature and response information mining, which will lead to inadequate use of some information. The main purpose of simulation application is to improve the traditional gray prediction from three dimensions, namely time, region and social factors, to reflect the application of the three-dimensional analysis model of communication data in social science.

According to the model theory of the above analysis and the introduction of simulation examples, the performance of the simulation computer includes processor of AMD A10-9600P RADEON R5,10COMPUTE CORES 4C+6G 2.8GHz and memory of 4.00GB, 64 Bit Operating System of Windows 10. And programming with MATLAB 2016a, Table 1 shows the total drug use in each state after data preprocessing.

Table 1. Total drug use in the counted states.

	2010	2011	2012	2013	2014	2015	2016	2017
VA	8685	6749	7831	11675	9037	8810	10195	10448
OH	19707	20330	23145	26846	30860	37127	42470	46104
KY	10453	10289	10722	11148	11081	9865	9093	9394
WV	2890	3271	3376	4046	3280	2571	2548	1614
PA	19814	19987	19959	20409	24904	25651	26164	27894

According to the mathematical model theory of time dimension analysis, the total amount of drug use in each state is analyzed. The predicted value of time dimension analysis in each state can be obtained by solving the first-order linear differential equation and least square method. According to the general equation of Verhulst model, the parameters of each state are shown in Table 2.

$$\begin{cases} x'(k+1) = \frac{ax_0'}{bx_0' + (a - bx_0')e^{ak}} \\ f_1(k+1) = x'(k+1) - x'(k) \end{cases} \tag{28}$$

Table 2. General Solution Parameter of One-Dimensional Verhulst Model for Each State.

	a	$b\left(10^{-4}\right)$	$x_0^{'}$
VA	-0.4676	-0.0493	8685
OH	-0.4331	-0.0106	19707
KY	-0.5143	-0.0540	10453
WV	-0.6279	-0.2501	2890
PA	-0.4446	-0.0174	19814

Table 3. Predicted value of drug use in time dimension in VA.

	2010	2011	2012	2013	2014	2015	2016	2017
VA	8685	6749	7831	11675	9037	8810	10195	10448
Prediction	8685	4460	6236	8197	9941	10945	10838	9663

Table 4. Error parameter for each state in time dimension.

	VA	OH	KY	WV	PA
relative errorε	0.0625	0.1220	0.0540	0.0408	0.0812

In order to make the errors more convincing and as different as possible, the relative errors mentioned above represent the integral multiple relationship between absolute errors and real values, that is, $\varepsilon Y \approx |Y - F|$, making the absolute error vector as real error vector ε times as possible, the smaller$\varepsilon$is, the smaller the overall error is.

5.2 Simulation experiment of Two-dimensional Data Analysis Model with spatial mobility factor

By providing the actual geographic location of five states according to Google Earth software, as shown in Figure 2. It can be seen that the geographical re-

lationship between the States is contiguous, and the geographical region has a certain role in the impact of drug volume. Ohio, Pennsylvania and Virginia are closely related to each other. They are influenced by diffusion. So it is necessary to consider spatial mobility factors.

Fig. 2. Geographical location of five states.

According to the model theory of two-dimensional data analysis with spatial mobility factor, the predicted value of the total drug use and time dimension in each state and the average drug use in other states are analyzed. The predicted value of two-dimensional data analysis with geographical factors in each state and the size of geographical factors can be obtained by (10) model and (23) formula calculation.

Table 5. Predictive values of drug use in VA on a two-dimensional model with geographic factors.

	2010	2011	2012	2013	2014	2015	2016	2017
VA	8685	6749	7831	11675	9037	8810	10195	10448
Prediction	9003	4784	6580	8572	10362	11397	11320	10173

From the calculation results, we can see that the magnitude of spatial mobility factor can be regarded as the sensitivity of drug use and regional mobility in the region. The factors are positive, indicating that regional mobility promotes

Table 6. Relevant Calculating Parameters of States on a two-dimensional model with geographical factors.

	VA	OH	KY	WV	PA
Geographic factor λ	0.0240	0.3944	0.0161	0.0014	0.1294
Relative error ε	0.0191	0.0135	0.0284	0.0326	0.0064

drug use, that is, drug use in other states leads to increased use in the region. OH is the most sensitive to regional liquidity, and WV is hardly affected by regional liquidity. OH can devote more energy and financial resources to border control ports to prevent drug invasion. Within the scope of policy control, strict drug inflow and outflow management at county or state borders can reduce drug use in the state. For the WV, space interaction is very small, the border control cannot have to invest a lot. The relative error is significantly lower than that of the time dimension analysis model, and the fitting effect is better.

5.3 Simulation experiment of three-dimensional data Analysis Model with social action factors

According to the model theory of three-dimensional data analysis, there are 149 social factors in all public years with the factor data source, and each social factor gives four different description forms, in order to facilitate the establishment and normalization of the model. Take the first description form as the description value. According to the trend vector theory, the similarity between each social factor and the original sequence change trend is calculated for each state. The similarity can be used to understand whether the factor promotes or inhibits or does not work.

Because the relevant indicators are still considered much, considering the influence of major social factors in each state, this paper selects the social factors with the most positive and negative correlations in each state as the fitting parameters to be incorporated into the two-dimensional data analysis model equation. The main social factors associated with each state are shown in Table 7.

Table 7. The correlation of the main social factors of positive and negative correlation in each state.

	Indicator number	Correlation	Indicator number	Correlation
VA	115	0.96	116	-0.83
OH	20	0.91	14	-0.93
KY	69	0.89	105	-0.92
WV	128	0.83	6	-0.98
PA	128	0.87	14	-0.90

For VA, the value vector corresponding to the 132th indicator is l_{132}, the corresponding vector of the 6th indicator is l_6, and the action factors are respectively p_{132}, p_6, where F_2 is the two-dimensional model prediction sequence with the geographical flow factor, F_3 is the three-dimensional data analysis model prediction sequence. And then

$$F_3 = F_2 + p_{132}l_{132} + p_6 l_6 + b \tag{29}$$

We can get

$$\begin{cases} [p_{132}, p_6] = [0.0374, -0.1197] \\ b = -1776.3 \end{cases} \tag{30}$$

The final three-dimensional data analysis model based on the obtained action factor is

$$\begin{cases} x_0' = 8685 \\ x'(k+1) = \frac{ax_0'}{bx_0' + (a - bx_0')e^{ak}} \\ D_1 = x'(k+1) - x'(k) \\ D_2 = 0.024g(k+1) \\ D_3 = 0.0374l_{132}(k+1) - 0.1197l_6(k+1) - 1776.3 \\ F_3(k+1) = D_1 + D_2 + D_3 \end{cases} \tag{31}$$

Similarly, the factors and error parameters of the main social factors in other states are calculated as Table 8 and Table 9.

Table 8. The indicator factors of the main social factors related to positive and negative correlation in each state.

	Indicator number	Indicator factor	Indicator number	Indicator factor	Constant
VA	132	0.7254	6	-0.1934	-1776.3
OH	20	0.0099	14	-0.0274	-6668.7
KY	3	0.0464	65	-0.0870	-2737.4
WV	59	0.0248	91	-0.2239	-600.4
PA	128	0.0374	15	-0.1197	-5784.2

Table 9. Relative error parameters for each state on the three-dimensional model.

	VA	OH	KY	WV	PA
Relative error ε	0.2038	0.2138	0.2624	0.1924	0.2535

Through the calculation of the above mathematical model, it is found that the positive correlation index are the positive and negative correlation factors are negative, thus verifying the accuracy of the model. The relative error of the two-dimensional data analysis is better than the time dimension relative error, but the relative error is relatively large after considering the three-dimensional

data analysis, mainly because the social environment is considered only when the two kinds of the most valued society are considered. There are still links between factors and social factors, and they all have more or less influence on the amount of drugs used. If only two extreme correlations are considered, the error will be larger.

In order to prove that the number of the social factors into the model can affect the error of the three-dimensional analysis model. In the case of social factors with maximum positive correlation and maximum negative correlation in the model, the sub-positive correlation factors and the sub-negative correlation factors continue to be considered. In formula (30), the predicted value after fitting is also obtained by the least square method, and the error is obtained by using the error formula, as shown in Table 10.

Table 10. The relative error parameters of each state in the three-dimensional model considering the four factors.

	VA	OH	KY	WV	PA
Relative error ε	0.0452	0.0540	0.0861	0.1270	0.0567

Compared with Table 9, the relative errors of all states have decreased after considering four factors. Therefore, it can be shown that the number of social factors considered in the model will affect the accuracy of the model. And when the number is large enough, the accuracy is high enough. But the more the number considered, the greater the cost of computing time. And in the real society, the most important social factors and the larger correlation factors are often considered, so that the calculation time is saved, and also provide specific guidance for the correct decision-making of the government, greater investment can be placed on some of the most critical factors.

In order to obtain more possible correlation factors for qualitative analysis, this paper establishes the advantage of the trend vector model, and averages the 149 social factors in all states, setting the positive correlation threshold to 0.6, that is, the average similarity is greater than 0.6 for positive correlation. The public social factor has a negative correlation threshold of -0.65 and an average similarity of less than -0.65 as a negative correlation public social factor. The public social factors of all regions are obtained by threshold classification. The positive and negative similarities of public social factors are shown in Table 11. and Table 12.

It can be found that each state is affected by many social factors. It is indeed inaccurate to consider the social factors of the two most valued cases. However, the degree of influence can be qualitatively compared by the size of the factors. Finding certain social factors from the size of relevance can also provide suggestions for relevant departments and make targeted strategies for the spread of drugs.

Table 11. Public social factors with positive similarities.

Indicator number	59	69	134
VA Similarity	0.798	0.405	0.304
OH Similarity	0.809	0.851	0.759
KY Similarity	0.606	0.702	0.752
WV Similarity	0.802	0.754	0.605
PA Similarity	0.574	0.927	0.682

Table 12. Public social factors with negative similarities.

Indicator number	6	11	91	94
VA Similarity	-0.875	-0.511	-0.37	-0.481
OH Similarity	-0.877	-0.932	-0.65	-0.652
KY Similarity	-0.659	-0.783	-0.66	-0.723
WV Similarity	-0.374	-0.606	-0.79	-0.705
PA Similarity	-0.749	-0.918	-0.82	-0.814

6 Conclusion

In this paper, we propose the theory of three-dimensional analysis of propagation data, and an improved three-dimensional data analyses model based on grayscale prediction is established. The simulation experiment is carried out by using the data of drug spread in the United States. The prediction equation is obtained by least squares method. The general solution and the spatial flow factor of each region, the size of the spatial flow factor can be used as a reference for the mobility of the drug area. Comparing the residuals after two data analysis, the overall error of the two-dimensional analysis model with geographic factors is smaller. In the establishment of a three-dimensional data analysis model with social factors, this paper proposes the concept of trend vector, eliminates the natural increase of the feature response itself, and makes the change of social factors completely correspond to the changes in drug use interfered by social factors. Through the calculation of similarity, the most important social factors and corresponding impact factors affecting the increase or decrease of drug use in social factors are found, and making suggestions for government macro-control. The discussion and establishment of the three-dimensional data analysis model has improved the shortcomings of most of the gray-scale predictions, and the data analysis is reflected by multiple angles. In the later period, we mainly discuss the impact of multiple social factors on the response, so as to improve the prediction precision after considering multiple social factors.

References

1. Fan J, Han F, Liu H. Challenges of big data analysis[J]. National science review, 2014, 1(2): 293-314.

2. Yu F, Xu X. A short-term load forecasting model of natural gas based on optimized genetic algorithm and improved BP neural network[J]. Applied Energy, 2014, 134: 102-113.
3. YANG Hai-min, PAN Zhi-song, BAI Wei. Review of Time Series Prediction Methods [J]. Computer Science, 2019, 46(1):21-28.
4. Xue-Qi C , Xiao-Long J , Yuan-Zhuo W , et al. Survey on Big Data System and Analytic Technology[J]. Journal of Software, 2014,(9):1889-1908.
5. Nogales F J, Contreras J, Conejo A J, et al. Forecasting next-day electricity prices by time series models[J]. IEEE Transactions on power systems, 2002, 17(2): 342-348.
6. Alvarez F M, Troncoso A, Riquelme J C, et al. Energy time series forecasting based on pattern sequence similarity[J]. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(8): 1230-1243.
7. Ke-Pei M , Lin-Lin G , Xun-Zhen Y . PREDICTION OF CHINA'S POPULATION BASED ON TWO NEW GREY MODELS[J]. Economic Geography, 2007, 27(6): 942-945.
8. Weiguo H , Jinfeng W , Xuhua L . BACK ANALYZING PARAMETERS AND PREDICTING TREND OF SARS TRANSMISSION[J]. Advance in Earth Sciences, 2004, 19(6):925-930.
9. Lin Y H, Chiu CC, Lee P C, et al. Applying fuzzy grey modification model on inflow forecasting[J]. Engineering Applications of Artificial Intelligence, 2012, 25(4): 734-743.
10. YONG-HUANG LIN, CHIH-CHIANG CHIU, PIN-CHAN LEE,et al. Applying fuzzy grey modification model on inflow forecasting[J]. Engineering Applications of Artificial Intelligence: The International Journal of Intelligent Real-Time Automation,2012,4(4):734-743.
11. ZHANG Chuang, PENG Zhenbing, PENG Wenxiang. Application of optimized grey discrete Verhulst model in settlement prediction of foundation pit[J].Journal of Central South University Science and Technology), 2017, 48(11):3030-3036.
12. Ping-Ping X , Yao-Guo D , Tianxiang Y , et al. The Research on the Modeling Method of Background Value Optimization in Grey Verhulst Model[J]. Chinese Journal of Management Science, 2012, 20(6):154-159.
13. Yonge AN, Xueying B, Qicai W. Railway freight volume forecasting based onunbiased grey Verhulst model[J]. Journal of Railway Science & Engineering, 2016, 13(1):181-186.
14. Wang Z X , Dang Y G , Liu S F . Unbiased Grey Verhulst Model and Its Application[J]. Systems Engineering-Theory & Practice, 2009, 29(10):138-144.

A Novel Spiking Neural P Systems for Image Recognition*

Xiantai Gou, Qifen Liu*, Gexiang Zhang, Meng Hu, Pirthwineel Paul, Fang Deng, Xihai Zhang, and Zhibin Yu

School of Electrical Engineering, Southwest Jiaotong University, Chengdu, 61003
email:1622680696@qq.com

Abstract. Spiking neural P systems (SNPS), a kind of distributed parallel bio-inspired model, has been a research hotspot in the field of membrane computing. SNPS has been widely concerned by scholars due to its powerful computing capacity and brain-like information transmission schema. Nevertheless, there are quite few research results about SNPS with learning ability applied for image recognition. In this research, the routing mechanism in capsule neural network is introduced into SNPS to update the weights between synapses of spiking neurons dynamically. The learning ability of SNPS is realized by the weight update algorithm, which represents the changes in the strength of neuronal synaptic connections. Moreover, this is the first attempt to construct a universal network model of SNPS with learning ability which extracts features though the image convolution. The experimental results demonstrate that the recognition accuracy of the Mixed National Institute of Standards and Technology database, namely MNIST, reaches 95.87% and the recognition accuracy of English letters with noise and rotation reaches 98.06% in SNPS, which verify the feasibility and effectiveness of the model we constructed.

Keywords: membrane computing · spiking neural P systems · image recognition · learning ability.

1 INTRODUCTION

Membrane computing is a systematic framework inspired by cell structure and function to research computational models or algorithms, which was proposed by the academician Gheorghe Păun in 1998 and belongs to the branch of natural computing[1]. The computational model of membrane computing research is called as membrane systems or P systems. The membrane system is a kind of distributed parallel bio-inspired model. The researches have proved that the computational power of most membrane systems is equivalent to the Turing machine [2-5]. Furthermore, the application research of membrane computing involves many fields, including ecosystem modeling [6-8], image processing [9,10],

* supported by the National Natural Science Foundation of China under Grant No.61972324 and No.61702428, and the Major Science and Technology Projects in Sichuan Province under Grant No. 2018GZDZX0043.

intelligent algorithm [11, 12] and so on. At present, the application of membrane computing in the field of image processing mainly focuses on simple image processing and image analysis, but it has not been applied in image high-level processing [13]. It is still challenging for membrane computing to process image recognition [14], therefore, it is significant to pay more attention on image recognition.

The SNPS, proposed by Ionescu et al in 2006, is a type of neurological computational model under the framework of membrane computing [15]. It is derived from the biological characteristics of neurons in the biological nervous system that transmit pulses and exchange information through synapses. The outstanding feature of SNPS is that information is encoded by time and transmitted by the pulses between neurons, which simulates the working mechanism of information interaction in biological neural network better. From the perspective of model characteristics, SNPS belongs to the third generation neural network models [16,17]. The third generation neural network represented by spiking neural network processes information through pulse sequences, which is the research frontier and hotspot in the field of current neural network [18-20]. In the past decade, SNPS has been widely concerned by scholars in the field of membrane computing due to its good mechanism of restoring biological information interaction and high computational parallelism, therefore, abundant research results in both theory and application has been obtained. Inspired by biological characteristics and mathematical computation models in the biological nervous system, many SNPS have emerged, such as SNPS with anti-spikes [21,22], SNPS with astrocyte [23], reversible SNPS [24], SNPS with structural plasticity [25], SNPS with rules on synapses [26-28], fuzzy SNPS [29], and so on. SNPS is applied to the practical application based on huge amounts of theoretical researches, and has achieved many research results, such as skeletonizing images [30], combinatorial optimization [31], fault diagnosis [32-36], etc. Illuminated by learning mechanisms in biological neural networks and artificial neural networks, Gutiérrez established a SNPS model with Hebbian learning [37]. However, there are quite few papers about SNPS with learning ability for image recognition. In 2019, A class of specific SNPS with simple Hebbian learning function is constructed to recognize English letters in a template matching manner that calculates the variance of the letter to be recognized and the standard letter output vector, which proved the potential of SNPS in pattern recognition. A high recognition accuracy is achieved in the absence of noise, but the recognition effect is poor in high noise [38]. In addition, the input data is only represented by a 0/1 character matrix of size 5*7 and the model is relatively simple, which is difficult to handle real-life applications.

SNPS for image recognition is still an open research problem, which has attracted the attention of a large number of scholars. In particular, there exist many application practices for image recognition and reach a high level of recognition in traditional artificial neural networks, while SNPS in the framework of membrane computing has hardly applied to research on pattern recognition, so it has fatal research significance to apply SNPS to image recognition. Because

there exist few research results about image recognition in SNPS at present, there is no suitable encoding method to encode the image into pulse numbers while retaining the pixel position information, the application of the simple Hebbian learning function is not sufficient to express the changes in the strength of neuronal synaptic connections, and there is no universal neuron connection topology in SNPS.

Aiming at the above problems, the time-coded image pixel data is converted into pulse numbers as the input of SNPS and the routing mechanism in capsule neural network [39] is introduced firstly. Then, a general network model of SNPS with learning ability is constructed. In order to verify the feasibility and effectiveness of the model, it is used to deal with the classical handwritten digit recognition and the English letters recognition with noise and rotation in this research.

2 SNPS WITH LEARNING ABILITY

Learning function is extremely important for pattern recognition in artificial neural networks. Therefore, it is the key to construct SNPS with learning ability for image recognition.

2.1 SYSTEM DEFINITION

The SNPS with learning ability of degree m ($m \geq 1$) is a construct of the form:

$$\Pi = (A, \{\sigma_1, \cdots, \sigma_m\}, syn, f_{pre}, f_{\text{learn}}, in, \text{ out }) \tag{1}$$

where:

(1) $A = \{a\}$ is an alphabet with single symbol, which denotes a pulse.

(2) $\{\sigma_1, \cdots, \sigma_{\text{m}}\}$ represents the neuron collection in SNPS. $\sigma_i = (n_i, R_i), 1 \leq i \leq m$ is a certain neuron in the system, where $n_i \geq 0$ is the pulse numbers contained in the neuron σ_i at the beginning of the computation, and R_i represents a finite set of all the rules in the neuron σ_i .There are two forms of rules, including the firing rule and the forgetting rule in SNPS.

Firing rule is shaped like $E/a^c \rightarrow a; d$, where E is a regular expression on the character a, $c \geq 1$ and $d \geq 0$ are integers. The forgetting rule is shaped like $a^s \rightarrow \lambda$. $a^s \notin L(E) \quad (s \geq 1)$ is required for any firing rule $E/a^c \rightarrow a; d$, where $L(E)$ is the regular language defined by E. Each neuron can contain one or more rules, but the firing rule and the forgetting rule cannot be triggered at the same time in a neuron. In a plurality of rules, a selection mechanism can be applied to trigger one of the rules to achieve an optimal level of system output.

(3) $Syn \subseteq \{1, 2, \cdots, m\} \times \{1, 2, \cdots, m\}$ is a directed graph of synaptic connections between neurons, with $(i, i) \notin syn$ for each $1 \leq i \leq m$.

(4) In, out $\in \{1, 2, \cdots, m\}$ represent the input and output neurons of system Π , respectively.

(5) F_{pre} is the data preprocessing function of the system. In this paper, the data preprocessing function of SNPS for image recognition contains convolution and pooling operations, which could extract image features. Different preprocessing functions can be adopted towards different applications.

(6) F_{learn} is a systematic learning function that characterizes the strength of neuronal synaptic connections by synaptic weights updating in training. The learning function adopted in this research is shown in Section 2.2.

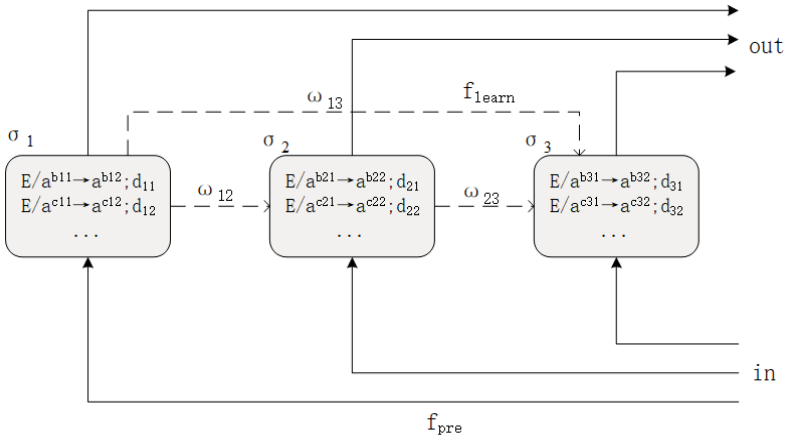


Fig. 1. An example of SNPS constitution.

2.2 LEARNING FUNCTION

In the SNPS with learning ability designed in this research, the routing mechanism, which core idea is the error back propagation algorithm, is used to update the weight of neuronal synaptic connections dynamically to imitate the information layered communication between the visual perception and comprehension neurons in the human brain [39].

Assume that the input of the nth layer neuron σ_i is u_i , and the input of the n+1th layer neuron σ_j is v_j in the SNPS. The routing mechanism updates the synaptic weights dynamically as follows [39]:

(1) Initializing the logarithmic prior probability $b_{ij} = 0$ of all neurons between σ_i in the nth layer and σ_j in the n+1th;

(2) Calculating the coupling coefficient c_i of the neuron σ_i in nth layer;

$$c_i = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})} \tag{2}$$

(3) Calculating the intermediate variable s_j of the neuron σ_j in n+1th layer;

$$s_j = \sum_i c_{ij} u_i \tag{3}$$

(4) Calculating the input v_j of the neuron σ_j in n+1th layer;

$$v_j = \text{squash}\left(s_j\right) = \frac{\left\|s_j\right\|^2}{1+\left\|s_j\right\|^2} \cdot \frac{s_j}{\left\|s_j\right\|} \tag{4}$$

(5) Updating the logarithmic prior probability b_{ij} by the input v_j of the neuron σ_j in n+1th layer obtained by the above process;

$$b_{ij} = b_{ij} + u_i \bullet v_j \tag{5}$$

(6) The input v_j of the neuron σ_j in n+1th layer can be obtained by repeating steps (2)-(5) r times for a given iteration step number r.

The coupling coefficient c_{ij} is the synaptic weight ω_{ij} between neuron σ_i and neuron σ_j in SNPS.

3 NETWORK MODEL OF SNPS WITH LEARNING ABILITY

Based on the above theoretical research, this paper shows the encoding method converting image pixel data into pulse numbers and the topology of neuron connections in SNPS. Moreover, a universal network model of SNPS with learning ability is constructed for image recognition, referred to as SNPSNet in this paper.

3.1 IMAGE DATA ENCODING

In general, the storage method of the image is pixel data, nevertheless SNPS processes input data in the form of pulses. Therefore, we need to encode the image into discrete pulse sequences by appropriate encoding method while preserving the pixel position of the image.

Spiking neurons in SNPS transmit information in the form of pulses [15]. Studies have shown that much of the information about a stimulus signal is actually transmitted within 20ms or 50ms in the start of the neuronal response [40-43]. Hence the neuron can be idealized as: the time of the first pulse generated after receiving the stimulus contains all the information of the stimulus [43].

Based on the above research results, the image pixel data is encoded as the pulse trigger time through the time-to-first spike coding [44]. For the time coding window T, the linear time coding formula is as follows.

$$F_{rc} = T \times \left(1 - \frac{p_{rc}}{p_{\max}}\right) \tag{6}$$

Where F_{rc} is the pulse trigger time in the rth row and the cth column of the image pixel matrix, p_{rc} is the pixel value in the rth row and the cth column of the image pixel matrix, and $\mathrm{p_{max}}$ is the maximum pixel value. It is obvious that the larger the pixel value, the earlier the trigger time, the greater the effect, and the smaller the pixel value, the later the trigger time, the less the effect from the formula above.

For the time-encoded data, further processing is required to convert into the discrete pulse numbers as the input of the network model designed in this research. Taking the MNIST data set as an example, the input tensor is 28*28, and the 28*28 input neurons are necessary accordingly. Suppose that for any N consecutive pixel points of the input tensor, the time-encoded image pixel data is a finite set $T \in \{t_1, t_2, \cdots, t_n\}, \quad t \in (a, b)$. And $b-a$ is divided into m shares, then the pulse numbers are saved in λ-th input neuron σ_λ , which trigger time fall in $\left[\frac{\mathrm{b-a}}{\mathrm{m}} \times (\lambda - 1), \frac{b-a}{\mathrm{m}} \times \lambda\right] (\lambda \geq 1)$. The larger m , the more precise position information of the original pixel is saved.

It is obvious that the original pixel data is more complicated than the tensor of discrete pulse sequences. The encoded tensor is adopted as the input of the network model we constructed, which makes the computer memory consumption less and improves the computational efficiency. In addition, the time cost of encoding consumption could be negligible for large image sample training and testing.

3.2 NETWORK MODEL OF SNPS

Considering that only neurons with synaptic connections in SNPS could trigger firing rule or forgetting rule, and it could be encoded only if neurons connected in a certain pattern. Of course, the random connections cannot guarantee the reliability of network output. Therefore, a topology of SNPS is constructed in this research shown in Figure 2.

In this figure, dim1 indicates the initial state of the input neurons, dim2 indicates that there are synaptic connections between two adjacent neurons, dim3 indicates that there are synaptic connections between three adjacent neurons, dim4 indicates that there are synaptic connections between four adjacent neurons, and dim5 indicates that there are synaptic connections between five adjacent neurons. All neurons in dim1 are spiking neurons of SNPS. Each neuron could be set with one or more firing rules or forgetting rules, such as $a^\alpha \rightarrow a^\beta; d$, where $\alpha \subseteq \{1, 2, 3, \cdots, \mathrm{n}\}$ is the initial pulse numbers of the neuron, $\beta \subseteq \{0, 1, 2, 3, \cdots, m\}$ represents the pulse numbers of neurons transmitted to the next layer, and d indicates the time of the rule triggered. Of course, this connection method could retain as much position information as possible towards the input pulse numbers in the model. It is that the input tensor is (dim1), and the output tensor is (dim1, dim2, ..., dim5) in SNPS. That is equivalent to which the dimension of the input tensor is increased after applying the rules of neurons in SNPS.

The structure of the whole SNPSNet model constructed in this paper is shown in Figure 3. The main components are the convolution layer, the pooling layer,

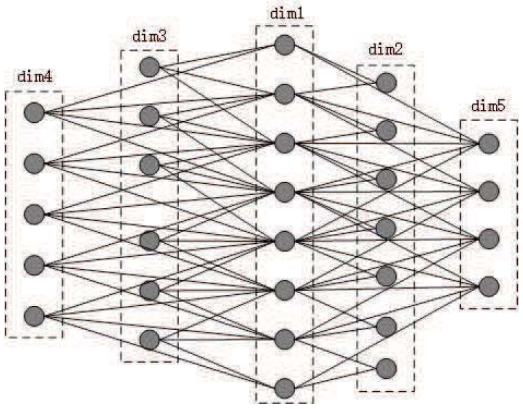


Fig. 2. Schematic diagram of neuron topology in SNPS.

the SNPS processing of the above topology connections and the decoder composed of three fully connected layers, which restore the input data and complete data classification.

The input of the SNPSNet is the pulse numbers obtained by time-coded image pixel data. First, the characteristics of the input data are extracted through the screening of each convolution kernel in the convolution layer. The specific parameters of the convolution layer are 64 convolution kernels with conv_size=3*3, and the activation function is relu. It economizes computational cost through the convolution to extract image features, so that large pixel data could not bring computer memory overflow easily during model operation. And then the data in the pooling area is compressed by the pooling layer with pooling_size=2*2. The purpose of pooling is to reduce the number of parameters. For example, if the dimension of input tensor is (20, 20, 64), the dimension of pooled output tensor is (10, 10, 64).

After that, it is the turn for the SNPS in Figure 2. The pulse information is transmitted and the synaptic weight is updated in learning process after rules triggered in spiking neurons. There are multiple rules could be selected for every neuron in SNPS. The rule selected in this paper is $a \to a$. The SNPS reacts under the global clock. The upper neurons satisfy the firing rule, consume the corresponding pulses immediately, and transmit pulses to neurons with synaptic connections in the next layer. Dynamic routing reflects the learning process of synaptic weights. SNPS is the core of the whole network model constructed in this paper and is the concentrated expression of learning capabilities. Furthermore, its input is the output tensor of the pooling layer. Toward the neuron $\sigma_{\mathbf{i}}$ in SNPS, a pulse is consumed, and $\omega_{ij} \times 1$ pulses are transmitted to the neurons $\sigma_{\mathbf{j}}$ with synaptic connections (ω_{ij} is derived from the learning function shown in Section 2.2).

Finally, the image recognition is completed in the fully connected layers to increase the receptive field. The first and the second fully connected layers are

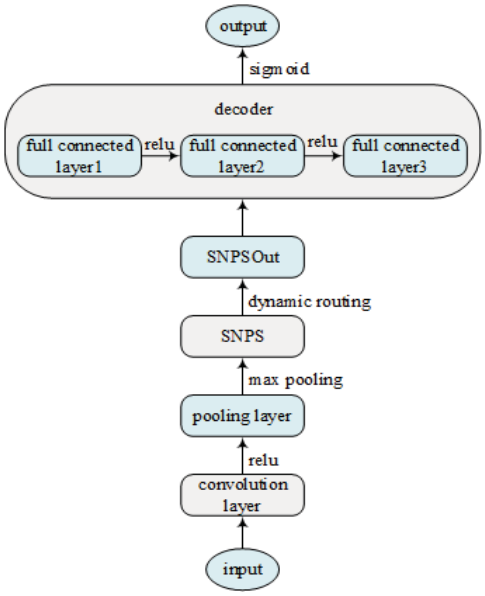


Fig. 3. Schematic diagram of SNPSNet model.

composed of 1024 and 512 neurons respectively which activation functions are relu. The number of neurons in the third fully connected layer is determined by the categories we expect about the images, and the activation function is sigmoid. The output of the model is stored in the form of a matrix, including recognition accuracy and loss values.

4 SIMULATION EXPERIMENTAL

In this paper, the feasibility and effectiveness of the SNPSNet model are verified through classical handwritten digit recognition and the English alphabet recognition with noise and rotation.

4.1 SNPSNET FOR MNIST RECOGNITION

In this research, the MNIST dataset is divided into training set and test set according to 6:1. The training set is 60,000 pictures and the test set is 10,000 pictures. And the 35 handwritten digital images selected randomly are shown in Figure 4.

First, the image pixel data is encoded into the pulse numbers through the data preprocessing method in Section 3.1, and saved to a file in mat format. For each MNIST image, 28*28 pixels, the original pixel data is encoded into the pulse trigger time after the time-to-first spike coding. Next, the time-encoded

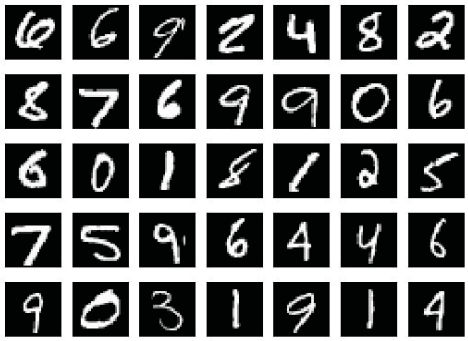


Fig. 4. Schematic diagram of MNIST handwritten digital.

data of each row in the mat are sorted and the pulse numbers are accumulated during a given time interval. In other words, the element of the rth row and the cth column towards every two-dimensional vector in the mat file indicates pulse numbers, which are triggered at time t $\in (c-1, c), (c > 0)$ in the rth row of the original pixel matrix. Then, the encoded data is fed in SNPSNet model in Section 3.2 for training, and the rule selected for each spiking neuron in SNPS is $a \to a$, with no delay. A pulse is consumed in spiking neurons that satisfy the firing rule at any moment and $\omega_{ij} \times 1$ pulses are delivered to neurons with synaptic connections in the next layer (ω_{ij} is derived from the learning function shown in Section 2.2). The size of input image is 28*28 in this model; the batch size is 60; the RMSProp is selected as the optimizer, which could be able to adjust the learning rate automatically [45]. Thus, the learning rate is no longer need to perform any attenuation operations once the initial learning rate is given. The initial learning rate is 0.0005 in this experiment. There are the accuracy and loss rate of 50 epochs shown in figure below.

After 50 epochs training, the recognition accuracy reached 95.87% for MNIST in SNPSNet designed in this research under the initial learning rate and batch size according to the above parameters. By modifying the epoch, batch size, and learning rate parameters, the recognition accuracies for MNIST in SNPSNet are shown in Table 1.

Table 1. Experimental results obtained by different parameters

epoch	batch size	initial learning rate	accuracy
30	60	0.001	92.85%
50	100	0.001	93.96%
60	100	0.001	93.55%
30	50	0.0005	93.14%
50	60	0.0005	95.87%
50	100	0.0005	94.32%

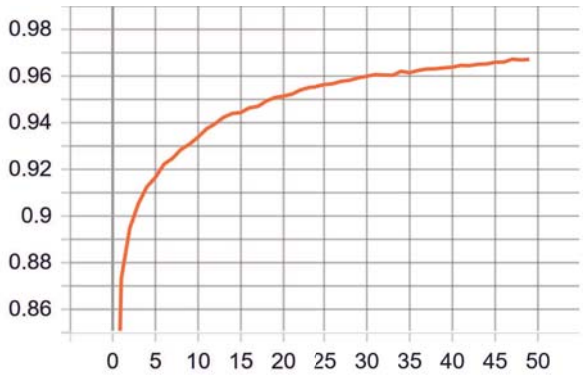


Fig. 5. Recognition accuracy graph for MNIST in training.

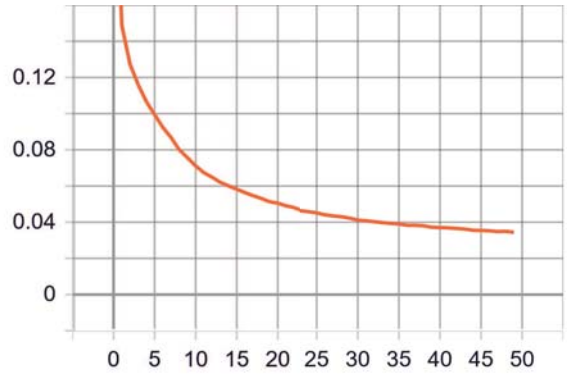


Fig. 6. Training loss graph for MNIST.

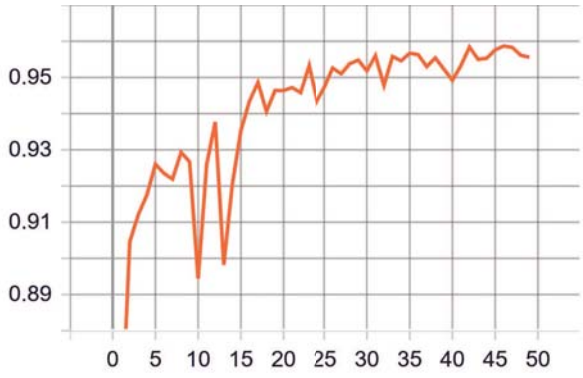


Fig. 7. Recognition accuracy graph for MNIST in test.

4.2 SNPSNET FOR ENGLISH LETTER RECOGNITION

In this paper, 26 classes of English letters are selected which are synthesized by different fonts of the computer in the chars 74k EnglishFnt dataset, including combinations of italic, bold and ordinary characters. Since there are only 1016 images of each type of English alphabet in the original dataset, the quantity of input data as the SNPSNst is too small. The image enhancement is used to expand the number of each class of English letter images to 5000 in this research. And the image enhancement methods adopted in this paper include rotating 30°, 60° and adding 10, 15, 20, 25 noises respectively, which is closer to practical applications. The training set and the test set are divided according to 12:1, the training set is 120,000 pictures, and the test set is 10,000 pictures. The 35 English letter images selected randomly are shown in Figure 8.

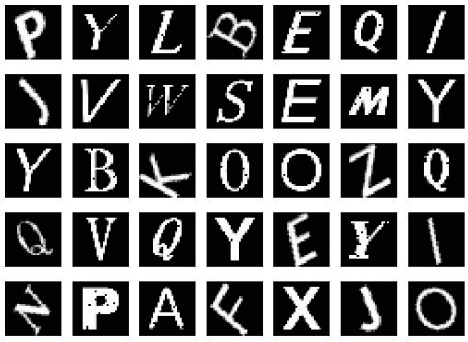


Fig. 8. Schematic diagram of English letter.

First, the image pixel data is encoded into the pulse numbers through the data preprocessing method in Section 3.1, and saved to a file in mat format. For each English letter image, 28*28 pixels, the original pixel data is encoded into the pulse trigger time after the time-to-first spike coding. Next, the time-encoded data of each row in the mat are sorted and the pulse numbers are accumulated during a given time interval. In other words, the element of the rth row and the cth column towards every two-dimensional vector in the mat file indicates pulse numbers, which are triggered at time t $\in (c-1, c), (\mathrm{c} > 0)$ in the rth row of the original pixel matrix. Then, the encoded data is fed in SNPSNet model in Section 3.2 for training, and the rule selected for each spiking neuron in SNPS is $a \to a$, with no delay. One pulse is consumed in spiking neurons that satisfy the firing rule at any moment and $\omega_{ij} \times 1$ pulses are delivered to neurons with synaptic connections in the next layer (ω_{ij} is derived from the learning function shown in Section 2.2). The size of input image is 28*28 in this model; the batch size is 60; the RMSProp is selected as the optimizer, which could be able to adjust the learning rate automatically [45]. Thus, the learning rate is no longer need to

perform any attenuation operations once the initial learning rate is given. The initial learning rate is 0.0005 in this experiment. There are the accuracy and loss rate of 50 epochs shown in figure below.

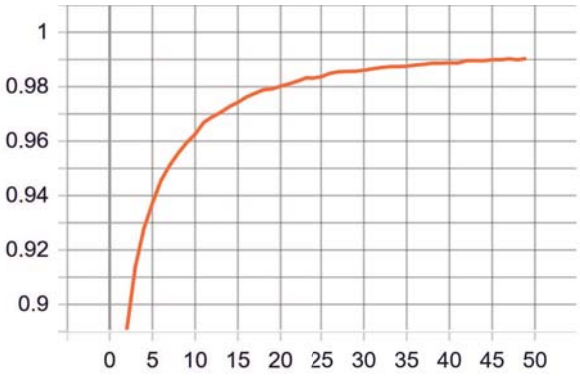


Fig. 9. Recognition accuracy graph for English letter in training.

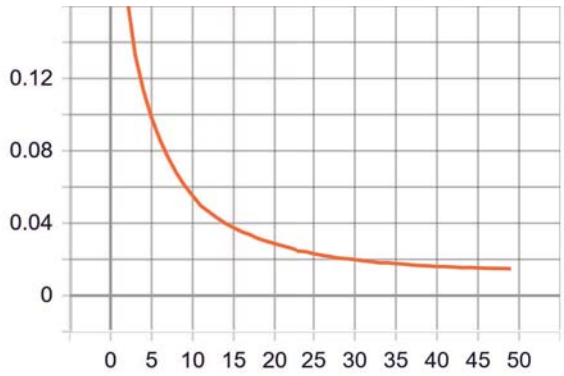


Fig. 10. Training loss graph for English letter.

After 50 epochs training, the recognition accuracy reached 98.06% for the English letter based on the chars 74k dataset in SNPSNet under the learning rate and batch size according to the above parameters.

In order to perform an experiment about the influence of noise on the recognition effect, the Gaussian white noise with signal-to-noise ratio (namely SNR) of 20db, 15db, 10db and 5db is added to the English alphabet test set respectively. The comparison between different noise and the original test set is shown in Figure 12. As shown in the figure, there are 35 images selected randomly from

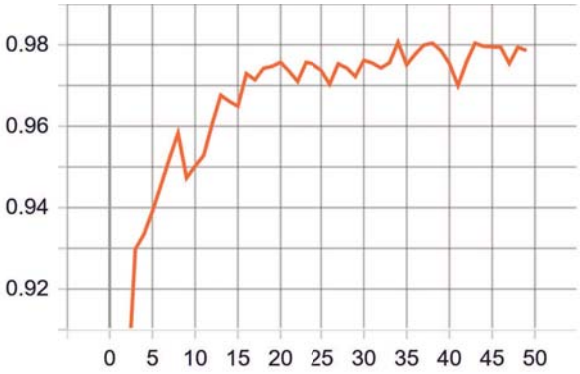


Fig. 11. Recognition accuracy graph for English letter in test.

the original test set and test set with noise which SNR is 20db, 15db, 10db and 5db respectively from top to bottom. And the recognition accuracies obtained are shown in Table 2.

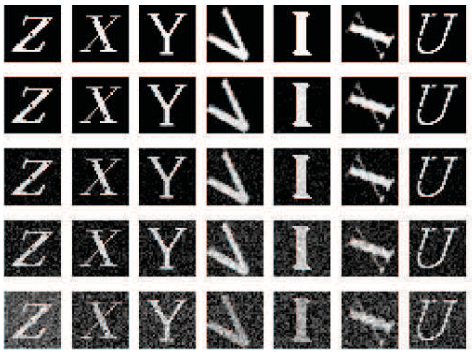


Fig. 12. Schematic diagram of different noise and original image.

Table 2. Experimental results for different noise recognition

SNR of Test set	∞	20db	15db	10db	5db
accuracy	98.06%	98.00%	96.93%	94.80%	86.17%

It is obvious that the test set with the SNR above 20db has little effect on the recognition accuracy towards English alphabet from the experimental

results. When the SNR of the test set is above 10db, the model can achieve high recognition accuracy. And it is proved that SNPSNet possesses a sort of anti-noise ability.

5 CONCLUSION AND PROSPECT

In this research, a novel universal model named SNPSNet is constructed to deal with the image recognition problem. This work is inspired by the routing mechanism in capsule neural network and the theoretical study of SNPS. The time-coded image pixel data is converted into pulse numbers as the input of SNPSNet model while retaining the position information. Besides, the routing mechanism is introduced to update the weights dynamically between the synapses of the neurons in SNPS. The experimental results show that the recognition accuracy for MNIST is 95.87% and the recognition accuracy of English letters after image enhancement reaches 98.06% in SNPSNet, which verify the feasibility and effectiveness of the model. For convolutional and BP neural networks with the same recognition accuracy, the time and space complexity of SNPSNet is reduced to a certain degree. Furthermore, the comparison experiments of the test set under different degrees of noise for English letters recognition prove that SNPSNet possesses a sort of anti-noise ability. The accuracy of SNPSNet is 94.80% and 86.17% when test set SNR is 10db and 5db respectively. Under the same experimental conditions, we did another test where the recognition accuracy is 94.56%, 76.81% respectively in the traditional convolutional neural network. From the experimental results it is obvious that when the SNR of test set is small, SNPSNet has better anti-noise performance in contrast with that of traditional convolutional neural network.

Although it does not reach the current highest recognition level in neural networks, this work is the first attempt to construct a novel SNPS for image recognition in the field of membrane computing which extracts features though the image convolution and applied to MNIST and English letters recognition, which provides some reflections and references for the pattern recognition problem of membrane system. The most prominent feature of SNPS is that different rules can be triggered in the same neuron at different time, and it could deal with different types of objects flexibly, such as texts, images, and voices. For further research, more rules could be adopted to the spiking neurons in SNPS, and corresponding delays could be set for the rules. In addition, a learning mechanism of rules could be introduced to select the disparate rules which achieve an optimal level in each neuron to attain better recognition accuracy and to make the SNPS system more universal.

References

1. Păun G.: Computing with Membranes.Journal of Computer & System Sciences **61**(1), 108-143 (2000)

2. Freund R , Gheorghe Păun, Mario J. Pérez-Jiménez.: Tissue P systems with channel states. Theoretical Computer Science **330**(1), 101–116 (2005)
3. Zhang X , Liu Y , Luo B , et al.: Computational power of tissue P systems for generating control languages. Information Sciences **278**, 285–297 (2014)
4. Freund R , Rogozhin Y , Verlan S .: Generating and accepting P systems with minimal left and right insertion and deletion. Natural Computing **(**2014)
5. Krishna S N, Gheorghe M, Ipate F, et al.: Further results on generalised communicating P systems. Theoretical Computer Science **701**, 146-160 (2017)
6. Besozzi D , Cazzaniga P , Pescini D , et al.: Modelling metapopulations with stochastic membrane systems. Biosystems **91**(3), 499–514 (2008)
7. Colomer, Maàngels, Margalida A , Valencia, Luís, et al.: Application of a computational model for complex fluvial ecosystems: The population dynamics of zebra mussel Dreissena polymorpha as a case study. Ecological Complexity **20**, 116–126 (2014)
8. García-Quismondo, Manuel, Levin M , Lobo D.: Modeling regenerative processes with membrane computing. Information Sciences **381**, 229–249 (2017)
9. Christinal H A , Daniel Díaz-Pernil, Real P.: Region-based segmentation of 2D and 3D images with tissue-like P systems. Pattern Recognition Letters **32**(16), 2206–2212 (2011)
10. Díaz-Pernil, Daniel, Berciano A , Pe$\tilde{n}$A-Cantillana F , et al.: Segmenting images with gradient-based edge detection using Membrane Computing. Pattern Recognition Letters **34**(8), 846–855 (2013)
11. Zhang X , Li J , Zhang L.: A multi-objective membrane algorithm guided by the skin membrane. Natural Computing **15**(4), 597–610 (2016)
12. Li Z , Zhang L , Su Y , et al.: A skin membrane-driven membrane algorithm for many-objective optimization. Neural Computing and Applications **(**2016)
13. Yuan J, Zhang G ,Guo D, et al.: Review of the application of membrane calculation in the field of image processing. Journal of Anhui University(Natural Science) **42**(03), 29–36 (2018)
14. M. Gheorghe, G. Păun, M. J. Perez-Jimenez, and G. Rozenberg.: Spiking neural P systems, research frontiers of membrane computing: Open problems and research topics. International Journal of Foundations of Computer Science **24**(05), 547–623 (2013)
15. Ionescu M, Păun G ,Yokomori T.: Spiking Neural P Systems. Fundamenta Informaticae **71**(2, 3), 279–308 (2006)
16. Maass W, Bishop C M. : Pulsed Neural Networks. Cambridge, MA: MIT Press, Location (2001)
17. Ghosh-Dastidar S , Adeli H: Spiking Neural Networks. International Journal of Neural Systems **19**(04), 295–308 (2009)
18. Maass, Wolfgang: Lower Bounds for the Computational Power of Networks of Spiking Neurons. Neural Computation **8**(1), 1–40 (1996)
19. Maass W.: Networks of Spiking Neurons: The Third Generation of Neural Network Models. Neural Networks **10**(9), 1659–1671 (1997)
20. Zhang G , Rong H , Neri F , et al.: An Optimization Spiking Neural P System for Approximately Solving Combinatorial Optimization Problems. International Journal of Neural Systems **24**(05), 1440006 (2014)
21. Pan L, Păun G.: Spiking neural P systems with anti-spikes. International Journal of Computers Communications & Control **4**(3), 273-282 (2009)
22. Song T, Pan L, Wang J, et al.: Normal forms of spiking neural P systems with anti-spikes. IEEE Transactions on NanoBioscience **11**(4), 352-359 (2012)

23. Pan L, Wang J, Hoogeboom H J.: Spiking neural P systems with astrocytes. Neural Computation **24**(3), 805-825 (2012)
24. Song T, Shi X, Xu J.: Reversible spiking neural P systems. Frontiers of Computer Science **7**(3), 350-358 (2013)
25. Cabarle F G C, Adorna H N, Perez-Jimenez M J, et al.: Spiking neural P systems with structural plasticity. Neural Computing and Applications **26**(8), 1905-1917 (2015)
26. Song T, Pan L, Păun G.: Spiking neural P systems with rules on synapses. Theoretical Computer Science **529**, 82-95 (2014)
27. Song T, Pan L.: Spiking neural P systems with rules on synapses working in maximum spikes consumption strategy. IEEE Transactions on NanoBioscience **14**(1), 38-44 (2015)
28. Song T, Pan L.: Spiking neural P systems with rules on synapses working in maximum spiking strategy. IEEE Transactions on NanoBioscience **14**(4), 465-477 (2015)
29. Wang J , Shi P , Peng H , et al.: Weighted Fuzzy Spiking Neural P Systems. IEEE Transactions on Fuzzy Systems **21**(2), 209–220 (2013)
30. Díaz-Pernil, Daniel, Pe$\tilde{n}$A-Cantillana F , Gutirrez-Naranjo, Miguel A.: A parallel algorithm for skeletonizing images by using spiking neural P systems. Neurocomputing **115**, 81–91 (2013)
31. Zhang G , Rong H , Neri F , et al.: An optimization spiking neural p system for approximately solving combinatorial optimization problems. International Journal of Neural Systems **24**(05), 1440006 (2014)
32. Peng H , Wang J , Pérez-Jiménez, Mario J, et al.: Fuzzy reasoning spiking neural P system for fault diagnosis. Information Sciences **235**(Complete), 106–116 (2013)
33. Zhao J B , Wang T , Zhang G , et al.: Fault Diagnosis of Electric Power Systems Based on Fuzzy Reasoning Spiking Neural P Systems. IEEE Transactions on Power Systems **30**(3), 1182–1194 (2014)
34. Xiong G, Shi D, Zhu L, et al.: A new approach to fault diagnosis of power systems using fuzzy reasoning spiking neural P systems. Mathematical Problems in Engineering **2013**, 1-13 (2013)
35. Wang T, Zhang G, Rong H, et al.: Application of fuzzy reasoning spiking neural P systems to fault diagnosis. International Journal of Computers Communications & Control **9**(6), 786-799 (2014)
36. Yahya Y, Qian A, Yahya A. Power transformer fault diagnosis using fuzzy reasoning spiking neural P systems. Journal of Intelligent Learning Systems and Applications.: Tissue P systems with channel states. Theoretical Computer Science **8**(04), 77-91 (2016)
37. Gutiérrez-Naranjo M A, Pérez-Jiménez M J.: A spiking neural P system based model for Hebbian learning. in: Proceedings of the 9th Workshop on Membrane Computing **1**89-207 (2008)
38. Song T , Pan L , Wu T , et al.: Spiking Neural P Systems with Learning Functions. IEEE Transactions on NanoBioscience **1**(1),(2019)
39. Sabour S , Frosst N , Hinton G E.: Dynamic Routing Between Capsules. Neural Information Processing Systems **3**856–3866 (2017)
40. Optican L M, Richmond B J.: Temporal encoding of two-dimensional patterns by single units in primate inferior temporal cortex. III. Information theoretic analysis. Journal of Neurophysiology **57**(1), 162 (1987)
41. Tovee M J , Rolls E T , Treves A , et al.: Information encoding and the responses of single neurons in the primate temporal visual cortex. Journal of Neurophysiology **70**(2), 640–654 (1993)

42. Kjaer T W , Hertz J A , Richmond B J.: Decoding cortical neuronal signals: Network models, information estimation and spatial tuning. Journal of Computational Neuroscience **1-2**(1), 109–139 (1994)
43. Maass W.: Computing with spiking neurons. Pulsed neural networks, Location (1999)
44. Bialek W, Rieke F, De R R V S, et al.: Reading a neural code. Science **252**(5014), 1854–1857 (1991)
45. Tieleman T, Hinton G.: Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning **4**(2), 26–3 (2012)

Reliability Evaluation of Distribution Network Based on Fuzzy Spiking Neural P System with Self-Synapse

YuLei Huang[1,2], Tao Wang[1,2], and Jun Wang[1,2] Hong Peng[3]

[1] School of Electrical Engineering and Electronic Information, Xihua University, Chengdu 610039, China
[2] Key Laboratory of Fluid and Power Machinery, Ministry of Education, Xihua University, Chengdu 610039, PR China
[3] School of Computer and Software Engineering, Xihua University, Chengdu 610039, PR China

Abstract. This paper proposes a fuzzy spiking neural P system with self-synapse (in short, FSNPSSs) which applied to the reliability assessment of distribution network. This method maps the operation or fault state of the distribution network component and the load to the excited or resting state of the neuron, and converts the electrical relationship between the component and the load and the system into a synaptic connection relationship. Then, the probability of occurrence of the state is transmitted in the form of a pulse value, and the reliability index of the distribution network is accumulated. Finally, the successful application of membrane system in reliability assessment of distribution network is realized.

Keywords: spiking neural P system, self-synapse, reliability assessment, distribution network

1 Introduction

The reliability of the distribution network of power system is directly related to the electricity consumption experience of the majority of users. It is of great practical significance to evaluate the reliability of the distribution network. The mainly methods for the reliability assessment of distribution network include the Monte Carlo method [1] and the analytical method [2]. These two methods can be used to calculate the reliability index of the system. However, these two methods cannot be used to calculate the failure probability of the component in the system failure, and cannot identify the weak link of the system. For this reason, many scholars have proposed Bayesian networks and other reasoning models to calculate the degree of influence of various components in the power grid on the system [3][4]. But there are certain difficulties in calculating the index of the system failure frequency when using a Bayesian network for non-sequential simulation of reliability.

The membrane computing, also known as P system, is a distributed parallel computing system derived from the structure and function of biological cells, first proposed by Păun in [5]. Spiking neural P system (in short, SNPs) is an important branches of the P system, which realize the information transfer and processing by simulating the transmission process of electrical impulses [6][7]. The neurons and the synapses are the basic composed of components SNPs. In SNPs, a neuron and a synapse represent a node and a connection among nodes(neurons), respectively. The synapses in original SNPs is static which limits its extension of function. So, in recent years, some studies focused on synapse modification and proposed a few new models through adding the learning

functions [8][9], considering the plasticity or timeliness of synapse [10][11], introducing the division and budding of neuron to the SNPs [12] and so on.

Under the guidance of the basic ideas of SNPs, many scholars have proposed fuzzy spiking neural P system (in short, FSNPs) by changing the transfer medium of the pulse in SNPs [13][14]. FSNPs is ability to process fuzzy information with dynamically and efficiently, and it has been successfully applied in the field of power system fault diagnosis [15][16][17][18]. At present, in general FSNPs and their extension systems, it is believed that the synapse connection can only from one neuron to another neuron and not to itself. Biological studies have shown that neuronal synapses can be directed not only from one neuron to another, but also from neuronal axons to their own dendrites or soma. These synapses are also called self-synapses [19][20], and the existence of self-synapse is of great significance to the normal physiological activities of living things. In addition, currently FSNPs is mainly used in the field of fault diagnosis of power systems, and there is still a lack of research data for the application of FSNPs in other fields of power systems.In addition, the introduce of self-synapse can improve the programming experience to convenient to observe the change of pulse value with time.

In this paper, a FSNPs with self-synapses(in short, FSNPSSs) is proposed. The FSNPSSs can accumulate the pulse value, and this feature is applied to calculate the reliability index of the distribution network. This method has the following two improvements to the FSNPs: (1) A new type of synapse(self-synapse) is added to the FSNPs. The FSNPSSs break through the limitations that the spiking of the neurons cannot be passed on to itself in FSNPs. As a result of, FSNPSSs realizing the accumulation of pulse values. (2) The forgetting rule is used for terminating system operations. In this paper, the RBTS-BUS2 system is used as an example to verify the feasibility of the algorithm. The self-synaptic neurons accumulate the probability of failure at each load point, and finally realize the calculation of the system reliability index. In the meantime, the analysis of the influence degree of components on the system can be simultaneously completed and provide an important data for improving the reliability of the distribution network.

2 Problem Description

Fig. 1 is a widely studied distribution network system(RBTS-BUS2-F1) which mainly includes 7 loads (LP1∼LP7), three segment switches (D1∼D3), 11 lines (L1∼L11), 7 transformers and a alternate supply(A) [21]. When the distribution network is in normal operation, all components are in normal working condition, and the segmentation switch is closed, and the load point is powered by the main power supply. When the line or transformer located on the load branch fails, due to the isolation of the fuse, only the load branch where the faulty component is located is power failure. And the power outage time is the component failure time, and other loads are not affected. For example, line L5 or transformer T5 failure will cause to load LP3 or load LP5 failure, respectively. If the fault occurs on the main feeder, all load points will be power failure. Then, the failure time of the load branch power which is in series with the faulty component is fault time of the component. However, the segment switch directly connected to the fault point will be disconnected, which is located at the front and rear ends of the load point. The load branch can be restored by the main power supply and the backup power supply respectively, and the failure time of the power is the switching operation time. For example, if line L7 is failure, then loads LP1∼LP7 will be outage of power.

The different state combinations of all components correspond to a certain probability of occurrence. Moreover, the probability of outage of each load point can be obtained by accumulating

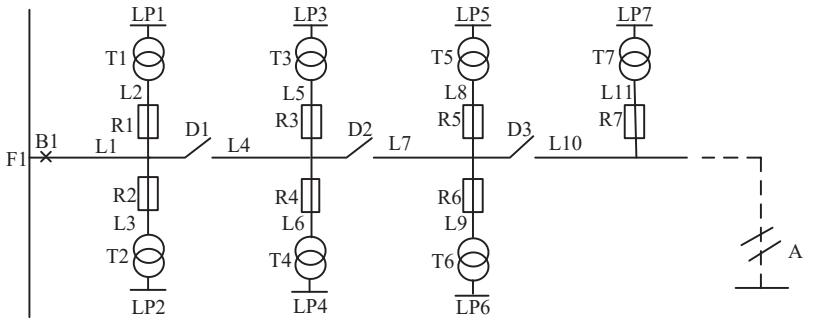


Fig. 1. RBTS-BUS2-F1 feeder system

the probability of failure of the load point under different operating states. Finally, the system reliability index can be calculated based on the corresponding data.

3 The fuzzy spiking neural P system

We briefly review the basic definition of the fuzzy spiking neural P system (FSNPs) [13].
Definition 1: An FSNPs (with degree $m \geq 1$) is a construct

$$\Pi = (A, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, I, O) \tag{1}$$

Here:

(1) A={a} is a singleton alphabet (a is called spike);
(2) $\sigma_1, \sigma_2, \ldots, \sigma_m$, are m neurons, of the form $\sigma_i = (\theta_i, c_i, r_i), i = 1, 2, \cdots, m$, where:
  (a) θ_i is a real number in [0, 1] representing the potential value of spikes(also called pulse value) contained in neuron σ_i;
  (b) c_i is a real number in [0, 1] representing the truth value associated with neuron σ_i;
  (c) r_i is a firing rule corresponding to neuron σ_i, of the form $E/a^{\theta} \rightarrow a^{\beta}$, where E is a regular expression,θ and β are real numbers in [0, 1];
(3) $syn \subseteq \{1, 2, \cdots, m\} \times \{1, 2, \cdots, m\}$ is the connection among neurons (synaptic directed graph), for all $(i, j) \in syn, i \leq m, j \leq m, i \neq j$;
(4) I and O are the input neuron set and the output neuron set, respectively.

The basic operation process of the FSNPs is as follow: firstly, the input neurons obtain input pulses from the environment. Then, if the neuron satisfies the firing condition, it sends pulses to all postsynaptic neurons which are connected to itself. Moreover the output neurons will output the final result. Since this system does not have a memory function, the neurons cannot accumulate the pulse values.

4 A fuzzy spiking neural P system with self-synapse

In this paper, we need to accumulate the pulse values obtained in each time. So, a fuzzy spiking neural P system with self-synapse system(FSNPSSs) is proposed.

Definition 2: An FSNPSSs (with degree $m \geq 1$) is a construct

$$\prod = (A, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, I, O) \tag{2}$$

Here:

(1) A, I and O are the same as definition 1.

(2) $\sigma_1, \sigma_2, \ldots, \sigma_m$, are m neurons, of the form $\sigma_i = (\theta_i, c_i, \overrightarrow{\omega}_i, \lambda_i, r_i), i = 1, 2, \cdots, m$, here:

(a) θ_i is a real number in [0, 1] representing the potential value of spikes (also called pulse value) contained in neuronσ_i;

(b) c_i is a real number in [0, 1] representing the truth value associated with neuron σ_i;

(c) $\overrightarrow{\omega}_i = (\omega_{i1}, \omega_{i2}, \cdots, \omega_{in})$ is a real vector in $[0, 1]^k$ representing the weight value of synaptic between neuron σ_i and postsynaptic neurons, where k is the dimension of $\overrightarrow{\omega}_i$ and n is the number of postsynaptic neurons with neuron σ_i.

(d) $\lambda_i = \{\lambda_i^r, \lambda_i^s\}$ is a pair of real numbers in [0, 1]. λ_i^r and λ_i^s represent the firing threshold and forgetting threshold of the neuron σ_i , respectively.

(e) r_i is a rule set. Here

1) $E/a^\theta \rightarrow a^\beta$ is a firing(spiking) rule, and $\theta > \lambda_i^r$ is the firing condition. If the θ received by σ_i is greater λ_i^r, the firing rule is activated.

2) $E/a^\theta \rightarrow \lambda$ is a forgetting rule, and $\theta \geq \lambda_i^s$ is the forgetting condition. If the θ received by σ_i is greater or equal to λ_i^s, the forgetting rule is activated. For the same neuron σ_i, $\lambda_i^s > \lambda_i^r$.

(f) $syn \subseteq \{1, \cdots, m\} \times \{1, \cdots, m\}$ is the connection among neurons(synaptic directed graph), for all $(i, j) \in syn, i \leq m, j \leq m$.

The FSNPSSs is similar in form to the FSNPs. The main difference between the FSNPs and the FSNPSSs is that the FSNPSSs remove the restriction $i \neq j$ in *syn*, and allow neurons to link synapses to themselves, thereby passing pulses to themselves. In addition, the forgetting rule is used to terminate the operation of the system.

4.1 *Fuzzy production rules*

According to different purposes, the neurons are divided into two types: the proposition neurons and the regular neurons. The regular neurons have two kinds of: the *or* regular neurons and the *and* regular neurons.

Definition 3: A proposition neuron, as shown in Fig. 2, represents a proposition in a production rule which is expressed by a symbol P.

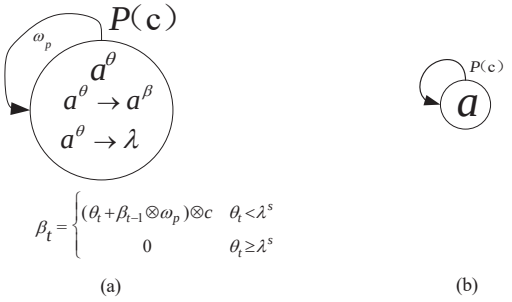


Fig. 2. (a) Proposition neuron and (b) its simplified form.

A proposition neuron can be represented as $\sigma = (\theta, c, \overrightarrow{\omega}, r)$, where θ, c are the pulse value and the true value of the proposition neuron, respectively. $\overrightarrow{\omega} \subseteq [0,1]^k$ is the k dimensions synaptic weight vector, where ω_p is the self-synaptic weight value which is a special elements in $\overrightarrow{\omega}$. If $\omega_p = 0$, then this proposition neuron is a neuron without self-synapse. r is a finite rule set in which the firing rule is $E/a^\theta \to a^\beta$ and the forgetting rule is $E/a^\alpha \to \lambda$.

(1) If $\omega_p \neq 0$, when the neuron receives a pulse value from the other neurons at time t meets the $\theta_t < \lambda^s$, the firing rule is triggered. And the neuron updated pulse value in accordance with the $\beta_t = \theta_t + \beta_{t-1} \otimes \omega_p$ and transmitted it to each of the postsynaptic neurons which connected itself. If $\theta_t > \lambda^s$, the forgetting rule is triggered and the pulse value is cleared.

(2) If $\omega_p = 0$, when the neuron receives a pulse with pulse value θ_t of a at time t, the pulse value is updated according to $\beta_t = \theta_t$ and it is transmitted to each postsynaptic neuron. This type of neuron has no forgetting rules. If the proposition neuron is an input neuron, then its pulse value is obtained from the environment. Otherwise, its pulse value is obtained from its presynaptic neurons.

Definition 4: An or rule neuron, as shown in Fig. 3, a rule that indicates an "or" relationship between presynaptic proposition neurons, represented by the symbol *or*. It has more than two presynaptic proposition neurons, but only one post-synaptic proposition neuron.

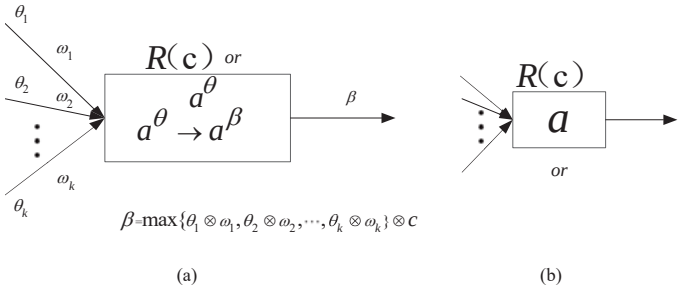


Fig. 3. An *or* rule neuron

When the *or* rule neuron receives k pulses of pulse value $\theta_1, \theta_2, \cdots, \theta_k$, if the firing condition is satisfied, a pulse with a pulse value of $\beta = max\{\theta_1 \otimes \omega_1, \theta_2 \otimes \omega_2, \cdots, \theta_k \otimes \omega_k\} \otimes c$ is outputted to its postsynaptic neurons. Otherwise the received pulse is used to update the neuron pulse value.

Definition 5: An and rule neuron, as shown in Fig. 4, a rule that indicates an "and" relationship between presynaptic proposition neurons, represented by the symbol *and*. It has more than one presynaptic proposition neurons, but only one postsynaptic proposition neuron.

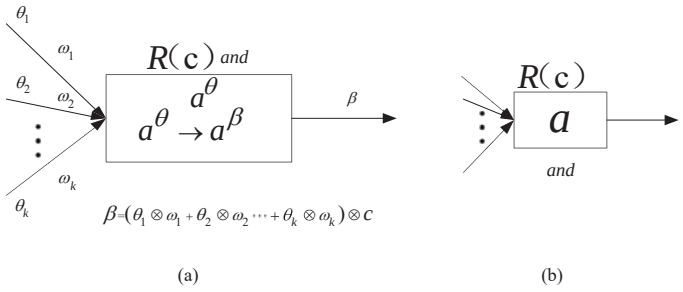


Fig. 4. An *and* rule neuron

When the *and* rule neuron receives k pulses of pulse value $\theta_1, \theta_2, \cdots, \theta_k$, if the firing condition is satisfied, a pulse with a pulse value of $\beta = (\theta_1 \otimes \omega_1 + \theta_2 \otimes \omega_2 + \cdots + \theta_k \otimes \omega_k) \otimes c$ is outputted to its postsynaptic neurons. Otherwise the received pulse is used to update the neuron pulse value.

In addition:

(1) θ_{jt} is the pulse value corresponding to the neuron θ_j at time t.

(2) All regular neurons do not have forgetting rules.

4.2 *FSNPSSs model for fuzzy production rules*

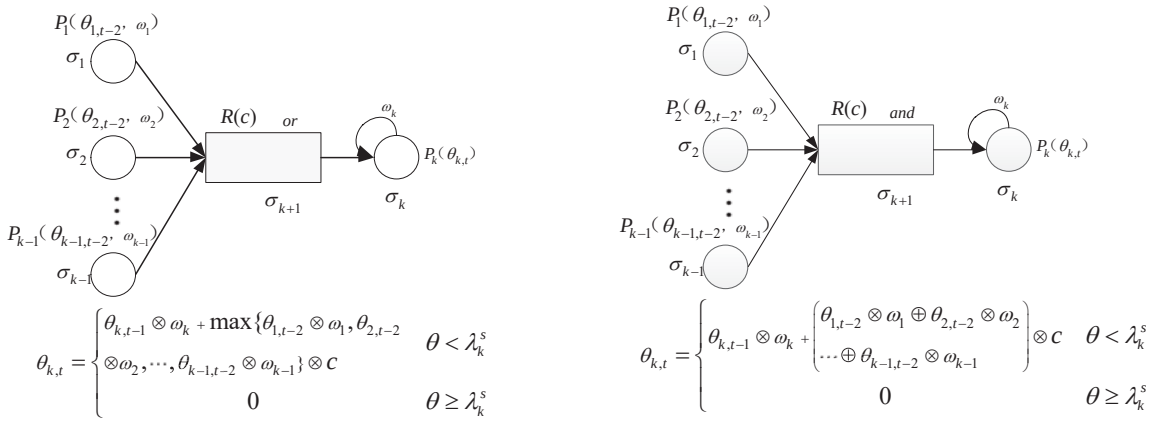
In this paper, the fault transfer process of the distribution network will be simulated, so the following fuzzy production rule models are established. *Type1: or* rule model. The modeling process is shown in Fig. 5(a), and the reasoning process is as follows:

Firstly, neuron σ_{k+1} obtains one group pulse with pulse value of $\theta_{1,t-2}, \theta_{2,t-2}, \cdots \theta_{k-1,t-2}$ from the environment or the presynaptic neurons. Then, the neuron σ_{k+1} transmits a pulse with value of $max\{\theta_{1,t-2} \otimes \omega_1 + \theta_{2,t-2} \otimes \omega_2 + \cdots + \theta_{k-1,t-2} \otimes \omega_{k-1}\} \otimes c$ to the postsynaptic proposition neuron σ_k. Finally, the proposition neuron σ_k decided to use the firing rule or the forgetting rule according to the received pulse value. If the firing rule is used, the pulse value is updated according to $\theta_{k,t} = \theta_{k+1,t-1} + \theta_{k,t-1} \otimes \omega_k$. And if the forgetting rule is used, the pulse value is cleared. For the neurons without self-synapse, no forgetting rules are used.

Type2: and rule model. The modeling process is shown in Fig. 5(b), and the reasoning process is similar to *type1*.

4.3 *Reasoning algorithm based on FSNPSSs*

Based on the definition of each neuron type and neuron production rule of the FSNPSSs, this section proposes a corresponding fuzzy reasoning algorithm. Including the definition of transfer



(a) *or* rule model (b) *and* rule model

Fig. 5. (a) *or* rule model and (b) *and* rule model

matrix, pulse vector, arithmetic operator, and the update method of neuron pulse value etc. The reasoning algorithm is as follows:

(1) Initialization the proposition neuron pulse vector is $\theta_t = (\theta_{1,t}, \theta_{2,t}, \cdots, \theta_{p,t})_p$ and the regular neuron pulse vector δ is $\delta_t = (\delta_{1,t}, \delta_{2,t}, \cdots, \delta_{r,t})_t$. The start time t=1 and the end time t=T. Synaptic weight value matrix is set to $W_{r1}, W_{r2}, W_{p1}, W_{p2}$.

(2) Corresponding to the input pulse sequence according to the pulse time series table.

(3) It is judged whether the neuron forgetting rule is satisfied. If it is satisfied, then the forgetting rule is executed and no pulse is generated,and the pulse value is 0. If the forgetting rule is not met, it is judged whether the firing rule of the neuron is satisfied. If it is satisfied, then a pulse is transmitted to all postsynaptic neurons of the regular neuron.

(4) Updating the regular neuron pulse vector with the following formula.

$$\delta_{t+1} = W_{r1}^T \odot \theta_t \tag{3}$$

(5) Updating the proposition neuron pulse vector with the following formula.

$$\theta_{t+1} = W_{p1}^T \odot (C_T \otimes \delta_t) + W_{p2}^T \otimes \theta_t \tag{4}$$

(6) Determining whether t is equal to the termination time T. If it is satisfied, the operation is terminated and the result is outputted. If not, then t=t+1 and return (2). Determining whether t is equal to the termination time T, and if it is satisfied, the operation is terminated and the result is output. If not, then t=t+1 and return (2).

Here:

(a) $W_{r1} = (\omega_{ij})_{p\times r}$ is the synaptic weight matrix representing the directed weight connection from proposition neurons to or rule neurons. If there is a synapse from the proposition neuron σ_i the or rule neuron to σ_j, then $\omega_{ij} \neq 0$, otherwise, $\omega_{ij} = 0$.

(b) $W_{r2} = (\omega_{ij})_{p\times r}$ is the synaptic weight matrix representing the directed weight connect-

ion from proposition neurons to and rule neurons. If there is a synapse from the proposition neuron σ_i to the and rule neuron σ_j, then $\omega_{ij} \neq 0$, otherwise, $\omega_{ij} = 0$.

(c) $W_{p1} = (\omega_{ij})_{r\times p}$ is the synaptic weight matrix representing the directed weight connection from regular neurons to proposition neurons. If there is a synapse from the regular neuron σ_i to the proposition neuronσ_j, then $\omega_{ij} \neq 0$, otherwise, $\omega_{ij} = 0$.

(d) $W_{p2} = (\omega_{ii})_{p\times p}$ is the synaptic weight matrix representing the directed weight connection from proposition neurons to proposition neurons. If there is a synapse from the proposition neuron σ_i to the proposition neuron σ_i ,then $\omega_{ii} \neq 0$, otherwise, $\omega_{ii} = 0$.

(e) $\theta_t = (\theta_{1,t}, \theta_{2,t}, \cdots, \theta_{p,t})_p$ is the pulse value vector of the proposition neuron, where $\theta_{i,t}$ is the pulse values of the ith proposition neuron at time t and it is a real numbers in [0, 1], i=1, 2, $\cdots$, p.

(f) $\delta_t = (\theta_{1,t}, \theta_{2,t}, \cdots, \theta_{r,t})_r$ is the pulse value vector of the regular neuron, where $\delta_{j,t}$ is the pulse values of the jth regular neuron at time t and it is a real numbers in [0, 1], j=1, 2, $\cdots$, r.

(g) $C = diag(c_1, c_2, \cdots, c_r)$, where c_j is the real number in [0, 1], j=1, 2, $\cdots$, r. Indicates the pulse truth value of the jth regular neuron, that is, its deterministic factor corresponding to the production rule.

Here:

$$W^T \odot \theta = (\delta_1, \delta_2, \cdots, \delta_p)^T, where,\ \delta_i = max\{w_{1i} \otimes \theta_1, w_{2i} \otimes \theta_2, \cdots, w_{pi} \otimes \theta_p\}, i = 1, 2, \cdots, p. \quad (5)$$

$$W^T \otimes \theta = (\delta_1, \delta_2, \cdots, \delta_r)^T, where,\ \delta_i = w_{1i} \otimes \theta_1 \oplus w_{2i} \otimes \theta_2 \oplus \cdots \oplus w_{1i} \otimes \theta_p, i = 1, 2, \cdots, p. \quad (6)$$

$$W^T \odot \delta = (\theta_1, \theta_2, \cdots, \theta_r)^T, where,\ \theta_i = max\{w_{1i} \otimes \delta_1, w_{2i} \otimes \delta_2, \cdots, w_{ri} \otimes \delta_r\}, i = 1, 2, \cdots, r. \quad (7)$$

$$C \odot \delta = (c_1 \otimes \delta_1, c_2 \otimes \delta_2, \cdots, c_r \otimes \delta_r)^T. \quad (8)$$

5 Distribution Network Reliability Evaluation Algorithm Based on FSNPSSs

In this paper, the process of the reliability assessment of distribution network is as follows. Firstly, accumulating the probability of power failure of each load point according to different state combinations of all components, and then calculates the probability of power failure of the feeder by the FSNPSSs. Finally, the reliability index is calculated according to the statistical data. Therefore, the fault information transmission sequence is from the component to the load point to the entire feeder. When mapping the FSNPSSs, each component is used as an input unit, and the load and the feeder are used as output units.

According to the above principles, the reliability evaluation method based on the FSNPSSs is as follows:

(1) Data initialization, which mainly includes calculating the failure probability of each component node and joint node. In this paper, we use 1 and 0 indicate the fault status and the normal operation status respectively. The calculation method is as follows:

(a) For line components, the probability of failure is:

$$P(L = 1) = l(\lambda_l r_l + \lambda'_l r'_l)/8760 \quad (9)$$

Here, l is the length of the line. λ_l and λ'_l, the annual average failure rate and the annual average planned maintenance rate of the line, respectively. r_l and r'_l are the annual fault repair time and planned maintenance time of the line, respectively.

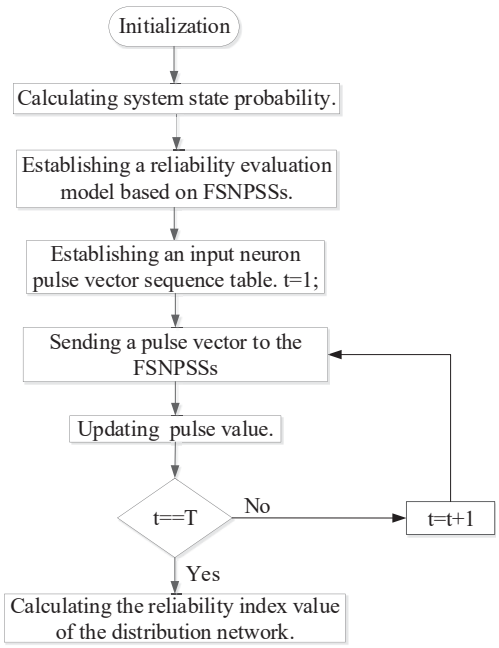


Fig. 6. Reliability Evaluation Flow of Distribution Network Based on FSNPSSs

(b) For transformer components, the probability of failure is:

$$P(T=1) = l(\lambda_T r_T + \lambda'_T r'_T)/8760 \tag{10}$$

Here, λ_T and λ'_T are the annual average failure rate of the transformer and the annual average planned maintenance rate, and r_T is the annual fault repair time or replacement time of the transformer. r'_T is the planned maintenance time of the transformer.

(c) For a joint node composed of a line and a transformer in series, the joint failure probability is:

$$P(LT=1) = 1 - P(L=0)P(T=0) \tag{11}$$

(d) For the joint node composed of the line and the front-end segment switch, the joint failure probability is:

$$P(LD=1) = l\lambda_l r_D/8760 \tag{12}$$

Here,r_D is the action time of the segment switch.

(e) For joint nodes composed of line and back-end sectional switches and alternate supply, t-

he joint failure probability is:

$$P(LDA=1) = l\lambda_l r_m/8760 \tag{13}$$

Where, r_m is the maximum operating time of the sectional switch and the standby power switch closing time.

(2) According to the relationship between components and loads, load and feeder in fault transmission. The network model based on FSNPSSs is established.
(3) The table of input pulse vectors sequence is calculated according to the probability obtained in (1). The impulse sequence of input neurons in (2) is in detail described in case analysis.
(4) Running the FSNPSSs according to the reasoning algorithm in Section 4.3.
(5) Determining whether the termination operation condition is satisfied. If it is satisfied, go to (6), and if not, return to (4).
(6) Obtaining the failure probability of each load point and feeder, and calculate all the reliability indicators in combination with the output unit's corresponding state change of the input units.

The calculation process is shown in Fig. 6.

6 Case analysis

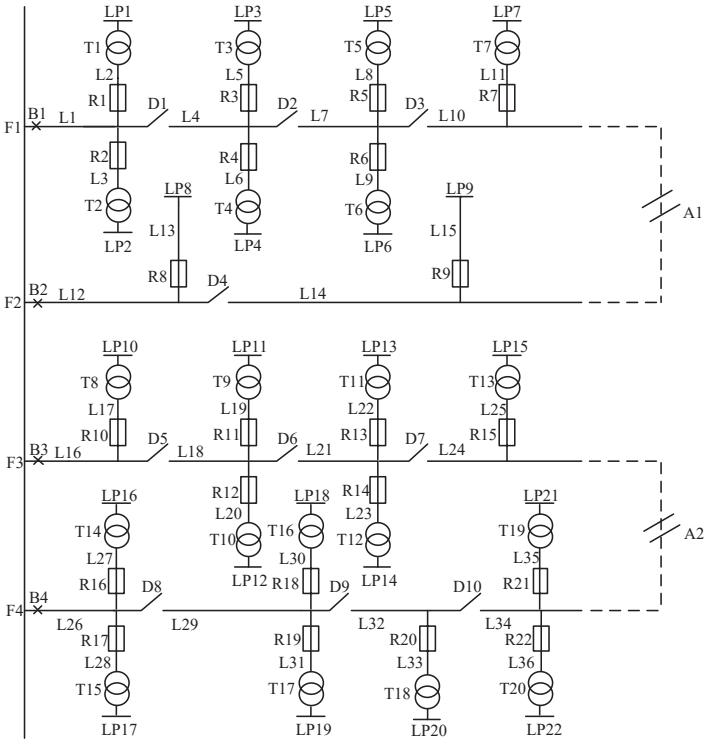


Fig. 7. The RBTS-BUS2 system

In this paper, RBTS-BUS2 is used as the test system. The system structure and parameters, such as line length, failure rate and repair time, are presented in Ref. [21] There are four feeders

(F1∼F4) in the system. The structure of BUS2 is shown in Fig. 7. Taking the feeder F1 as an example, the rationality of this method is illustrated. The calculation process of other feeders and the whole bus are similar to the F1. According to the discussion in Sections 1 to 5, the failure probability of each node in F1 is calculated and its FSNPSS network is established as shown in Fig. 8.

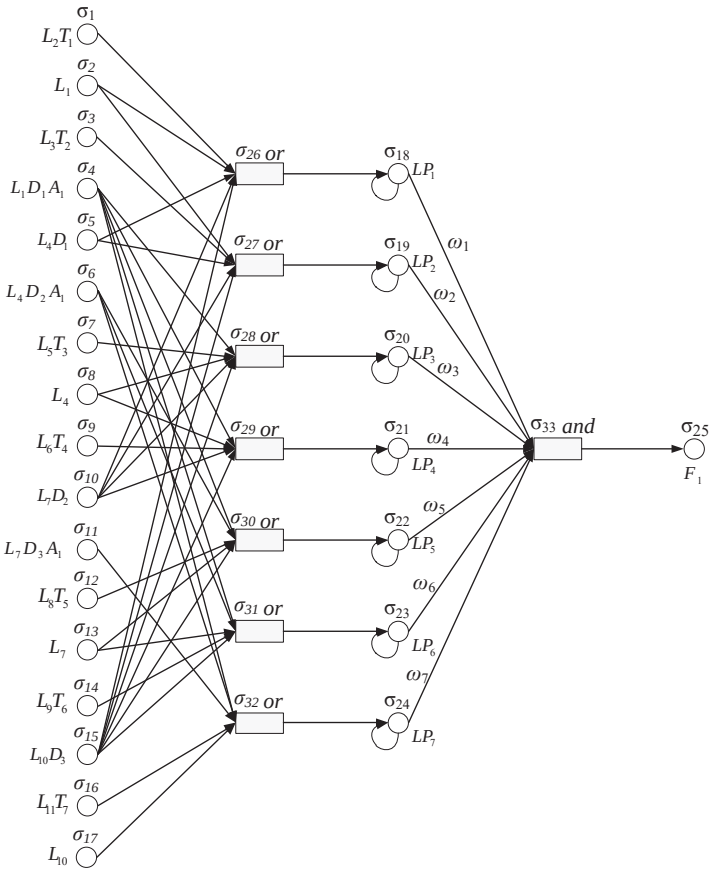


Fig. 8. FRSNPSS network model corresponding to BUS2-F1

Here:

(1) The input neuron is $\sigma_1 \sim \sigma_{17}$, corresponding to the component node, and the output neuron is $\sigma_{18} \sim \sigma_{25}$, corresponding to the load points and the feeder. Among them, $\sigma_{18} \sim \sigma_{24}$ neurons are the neurons with self-synapse.

(2) According to the electrical principle, the relationship among the input nodes and the load nodes are mapped to the neuron $\sigma_{26} \sim \sigma_{32}$ in the graph. For example, synapses from the input neurons $\sigma_1, \sigma_2, \sigma_5, \sigma_{10}$ and σ_{15} to the neuron σ_{26} means that for nodes L2T1, L1,

L4D1, L7D2 and L10D3, as long as there is one fault, the load point LP1(σ_{18}) will be cut off, so these neurons($\sigma_1, \sigma_2, \sigma_5, \sigma_{10}$ and σ_{15}) are connected by or regular neuron(σ_{26}). Theirs synaptic weights are all 1 (Not shown in the Fig. 8).

(3) Each load acts on the feeder through different influence factors, which are mapped to neuron σ_{33} in the graph. For these synapses from neurons $\sigma_{18} \sim \sigma_{24}$ to neuron σ_{33} means that for load points LP1∼LP7 whose fault influence on feeder F1(σ_{25}) by a certain weight value($\omega_1 \sim \omega_7$) respectively, so these neurons($\sigma_{18} \sim \sigma_{24}$) are connected by and regular neuron(σ_{33}). And the formula of synaptic weight is:

$$\omega_i = N_i / \sum_{i=1}^{n} N_i \tag{14}$$

Where, Ni is the number of users of ith load point and n is the number of load points.

(4) The last input of the input neuron is a pulse with a pulse value of 1. The purpose is to stimulate the forgetting rule in the neurons with self-synapse, so that the output is zero, and the system is prevented from running unrestricted. The threshold of the forgetting rule within the neurons with self-synapse is $\lambda^s = 1$.

(5) The input neuron pulse sequence is shown in Table 1.

Table 1. Input neuron status sequence

t	σ_1	σ_2	σ_3	σ_4	σ_5	σ_6	σ_7	σ_8	$\cdots$	σ_{17}	θ
0	0	0	0	0	0	0	0	0	$\cdots$	0	θ_0
1	0	0	0	0	0	0	0	0	$\cdots$	1	θ_1
2	0	0	0	0	0	0	0	0	$\cdots$	0	θ_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
2^n-1	1	1	1	1	1	1	1	1	$\cdots$	1	θ_{2^n-1}
2^n	1	1	1	1	1	1	1	1	$\cdots$	1	θ_{2^n}

Where, 1 and the 0 represents the excitation or rest of the neuron, indicating that the element is in the fault/disconnection state or in the normal/closed state, respectively.

(a) The formula for calculating the pulse value is as follows:

$$\theta_{t,i} = \begin{cases} S_{t,i} \prod_{i=1}^{n} P_{t,i} & if \quad t \leq 2^n - 1 \\ 1 & if \quad t = 2^n \\ 0 & if \quad t > 2^n \end{cases} \tag{15}$$

Here:

$$P_{t,i} = \begin{cases} P & if \quad S_{t,i} = 0 \\ 1 - P & if \quad S_{t,i} = 1 \end{cases} \tag{16}$$

P is the normal working probability of the corresponding element of the neuron. $S_{t,i}$ is the state of the component i at time t. n is the number of all elements. 1 and 0 represents the state of fault/action or the state of normal/non-action, respectively.

6.1 The reliability index of the system

In this paper, the simulation operation is performed on matlab2010b, and the reasoning algorithm is programmed according to the Section 4. According to the output result, the reliability index of the system can be calculated, including ASUI (system average unavailability index) ASAI (system average availability index), SAIDI (system average interruption duration index), EENS (expected energy not supplied), SAIFI (system average interruption frequency index). Table 2 shows the comparison of FSNPSS results with those in Ref.[21]. (The ASUI can be derived directly from the final output of the system. The ASAI and SAIDI can be derived from ASUI. The SAIFI can be calculated according to the change of load point failure rate with component state, and EENS can be calculated from the energy shortage at each load point.These indexes are calculated by formulas provided in [22])

Table 2. The results calculated from the FSNPSS model and Ref.[21]

	ASAI		SAIFI		EENS		ASUI		SAIDI	
	FSNPSS	Ref.[21]	FSNPSS	Ref.[21]	FSNPSS	Ref.[21]	FSNPSS	Ref.[21]	FSNPSS	Ref.[21]
F1	0.999912	0.999912	0.247	0.248	13171	13172	0.000088	0.000088	3.61	3.62
F2	0.999940	0.999940	0.139	0.140	1122	1122	0.000060	0.000060	0.52	0.52
F3	0.999586	0.999586	0.230	0.250	11203	11203	0.000412	0.000412	3.60	3.61
F4	0.999588	0.999588	0.248	0.247	12248	12248	0.000412	0.000412	3.60	3.61
BUS2	0.999588	0.999588	0.241	0.248	37743	37746	0.000412	0.000412	3.62	3.61

It can be seen from the data in the table that the results obtained by the FSNPSSs method and the Ref.[16] are consistent, and the correctness of the method is verified.

6.2 The failure probability of component during system failure

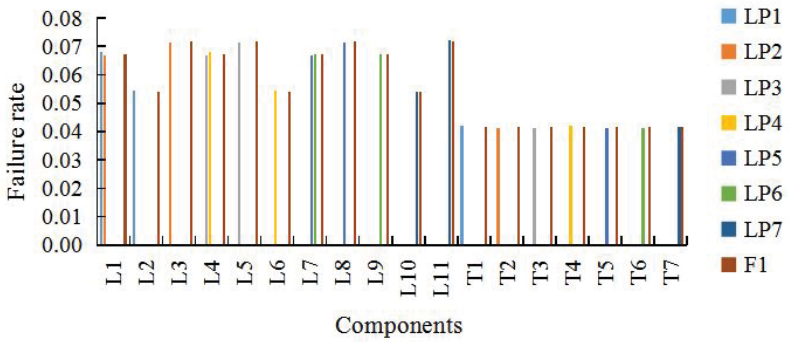


Fig. 9. Possible failure rate of components at each load point or system failure of F1

In FSNPSSs, the output pulse varies with different input pulse. By comparing the relationship between input pulse and output pulse, the probability of simultaneous failure of load point and component is counted. Then the probabilities of possible failure of each component can be calculated according to the Bayesian conditional probability formula, and the influence of components on load and feeder reliability can be analyzed by fault probability. Fig. 9 shows the possible failure rate of each component in each load point or system failure in F1.

From the Fig. 9, we can see that when the feeder F1 is failure, the failure rates of components L3, L5, L8 and L11 are higher relative to the others components. So, in order to reduce the failure rate of feeder F1, it is necessary to give priority to improving the reliability parameters of components L3, L5, L8 and L11.

7 Conclusion

(1)This method proposed in this paper achieves the accumulation of pulse values by introducing self-synapses into the FSNPs. Combined with the fault information transmission principle of distribution network, the system reliability index is accumulated. A reliability assessment method of distribution network based on FSNPSSs is proposed.

(2) The method has the characteristics that the graphical interface structure is simple and clear, and the states of the load point and the feeder line in the component failure can be obtained in parallel. In addition, since the FSNPSSs can obtain the output state of the corresponding system under different input states. The calculation of reliability index and component impact analysis on system can be performed simultaneously.

(3) Because this method converts the transmission relationship of fault information into the process of neuron excitation on the synapse, it has strong applicability. When the component connection mode is changing, such as absence of the fuse or the isolating switch etc. The system reliability also can be calculated by changing the synaptic connection mode or inputting pulse sequence.

(4) Compared with the original FSNPs, the complexity of FSNPSSs programming has not increased much, but its operation efficiency and function have been greatly improved. So it can obtain a better programming experience.

(5) Compared with the traditional methods and Bayesian network, the method proposed by this paper can not only calculating all reliability indexes but also synchronous obtaining the importance of components in the system.

Acknowledgment This work was supported by a grant from Sichuan Provincial Department of Science and Technology (No. 2019122).

References

1. Heydt, G.T., Graf, T.J.: Distribution system reliability evaluation using enhanced samples in a monte carlo approach. IEEE Transactions on Power Systems 25(4), 2006–2008 (2010)
2. Hou, K., Jia, H., Xu, X., Liu, Z., Jiang, Y.: A continuous time markov chain based sequential analytical approach for composite power system reliability assessment. IEEE Transactions on Power Systems 31(1), 738–748 (2015)
3. Yu, D.C., Nguyen, T.C., Haddawy, P.: Bayesian network model for reliability assessment of power systems. IEEE Transactions on Power Systems 14(2), 426–432 (1999)

4. Zhu, Y., Huo, L., Zhang, L., Yan, W.: Bayesian network based time-sequence simulation for power system reliability assessment. In: Mexican International Conference on Artificial Intelligence (2008)
5. Păun, G.: Computing with membranes. Journal of Computer System Sciences 61(1), 108–143 (2000)
6. Ionescu, M., Yokomori, T.: Spiking Neural P Systems (2006)
7. Păun, G., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing (2010)
8. Song, T., Pan, L., Wu, T., Zheng, P., Wong, M.D., Rodríguez-Patón, A.: Spiking neural p systems with learning functions. IEEE transactions on nanobioscience 18(2), 176–190 (2019)
9. Gutirrez-Naranjo, M.A., Prez-Jimnez, M.J.: Hebbian learning from spiking neural p systems view. In: Membrane Computing - 9th International Workshop, WMC 2008, Edinburgh, UK, July 28-31, 2008, Revised Selected and Invited Papers (2008)
10. Cabarle, F.G.C., Adorna, H.N., Pérez-Jiménez, M.J., Song, T.: Spiking neural p systems with structural plasticity. Neural Computing and Applications 26(8), 1905–1917 (2015)
11. Fgc, C., Adorna, H.N., Jiang, M., Zeng, X.: Spiking neural p systems with scheduled synapses PP(99), 1–1 (2017)
12. Pan, L., Păun, G., Pérez-Jiménez, M.J.: Spiking neural p systems with neuron division and budding. Science China Information Sciences 54(8), 1596 (2011)
13. Hong, P., Wang, J., Prez-Jimnez, M.J., Hao, W., Jie, S., Tao, W.: Fuzzy reasoning spiking neural P system for fault diagnosis. Information Sciences 235(6), 106–116 (2013)
14. Wang, J., Shi, P., Peng, H., Prez-Jimnez, M.J., Wang, T.: Weighted fuzzy spiking neural P systems. IEEE Transactions on Fuzzy Systems 21(2), 209–220 (2013)
15. Min, T.U., Wang, J., Hong, P., Peng, S.: Application of adaptive fuzzy spiking neural P systems in fault diagnosis of power systems. Chinese Journal of Electronics 23(1), 87–92 (2014)
16. Zhao, J.B., Tao, W., Zhang, G., He, Z., PErez-JimEnez, M.J.: Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems. IEEE Transactions on Power Systems 30(3), 1182–1194 (2015)
17. Wang, J., Hong, P., Min, T.U., Mario, P.J.J., Peng, S.: A fault diagnosis method of power systems based on an improved adaptive fuzzy spiking neural P systems and PSO algorithms. Chinese Journal of Electronics 25(2), 320–327 (2016)
18. Xiong, G., Shi, D., Lin, Z., Duan, X.: A new approach to fault diagnosis of power systems using fuzzy reasoning spiking neural P systems. Mathematical Problems in Engineering 2013(1), 211–244 (2013)
19. Yin, L., Zheng, R., Ke, W., He, Q., Zhang, Y., Li, J., Wang, B., Mi, Z., Long, Y.s., Rasch, M.J., et al.: Autapses enhance bursting and coincidence detection in neocortical pyramidal cells. Nature communications 9(1), 4890 (2018)
20. Bekkers, J.M.: Synaptic transmission: excitatory autapses find a function? Current Biology 19(7), R296–R298 (2009)
21. Allan, R.N., Billinton, R., Sjarief, I., Goel, L., So, K.S.: A reliability test system for educational purposes–basic distribution system data and results. IEEE Transactions on Power Systems 6(2), 813–820 (1991)
22. Li, W.: Risk assessment of power systems : models, methods, and applications 36(2), 179–180 (2014)

Parallel Contextual Array Insertion Deletion P Systems and Siromoney Matrix Grammars

S. Jayasankar[(A)] D. Gnanaraj Thomas[(B)]
S. James Immanuel[(B)] Meenakshi Paramasivan[(C)]
T. Robinson[(D)] Atulya K. Nagar[(E)]

[(A)]Department of Mathematics, Ramakrishna Mission Vivekananda College, Mylapore, Chennai - 600004, India
ksjayjay@gmail.com

[(B)]Department of Science and Humanities (Mathematics Division), Saveetha School of Engineering, Chennai - 602105, India
dgthomasmcc@yahoo.com , james_imch@yahoo.co.in

[(C)]Institut für Informatik, Universität Leipzig, D-04009 Leipzig, Germany
meena_maths@yahoo.com

[(D)]Department of Mathematics, Madras Christian College, Tambaram, Chennai - 600059, India
robinson@mcc.edu.in

[(E)]Department of Mathematics and Computer Science, Liverpool Hope University, Liverpool, United Kingdom
nagara@hope.ac.uk

Abstract
A variant of P systems named as parallel contextual array insertion deletion P system was introduced and some of its properties were studied in [14]. The family of array languages generated by this variant $PCAIDPS$ includes families of array languages like recognizable picture languages (REC) and context-sensitive matrix languages (CSML). In this paper we show that another interesting family $\mathfrak{L}(CF : RIR)$ of Siromoney matrix languages [12] is included in the family of array languages generated by $PCAIDPS$ with two membranes.

1. Introduction

A P system or a membrane system introduced by Gh. Paun [10] evolves in parallel. A computation starts from an initial configuration of a system, defined by a membrane structure with objects and evolution rules in each membrane and terminates when no further rule can be applied. Various types of P systems have been introduced in the literature and their properties,

computing power, normal forms and basic decision problems have been studied [10, 11]. In [9] the contextual way of handling string objects in P systems has been considered and the contextual P systems are found to be more powerful. In [1], a P system model, called contextual array P system with array objects and array contextual rules has been introduced. An interesting model of array P systems using contextual array grammars [5] with an application of Kolam patterns has been considered in [4]. In [6], another P system model namely, external and internal parallel contextual array P systems have been defined and examined. A new variant of P system model called parallel contextual array insertion deletion P system (PCAIDPS) has been studied in [14] based on array insertion and deletion operations and parallel contextual array insertion deletion grammar [15]. It is proved that the family of languages generated by PCAIDPS includes families of array languages like recognizable picture languages (REC) [2, 3] and context-sensitive matrix languages CSML [12]. For an application of insertion and deletion operations in natural computing, we refer to [7] and in P systems, we refer to [8].

Insertion Deletion System has the generative power equal to the class of recursive enumerable string languages [16]. The primary motivation for studying PCAIDPS is not only to develop the 2D counterpart of Insertion Deletion System available for one dimensional string languages but also to find the status of $\mathfrak{L}(PCAIDPS)$ among the families of 2D languages available in the literature. As far as our knowledge goes, there is no definite hierarchy for the families of 2D languages like Chomsky hierarchy for the classes of string languages. Hence, we are motivated to show that PCAIDPS has more generative power than any other class of 2D grammars generating languages having intersection with REC and CSML. One such family is $\mathfrak{L}(CF : RIR)$ of Subramanian et al [13] who have considered the Siromoney matrix grammar (SMG) [12] and examined the notion of attaching indices to nonterminals in the vertical derivations. The system (CF:RIR)SMG has greater generative power compared to context free matrix languages CFML [12] with interesting applications in the studies of tilings, polyominoes, noisy patterns and parquet deformations. In this paper we prove $\mathfrak{L}(PCAIDPS_2)$ includes $\mathfrak{L}(CF : RIR)$ [13].

2. Preliminaries

In this section, we recall some notions of parallel contextual array insertion deletion P systems and give an example. For further details of the P system we can refer to [14].

Let V be a finite alphabet, V^*, the set of words over V including the empty word λ. $V^+ = V^* - \{\lambda\}$. For $w \in V^*$ and $a \in V$, $|w|_a$ denotes the number of occurrences of a in w. An array consists of finitely many symbols from V that are arranged as rows and columns in some particular order and is written in the form, $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$ or in short $A = [a_{ij}]_{m\times n}$, for all $a_{ij} \in V$, $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. The set of all arrays over V is denoted by V^{**} which also includes the empty array Λ (zero rows and zero columns). $V^{++} = V^{**} - \{\Lambda\}$. The column

concatenation of $A = \begin{bmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} \end{bmatrix}$ and $B = \begin{bmatrix} b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nq} \end{bmatrix}$, defined only when $m = n$, is given by $A \bigoplus B = \begin{bmatrix} a_{11} & \cdots & a_{1p} & b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} & b_{n1} & \cdots & b_{nq} \end{bmatrix}$. As $1 \times n$-dimensional arrays can be easily interpreted as words of length n (and vice versa), we will then write their column catenation by juxtaposition (as usual). Similarly, the row concatenation of A and B, defined only when $p = q$, is given by $A \bigoplus\!\!\!\!- B = \begin{bmatrix} a_{11} & \cdots & a_{1p} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mp} \\ b_{11} & \cdots & b_{1q} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nq} \end{bmatrix}$. The empty array acts as the identity for column and row catenation of arrays of arbitrary dimensions.

Definition 2.1 *Let V be a finite alphabet. A column array context over V is of the form, $c = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \in V^{**}$, u_1, u_2 are of size $1 \times p$, $p \geq 1$.*

A row array context over V is of the form, $r = \begin{bmatrix} u_1 & u_2 \end{bmatrix} \in V^{**}$, u_1, u_2 are of size $p \times 1$, $p \geq 1$.

Definition 2.2 *The parallel column contextual insertion (deletion) operation is defined as follows: Let V be an alphabet, C be a finite subset of V^{**} whose elements are the column array contexts and $\varphi_c^i(\varphi_c^d) : V^{**} \times V^{**} \to 2^C$ be a choice mapping. We define $\varphi_c^i(\varphi_c^d) : V^{**} \times V^{**} \to 2^{V^{**}}$ such that, for arrays $A = \begin{bmatrix} a_{1j} & \cdots & a_{1(k-1)} \\ \vdots & \ddots & \vdots \\ a_{mj} & \cdots & a_{m(k-1)} \end{bmatrix}, B = \begin{bmatrix} a_{1k} & \cdots & a_{1(l-1)} \\ \vdots & \ddots & \vdots \\ a_{mk} & \cdots & a_{m(l-1)} \end{bmatrix}, j < k < l, a_{ij} \in V$, $\left(B = \begin{bmatrix} a_{1(k-p)} & \cdots & a_{1(l-1)} \\ \vdots & \ddots & \vdots \\ a_{m(k-p)} & \cdots & a_{m(l-1)} \end{bmatrix} \right)$, $I_c \in \varphi_c^i(A,B)(\varphi_c^d(A,B)), I_c(D_c) = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}$, if $c_i = \begin{bmatrix} u_i \\ u_{i+1} \end{bmatrix} \in \varphi_c^i \left(\begin{smallmatrix} a_{ij} & \cdots & a_{i(k-1)} \\ a_{(i+1)j} & \cdots & a_{(i+1)(k-1)} \end{smallmatrix}, \begin{smallmatrix} a_{ik} & \cdots & a_{i(l-1)} \\ a_{(i+1)k} & \cdots & a_{(i+1)(l-1)} \end{smallmatrix} \right) \left(\varphi_c^d \left(\begin{smallmatrix} a_{ij} & \cdots & a_{i(k-1)} \\ a_{(i+1)j} & \cdots & a_{(i+1)(k-1)} \end{smallmatrix}, \begin{smallmatrix} a_{ik} & \cdots & a_{i(l-1)} \\ a_{(i+1)k} & \cdots & a_{(i+1)(l-1)} \end{smallmatrix} \right) \right)$, $c_i \in C, 1 \leq i \leq m-1$, not all need to be distinct.*

Given an array $X = [a_{ij}]_{m \times n}, a_{ij} \in V$ $X = X_1 \bigoplus A \bigoplus B \bigoplus X_2$ ($X = X_1 \bigoplus A \bigoplus D_c \bigoplus B \bigoplus X_2$), $X_1 = \begin{bmatrix} a_{11} & \cdots & a_{1(j-1)} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{m(j-1)} \end{bmatrix}, A = \begin{bmatrix} a_{1j} & \cdots & a_{1(k-1)} \\ \vdots & \ddots & \vdots \\ a_{mj} & \cdots & a_{m(k-1)} \end{bmatrix}, B = \begin{bmatrix} a_{1k} & \cdots & a_{1(l-1)} \\ \vdots & \ddots & \vdots \\ a_{mk} & \cdots & a_{m(l-1)} \end{bmatrix}, X_2 = \begin{bmatrix} a_{1l} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{ml} & \cdots & a_{mn} \end{bmatrix}$, $1 \leq j \leq k < l \leq n+1$ (or) $1 \leq j < k \leq l \leq n+1$, we write $X \Rightarrow^{col_i(col_d)} Y$ if $Y = X_1 \bigoplus A \bigoplus I_c \bigoplus B \bigoplus X_2$ ($Y = X_1 \bigoplus A \bigoplus B \bigoplus X_2$), such that $I_c \in \varphi_c^i(A,B)$ ($D_c \in \varphi_c^d(A,B)$). $I_c(D_c)$ is called as the inserted (deleted) column context. We say that Y is obtained from X by parallel column contextual insertion (deletion) operation. The following 4 special cases for $X = X_1 \bigoplus A \bigoplus B \bigoplus X_2$ are also considered,

1. *For $j = 1$ we have $X_1 = \Lambda$.*
2. *For $j = k$, we have $A = \Lambda$. If $j = k = 1$, then $X_1 = \Lambda$ and $A = \Lambda$.*

3. *For $k = l$ ($For\ k + p = l$), we have $B = \Lambda$.*
4. *For $l = n + 1$, we have $X_2 = \Lambda$. If $k = l = n + 1$ ($If\ \ (k + p) = l = n + 1$), then $B = \Lambda$ and $X_2 = \Lambda$.*

The case $j = k = l$ is not considered for parallel column contextual insertion (deletion) operation.

Similarly, we can define parallel row contextual insertion (deletion) operation by inserting (deleting) row context $\mathbf{I_r}(\mathbf{D_r})$ in between two sub-arrays A and B with the help of row operation $\ominus$ and set of row array contexts R. We have $X \Rightarrow^{row_i(row_d)} Y$ if $X = X_1 \ominus A \ominus B \ominus X_2$ $(X_1 \ominus A \ominus D_r \ominus B \ominus X_2)$ and $Y = X_1 \ominus A \ominus I_r \ominus B \ominus X_2$ $(X_1 \ominus A \ominus B \ominus X_2)$.

Definition 2.3 *A parallel contextual array insertion deletion P system with h membranes ($PCAIDPS_h$) is a construct,*

$$\prod = (V, T, \mu, C, R, (M_1, I_1, D_1), \ldots, (M_h, I_h, D_h), \varphi_c^i, \varphi_r^i, \varphi_c^d, \varphi_r^d, i_0)$$

where,

– *V is the finite nonempty set of symbols called alphabet;*

– *$T \subseteq V$ is the output alphabet;*

– *μ is the membrane structure with h membranes or regions;*

– *C is the finite subset of V^{**} called set of column array contexts;*

– *R is the finite subset of V^{**} called set of row array contexts;*

– *M_i is the finite set of arrays over V called as axioms associated with the region $\mu_{\mathbf{i}}$ of μ;*

– *$\varphi_c^i : V^{**} \times V^{**} \to 2^C$ is the choice mapping performing parallel column contextual insertion operations;*

– *$\varphi_r^i : V^{**} \times V^{**} \to 2^R$ is the choice mapping performing parallel row contextual insertion operations;*

– *$\varphi_c^d : V^{**} \times V^{**} \to 2^C$ is the choice mapping performing parallel column contextual deletion operations;*

– *$\varphi_r^d : V^{**} \times V^{**} \to 2^R$ is the choice mapping performing parallel row contextual deletion operations;*

– *$I_i = \emptyset$ (or) $\left\{ \left(\left\{ \varphi_c^i(A_i, B_i) = \left[\begin{smallmatrix} u_i \\ u_{i+1} \end{smallmatrix} \right] \middle| i = 1, 2, \ldots, m-1 \right\}, \alpha \right) \right\}$ where*

$A_i = \left[\begin{smallmatrix} a_{ij} & \cdots & a_{i(k-1)} \\ a_{(i+1)j} & \cdots & a_{(i+1)(k-1)} \end{smallmatrix}\right], B_i = \left[\begin{smallmatrix} a_{ik} & \cdots & a_{i(l-1)} \\ a_{(i+1)k} & \cdots & a_{(i+1)(l-1)} \end{smallmatrix}\right]$, $1 \leq j \leq k < l \leq n+1$ *(or)* $1 \leq j < k \leq l \leq n+1$, $\alpha \in \{here, out, in_t\}$, u_i *and* u_{i+1} *are of size* $1 \times p$ *with* $p \geq 1$.

(or)

$\left\{\left(\left\{\varphi_r^i(C_i, E_i) = \left[\begin{smallmatrix} u_i & u_{i+1} \end{smallmatrix}\right] \middle| i = 1, 2, \ldots, n-1\right\}, \alpha\right)\right\}$ *where*

$C_i = \begin{bmatrix} a_{ji} & a_{j(i+1)} \\ \vdots & \vdots \\ a_{(k-1)i} & a_{(k-1)(i+1)} \end{bmatrix}, E_i = \begin{bmatrix} a_{ki} & a_{k(i+1)} \\ \vdots & \vdots \\ a_{(l-1)i} & a_{(l-1)(i+1)} \end{bmatrix}$, $1 \leq j \leq k < l \leq m+1$ *(or)* $1 \leq j < k \leq l \leq m+1$, $\alpha \in \{here, out, in_t\}$, u_i *and* u_{i+1} *are of size* $p \times 1$ *with* $p \geq 1$.

$D_i = \emptyset$ *(or)* $\left\{\left(\left\{\varphi_c^d(A_i, B_i) = \left[\begin{smallmatrix} u_i \\ u_{i+1} \end{smallmatrix}\right] \middle| i = 1, 2, \ldots, m-1\right\}, \alpha\right)\right\}$ *where*

$A_i = \left[\begin{smallmatrix} a_{ij} & \cdots & a_{i(k-1)} \\ a_{(i+1)j} & \cdots & a_{(i+1)(k-1)} \end{smallmatrix}\right], B_i = \left[\begin{smallmatrix} a_{i(k+p)} & \cdots & a_{i(l-1)} \\ a_{(i+1)(k+p)} & \cdots & a_{(i+1)(l-1)} \end{smallmatrix}\right]$, $1 \leq j \leq k < l \leq n+1$, $\alpha \in \{here, out, in_t\}$, u_i *and* u_{i+1} *are of size* $1 \times p$ *with* $p \geq 1$.

(or)

$\left\{\left(\left\{\varphi_r^d(C_i, E_i) = \left[\begin{smallmatrix} u_i & u_{i+1} \end{smallmatrix}\right] \middle| i = 1, 2, \ldots, n-1\right\}, \alpha\right)\right\}$ *where*

$C_i = \begin{bmatrix} a_{ji} & a_{j(i+1)} \\ \vdots & \vdots \\ a_{(k-1)i} & a_{(k-1)(i+1)} \end{bmatrix}, E_i = \begin{bmatrix} a_{(k+p)i} & a_{(k+p)(i+1)} \\ \vdots & \vdots \\ a_{(l-1)i} & a_{(l-1)(i+1)} \end{bmatrix}$, $1 \leq j \leq k < l \leq m+1$, $\alpha \in \{here, out, in_t\}$, u_i *and* u_{i+1} *are of size* $p \times 1$ *with* $p \geq 1$.
$-$ i_0 *is the output membrane.*

The array language generated by $\prod$ is denoted by $L(\prod)$ and the family of array languages generated by $PCAIDPS$ with h membranes is denoted by $\mathfrak{L}(PCAIDPS_h)$.

Example 2.4 *Consider a P system given by* $PCAIDPS_2$

$$\prod = (\mathbf{V}, \mathbf{T}, \mu, \mathbf{C}, \mathbf{R}, (\mathbf{M_1}, \mathbf{I_1}, \mathbf{D_1}), (\mathbf{M_2}, \mathbf{I_2}, \mathbf{D_2}), \varphi_{\mathbf{c}}^{\mathbf{i}}, \varphi_{\mathbf{r}}^{\mathbf{i}}, \varphi_{\mathbf{c}}^{\mathbf{d}}, \varphi_{\mathbf{r}}^{\mathbf{d}}, \mathbf{1}), where$$

$\mathbf{V} = \{\bullet, X, Y\}$; $\mathbf{T} = \{\bullet, X\}$; $\mu = [_1[_2]_2]_1$;
$\mathbf{C} = \left\{\begin{bmatrix} \bullet \\ X \end{bmatrix}, \begin{bmatrix} X \\ \bullet \end{bmatrix}, \begin{bmatrix} \bullet \\ \bullet \end{bmatrix}, \begin{bmatrix} Y \\ Y \end{bmatrix}\right\}$;
$\mathbf{R} = \left\{\begin{bmatrix} X & Y \\ Y & Y \end{bmatrix}, \begin{bmatrix} Y & \bullet \\ Y & Y \end{bmatrix}, \begin{bmatrix} \bullet & X \\ Y & Y \end{bmatrix}, \begin{bmatrix} \bullet & \bullet \\ Y & Y \end{bmatrix}, \begin{bmatrix} Y & Y \\ X & Y \end{bmatrix}, \begin{bmatrix} Y & Y \\ Y & \bullet \end{bmatrix}, \begin{bmatrix} Y & Y \\ \bullet & X \end{bmatrix}, \begin{bmatrix} Y & Y \\ \bullet & \bullet \end{bmatrix}, \begin{bmatrix} Y & Y \end{bmatrix}\right\}$; $\mathbf{M_1} = \emptyset$;
$\mathbf{M_2} = \begin{bmatrix} X & \bullet & X \\ X & X & X \\ X & \bullet & X \end{bmatrix}$; $\mathbf{I_1} = \emptyset$; $\mathbf{D_1} = \emptyset$;
$\mathbf{I_2} = \left\{\left(\left\{\varphi_{\mathbf{c}}^{\mathbf{i}} \begin{bmatrix} X & \bullet \\ X & , X \end{bmatrix} = \begin{bmatrix} \lambda & \lambda \\ \lambda & \lambda \end{bmatrix}, \varphi_{\mathbf{c}}^{\mathbf{i}} \begin{bmatrix} X & \bullet \\ X & , \bullet \end{bmatrix} = \begin{bmatrix} \lambda & \lambda \\ \lambda & \lambda \end{bmatrix}, \varphi_{\mathbf{c}}^{\mathbf{i}} \begin{bmatrix} X & X \\ X & , \bullet \end{bmatrix} =\right.\right.$

$$\left.\begin{bmatrix}\lambda\ \lambda\\ \lambda\ \lambda\end{bmatrix}\right\}, out\Bigg),$$

$$\left(\left\{\varphi_{\mathbf{c}}^{\mathbf{i}}\begin{bmatrix}X & \bullet\\ X\ , & X\end{bmatrix}=\begin{bmatrix}\bullet\\ X\end{bmatrix}, \varphi_{\mathbf{c}}^{\mathbf{i}}\begin{bmatrix}X & \bullet\\ X\ , & \bullet\end{bmatrix}=\begin{bmatrix}\bullet\\ \bullet\end{bmatrix}, \varphi_{\mathbf{c}}^{\mathbf{i}}\begin{bmatrix}X & X\\ X\ , & \bullet\end{bmatrix}=\begin{bmatrix}X\\ \bullet\end{bmatrix}\right\}, here\right),$$

$$\left(\left\{\varphi_{\mathbf{c}}^{\mathbf{i}}\begin{bmatrix}X & \bullet\\ X\ , & X\end{bmatrix}=\begin{bmatrix}Y\\ Y\end{bmatrix}, \varphi_{\mathbf{c}}^{\mathbf{i}}\begin{bmatrix}X & \bullet\\ X\ , & \bullet\end{bmatrix}=\begin{bmatrix}Y\\ Y\end{bmatrix}, \varphi_{\mathbf{c}}^{\mathbf{i}}\begin{bmatrix}X & X\\ X\ , & \bullet\end{bmatrix}=\begin{bmatrix}Y\\ Y\end{bmatrix}\right\}, here\right),$$

$$\Bigg(\Bigg\{\varphi_{\mathbf{r}}^{\mathbf{i}}\left[X\ Y\ ,\ X\ Y\right]=\begin{bmatrix}X\ Y\\ Y\ Y\end{bmatrix}, \varphi_{\mathbf{r}}^{\mathbf{i}}\left[Y\ \bullet\ ,Y\ X\right]=\begin{bmatrix}Y\ \bullet\\ Y\ Y\end{bmatrix}, \varphi_{\mathbf{r}}^{\mathbf{i}}\left[\bullet\ X\ ,\ X\ X\right]=$$

$$\begin{bmatrix}\bullet\ X\\ Y\ Y\end{bmatrix}, \varphi_{\mathbf{r}}^{\mathbf{i}}\left[\bullet\ \bullet\ ,\ X\ X\right]=\left.\begin{bmatrix}\bullet\ \bullet\\ Y\ Y\end{bmatrix}\right\}, here\Bigg),$$

$$\Bigg(\Bigg\{\varphi_{\mathbf{r}}^{\mathbf{i}}\left[X\ Y\ ,\ X\ Y\right]=\begin{bmatrix}Y\ Y\\ X\ Y\end{bmatrix}, \varphi_{\mathbf{r}}^{\mathbf{i}}\left[Y\ X\ ,Y\ \bullet\right]=\begin{bmatrix}Y\ Y\\ Y\ \bullet\end{bmatrix}, \varphi_{\mathbf{r}}^{\mathbf{i}}\left[X\ X\ ,\bullet\ X\right]=\begin{bmatrix}Y\ Y\\ \bullet\ X\end{bmatrix},$$

$$\varphi_{\mathbf{r}}^{\mathbf{i}}\left[X\ X\ ,\ \bullet\ \bullet\right]=\left.\begin{bmatrix}Y\ Y\\ \bullet\ \bullet\end{bmatrix}\right\}, here\Bigg)\Bigg\};$$

$$\mathbf{D_2}=\Bigg\{\Bigg(\Bigg(\Bigg\{\varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix}X & \bullet\\ X\ , & \bullet\end{bmatrix}=\begin{bmatrix}Y\\ Y\end{bmatrix}, \varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix}X & \bullet\\ Y\ , & Y\end{bmatrix}=\begin{bmatrix}Y\\ Y\end{bmatrix}, \varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix}Y & Y\\ X\ , & X\end{bmatrix}=\begin{bmatrix}Y\\ Y\end{bmatrix},$$

$$\varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix}X & X\\ Y\ , & Y\end{bmatrix}=\begin{bmatrix}Y\\ Y\end{bmatrix}, \varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix}Y & Y\\ X\ , & \bullet\end{bmatrix}=\left.\begin{bmatrix}Y\\ Y\end{bmatrix}\right\}, here\Bigg),$$

$$\Bigg(\Bigg\{\varphi_{\mathbf{r}}^{\mathbf{d}}\left[X\ \bullet\ ,\ X\ X\right]=\left[Y\ Y\right], \varphi_{\mathbf{r}}^{\mathbf{d}}\left[\bullet\ \bullet\ ,X\ X\right]=\left[Y\ Y\right],$$

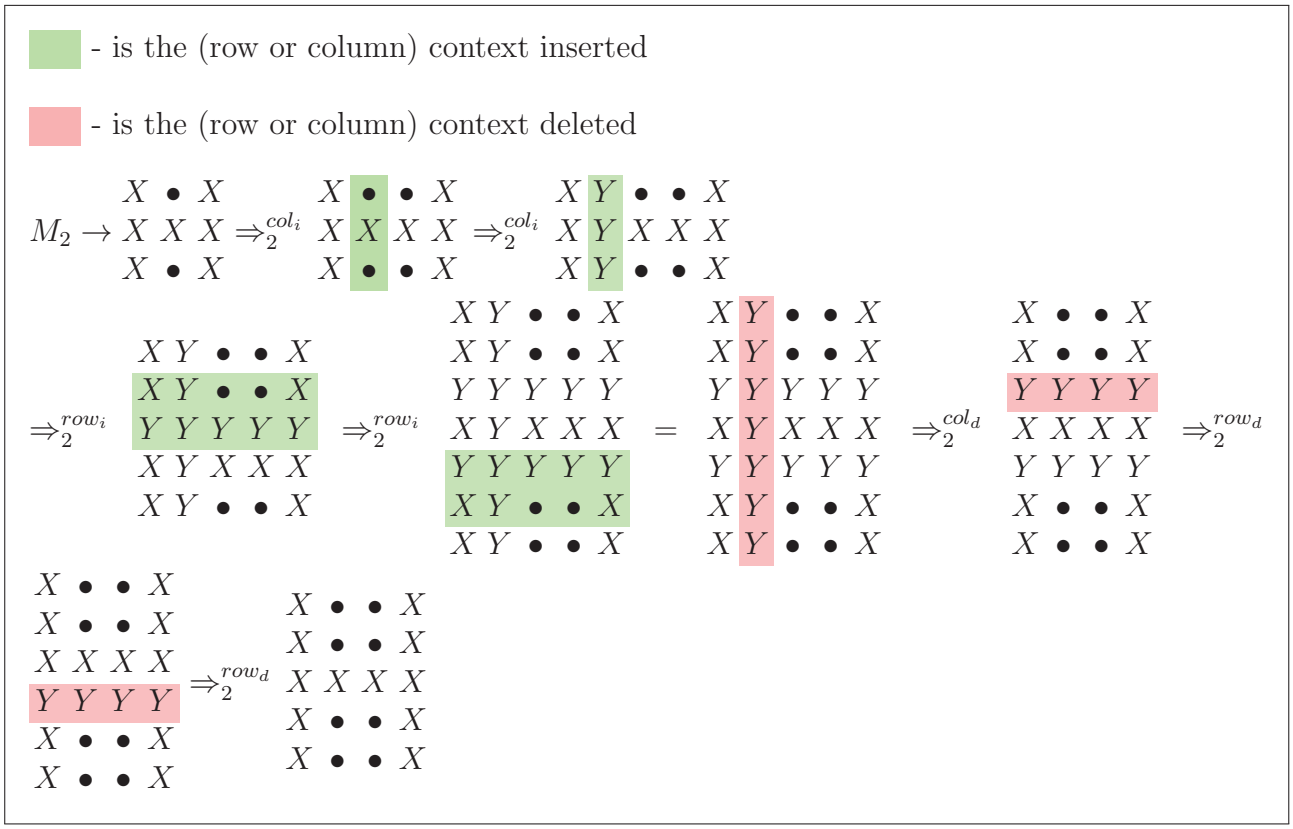
$$\varphi_{\mathbf{r}}^{\mathbf{d}}\left[\bullet\ X\ ,\ X\ X\right]=\left.\left[Y\ Y\right]\right\}, here\Bigg),$$

$$\Bigg(\Bigg\{\varphi_{\mathbf{r}}^{\mathbf{d}}\left[X\ X\ ,\ X\ \bullet\right]=\left[Y\ Y\right], \varphi_{\mathbf{r}}^{\mathbf{d}}\left[X\ X\ ,\bullet\ \bullet\right]=\left[Y\ Y\right],$$

$$\varphi_{\mathbf{r}}^{\mathbf{d}}\left[X\ X\ ,\ \bullet\ X\right]=\left.\left[Y\ Y\right]\right\}, here\Bigg)\Bigg)\Bigg\};$$

Clearly $L(\prod)$ is $\left\{\begin{matrix}X\ \bullet\ X\\ X\ X\ X,\\ X\ \bullet\ X\end{matrix}\ \begin{matrix}X\ \bullet\ \bullet\ X\\ X\ \bullet\ \bullet\ X\\ X\ X\ X\ X\\ X\ \bullet\ \bullet\ X\\ X\ \bullet\ \bullet\ X\end{matrix}, \cdots\right\}$, the set of tokens of H of $X's$ with the horizontal row of $X's$ exactly in the middle. A sample computation is shown below.

In the picture given below, $A \Rightarrow_s^{\alpha_k} B$ means A is transformed into B by k ($k = i$ or d) rewriting rules of α (α = row or col) rewriting in the membrane s using the appropriate rules of the membrane s. Also, $M \rightarrow A$ means A is in the membrane M.



□

3. Main Result

In this section, we recall the definition of the Siromoney matrix grammar $(CF : RIR)$SMG and an example [13]. We show that $\mathfrak{L}(CF : RIR) \subseteq \mathfrak{L}(PCAIDPS_2)$

Definition 3.1 *A right-linear indexed right-linear(RIR) grammar G is a five-tuple (N,T,F,P,S) where N, T, F are finite, pairwise disjoint sets of nonterminals, terminals and indices respectively; P consists of two types of productions, namely*

1. *right-linear productions of the form* $A \rightarrow xBf$, $A \in N$, $x \in T \cup \{\lambda\}$, $f \in F \cup \{\lambda\}$, $B \in N \cup \{\lambda\}$, *not all* λ, *and*
2. *indexed productions of the form* $Af \rightarrow xB$, $A \in N$, $B \in N \cup \{\lambda\}$, $x \in T \cup \{\lambda\}$, $f \in F$, *and* $S \in N$ *is the start symbol.*

Derivations are as in a Chomskian right-linear grammar except that a nonterminal A is replaced using a production of the form (1) or an index f is consumed by a production of the

form (2).

Definition 3.2 *A (CF:RIR)SMG $G = (G_1, G_2)$ is defined as follows: $G_1 = (N, I, P, S)$ is a context-free (CF) grammar where N is a finite set of nonterminals, I is a finite set of k intermediate symbols $S_1, S_2, \cdots, S_k$ with $N \cap I = \emptyset$, P is a finite set of production rules and $S \in N$ is the start symbol of G_1. G_1 is called the horizontal grammar.*

$G_2 = \{G_{21}, G_{22}, G_{23}, \cdots, G_{2k}\}$, where each $G_{2i}, i = 1, \cdots, k$ is called a vertical grammar where $G_{2i} = (N_i, T, F_i, P_i, S_i)$ is an RIR grammar with N_i, F_i being distinct from N_j, F_j for $i \neq j (1 \leq i, j \leq k)$. The intermediate symbol S_i of G_1 is in N_i and is the start symbol of G_{2i}.

The derivations are as follows: G_i generates finite strings of intermediates as in a Chomskian CF grammar. The vertical derivation starts with a string of intermediates generated by G_2 and proceeds in parallel as in Siromoney matrix grammar [12]. Rules of the same type - either CF productions $A \to \alpha$ or indexed productions $Af \to \alpha$ with the length of α being the same $(3, 2, 1$ or $0)$, are applied in the vertical derivation. At every step of the vertical derivation, a row $A_1, A_2, \cdots, A_m$ of nonterminals is expanded or a string of indexed nonterminals $\begin{matrix} A_1 & \cdots & A_m \\ f_1 & \cdots & f_m \end{matrix}$ is rewritten, consuming the indices. A derivation successfully ends on generating an array M over T by rewriting a row of nonterminals with(without) indices by rules of the form $Af \to \alpha$ $(A \to \alpha)$. The array language generated by the grammar G is given by

$$L(G) = \{M \in T^{++} / S \Rightarrow^*_{G_1} \alpha \text{ and } \alpha \Rightarrow^*_{G_2} M, \alpha \in I^{++}\}.$$

Example 3.3 *The $(R : RIR)SMG$ which generates the tokens H of $x's$ with the horizontal row of $x's$ exactly in the middle is as follows:*

$$G = (G_1, G_2)$$

where

$$G_1 = (\{S, A\}, \{S_1, S_2\}, \{S \to S_1A, A \to S_2A, A \to S_2S_1\}, S)$$

generating strings of intermediates $S_1S_2^nS_1$ for $n \geq 1$ and $G_2 = \langle G_{21}, G_{22} \rangle$
where

$$G_{21} = (\{S_1, A_1, A_2, A_3\}, \{x\}, \{g_1, g_2\}, P_{21}, S_1)$$

with $P_{21} = \{S_1 \to xA_1g_2, A_1 \to A_2g_1, A_1 \to xA_1g_1, A_2g_1 \to xA_3,$
$A_3g_1 \to xA_3, A_3g_2 \to x\}$,

$$G_{22} = (\{S_2, B_1, B_2, B_3, B_4\}, \{\cdot, x\}, \{f_1, f_2\}, P_{22}, S_2)$$

with $P_{22} = \{S_2 \to \cdot B_1f_2, B_1 \to \cdot B_1f_1, S_2 \to \cdot B_2f_2, B_1 \to \cdot B_2f_1,$
$B_2 \to B_3f_1, B_3f_1 \to xB_4, B_4f_1 \to \cdot B_4, B_4f_2 \to \cdot\}$.

A sample vertical derivation is as follows:

$$S_1S_2S_1 \Rightarrow_{G_2} \begin{matrix} x & \cdot & x \\ A_1 & B_2 & A_1 \\ g_2 & f_2 & g_2 \end{matrix} \Rightarrow_{G_2} \begin{matrix} x & \cdot & x \\ A_2 & B_3 & A_2 \\ g_1 & f_1 & g_1 \\ g_2 & f_2 & g_2 \end{matrix} \Rightarrow_{G_2} \begin{matrix} x & \cdot & x \\ x & x & x \\ A_3 & B_4 & A_3 \\ g_2 & f_2 & g_2 \end{matrix} \Rightarrow_{G_2} \begin{matrix} x \cdot x \\ x\ x\ x \\ x \cdot x \end{matrix}$$

Example 2.4 and example 3.3 exhibit the fact that $\mathfrak{L}(R : RIR) \cap \mathfrak{L}(PCAIDPS_2) \neq \emptyset$. We now proceed to prove the main result.

Theorem 3.4 $\mathfrak{L}(CF : RIR) \subset \mathfrak{L}(PCAIDPS_2)$

Proof. This result can be proved by showing that for any grammar generating a language L, a P-system can be constructed to generate L by introducing insertion deletion rules corresponding to the rules of the grammar G.

$\mathfrak{L}(CF : RIR) \subset \mathfrak{L}(PCAIDPS_2)$ can be proved by the following construction:

For every $(CF : RIR)SMG$, with $G = (G_1, G_2)$ with $G_1 = (N, I, P, S)$, $G_2 = \langle G_{21}, \cdots, G_{2k} \rangle$, $G_{2i} = (N_i, T, F_i, P_i, S_i), N_i, F_i$ being distinct from N_j, F_j with $i \neq j$, $i = 1, 2, \cdots, k$ $(1 \leq i, j \leq k)$, we can construct a $PCAIDPS_2$

$$\prod = \left(\mathbf{V}, \mathbf{T}, [_{\mathbf{1}}[_{\mathbf{2}}]_{\mathbf{2}}]_{\mathbf{1}}, \mathbf{C}, \mathbf{R}, (\mathbf{M_1}, \mathbf{I_1}, \mathbf{D_1}), (\mathbf{M_2}, \mathbf{I_2}, \mathbf{D_2}), \varphi^{\mathbf{i}}_{\mathbf{c}}, \varphi^{\mathbf{i}}_{\mathbf{r}}, \varphi^{\mathbf{d}}_{\mathbf{c}}, \varphi^{\mathbf{d}}_{\mathbf{r}}, \mathbf{1}\right)$$

such that $L(\prod) = L(G)$ where $\mathbf{V} = N \cup (\bigcup_{i=1}^{k} N_i) \cup T \cup (\bigcup_{i=1}^{k} F_i) \cup I \cup \{\#\}$;
$\mathbf{T} = T$; $\mathbf{M_1} = \emptyset$; $\mathbf{M_2} = \left\{ \begin{matrix} \# & \# & \# \\ \# & S & \# \end{matrix} \right\}$;

$$\mathbf{C} = \left\{ \begin{matrix} \# \\ \alpha \end{matrix} \middle| S \to \alpha \in P \right\}$$
$$\cup \left\{ \begin{matrix} \# \\ S \end{matrix} \middle| \ S \to \alpha \in P \right\}$$
$$\cup \left\{ \begin{matrix} \# \\ \gamma \end{matrix} \middle| \beta \to \gamma \in P, \beta \in N, \gamma \in (N \cup T)^* \right\}$$
$$\cup \left\{ \begin{matrix} \# \\ \beta \end{matrix} \middle| \beta \to \gamma \in P, \beta \in N, \gamma \in (N \cup T)^* \right\};$$

$$\mathbf{R} = \left\{ \begin{matrix} \# & c \\ \# & C_i \\ \# & f_i \end{matrix} \middle| C \to cC_if_i \in P_i, C \in N_i \right\}$$
$$\cup \left\{ \begin{matrix} \# & c \\ \# & C_i \end{matrix} \middle| C \to cC_i \in P_i, C \in N_i \right\}$$
$$\cup \left\{ \begin{matrix} \# & c \\ \# & \# \end{matrix} \middle| C \to c \in P_i, C \in N_i \right\}$$

$$\cup \left\{ \begin{matrix} c & d \\ C_i & D_j \\ f_i & f_j \end{matrix} \middle| C \to cC_if_i \in P_i, D \to dD_jf_j \in P_j, C \in N_i, D \in N_j \right\}$$
$$\cup \left\{ \begin{matrix} c & d \\ C_i & D_j \end{matrix} \middle| C \to cC_i \in P_i, D \to dD_j \in Pj, C \in N_i, D \in N_j \right\}$$
$$\cup \left\{ \begin{matrix} c & d \\ \# & \# \end{matrix} \middle| C \to c \in P_i, D \to d \in P_j \right\}$$
$$\cup \left\{ \begin{matrix} c & \# \\ C_i & \# \\ f_i & \# \end{matrix} \middle| C \to cC_if_i \in P_i, C \in N_i \right\}$$
$$\cup \left\{ \begin{matrix} c & \# \\ C_i & \# \end{matrix} \middle| C \to cC_i \in P_i, C \in N_i \right\}$$
$$\cup \left\{ \begin{matrix} c & \# \\ \# & \# \end{matrix} \middle| C \to c \in P_i, C \in N_i \right\}$$
$$\cup \left\{ \begin{matrix} c & d \\ C_i & D_j \end{matrix} \middle| Cf_i \to cC_i \in P_i, Df_j \to dD_j \in P_j \right\}$$
$$\cup \left\{ \begin{matrix} \# & c \\ \# & C_i \end{matrix} \middle| Cf_i \to cC_i \in P_i \right\}$$
$$\cup \left\{ \begin{matrix} c & \# \\ C_i & \# \end{matrix} \middle| Cf_i \to cC_i \in P_i \right\}$$
$$\cup \left\{ \begin{matrix} \# & c \\ \# & \# \end{matrix} \middle| Cf_i \to c \in P_i \right\}$$
$$\cup \left\{ \begin{matrix} c & d \\ \# & \# \end{matrix} \middle| Cf_i \to c \in P_i, Df_j \to d \in P_j \right\}$$
$$\cup \left\{ \begin{matrix} c & \# \\ \# & \# \end{matrix} \middle| Cf_i \to c \in P_i \right\}$$
$$\cup \left\{ \begin{matrix} \# & C \end{matrix} \middle| C \to cC_if_i \text{ or } C \to cC_i \text{ or } C \to c \in P_i \right\}$$
$$\cup \left\{ \begin{matrix} C & D \end{matrix} \middle| C \to cC_if_i \text{ or } C \to cC_i \text{ or } C \to c \in P_i, D \to dD_jf_j \text{ or } D \to dD_j \text{ or } D \to d \in P_j \right\}$$
$$\cup \left\{ \begin{matrix} C & \# \end{matrix} \middle| C \to cC_i \text{ or } C \to c \in P_i \right\}$$
$$\cup \left\{ \begin{matrix} C & D \\ f_i & f_j \end{matrix} \middle| Cf_i \to cC_i \text{ or } Cf_i \to c \in P_i, Df_j \to dD_j \in P_j \right\}$$
$$\cup \left\{ \begin{matrix} \# & C \\ \# & f_i \end{matrix} \middle| Cf_i \to cC_i \in P_i \text{ or } Cf_i \to c \in P_i \right\}$$
$$\cup \left\{ \begin{matrix} C & \# \\ f_i & \# \end{matrix} \middle| Cf_i \to cC_i \in P_i \text{ or } Cf_i \to c \in P_i \right\}$$
$$\cup \left\{ \begin{matrix} \# & \# \end{matrix} \right\}.$$

$\mathbf{I_1} = \emptyset$, $\mathbf{D_1} = \emptyset$;

In the membrane 2, three column insertion rules and twenty eight row insertion rules are defined in $\mathbf{I_2}$ based on the rules of G_1 and G_2 respectively of the grammar G (See the Appendix). For every rule $S \to \alpha \in P$ in G_1, column insertion rule(s) to insert the context $\begin{smallmatrix}\# \\ \alpha\end{smallmatrix}$ between $\begin{smallmatrix}\# \\ \#\end{smallmatrix}$ and $\begin{smallmatrix}\# \\ S\end{smallmatrix}$ are defined.

For every rule of the form $\beta \to \gamma \in P$ in G_1 the corresponding column insertion rules are defined to insert the context $\begin{smallmatrix}\# \\ \gamma\end{smallmatrix}$ between $\begin{smallmatrix}\# \\ \#\end{smallmatrix}$ and $\begin{smallmatrix}\# \\ \beta\end{smallmatrix}$ (or) $\begin{smallmatrix}\# \\ A\end{smallmatrix}$ and $\begin{smallmatrix}\# \\ \beta\end{smallmatrix}$.
In membrane 2, three column deletion rules and thirty three row deletion rules are defined in $\mathbf{D_2}$ based on the rules of G_1 and G_2 respectively of the grammar G (See the Appendix). For every rule $S \to \alpha \in P$ in G_1 deletion rule(s) to delete the contexts $\begin{smallmatrix}\# \\ S\end{smallmatrix}$ between $\begin{smallmatrix}\# \\ \alpha\end{smallmatrix}$ and $\begin{smallmatrix}\# \\ \#\end{smallmatrix}$ are defined.

For every rule of the form $\beta \to \gamma \in P$ in G_1 the corresponding column deletion rules are defined to delete the context $\begin{smallmatrix}\# \\ \beta\end{smallmatrix}$ between $\begin{smallmatrix}\# \\ \alpha\end{smallmatrix}$ and $\begin{smallmatrix}\# \\ \#\end{smallmatrix}$ (or) $\begin{smallmatrix}\# \\ \gamma\end{smallmatrix}$ and $\begin{smallmatrix}\# \\ \#\end{smallmatrix}$.

Using these column insertion and column deletion rules of the P-system the same horizontal derivation of the horizontal grammar G_1 of G can be achieved.
It is to be noted that the rules for the horizontal growth in I_2 and D_2 of the P-system are defined in such a manner that the column insertion rules and column deletion rules are applied alternatively.

To simulate the vertical derivation of the grammar G, row insertion and row deletion rules are defined based on the rules of the vertical grammars $G_{2i}(1 \leq i \leq k)$ of G_2 in G.

For the rules of the form $C \to cC_if_i$, $C \to cC_i$, $C \to c$, $Cf_i \to cC_i$ in P_i of G_{2i} the corresponding row insertion and row deletion rules are defined in $\mathbf{I_2}$ and $\mathbf{D_2}$ respectively to replicate the vertical derivation of the vertical grammars G_{2i}.

It is again to be noted that the rules for vertical growth in $\mathbf{I_2}$ and $\mathbf{D_2}$ of the P-system are defined in such a manner that the row insertion and row deletion rules are applied alternatively. A row insertion rule in $\mathbf{I_2}$ is defined in to send the generated picture to the skin membrane which is the output membrane.

The working of the P-system in membrane 2 is as follows: The axiom set consists of the array $\begin{smallmatrix}\# & \# & \# \\ \# & S & \#\end{smallmatrix}$ based on the starting symbol S of any $(CF : RIR)SMG$ G. We consider the rules in $\mathbf{I_2}$ and $\mathbf{D_2}$ to perform the parallel contextual column insertion and column deletion operations and these operations are performed alternatively to simulate the generation of horizontal strings

of intermediates based on G_1 of G. Now we consider the rules in $\mathbf{I_2}$ and $\mathbf{D_2}$ to perform the parallel contextual row insertion operation and row deletion operations. Parallel contextual row insertion and deletion operations are performed alternatively to simulate the vertical generation of the picture based on the G_{2i}. Then using the parallel contextual column deletion rules in $\mathbf{D_2}$ the #'s along the borders of the columns are deleted and finally using the parallel contextual row deletion rules in $\mathbf{D_2}$ the #'s along the borders of the rows are deleted. The resulting arrays belong to $L(G)$. Finally, using the rule $\left(\left\{\varphi_r^i\left[a\ b\ ,\ \lambda\ \lambda\right] = \left\{\lambda\ \lambda \middle| a, b \in T\right\}\right\}, out\right) \in \mathbf{I_2}$, the resultant arrays are sent to membrane 1, which is the output membrane.

□

4. Conclusion

PCAIDPS has greater generative capacity to obtain some known families of two dimensional array languages. So far, we could show that $\mathfrak{L}(PCAIDPS)$ includes REC, CSML and $\mathfrak{L}(CF : RIR)$. We note that these three classes are incomparable but not disjoint. It is worth investigating to form a hierarchy among the families of two dimensional array languages. This paper strives in this direction.

References

[1] K.S. Dersanamiba, and K. Krithivasan: Contextual Array P systems, International Journal of Computer Mathematics, Vol.81, No.8, 955–969 (2004)

[2] D. Giammarresi and A. Restivo: Two- Dimensional Languages, Handbook of formal languages, vol. 3, 215–267 (1997)

[3] D. Giammarresi and A. Restivo: Recognizable Picture Languages, International Journal of Pattern Recognition and Artificial Intelligence 6, 241–256 (1992)

[4] H. Fernau, R. Freund, Markus L. Schmid, K.G. Subramanian, and P. Wielderhold: Contextual array grammars and array P systems, Annals of Mathematics and Artificial Intelligence, Vol. 75, 5–26 (2013)

[5] R. Freund, Gh. Păun, G. Rozenberg: Contextual Array Grammars, Formal Models, Languages and Applications, Series in Machine Perception and Artificial Intelligence, Vol. 66, World Scientific, 112–136, (2007)

[6] S. James Immanuel, D.G. Thomas, T. Robinson and Atulya K Nagar: Parallel Contextual Array P Systems, in the Proceedings of Asian Conference on Membrane Computing - ACMC 2014, IEEE Xplore, 1–9 (2014)

[7] L. Kari and G. Thierrin: Contextual insertions/deletions and computability. Information and Computation, 131(1):47-61 (1996)

[8] S.N. Krishna and R. Rama: Insertion-deletion P systems, Lecture Notes in Computer Science 2340, Springer, 360-370 (2002).

[9] M. Madhu and K. Krithivasan: Contextual P Systems, Fund. Info., 49, 179–189 (2002)

[10] Gh. Păun: Computing with membranes, Journal of Computer and System Sciences, 61, 108–143 (2000)

[11] Gh. Păun, G. Rozenberg and A. Salomaa(Editors): The Oxford Handbook of Membrane Computing, Oxford Univ. Press (2010)

[12] G. Siromoney, R. Siromoney, and K. Krithivasan: Abstract families of matrices and picture languages, Computer Graphics and Image Processing 1, 234–307 (1972)

[13] K.G. Subramanian, L. Revathi and R. Siromoney: Siromoney Array Grammars and Applications, International Journal of Pattern Recognition and Artificial Itelligence, Vol. 3 No.3 & 4 333-351, (1989)

[14] S. James Immanuel, D.G. Thomas, T. Robinson and Atulya K. Nagar: Parallel Contextual Array Insertion Deletion P System, Proceedings of IWCIA 2017, Lecture Notes in Computer Science 10256, Springer, 170-183 (2017)

[15] D.G Thomas, S. James Immanuel, Atulya K. Nagar and T. Robinson: Parallel Contextual Array Insertion Deletion Grammar, Lecture Notes in Computer Science 11255, 28-42 (2018)

[16] S. Verlan: Recent Developments on Insertion Deletion Systems , Computer Science Journal of Maldova, vol.18, no.2(53), 2010

Appendix

The definitions of $\mathbf{I_2}$ and $\mathbf{D_2}$ of theorem 3.4 are given below:

Definition of $\mathbf{I_2}$:

$$\mathbf{I_2} = \left\{ \left(\left\{ \varphi_{\mathbf{c}}^{\mathbf{i}} \begin{bmatrix} \# & \# \\ \# \ , & S \end{bmatrix} = \left\{ \begin{matrix} \# \\ \alpha \end{matrix} \middle| S \to \alpha \in P \right\}, here \right), \right.$$

$$\left(\left\{ \varphi_{\mathbf{c}}^{\mathbf{i}} \begin{bmatrix} \# & \# \\ \# \ , & \beta \end{bmatrix} = \left\{ \begin{matrix} \# \\ \gamma \end{matrix} \middle| \beta \to \gamma \in P \right\}, here \right),$$

$$\left(\left\{ \varphi_{\mathbf{c}}^{\mathbf{i}} \begin{bmatrix} \# & \# \\ A \ , & \beta \end{bmatrix} = \left\{ \begin{matrix} \# \\ \gamma \end{matrix} \middle| \beta \to \gamma \in P, A \in N \cup I \right\}, here \right),$$

$$\left(\left\{ \varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , \# \ C \right] = \left\{ \begin{matrix} \# & c \\ \# & C_i \\ \# & f_i \end{matrix} \middle| C \to cC_if_i \in P_i, C \in I, C_i \in N_i \right\} \right.,$$

$$\varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , C \ D \right] = \left\{ \begin{matrix} c & d \\ C_i & D_j \\ f_i & g_j \end{matrix} \middle| C \to cC_if_i \in P_i, C \in I, C_i \in N_i, D \to dD_jf_j \in P_j, D \in I, D_j \in N_j \right\},$$

$$\varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , C \ \# \right] = \left\{ \begin{matrix} c & \# \\ C_i & \# \\ f_i & \# \end{matrix} \middle| C \to cC_if_i \in P_i, C \in I, C_i \in N_i, \right\} \right\}, here \right),$$

$$\left(\left\{ \varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , \# \ C \right] = \left\{ \begin{matrix} \# & c \\ \# & C_i \end{matrix} \middle| C \to cC_i \in P_i, C \in I, C_i \in N_i \right\} \right.,$$

$$\varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , C \ D \right] = \left\{ \begin{matrix} c & d \\ C_i & D_j \end{matrix} \middle| C \to cC_i \in P_i, C \in I, C_i \in N_i, D \to dD_j \in P_j, D \in I, D_j \in N_j \right\},$$

$$\varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , C \ \# \right] = \left\{ \begin{matrix} c & \# \\ C_i & \# \end{matrix} \middle| C \to cC_i \in P_i, C \in I, C_i \in N_i, \right\} \right\}, here \right),$$

$$\left(\left\{ \varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , \# \ C \right] = \left\{ \begin{matrix} \# & c \\ \# & \# \end{matrix} \middle| Cf_i \to c \in P_i, C \in I, c \in T \right\} \right.,$$

$$\varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , C \ D \right] = \left\{ \begin{matrix} c & d \\ \# & \# \end{matrix} \middle| Cf_i \to c \in P_i, Dg_i \to d \in P_j, c, d \in T, C, D \in I \right\},$$

$$\varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , C \ \# \right] = \left\{ \begin{matrix} c & \# \\ \# & \# \end{matrix} \middle| Cf_i \to c \in P_i, C \in I, c \in T, \right\} \right\}, here \right),$$

$$\left(\left\{ \varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , \# \ C \right] = \left\{ \begin{matrix} \# & c \\ \# & \# \end{matrix} \middle| C \to c \in P_i, C \in I, c \in T \right\} \right.,$$

$$\varphi_{\mathbf{r}}^{\mathbf{i}} \left[\# \ \# \ , C \ D \right] = \left\{ \begin{matrix} c & d \\ \# & \# \end{matrix} \middle| C \to c \in P_i, c \in T, C \in I, , D \to d \in P_j, d \in T, D \in I \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\left[\# \ \# \ , C \ \#\right] = \left\{ \begin{matrix} c & \# \\ \# & \# \end{matrix} \middle| C \to c \in P_i, C \in I, c \in T, \right\}\Bigg\}, here\Bigg),$$

$$\Bigg(\Bigg\{\varphi^{\mathbf{i}}_{\mathbf{r}}\begin{bmatrix} a & b & C & D \\ c & d & f_i & f_j \end{bmatrix} = \left\{ \begin{matrix} c & d \\ C_i & D_j \end{matrix} \middle| Cf_i \to cC_i \in P_i, a, b, c, d \in T, C \in I, C_i \in N_i, Df_j \to dD_j \in \right.$$

$$\left. P_j, D \in I, D_j \in N_j \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\begin{bmatrix} \# & a & \# & C \\ \# & b, & \# & f_i \end{bmatrix} = \left\{ \begin{matrix} \# & c \\ \# & C_i \end{matrix} \middle| Cf_i \to cC_i \in P_i, a, b \in T, C \in I, C_i \in N_i \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\begin{bmatrix} a & \# & C & \# \\ b & \#, & f_i & \# \end{bmatrix} = \left\{ \begin{matrix} c & \# \\ C_i & \# \end{matrix} \middle| Cf_i \to cC_i \in P_i, a, b \in T, C \in I, C_i \in N_i \right\}\Bigg\}, here\Bigg),$$

$$\Bigg(\Bigg\{\varphi^{\mathbf{i}}_{\mathbf{r}}\left[\# \ a \ , \# \ E\right] = \left\{ \begin{matrix} \# & e \\ \# & \# \end{matrix} \middle| E \to e \in P_i, a, e \in T, E \in N_i \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\left[a \ b \ , E \ F\right] = \left\{ \begin{matrix} e & f \\ \# & \# \end{matrix} \middle| E \to e \in P_i, a, e, f \in T, E \in N_i, F \in N_j \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\left[a \ \# \ , E \ \#\right] = \left\{ \begin{matrix} e & \# \\ \# & \# \end{matrix} \middle| E \to e \in P_i, a, e \in T, E \in N_i, \right\}\Bigg\} here\Bigg),$$

$$\Bigg(\Bigg\{\varphi^{\mathbf{i}}_{\mathbf{r}}\left[\# \ a \ , \# \ E\right] = \left\{ \begin{matrix} \# & e \\ \# & E_i \\ \# & f_i \end{matrix} \middle| E \to eE_if_i \in P_i, E, E_i \in N_i, e \in T \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\left[a \ b \ , E \ F\right] = \left\{ \begin{matrix} e & f \\ E_i & F_j \\ f_i & f_j \end{matrix} \middle| E \to eE_if_i \in P_i, E, E_i \in N_i, e, f \in T, F \to fF_jf_j \in P_j, F, F_j \in N_j \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\left[a \ \# \ , E \ \#\right] = \left\{ \begin{matrix} e & \# \\ E_i & \# \\ f_i & \# \end{matrix} \middle| E \to eE_if_i \in P_i, E, E_i \in N_i, e \in T \right\}\Bigg\}, here\Bigg),$$

$$\Bigg(\Bigg\{\varphi^{\mathbf{i}}_{\mathbf{r}}\left[\# \ a \ , \# \ E\right] = \left\{ \begin{matrix} \# & e \\ \# & E_i \end{matrix} \middle| E \to eE_i \in P_i, E, E_i \in N_i, e \in T \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\left[a \ b \ , E \ F\right] = \left\{ \begin{matrix} e & f \\ E_i & F_j \end{matrix} \middle| E \to eE_i \in P_i, E, E_i \in N_i, e, f \in T, F \to fF_j \in N_j, F, F_j \in N_j \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\left[a \ \# \ , E \ \#\right] = \left\{ \begin{matrix} e & \# \\ E_i & \# \end{matrix} \middle| E \to eE_i \in P_i, E, E_i \in N_i, e \in T \right\}\Bigg\}, here\Bigg)\Bigg\},$$

$$\Bigg(\Bigg\{\varphi^{\mathbf{i}}_{\mathbf{r}}\left[\# \ a \ , \# \ E\right] = \left\{ \begin{matrix} \# & e \\ \# & \# \end{matrix} \middle| E \to e \in P_i, a, e \in T, E \in N_i \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\left[a \ b \ , E \ F\right] = \left\{ \begin{matrix} e & f \\ \# & \# \end{matrix} \middle| E \to e \in P_i, F \to f \in P_j, a, e, f \in T, E \in N_i, F \in N_j \right\},$$

$$\varphi^{\mathbf{i}}_{\mathbf{r}}\left[a \ \# \ , E \ \#\right] = \left\{ \begin{matrix} e & \# \\ \# & \# \end{matrix} \middle| E \to e \in P_i, a, e \in T, E \in N_i, \right\}\Bigg\}, here\Bigg),$$

$$\left(\left\{\varphi_{\mathbf{r}}^{\mathbf{i}}\left[a\ b\ ,\ \lambda\ \lambda\right]=\left\{\lambda\ \lambda\middle| a,b\in T\right\}\right\},out\right).$$

Definition of $\mathbf{D_2}$:

$$\mathbf{D_2}=\left\{\left(\left\{\varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix}\#&\#\\ \alpha\ ,&\#\end{bmatrix}=\left\{\begin{matrix}\#\\ s\end{matrix}\middle| s\to\alpha\in P\right\},here\right),\right.$$
$$\left(\left\{\varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix}\#&\#\\ \gamma\ ,&\#\end{bmatrix}=\left\{\begin{matrix}\#\\ \beta\end{matrix}\middle|\beta\to\gamma\in P\right\},here\right),$$
$$\left(\left\{\varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix}\#&\#\\ \gamma\ ,&B\end{bmatrix}=\left\{\begin{matrix}\#\\ B\end{matrix}\middle|\beta\to\gamma\in P,B\in N\cup I\right\},here\right),$$
$$\left(\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}\#&c\\ \#\ C_i&,\ \lambda\ \lambda\\ \#&f_i\end{bmatrix}=\left\{\#\ C\middle| C\to cC_if_i\in P_i,C_i\in N_i,C\in I\right\}\right.,$$
$$\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}c&d\\ C_i\ D_j&,\ \lambda\ \lambda\\ f_i&g_i\end{bmatrix}=\left\{\begin{matrix}C\ D\end{matrix}\middle| C\to cC_if_i\in P_i,C_i\in N_i,C\in I,D\to dD_jf_j\in P_j,D_j\in N_j,D\in\right.\right.$$
$$\left.\left.I\right\},\right.$$
$$\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}c&\#\\ C_i\ \#&,\ \lambda\ \lambda\\ f_i&\#\end{bmatrix}=\left\{C\ \#\middle| C\to cC_if_i\in P_i,C_i\in N_i,C\in I\right\}\right\},here\right),$$
$$\left(\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}\#&c\\ \#\ C_i&,\ \lambda\ \lambda\end{bmatrix}=\left\{\#\ C\middle| C\to cC_i\in P_i,C_i\in N_i,C\in I\right\}\right.,$$
$$\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}c&d\\ C_i\ D_j&,\ \lambda\ \lambda\end{bmatrix}=\left\{C\ D\middle| C\to cC_i\in P_i,C_i\in N_i,C\in I,D\to dD_j\in P_j,D_j\in N_j,D\in I\right\},\right.$$
$$\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}c&\#\\ C_i\ \#&,\ \lambda\ \lambda\end{bmatrix}=\left\{C\ \#\middle| C\to cC_i\in P_i,C_i\in N_i,C\in I\right\}\right\},here\right),$$
$$\left(\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}\#&c\\ \#\ \#&,\ \lambda\ \lambda\end{bmatrix}=\left\{\#\ C\middle| C\to c\in P_i,C\in I,c\in T\right\}\right.,$$
$$\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}c&d\\ \#\ \#&,\ \lambda\ \lambda\end{bmatrix}=\left\{C\ D\middle| C\to c\in P_i,c\in T,C\in I,D\to d\in P_j,d\in T,D\in I\right\},\right.$$
$$\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}c&\#\\ \#\ \#&,\ \lambda\ \lambda\end{bmatrix}=\left\{C\ \#\middle| C\to c\in P_i,c\in T,C\in I\right\}\right\},here\right),$$
$$\left(\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}\#&c\\ \#\ C_i&,\ \lambda\ \lambda\end{bmatrix}=\left\{\begin{matrix}\#\ C\\ \#\ f_i\end{matrix}\middle| Cf_i\to cC_i\in P_i,a,b\in T,C\in I,C_i\in N_i\right\}\right.,$$
$$\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}c&d\\ C_i\ D_j&,\ \lambda\ \lambda\end{bmatrix}=\left\{\begin{matrix}C\ D\\ f_i\ f_j\end{matrix}\middle| Cf_i\to cC_i\in P_i,C\in I,C_i\in N_i,Df_j\to dD_j\in P_j,D\in I,D_j\in\right.\right.$$

$$N_j\Bigg\},$$
$$\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix} c & \# \\ C_i\ \# & ,\ \lambda\ \lambda \end{bmatrix} = \left\{\begin{matrix} c & \# \\ f_i & \# \end{matrix}\middle| Cf_i \to cC_i \in P_i, c \in T, C \in I, C_i \in N_i\right\}\right\}, here\Bigg),$$
$$\Bigg(\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix} \# & e \\ \#\ E_i & ,\ \#\ g_i \end{bmatrix} = \left\{\# \ E \middle| E \to eE_i \in P_i,\ E, E_i \in N_i, e \in T, g_i \in F_i\right\},$$
$$\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix} e & f \\ E_i\ F_j & ,\ g_i\ g_j \end{bmatrix} = \left\{E\ F \middle| E \to eE_i \in P_i, E, E_i \in N_i, g_i \in \mathcal{F}_i, F \to fF_j \in P_j, F, F_j \in N_j, g_j \in \right.$$
$$\left.\mathcal{F}_j, e, f \in T\right\},$$
$$\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix} e & \# \\ E_i\ \# & ,\ g_i\ \# \end{bmatrix} = \left\{E\ \# \middle| E \to eE_i \in P_i, E, E_i \in N_i, e \in T, g_i \in \mathcal{F}_i\right\}\right\}, here\Bigg),$$
$$\Bigg(\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix} \# & e \\ \#\ \# & ,\ \lambda\ \lambda \end{bmatrix} = \left\{\# \ E \middle| E \to e \in P_i, a, e \in T, E \in N_i\right\},$$
$$\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix} e & f \\ \#\ \# & ,\ \lambda\ \lambda \end{bmatrix} = \left\{E\ \# \middle| E \to e \in P_i, a, e, f \in T, E \in N_i, F \in N_j\right\},$$
$$\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix} e & \# \\ \#\ \# & ,\ \lambda\ \lambda \end{bmatrix} = \left\{E\ \# \middle| E \to e \in P_i, a, e \in T, E \in N_i\right\}\right\}, here\Bigg),$$
$$\Bigg(\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\left[\# \ a\ ,\ \lambda\ \lambda\right] = \left\{\# \ \# \middle| a \in T\right\},\right.$$
$$\varphi_{\mathbf{r}}^{\mathbf{d}}\left[a\ b\ ,\ \lambda\ \lambda\right] = \left\{\# \ \# \middle| a, b \in T\right\},$$
$$\varphi_{\mathbf{r}}^{\mathbf{d}}\left[a\ \#\ ,\ \lambda\ \lambda\right] = \left\{\# \ \# \middle| a \in T\right\}\right\}, here\Bigg),$$
$$\Bigg(\left\{\varphi_{r}^{d}\left[\lambda\ \lambda\ ,\ \#\ a\right] = \left\{\# \ \# \middle| a \in T\right\},\right.$$
$$\varphi_{\mathbf{r}}^{\mathbf{d}}\left[\lambda\ \lambda\ ,\ a\ b\right] = \left\{\# \ \# \middle| a, b \in T\right\},$$
$$\varphi_{\mathbf{r}}^{\mathbf{d}}\left[\lambda\ \lambda\ ,\ a\ \#\right] = \left\{\# \ \# \middle| a \in T\right\}\right\}, here\Bigg),$$
$$\Bigg(\left\{\varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix} \lambda & a \\ \lambda\ , & b \end{bmatrix} = \left\{\begin{matrix} \# \\ \# \end{matrix}\middle| a, b \in T\right\}\right\}, here\Bigg),$$
$$\Bigg(\left\{\varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix} a & \lambda \\ b\ , & \lambda \end{bmatrix} = \left\{\begin{matrix} \# \\ \# \end{matrix}\middle| a, b \in T\right\}\right\}, here\Bigg),$$

$$\left(\left\{\varphi_{\mathbf{c}}^{\mathbf{d}}\begin{bmatrix}a & \lambda\\ b\ , & \lambda\end{bmatrix} = \left\{\begin{matrix}\#\\ \#\end{matrix}\middle| a,b\in T\right\}\right\}, here\right),$$

$$\left(\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}\# & c\\ \# & C_i\ , \# & g_i\end{bmatrix} = \left\{\begin{matrix}\# & C\\ \# & f_i\end{matrix}\middle| Cf_i\to cC_i\in P_i, g_i\in F_i, C\in I, C_i\in N_i\right\}\right.,$$

$$\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}c & d\\ C_i & D_j\ , g_i & g_j\end{bmatrix} = \left\{\begin{matrix}C & D\\ f_i & f_j\end{matrix}\middle| Cf_i\to cC_i\in P_i, g_i\in\mathcal{F}_i, C\in I, C_i\in N_i, Df_j\to dD_j\in P_j, g_j\in\right.$$

$$\left.\mathcal{F}_j, D_j\in N_j, D\in I\right\},$$

$$\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}c & \#\\ C_i & \#\ , g_i & \#\end{bmatrix} = \left\{\begin{matrix}C & \#\\ f_i & \#\end{matrix}\middle| Cf_i\to cC_i\in P_i, g_i\in\mathcal{F}_i, C\in I, C_i\in N_i\right\}\right\}, here\right),$$

$$\left(\left\{\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}\# & e\\ \# & E_i & \# & g_i\\ \# & f_i\ , & & \end{bmatrix} = \left\{\begin{matrix}\# & E\end{matrix}\middle| E\to eE_if_i\in P_i, E, E_i\in N_i, g_i\in\mathcal{F}_i, e\in T\right\}\right.,$$

$$\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}e & f\\ E_i & F_j & g_i & g_j\\ f_i & f_j\ , & & \end{bmatrix} = \left\{\begin{matrix}E & F\end{matrix}\middle| E\to eE_if_i\in P_i, E, E_i\in N_i, e, f\in T, g_i\in\mathcal{F}_i, F\to fF_jf_j\in\right.$$

$$\left.P_j, F, F_j\in N_j, g_j\in\mathcal{F}_j\right\},$$

$$\varphi_{\mathbf{r}}^{\mathbf{d}}\begin{bmatrix}e & \#\\ E_i & \# & g_i & \#\\ f_i & \#\ , & & \end{bmatrix} = \left\{\begin{matrix}E & \#\end{matrix}\middle| E\to eE_if_i\in P_i, E, E_i\in N_i, e\in T\right\}\right\}, here\right)\right\}.$$

Solving the feasibility problem in robotic motion planning by means of Enzymatic Numerical P systems

Ignacio Pérez-Hurtado[1], Miguel Á. Martínez-del-Amor[1],
Gexiang Zhang[2], Ferrante Neri[3], and Mario J. Pérez-Jiménez[1]

[1] Research Group on Natural Computing, Dpt. Computer Science and Artificial Intelligence, School of Computer Engineering, Universidad de Sevilla, Seville, Spain
[2] School of Electrical Engineering, Southwest Jiaotong University, Chengdu, People's Republic of China
[3] COL Laboratory, School of Computer Science, University of Nottingham, Nottingham, United Kingdom

Abstract. Solving the feasibility problem in robotic motion planning means to find feasible trajectories for specific mobile robots acting in environments with obstacles whose positions are known a priori. The Rapidly-exploring Random Tree (RRT) algorithm is a classical algorithm to solve such a problem in real-life applications. In this paper, we provide a model in the framework of Enzymatic Numerical P systems to reproduce the behaviour of the RRT algorithm. A C++ ad-hoc simulator is also provided to validate the model.

Keywords: Motion Planning · Rapidly-exploring Random Tree · Membrane Computing · Enzymatic Numerical P systems

1 Introduction

This article studies a crucially important problem related to motion tracking, i.e. motion planning. The motion planning problem consists of finding a trajectory to move an agent trough a complex environment from a starting point to a desired area avoiding any obstacle while considering constraints related to the agent such as shape, kinematics and others. This problem is critical in almost all robot applications since, by definition, a robot is a machine developing tasks in the real world.

Several approximated solutions have been proposed to the motion planning problem. An special mention should be given to a category of algorithms to build Rapidly-exploring Random Trees (RRTs) [5]. They are based on the randomized exploration of the configuration space by building a tree where nodes represent reachable states and edges represent transitions.

The RRT algorithms are inherently sequential, but there are modules that can be parallelized such as the obstacle collision detection [2].

Membrane Computing [15] is a computing paradigm inspired from the living cells and it provides distributed, massively parallel devices. Models in Membrane Computing are generically called P systems and they have been used in different contexts. Among others, we stress the following: (a) showing the ability of some models to give polynomial time solutions to computationally hard problems, by trading space for time; (b)

providing a new methodology to tackle the **P** versus **NP** problem [14, 9, 7]; (c) being a framework for modelling biomolecular processes as well as real ecosystems [17, 4, 1, 3]; (d) incorporating fuzzy reasoning in models that mimic the way that neurons communicate with each other by means of short electrical impulses, and applying them to many different industrial applications related to fault diagnosis [18].

A variant of P systems called Enzimatic Numerical P systems (ENPS, for short) has been used to model and simulate robot controllers [11, 12, 20]. In this work, the ENPS framework is used to design a bio-inspired parallel RRT model. It is worth pointing out that no additional ingredient to the ENPS framework has been included, as in [13]. In consequence, the presented models can be compatible with existing ENPS robot controllers. In order to validate the model, an ad-hoc simulator has been implemented in C++.

The rest of this paper is structured as follows. Next section summarizes some preliminary concepts. Section 3 is devoted to present a novel ENPS model for the RRT algorithm. The simulator implemented in C++ is described in Section 4. The paper ends with conclusions and some ideas for future work.

2 Preliminaries

This Section provides the reader with the basic concepts and notation used throughout this paper.

2.1 Motion planning

In general terms, the problem of motion planning can be defined in the configuration space of a mobile agent as follows. Given:

- An initial configuration state.
- A set of valid final configuration states.
- A map of obstacles in the environment.
- A description of the agent shape.
- A description of the agent kinematics.

Find a sequence of configuration states through the configuration space, a.k.a. trajectory or plan, from the initial state to one of the final states, which does not touch any obstacle in the environment considering the agent shape and kinematics.

There are two variants of the problem:

- **The feasibility problem** is to find a feasible trajectory, if one exists, and report failure otherwise.
- **The optimality problem** is to find a feasible trajectory with minimal cost where the cost of a trajectory is given by a computable function.

2.2 Rapidly-exploring random trees

AN RRT [5] is a randomized tree structure for rapidly exploring the obstacle-free configuration space. It has successfully been used to solve nonholonomic and kinodynamic motion planning problems [6]. Nodes in an RRT represent possible reachable states, edges represent transitions between states. The root of an RRT is the initial state. Each state in an RRT can be reached by following the trajectory from the root to the corresponding node, as can be seen in Figure 1. Algorithms used in robotics to generate RRTs are known to be anytime and any-angle, i.e, the produced trajectories could contain turns in any valid angle considering the robot constraints and kinodynamics.

Fig. 1: A trajectory conducted by an RRT

The RRT algorithm The RRT algorithm [5] is used to generate an RRT structure in order to solve the feasibility problem for motion planning in robotics. Algorithm 1 shows the pseudocode.

Algorithm 1 The RRT algorithm

$V \leftarrow \{x_{init}\}$; $E \leftarrow \emptyset$; $i \leftarrow 0$;
while $i < N$ **do**
  $G \leftarrow (V, E)$;
  $x_{rand} \leftarrow Sample()$; $i \leftarrow i + 1$;
  $(V, E) \leftarrow Extend(G, x_{rand})$;
end while

where

- x_{init} is the initial robot state.
- $Sample()$ returns independent identically distributed (i.i.d.) samples from the state space.

Algorithm 2 $Extend(G,x)$

$V' \leftarrow V$; $E' \leftarrow E$;
$x_{nearest} \leftarrow Nearest(G,x)$;
$x_{new} \leftarrow Steer(x_{nearest}, x_{new})$;
if $ObstacleFree(x_{nearest}, x_{new})$ **then**
  $V' \leftarrow V' \cup \{x_{new}\}$;
  $E' \leftarrow E' \cup \{(x_{nearest}, x_{new})\}$;
end if
return $G' = (V', E')$

where

- $Nearest(G,x)$ returns the nearest node in G to x according to a distance metric. In this paper we will consider the Euclidean distance.
- $Steer(x,y)$ simulates a motion from x to y considering the robot constraints and returns the computed state after a fixed Δt time. In this paper we will consider holonomic wheeled robots in 2D environments able to turn and move forward in any angle. Thus, the simulated motions can be considered as straight lines.
- $ObstacleFree(x,y)$ returns $true$ if the trajectory in straight line from x to y is free of obstacles; $false$ otherwise.

2.3 Numerical P systems

Numerical P systems (NPS) are P systems, see [10, 19], introduced in [16] to model economical and business processes. In NPS, the concept of multisets of objects is replaced by numerical variables that evolve from initial values by means of production functions and repartition protocols. A numerical P system is formally expressed by:

$$\Pi = (m, H, \mu, (Var_1, Pr_1, Var_1(0)), \ldots, (Var_m, Pr_m, Var_m(0)))$$

where

- m is the number of membranes; $m \geq 1$;
- H is an alphabet of labels, containing m symbols;
- μ is the membrane structure;
- Var_i is a set of variables for compartment i, being $Var_i(0)$ their initial values;
- Pr_i is a set of programs for compartment i. A program has the following syntax:

$$F(x_1, \ldots, x_k) \rightarrow c_1|v_1 + \ldots + c_n|v_n$$

  where
  - The left-hand-side of the program is called *production function* and the right-hand-side is called *repartition protocol*.
  - $F(x_1, \ldots, x_k)$ is a function $\mathbb{R}^k \rightarrow \mathbb{R}$ using variables $x_1, \ldots, x_k$.
  - $v_1, \ldots, v_n$ are output variables. The output value of F will be distributed among the output variables according to the repartition protocol given by $c_1, \ldots, c_n$.

- $c_1, \ldots, c_n$ are numeric values representing the portion of the output which is going to be assigned to the corresponding variable.

For the sake of simplicity, in the rest of this paper, we will use the next syntax:

$$F(x_1, \ldots, x_k) \rightarrow v$$

when there is one and only one output variable in the repartition protocol.

2.4 Enzymatic numerical P systems

Enzimatic numerical P systems (ENPS) are an extension of NPS introduced in [11] for modeling and simulation of membrane controllers for autonomous mobile robots. The main difference of ENPS with respect to NPS is the concept of *enzyme* which is used to write conditions related to programs. The formal definition of the ENPS is the following:

$$\Pi = (m, H, \mu, (Var_1, E_1, Pr_1, Var_1(0)), \ldots, (Var_m, E_m, Pr_m, Var_m(0)))$$

where

- m, H, μ, Var_i and $Var_i(0)$ have the same meaning than explained in Subsection 2.3.
- E_i is a set of variables $E_i \subseteq Var_i$ called enzymes.
- Pr_i is a set of programs for compartment i. The syntax of a program is the following:

$$F(x_1, \ldots, x_k)|_{Cond(e_1, \ldots, e_r)} \rightarrow c_1|v_1 + \ldots + c_n|v_n$$

where

- $F(x_1, \ldots, x_k)$ and $c_1|v_1 + \ldots + c_n|v_n$ have the same meaning than explained in Subsection 2.3.
- $Cond(e_1, \ldots, e_r)$ is a condition function $\mathbb{R}^r \rightarrow \{true, false\}$ using enzymes $e_1, \ldots, e_r$. The program is disabled when the output of such a function is $false$.

For the sake of simplicity, it is not necessary to write the condition function when it is the constant function $Cond() = true$.

The membrane structure is designed to organize programs and variables in modules. All the variables are considered global, i.e, any program can read or write a variable regardless of the compartment where the variable is defined. The definition of variables as well as their initial values is given by the syntax $x[v]$ where x is a variable and v is a numeric value. The reserved word *input* can be written instead of a numeric value for v when the initial value should be read from external inputs, e.g, robot sensors. In this paper, we will use the syntax $a_{[b]}$ in order to refer to variable a_i where i is the value stored in variable b.

3 An ENPS model for the RRT algorithm

In this section, an ENPS model is presented in order to simulate the behaviour of the RRT algorithm taking advantage of the inherent parallelism level existing in the membrane computing framework. We have divided the whole problem in the next sub-problems:

- Find the nearest point to a given point according to the Euclidean distance.
- Determine if a given trajectory is obstacle free.
- Simulate the RRT algorithm.

3.1 Finding the nearest point

Given a set of points $X = \{(x_i, y_i)\} : 1 \leq i \leq 2^n$ and a target point (x_t, y_t), find the nearest point (x, y) in X to the target point according to the Euclidean distance.

Solution for n=3 The solution of the nearest point for $n = 2$ is shown in Figure 2 Where

$$min(a,b) = \begin{cases} a & a < b \\ b & a \geq b \end{cases}$$

$$min^*(a,b,c,d) = \begin{cases} a & c < d \\ b & c \geq d \end{cases}$$

The P system computes in one step of computation the squared Euclidean distance for all the points in X to the target point. After that, a reduction operation is conducted in 3 steps to compute the minimum distance as well as the nearest point in X. The computation stops after 4 steps, then *halt* is set to 1 and the nearest point is stored in $(x_{nearest}, y_{nearest})$. Variable α is used as an step counter.

General solution The general solution, for n points is given in Figure 3
After $n+1$ steps, the nearest point is stored in $(x_{nearest}, y_{nearest})$.

3.2 Obstacle free trajectories

Given a set of obstacle points $O = \{(a_i, b_i)\} : 1 \leq i \leq 2^m$, a starting point (x_0, y_0) and an ending point (x_1, y_1), determine if the trajectory following a straight line from (x_0, y_0) to (x_1, y_1) is obstacle free. A trajectory is obstacle free if the distance from the nearest obstacle to the trajectory is greater or equal than a given parameter ξ, see Figure 4.

With reference to Figure 4:

- $pDist(c_x, c_y, a_x, a_y, b_x, b_y)$ returns the squared Euclidean distance from the point (c_x, c_y) to the segment $[(a_x, a_y), (b_x, b_y)]$.
- After $m+1$ steps, the variable *collision* contains a value equal or less than zero if the trajectory is obstacle free.

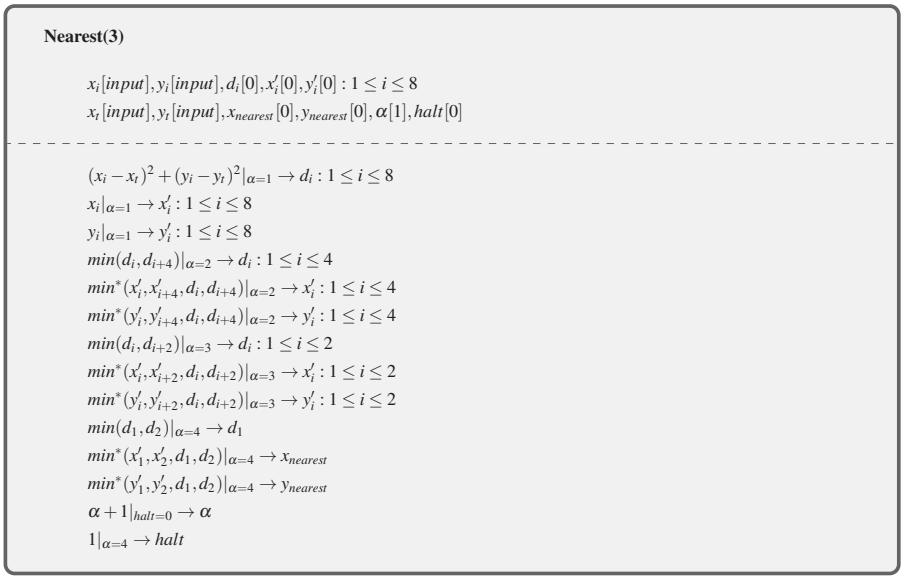


Fig. 2: Nearest(3) procedure

Algorithm 3 shows the pseudocode of the *pDist* function.

The P system computes in one step of computation the squared Euclidean distance for all the obstacles to the segment given by $[(x_0, y_0), (x_1, y_1)]$. After that, a reduction operation is conducted in m steps of computation, obtaining the minimum distance. In the last step, variables $collision$ and *halt* are set.

3.3 The RRT algorithm

For the following set of parameters

- An initial robot position (x_1, y_1).
- A set of obstacle points $\{(a_i, b_i)\} : 1 \leq i \leq 2^m$.
- The size (p, q) of the scenario.
- Parameter n, where the number of points in the RRT will be 2^n.
- Parameter ξ as explained in subsection 3.2.
- Parameter δ giving the length of the edges in the RRT.

the RRT algorithm is illustrated in Figure 5.

The inner modules are defined as shown in Figures 6, 7, and 8 where

- $random()$ returns a random number independent identically distributed (i.i.d.) in $[0, 1] \cap \mathbb{R}$
- $rm(x, y)$ returns the remainder of the integer division between x and y.
- The coordinates of the RRT nodes will be $\{(x_i, y_i)\} : 1 \leq i \leq 2^n$

Nearest(n)

$x_i[input], y_i[input], d_i[0], x'_i[0], y'_i[0] : 1 \leq i \leq 2^n$
$x_t[input], y_t[input], x_{nearest}[0], y_{nearest}[0], \alpha[1], halt[0]$

$(x_i - x_t)^2 + (y_i - y_t)^2|_{\alpha=1} \rightarrow d_i : 1 \leq i \leq 2^n$
$x_i|_{\alpha=1} \rightarrow x'_i : 1 \leq i \leq 2^n$
$y_i|_{\alpha=1} \rightarrow y'_i : 1 \leq i \leq 2^n$
$min(d_i, d_{i+2^{n-j}})|_{\alpha=j+1} \rightarrow d_i : 1 \leq i \leq 2^{n-j}, 1 \leq j \leq n$
$min^*(x'_i, x'_{2^{n-j}}, d_i, d_{2^{n-j}})|_{\alpha=j+1} \rightarrow x'_i : 1 \leq i \leq 2^{n-j}, 1 \leq j < n$
$min^*(y'_i, y'_{2^{n-j}}, d_i, d_{2^{n-j}})|_{\alpha=j+1} \rightarrow y'_i : 1 \leq i \leq 2^{n-j}, 1 \leq j < n$
$min^*(x'_1, x'_2, d_1, d_2)|_{\alpha=n+1} \rightarrow x_{nearest}$
$min^*(y'_1, y'_2, d_1, d_2)|_{\alpha=n+1} \rightarrow y_{nearest}$
$\alpha + 1|_{halt=0} \rightarrow \alpha$
$1|_{\alpha=n+1} \rightarrow halt$

Fig. 3: General Nearest(n) procedure

Algorithm 3 $pDist(c_x, c_y, a_x, a_y, b_x, b_y)$

$u \leftarrow (c_x - a_x) \cdot (b_x - a_x) + (c_y - a_y) \cdot (b_y - a_y)$;
$u \leftarrow u/[(b_x - a_x)^2 + (b_y - a_y)^2]$;
if $u < 0$ **then**
  return $(a_x - c_x)^2 + (a_y - c_y)^2$
end if
if $u > 1$ **then**
  return $(b_x - c_x)^2 + (b_y - c_y)^2$
end if
$p_x \leftarrow a_x + u \cdot (b_x - a_x)$;
$p_y \leftarrow a_y + u \cdot (b_y - a_y)$;
return $(p_x - c_x)^2 + (p_y - c_y)^2$

- The coordinates of the RRT parent nodes will be $\{(px_i, py_i)\} : 2 \leq i \leq 2^n$
- The RRT is completely generated and the computation stops when $halt = 1$.

The P system generates the point (x_{rand}, y_{rand}) in one step of computation, after that, the module $Nearest_{RRT}(n)$ computes the point $(x_{nearest}, y_{nearest})$ in $n+1$ steps of computation as explained in subsection 3.1. Then, the (x_{new}, y_{new}) point is computed according to the δ parameter. The next module to be executed is the $ObstacleFree_{RRT}(n, \xi)$ module, see subsection 3.2, computing the squared Euclidean distance of the nearest obstacle point to the segment $[(x_{nearest}, y_{nearest}), (x_{new}, y_{new})]$, if this value is less than ξ parameter, then the variable $collision$ will contain a value greater than 0 after $m+1$ steps of computation. Finally, if the segment is obstacle free, the module $Extend(n, m)$ updates the RRT. In this way, each node is added to the RRT in $m+n+6$ computation steps if the corresponding edge is obstacle free.

ObstacleFree(m,ξ)

$a_i[input], b_i[input], d_i'[0] : 1 \leq i \leq 2^m$
$x_0[input], y_0[input], x_1[input], y_1[input]$
$collision[0], \alpha[1], halt[0]$

$pDist(a_i, b_i, x_0, y_0, x_1, y_1)|_{\alpha=1} \rightarrow d_i' : 1 \leq i \leq 2^m$
$min(d_i', d_{i+2^{m-j}}')|_{\alpha=j+1} \rightarrow d_i' : 1 \leq i \leq 2^{m-j}, 1 \leq j < m$
$min(\xi - d_1', \xi - d_2')|_{\alpha=m+1} \rightarrow collision$
$\alpha + 1|_{halt=0} \rightarrow \alpha$
$1|_{\alpha=m+1} \rightarrow halt$

Fig. 4: Procedure for the detection of an obstacle free trajectory

RRT(n,m,p,q,δ,ξ)

$x_1[input], y_1[input]$
$a_i[input], b_i[input] : 1 \leq i \leq 2^m$
$x_i[3 \cdot p], y_i[3 \cdot q] : 2 \leq i \leq 2^n$
$px_i[0], py_i[0] : 2 \leq i \leq 2^n$
$x_{rand}[0], y_{rand}[0], x_{new}[0], y_{new}[0],$
$x_{nearest}[0], y_{nearest}[0], collision[0], \alpha[1], index[2], halt[0]$

$0|_{\alpha=1} \rightarrow collision$
$p \cdot random()|_{\alpha=1} \rightarrow x_{rand}$
$q \cdot random()|_{\alpha=1} \rightarrow y_{rand}$
$x_{nearest} + \delta \cdot \frac{(x_{rand} - x_{nearest})}{\sqrt{d_1}}|_{\alpha=n+3} \rightarrow x_{new}$
$y_{nearest} + \delta \cdot \frac{(y_{rand} - y_{nearest})}{\sqrt{d_1}}|_{\alpha=n+3} \rightarrow y_{new}$
$rm(\alpha, m+n+6) + 1|_{collision \leq 0} \rightarrow \alpha$
$1|_{collision>0} \rightarrow \alpha$
$index + 1|_{\alpha=m+n+6} \rightarrow index$
$1|_{index=2^n+1} \rightarrow halt$

Nearest$_{RRT}$(n)

ObstacleFree$_{RRT}$(m,ξ)

Extend$_{RRT}$(n,m)

Fig. 5: Rapidly-exploring Random Tree

Nearest$_{RRT}$(n)

$d_i[0], x'_i[0], y'_i[0] : 1 \leq i \leq 2^n$

$(x_i - x_{rand})^2 + (y_i - y_{rand})^2|_{\alpha=2} \rightarrow d_i : 1 \leq i \leq 2^n$
$x_i|_{\alpha=2} \rightarrow x'_i : 1 \leq i \leq 2^n$
$y_i|_{\alpha=2} \rightarrow y'_i : 1 \leq i \leq 2^n$
$min(d_i, d_{i+2^{n-j}})|_{\alpha=j+2} \rightarrow d_i : 1 \leq i \leq 2^{n-j}, 1 \leq j \leq n$
$min^*(x'_i, x'_{2^{n-j}}, d_i, d_{2^{n-j}})|_{\alpha=j+2} \rightarrow x'_i : 1 \leq i \leq 2^{n-j}, 1 \leq j < n$
$min^*(y'_i, y'_{2^{n-j}}, d_i, d_{2^{n-j}})|_{\alpha=j+2} \rightarrow y'_i : 1 \leq i \leq 2^{n-j}, 1 \leq j < n$
$min^*(x'_1, x'_2, d_1, d_2)|_{\alpha=n+2} \rightarrow x_{nearest}$
$min^*(y'_1, y'_2, d_1, d_2)|_{\alpha=n+2} \rightarrow y_{nearest}$

Fig. 6: Nearest$_{RRT}$ Procedure

ObstacleFree$_{RRT}$(m,ξ)

$d'_i[0] : 1 \leq i \leq 2^m$

$pDist(a_i, b_i, x_{nearest}, y_{nearest}, x_{new}, y_{new})|_{\alpha=n+4} \rightarrow d'_i : 1 \leq i \leq 2^m$
$min(d'_i, d'_{i+2^{m-j}})|_{\alpha=j+n+4} \rightarrow d'_i : 1 \leq i \leq 2^{m-j}, 1 \leq j < m$
$min(\xi - d'_1, \xi - d'_2)|_{\alpha=m+n+4} \rightarrow collision$

Fig. 7: ObstacleFree$_{RRT}$ Procedure

Extend$_{RRT}$(n,m)

$x_{new}|_{\alpha=m+n+6} \rightarrow x_{[index]}$
$y_{new}|_{\alpha=m+n+6} \rightarrow y_{[index]}$
$x_{nearest}|_{\alpha=m+n+6} \rightarrow px_{[index]}$
$y_{nearest}|_{\alpha=m+n+6} \rightarrow py_{[index]}$

Fig. 8: Extend$_{RRT}$ Procedure

4 Developed software

We have implemented a specific (ad-hoc) simulator using C++, which is able to simulate the ENPS-RRT model described in section 3. Let us recall that a specific simulator aims at simulating a certain P system model (as in this case), instead of simulating a whole P system variant (i.e. a generic simulator) [8]. In a specific simulator, the developer encode the P system directly in the source code, and can implement assumptions for simplicity and efficiency. In any case, the simulator must simulate the model; that is, there should be a way to know the state of the P system at any time.

The simulator is able to manage image files in PGM format defining the obstacle maps, two predefined scenarios have been included with the software. For each one, a PGM file is included along with a fixed initial position for the robot. The resolution for all maps is $5cm^2/pixel$. In Figure 9, the two scenarios and their corresponding output are shown.

The parameters of the models are fixed:

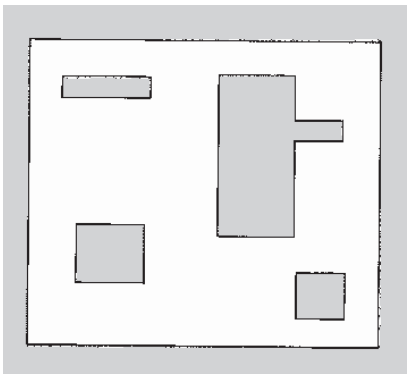
- $n = 12$, i.e, 2^{12} RRT nodes will be generated.
- $\delta = 15cm$, i.e, the RRT edges will have $15cm$ length.
- $\varepsilon = 20cm$, the robot radius.
- m, p, q depend on the specific PGM file.

The simulator uses data structures to store the value of the ENPS variables in runtime and programming sentences in C++ in order to simulate the behaviour of the ENPS programs.

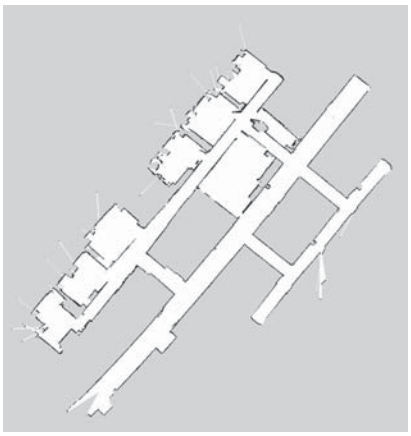
5 Conclusions

In this work, we present an ENPS model emulating the computation of the RRT algorithm in order to solve the feasibility problem for robotic motion planning. The ENPS framework has been successfully used to design and simulate robot controllers, but there is a lack of solutions for global path planning using the framework. An extension of ENPS, called Random Enzymatic Numerical P systems with Shared Memory (RENPSM for short), was introduced in [13] to simulate for the first time RRT algorithms but such an extension contains several new ingredients, on the contrary, the model presented in this paper is based on the original ENPS framework, being compatible with existing membrane computing robot controllers.

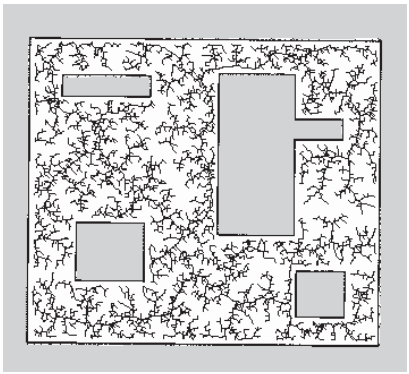
Several research lines are proposed as future work. We would like to study other RRT algorithms, as the RRT* algorithm which provides an asymptotically-optimal solution to the motion planning problem. We also would like to extend our current simulator to existing parallel hardware/software architectures such a CUDA, OpenMP or FPGA in order to take advantage of the massively parallelism level in the model. Finally, we would like to integrate our ENPS-RRT models with existing ENPS robot controllers in order to get a navigation stack based on membrane computing for real robots.



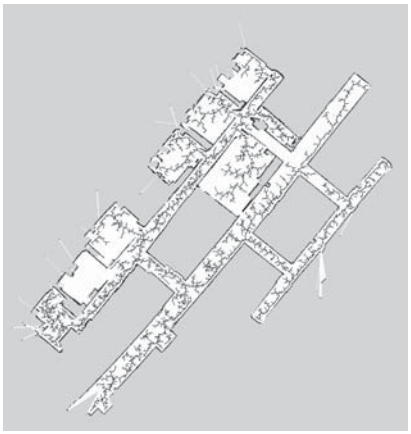
(a) map1



(b) ccia_h



(c) Output map1



(d) Output ccia_h

Fig. 9: Maps employed in the experiments and the obtained outputs from the ENPS-RRT model: (a) is **map1** (a simple room), (b) is **ccia_h** (scanned map of the computer science department's wing H at University of Seville), (c) is the output of **map1**, (d) is the output of **ccia_h**.

Acknowledgements

This work is supported by the research project TIN2017-89842-P (MABICAP), co-financed by *Ministerio de Economía, Industria y Competitividad (MINECO)* of Spain, through the *Agencia Estatal de Investigación (AEI)*, and by *Fondo Europeo de Desarrollo Regional (FEDER)* of the European Union.

This work is supported by the *National Natural Science Foundation of China* (61972324, 61672437, 61702428), the New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0043) and the Sichuan Science and Technology Program (2018GZ0185, 2018GZ0086) and Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06).

References

1. Barbuti, R., Bove, P., Milazzo, P., Pardini, G.: Minimal probabilistic P systems for modelling ecological systems. Theoretical Computer Science **608**, 36 – 56 (2015). https://doi.org/10.1016/j.tcs.2015.07.035
2. Bialkowski, J., Karaman, S., Frazzoli, E.: Massively parallelizing the RRT and the RRT*. In: Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 3513–3518 (Sep 2011)
3. Colomer, M.A., Margalida, A., Pérez-Jiménez, M.J.: Population Dynamics P System (PDP) Models: A Standardized Protocol for Describing and Applying Novel Bio-Inspired Computing Tools. PLOS ONE **8**(5), e60698 (2013). https://doi.org/10.1371/journal.pone.0060698
4. Gheorghe, M., Krasnogor, N., Camara, M.: P systems applications to systems biology. Biosystems **91**(3), 435 – 437 (2008). https://doi.org/10.1016/j.biosystems.2007.07.002
5. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Tech. rep. (1998)
6. LaValle, S.M., Kuffner, J.J.: Randomized Kinodynamic Planning. The International Journal of Robotics Research **20**(5), 378–400 (2001)
7. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: A survey on space complexity of P systems with active membranes. International Journal of Advances in Engineering Sciences and Applied Mathematics **10**(3), 221–229 (2018). https://doi.org/10.1007/s12572-018-0227-8
8. Martínez-del-Amor, M.Á., Orellana-Martín, D., Pérez-Hurtado, I., Valencia-Cabrera, L., Riscos-Núñez, A., Pérez-Jiménez, M.J.: Design of Specific P Systems Simulators on GPUs. In: Hinze, T., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) Membrane Computing. Lecture Notes in Computer Science, vol. 11399, pp. 202–207. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-12797-8_14
9. Orellana-Martín, D., Valencia-Cabrera, L., Riscos-Núñez, A., Pérez-Jiménez, M.J.: A path to computational efficiency through membrane computing. Theoretical Computer Science **777**, 443 – 453 (2019). https://doi.org/10.1016/j.tcs.2018.12.024
10. Pan, L., Puaun, G., Zhang, G., Neri, F.: Spiking neural *P* systems with communication on request. Int. J. Neural Syst. **27**(8), 1–13 (2017)
11. Pavel, A., Arsene, O., Buiu, C.: Enzymatic numerical p systems - a new class of membrane computing systems. In: 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA). pp. 1331–1336 (Sep 2010)
12. Pavel, A., Buiu, C.: Using enzymatic numerical p systems for modeling mobile robot controllers. Natural Computing **11**(3), 387–393 (2012). https://doi.org/10.1007/s11047-011-9286-5

13. Pérez-Hurtado, I., Pérez-Jiménez, M.J., Zhang, G., Orellana-Martín, D.: Simulation of Rapidly-Exploring Random Trees in Membrane Computing with P-Lingua and Automatic Programming. International Journal of Computers, Communications and Control **13**(6), 1007–1031 (2019)
14. Pérez-Jiménez, M.J., Riscos-Núñez, A., Valencia-Cabrera, L., Orellana-Martín, D.: Results on Computational Complexity in Bio-inspired Computing, chap. 2, pp. 33–73. World Scientific (2019). https://doi.org/10.1142/9789813143180_0002
15. Păun, Gh.: Computing with Membranes. Journal of Computer and System Sciences **61**(1), 108 – 143 (2000). https://doi.org/10.1006/jcss.1999.1693
16. Păun, Gh., Păun, R.: Membrane Computing and Economics: Numerical P Systems. Fundamenta Informaticae **73**(1,2), 213–227 (2006)
17. Păun, Gh., Romero-Campero, F.J.: Membrane Computing as a modeling framework. Cellular systems case studies. In: Bernardo, M., Degano, P., Zavattaro, G. (eds.) Formal Methods for Computational Systems Biology, Lecture Notes in Computer Science, vol. 5016, pp. 168–214. Springer Berlin Heidelberg (2008). https://doi.org/10.1007/978-3-540-68894-5_6
18. Wang, T., Zhang, G., Zhao, J., He, Z., Wang, J., Pérez-Jiménez, M.J.: Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural p systems. IEEE Transactions on Power Systems **30**(3), 1182–1194 (2015)
19. Wu, T., Bîlbîe, F., Paun, A., Pan, L., Neri, F.: Simplified and yet turing universal spiking neural P systems with communication on request. Int. J. Neural Syst. **28**(8), 1850013 (2018)
20. Zhang, G., Pérez-Jiménez, M., Gheorghe, M.: Real-life Applications with Membrane Computing, vol. 25. Springer (2017). https://doi.org/10.1007/978-3-319-55989-6

A survey of learning SNP systems and some new ideas

Yunhui Chen[1], Gexiang Zhang[2*], Ying Chen[1], Prithwineel Paul[2], Tianbao Wu[1], Xihai Zhang[2], and Haina Rong[2]

[1] State Grid Sichuan Electric Power Company, Chengdu 610094, China
[2] School of Electrical Engineering, Southwest Jiaotong University, Chengdu 611756, China

Abstract. In last few decades membrane computing has established itself as an important branch of natural computing. Investigating computational power, complexity aspects and real-world applications of different variants of membrane computing models have been a successful direction of research. In recent years with the invention of efficient learning algorithms, many researchers have concentrated their research into construction of intelligent biological computing systems inspired by the working of neurons in human brains to emulate human thinking. Spiking neural P systems are such type of computing systems. In this paper we survey spiking neural P systems (i.e., neural-like membrane computing models) with learning ability, their architecture, learning mechanism and compare these models, discuss their advantages and disadvantages and application of these models in solving real-world problems. We further discuss learning mechanism of associative memory network based on spiking neural P systems with white holes and weights. At the end, we discuss about some new ideas to further extend the study of membrane computing models having learning ability.

Keywords: Membrane computing · Spiking neural networks · Spiking neural P systems · Neural plasticity · Structural plasticity · Machine learning.

1 Introduction

The science and engineering of creating intelligent machines is well-known as artificial intelligence. Intelligent machines learn from their experience and this ability to learn is known as machine learning. Machine learning is an integral part of artificial intelligence. In machine learning (ML) [20] the performance of the computer systems for solving a specific task can be improved by studying the sample data set or training data set. Machine learning algorithms also can make predictions or decisions even if the system is not explicitly programmed to perform the task. It has been used in statistics, pattern recognition, neural networks, artificial intelligence, signal processing, control, data mining, email filtering, detection of network intruders and computer vision. Learning algorithms in machine learning can be divided into mainly three categories: supervised learning, unsupervised learning and reinforcement learning. Supervised learning algorithms are popular for training artificial neural networks. The main purpose of supervised learning is to learn functions to map an input to desired output by observing a set of input-output data. The main purpose of unsupervised learning is to find the similarities between the given data.

Over the years machine learning and artificial intelligence have been used as one of the most important tools to solve problems in industry as well as in academics. However until now no known efficient learning algorithm has been introduced which can emulate human thinking and achieve human-level performance at solving difficult tasks. P systems are powerful parallel and distributed computing models and these systems have been used significantly in solving computationally hard problems [1] and many problems in real-life applications [36]. Although the use of machine learning algorithms for training of artificial neural networks is extensively studied, it is relatively new for training of membrane computing models. Furthermore, spiking neural P systems are inspired by the structure and functioning of biological neurons. So the construction of machine learning algorithms in spiking neural P systems framework which can emulate human thinking and expansion of the scope of the membrane computing models for applications in the areas such as pattern recognition, image processing, robot locomotion, computer vision, sequence mining etc have become an important direction of research. The learning property is not a built-in feature in P systems, however, formal language theory itself does not exclude such features. In fact, investigation of the learning ability of the membrane computing models was listed as one of the open problems in [15] by Gh. Păun.

In 1998 inspired by the structure and functioning of living cells, Gh. Păun initiated the study of membrane computing. In last few decades, membrane computing has grown as a prominent branch of natural computing and been a popular research area for researchers in computer science, mathematics, biology, economics. Moreover, in 2003, the seminal paper by Păun [17] was mentioned by the Institute for Scientific Information, ISI, as fast breaking in the area of computer science and membrane computing was termed as *Emerging Research Front* in computer science. The fundamental component of membrane computing models is a parallel and distributive computing model known as P systems. Since the introduction of first model, different variants of P systems and their computing power, complexity aspects, real-life applications have been studied extensively [1, 36]. These models have different membrane structures and the objects inside the membranes can be also of different data structures such as multiset, strings, arrays etc. These models also can mimic the working of the living cells and based on the membrane structure, can be divided into three categories: cell-like, tissue-like and neural-like. Neural-like membrane computing models have gained popularity among researchers in recent years because of its similarities with neural networks and learning ability.

Artificial neural networks(ANNs) are computing models inspired from the structure and functioning of biological neurons. Neural networks have been a very useful tool in processing data and with learning mechanisms these models can make useful predictions. Based on the computational units, neural network models can be classified into three categories: first generation, second generation and third generation. Spiking neural networks(SNNs) are considered as third generation neural networks and SNNs are fundamentally different from first and second generation neural networks. However its functioning is more closer to the working of biological neurons. The neurons in ANNs use non-linear continuous valued activation functions to communicate with each other and a threshold is associated with the neurons such that the transmission of signals is possible only if the threshold is crossed [20]. SNNs are computationally more powerful and do not fire at each propagation cycle. Also in SNNs, neurons communicate with each other with electrical pulses known as spikes and spikes are sent when the membrane potential of a neuron is crossed [46]. Moreover, the neurons in ANNs are single, static and have continuous valued action. But in biological neurons discrete spikes are used to compute and transmission of information. This idea of spiking has been used in SNNs where the time of the spiking is more important than the spike rates. These reasons help the working of SNNs to be more closer to biological neurons. Moreover, third generation neural networks use the idea of individual spikes which is a property of biological neurons and are more hardware friendly and energy efficient [58]. Inspired by the information processing and communication of the neurons and features of spiking in spiking neural networks, Păun, Ionescu and Yokomori initiated the study of spiking neural P systems (SNP systems) in [14]. Following this many variants of SNP systems [38–41] have been introduced inspired from the features of neuroscience such as asynchronous systems in [21], astrocytes in [22], rules on synapse in [24], thresholds in [25], communication on request in [26], and synapses with schedules in [28]. Spiking neural P systems also have been used to solve computationally hard problems [29–31]. Also, SN P systems and their variants have been used to solve problems in real-life applications such as fuzzy reasoning SN P systems for fault diagnosis, image processing, fault diagnosis of electric power systems, solving combinatorial optimization problems [32–35].

SN P systems have been very effective in solving problems in theory and application because some important properties of SNNs have been incorporated into SNP systems within formal language theory framework which helped these systems to be more powerful. Also, there exist some properties which distinguish them. Like SNNs, the individual spikes are considered in the SNP systems where it is represented by an alphabet with single symbol. Moreover, the informations are encoded in the form of time difference between the firing of a specified neuron which is called as output neuron. The major difference between SNNs and SNPs is the rules present in the neurons of SNPs. The rules are written using the concepts of formal language theory. In SNNs, the current activation level of a neuron is modeled by differential equation [46]. Moreover, unlike SNNs, SNPs work as language generating and accepting device. These models can have language generating and accepting power equivalent to Turing machines [38–40]. The training of SNNs is considered very challenging. Although many supervised as well as unsupervised learning algorithms have been introduced for SNNs [46], still investigations regarding construction of training algorithms is considered as an open area of research. The intrinsic similarities with the SNNs inspired researchers to investigate the learning ability of the SNP systems. Followed by these many researchers have introduced supervised and unsupervised learning algorithms for SNP systems in [2–5, 37].

Investigating learning mechanism of membrane computing models is relatively new and promising branch of natural computing. The first model discussing the learning mechanism for membrane computing

models was introduced in 2008 by Gutiérrez-Naranjo and Pérez-Jiménez. In this model the rules in presynaptic neurons having higher probability for firing of postsynaptic neuron is learnt for Hebbian SN P unit [2]. Following this Adaptive SN P systems [3], Adaptive fuzzy SN P systems [4], SN P systems with learning function [37] were introduced. Moreover SN P systems with Hebbian learning strategy has been used for identification of nuclear export signals [11] and SN P systems with Hebbian learning function has been used for recognition of digital English letters [37]. Also the Widrow-Hoff learning algorithm introduced for AFSNP systems have been used to solve the fault diagnosis problem in power systems [5]. Although only very few learning models have been constructed in membrane computing, most of these models are simple and not capable of solving complex problems. Hence it requires further investigation.

In recent years because of the increasing popularity and usefulness of machine learning algorithms in solving complex problems, it became important to prepare a survey of the learning models discussing the architecture of membrane computing learning models, study their comparisons, advantages and disadvantages and applications in solving real world problems. Followings are motivation for preparing this survey:

1. Until now there has been no known survey on membrane computing models with learning ability. This survey will help the researchers in membrane computing community to easily understand these models along with their learning methods and application in solving real-world problems. Also will be able to compare these models easily.
2. This survey extends the scope of study of machine learning for bio-computing models. Moreover, researchers will be able to understand the difference between the learning in SNNs and SNPs having completely different mathematical structures.
3. Address some new research problems in membrane computing and machine learning.

The contributions of this paper are as follows:

1. We survey all membrane computing models with learning mechanism and analyze them in detail. Moreover, we do a throughout comparison of these models and discuss their advantages and disadvantages.
2. We discuss the real-world problems which can be solved using the membrane computing models and their learning mechanism and compare the performance of these models with traditional models.
3. Discuss an example of learning in associative memory network based on extended spiking neural P systems with white holes and weight (AMN-EWSNP).
4. Discuss future research direction by proposing methodology for extending the study of single layer network to multilayer network for membrane computing models.

This article is organized in the following manner. In Section 2, we discuss about membrane computing models with neural-like structure and their learning ability. Section 3 discusses the example of learning in associative memory network based on extended spiking neural P systems with white holes and weights (AMN-EWSNP). Section 4 discusses the conclusion and future directions of research.

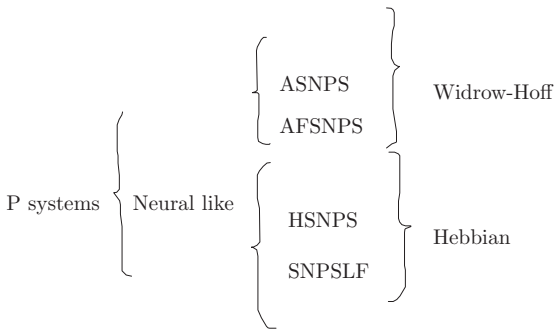


Fig. 1: Membrane computing models with learning

2 Membrane computing models with learning

Most of the learning processes in the biologically motivated computing models are inspired from the Hebbian learning rules. SNP systems are inspired from the structure and functioning of SNNs and hence studying Hebbian learning in these systems was of primary importance. In this section, we discuss Hebbian and Widrow-Hoff learning in Hebbian SN P system unit, adaptive SN P systems, and adaptive fuzzy SN P systems and we also study comparisons, advantages and disadvantages of these models. We also discuss the application of the Hebbian learning strategy in SN P systems for solving NES identification problem, Widrow-Hoff learning algorithm for SN P systems for fault diagnosis of power systems and recognition of English letters by SN P systems with Hebbian learning function. Moreover, we discuss these learning models in SN P systems according to their year of inception. We start with learning in Hebbian SN P systems unit [2] introduced in 2008. It is also the first known article discussing the concept of machine learning in membrane computing.

2.1 Unsupervised (Hebbian) Learning with SN P systems:

Hebbian theory or Hebb's postulate was introduced by D.O. Hebb in 1949 in his book *The Organization of Behavior* [19]. In this theory, he claimed that the efficiency of the synapse increases when a presynaptic cell is responsible for repeated and persistent stimulation of postsynaptic cells. This theory paved the way for new learning algorithm called as "Hebbian learning algorithm". In Hebbian learning the plasticity between the presynaptic and postsynaptic neuron is taken into account. Also, the weight associated with the synapse is updated according to the Hebbian rules in artificial neural networks. Although Hebb introduced the idea, he did not mathematically formulated the concept. There exist several mathematical formulation of Hebbian learning. Some of the popular mathematical formulation of Hebbian learning are (1) Rate-based Hebbian learning [57]; (2) Spike-based models of Hebbian plasticity [57]. Also one of the popular mathematical generalization of Hebb's rule is as follows: Let $\{\boldsymbol{p_1}, t_1\}, \{\boldsymbol{p_2}, t_2\}, \ldots, \{\boldsymbol{p_n}, t_n\}$, be the training data where $\boldsymbol{p_i} \in \mathbb{R}^d$ and the corresponding output of each input pattern is $O_i = sgn(\boldsymbol{w}^T \boldsymbol{p_i} + b)$. Then the weights are updated in the following manner until the weights converge:

$\boldsymbol{w}' = \boldsymbol{w} + \Delta \boldsymbol{w}$;

$$\Delta w = \begin{cases} \eta t_i \boldsymbol{p_i} & \text{if } O_i \neq t_i \\ 0 & \text{Otherwise} \end{cases}$$

where η is the learning rate.

Hebbian learning is an unsupervised learning where good representations are learned from unlabelled input data and it has different weight updating strategies. Naranjo and Pérez-Jiménez introduced the idea of Hebbian learning with membrane computing in the year 2008 [2].

The Hebbian learning for SN P systems is discussed in [2] with the help of the computing device called as Hebbian SN P systems unit. The main purpose of this device is transmission of the information. For any given input if the device is able to send a spike to the environment, then the device has successfully transmitted the information. Otherwise it has failed. The learning problem of the SN P system can be formulated using the Hebbian SN P unit in the following manner. A Hebbian SN P system unit of degree (m, k, p) [2] is a construct $H\Pi = (O, u_1, \ldots, u_m, v)$, where each decaying rule R_{ij} is of the form $a^k \to (a^{n_{ij}}, S)$; d_{ij} where $k \geq n_{ij} \geq 0$ and $d_{ij} \geq 0$. The decaying sequence $S = (s_1, s_2, \ldots, s_r)$ is a finite non increasing sequence of natural numbers where $s_1 = k$ and $s_r = 0$.

The idea of *decaying sequence* associated with each spiking rule of Hebbian SN P systems unit is a new concept. Moreover, after application of a rule $E/a^k \to (a^p, S); d$ where $S = (s_1, s_2, \ldots, s_r)$, for the least l such that $p \geq s_l$, the number of spikes sent to the connected neurons is s_l. The idea of *decaying sequence* is inspired from the biological behaviour of spikes, i.e., whenever any spike is reached to a neuron, the potential of the neuron increases. Moreover, the spikes inside the neuron decay with time.

Now for an input x to the Hebbian SN P unit, the success and failure of the Hebbian SN P unit depend on the non-deterministic selection of the rules present in the system. Some rules are better than the other rules, i.e., applications of some rules are correlated with the successful computation in Hebbian SN P unit. The learning problem is to identify these rules. In this model, a ***weight*** or ***efficacy*** is associated with each rule of presynaptic neuron. The rule in presynaptic neuron is of the form $a^k \to (a^{n_{ij}}, S); d$ where $S = (s_1, s_2, \ldots, s_r)$ and is associated with a weight w_{ij}. If the rule is triggered for a least l such

that $n_{ij} \times w_{ij} \geq s_l$, then s_l spikes are sent to the postsynaptic neuron after d time unit. If the time difference between the firing of a presynaptic neuron and the firing of postsynaptic neuron is less, i.e., the contribution of the presynaptic neuron initiates the firing of the postsynaptic neuron, then the efficacy or weight of the rule is more and the rules with less efficacy or weight are removed from the system.

Remarks: (1) In the original definition of SNP systems, the *decaying sequence* is not added with the firing rules. Also the idea of decay has not been used in the original definition of SN P systems and its variants. In fact in the original definition, the spikes inside the neurons in SN P systems can stay for a long period of time until consumed by a rule.

(2) The Hebbian learning strategy has been used to train a single layer neural network model in [2] where m presynaptic neurons are associated with one postsynaptic neuron. Since single layer neural networks can identify only linearly separable data, the extension of this model for complex multilayer networks and training of these networks using same Hebbian learning strategy can be a topic of further investigation.

(3) It is important to note that in this model the weights are not associated with the synapses connecting presynaptic and postsynaptic neurons, instead weight of the rules are updated and rules with less weights are removed from the system, i.e., most effective rules are learnt in this Hebbian learning algorithm.

Following the introduction of Hebbian learning algorithms for spiking neural P systems, a supervised learning algorithm, i.e., Widrow-Hoff learning algorithm for spiking neural P systems was introduced in 2010.

2.2 Supervised (Widrow-Hoff)Learning with spiking neural P systems:

Widrow-Hoff /LMS algorithm is a popular supervised learning algorithm. Least-mean square algorithm (LMS) / Widrow-Hoff algorithm was introduced by Widrow and Hoff in 1960. It is also well-known as linear perceptron training algorithm and delta rule. LMS algorithms have very simple design and it operates with single linear neuron model. Also LMS algorithms have applications in adaptive signal processing and is popular in adaptive filtering because of low computational complexity, proof of convergence in stationary environment etc [56].

In Widrow-Hoff algorithm the performance error of the system is measured by the cost function $E(W)$, i.e., $E(W) = \frac{1}{2}\sum_{k=1}^{p}(d^{(k)} - \sum_{j=1}^{m} w_j x_j^{(k)})^2$
where the input is $x_1^{(k)}, x_2^{(k)}, \ldots, x_m^{(k)}$ and $y^{(k)} = \sum_{j=1}^{m} w_j x_j^{(k)}$, $E(W) \geq 0$ and it approaches to 0 whenever $y^{(k)}$ approaches to $d^{(k)}$ where $k = 1, 2, \ldots, p$ and p represents the number of applied patterns. The LMS/Widrow-Hoff algorithm minimizes the cost function which is also known as mean square error.

In this section we discuss about Widrow-Hoff learning algorithms in membrane computing models. At first we discuss about Widrow-Hoff learning algorithm for adaptive spiking neural P systems introduced in 2010. Then we discuss Widrow-Hoff learning algorithm in adaptive fuzzy spiking neural P systems framework which was introduced in 2013. We further compare these models and discuss their advantages and disadvantages.

Adaptive spiking neural P systems: An ASN P system (adaptive spiking neural P systems) of degree $(m+1)$ [3], is a construct of the form $\Pi = (A, \sigma_1, \ldots, \sigma_m, \sigma_{m+1}, syn, I, O)$ where

- $A = \{a\}$ is the singleton alphabet (the object a is called spike);
- $\sigma_1, \ldots, \sigma_m, \sigma_{m+1}$ represent $(m+1)$ neurons where each neuron $\sigma_i = (\alpha_i, \omega_i, R_i)$ $(i \in \{1, \ldots, m+1\})$ has (i) $\alpha_i (\geq 0) \in \mathbb{R}$, and it represents the value of the spikes contained in neuron i; (ii) $\omega_i \in \mathbb{R}$ and it represents the weights on the synapses connecting the m input neurons and output neuron.
  (iii) R_i represents the rules in the neuron i. In this model each neuron has a rule of the form $\{a^n | n \geq 1\}/a^\alpha \rightarrow a^\alpha$ where $\{a^n | n \geq 1\}$ is the firing condition of neuron σ_i.
- $syn = \{(\sigma_1, \sigma_{m+1}), \ldots, (\sigma_m, \sigma_{m+1})\}$, represents the synaptic connection between neurons.
- $I = \{\sigma_1, \ldots, \sigma_m\}$ represents the input neuron set and $O = \{\sigma_{m+1}\}$ represents the output neuron set.

In this model, m neurons $\sigma_1, \sigma_2, \ldots, \sigma_m$ are connected with the neuron σ_{m+1} and each synapse between the neurons $\sigma_1, \sigma_2, \ldots, \sigma_m$ and σ_{m+1} are associated with a weight ω_i. Since the neurons only contains

firing rules of the form $\{a^n|n \geq 1\}/a^\alpha \rightarrow a^\alpha$ along with the firing condition $\{a^n|n \geq 1\}$, whenever the inputs $x_1, x_2, \ldots, x_m$ are given to the neuron $\sigma_1, \sigma_2, \ldots, \sigma_m$ respectively, all the neurons fire immediately. Moreover, the inputs are multiplied by the weights of the synapses and reach the neuron σ_{m+1} in the next moment. Also, since the neuron σ_{m+1} has a firing rule $\{a^n|n \geq 1\}/a^\alpha \rightarrow a^\alpha$, after receiving the spikes with values $\omega_i x_i (i = 1, 2, \ldots, m)$ through each synapse, the neuron will fire. The value of spike, i.e., $\omega_1 x_1 + \omega_2 x_2 + \ldots + \omega_n x_n$ is considered as the output of the adaptive spiking neural P systems. Moreover the output of the system helps to learn the weights $\omega_1, \omega_2, \ldots, \omega_n$.

The learning problem of ASNP systems is associated with learning of the weights $\omega_1, \omega_2, \ldots, \omega_m$ associated with the synapses connecting the input neurons and the output neuron from the training data. Moreover Widrow-Hoff learning algorithm is used to learn the weights in a single layer network of spiking neural P systems where m input neurons are connected with an output neuron and the learning process is done by processing training data in the adaptive spiking neural P system framework. Note that unlike in unsupervised learning, in supervised learning a function is trained to obtain desired output from the available labeled input data. The learning of the weights of the synapses using the Widrow-Hoff learning algorithm is done in the following manner:

Let $x = (x_1, x_2, \ldots, x_m) \in R_m$ be the input to the neurons where each x_i represents the input to the neuron σ_i and the training data set is $D = \{((x(1), t(1)), (x(2), t(2)), \ldots, (x(n), t(n))\}$, where the output to the system is represented by the symbol $t(i)$. Then, the Widrow-Hoff learning law is used to update the weights using the following formulas: $\omega_i(k+1) = \omega_i(k) + 2\delta e(k) x_i(k)$, and $e(k) = t(k) - y(k)$, $i = 1, 2, \ldots, m$ where the desired output, original output and the learning rate are denoted by $y(k), t(k)$ and δ respectively.

Remarks: (1) The learning problem in ASNP systems is different from the learning problem of the Hebbian SN P system unit where the weights are associated with the rules and the weights are learnt using Widrow-Hoff algorithm.

(2) In ASNP systems, the firing rules are of the form $E/a^\alpha \rightarrow a^\alpha$ where E represents a firing condition of the neuron instead of regular expression;

(3) In the original definition of SN P systems no synaptic weight is associated with the synapses connecting the neurons unlike ASNP systems;

(4) The rules in ASNP systems do not have delay;

(5) In ASNP systems the content of the neurons are not represented by the number of spikes in the neuron. Instead a real number is associated with the neurons which further represents the value of the spikes in the neuron.

(6) In the above discussed models, Hebbian and Widrow-Hoff learning algorithm have been used to train the single layer network. So the Widrow-Hoff learning algorithm in ASNP systems are only be capable of solving simple problems. Hence expanding the use of supervised learning algorithms for training of the multilayer network in ASNP systems framework can be a future direction of research.

Next we discuss learning in another variant of spiking neural P systems known as adaptive fuzzy spiking neural P systems.

Adaptive fuzzy spiking neural P systems: Adaptive fuzzy spiking neural P systems were introduced in 2013 by Wang and Peng with the motivation to solve learning problems in dynamic fuzzy knowledge prevalent in many areas of application. It is a variant of SN P systems which is capable of modelling fuzzy production rules. In [4], a dynamic fuzzy reasoning algorithm and learning algorithm using AFSN P system was developed based on the firing mechanisms of the neurons.

An AFSN P system (of degree $m \geq 1$) [4] is a 6-tuple of the form $\Pi = (A, N_p, N_r, syn, I, O)$, where

- $A = \{a\}$ is the alphabet;
- $N_p = \{\sigma_{p1}, \sigma_{p2}, \ldots, \sigma_{pm}\}$ is the proposition neuron set. For each $i (1 \leq i \leq m)$, the ith proposition neuron, i.e., σ_{pi} can be associated with a fuzzy proposition in weighted fuzzy production rules. It has the mathematical structure $\sigma_{pi} = (\alpha_i, \boldsymbol{\omega_i}, \lambda_i, r_i)$ ($1 \leq i \leq m$) where (a) $\alpha_i \in [0, 1]$ is called the (potential) value of pulse contained in proposition neuron σ_{pi}. Moreover it is used to express fuzzy truth value of the proposition associated with proposition neuron σ_{pi};

(b) $\boldsymbol{\omega_i} = (\omega_{i1}, \omega_{i2}, \ldots, \omega_{is_i})$ is the output weight vector of the neuron σ_{pi}. Each component $\omega_{ij} \in [0,1](1 \leq j \leq s_i)$ represents the weight on the j th output synapse of the neuron. The number of all output synapses of the neuron is represented by s_i;
(c) The firing/spiking rule r_i is of the form $E/a^\alpha \to a^\alpha$, where $\alpha \in [0,1]$. The firing condition is $E = \{\alpha \geq \lambda_i\}$. It means that if $\alpha \geq \lambda_i$, then the firing rule will be applicable where $\lambda_i \in [0,1)$ is called the firing threshold.

- $N_r = \{\sigma_{r1}, \sigma_{r2}, \ldots, \sigma_{rn}\}$ is the rule neuron set. The i-th rule neuron associated with a weighted fuzzy production rule is denoted by $\sigma_{ri} = (\alpha_i, \gamma_i, \tau_i, r_i)(1 \leq i \leq n)$.
(a) $\alpha_i \in [0,1]$ represents the (potential) value of pulse contained in rule neuron σ_{ri} ;
(b) $\gamma_i \in [0,1]$ represents the certainty factor. It represents the strength of belief of the weighted fuzzy production rule associated with rule neuron σ_{ri} and γ_i also represents the weight on the output synapse of the neuron;
(c) r_i is a firing/spiking rule, of the form $E/a^\alpha \to a^\beta$, where $\alpha, \beta \in [0,1]$. $E = \{\alpha \geq \tau_i\}$ is called the firing condition, that is, if $\alpha \geq \tau_i$, then the firing rule will be enabled, where $\tau_i \in [0,1)$ is called the firing threshold.
- $syn \subseteq (N_p \times N_r) \cup (N_r \times N_p)$ represents the synaptic connection between proposition neurons and rule neurons.
- $I, O \subseteq N_p$ represent input neuron set and output neuron set, respectively.

The learning problem in AFSNP systems deals with the weights associated with the synapses connecting proposition neurons with rule neurons. The synapses connecting the proposition neurons with rule neurons are associated with weights w_i. Also, weights are associated with each rule neuron. But Widrow-Hoff learning learning algorithm is used in [4] only to update the weights connecting the proposition neuron with rule neurons. Moreover, this learning problem in the framework of AFSNP systems can be transformed into a learning problem in single layer neural networks. Also the division of the rules in AFSNP systems help the learning problem to be divided into three structures which further reduces the complexity of it.

Remarks: AFSN P systems have some properties which distinguish them from SNP systems, ASNP systems and Hebbian SN P system unit:

(1) In ASNP systems and Hebbian SN P system unit input neurons are connected with an output neuron. But AFSN P systems contain two types of neurons, i.e., proposition neurons and rule neurons. Also there are no synaptic connection between any two proposition neurons or between any two rule neurons;

(2) The proposition neuron of AFSNP systems contain firing rules of the form $E/a^\alpha \to a^\alpha, \alpha \in [0,1]$ and firing rules in the rule neuron are of the form $E/a^\alpha \to a^\beta, \alpha, \beta \in [0,1]$ respectively where E represents a firing threshold instead of regular expression. These rules are different from the rules present in the original definition of SNP systems, ASNP systems and Hebbian SN P systems unit;

(3) Proposition neurons simulate the fuzzy truth values of the fuzzy production rules andAFSN P systems are constructed by simulating weighted fuzzy production rules. Moreover in ASNP and AFSNP systems the firing rules do not have any delay, and the neurons contain real numbers instead of the number of spikes in the neuron, which further represents the potential of the spikes in the neuron. Moreover mainly there exist three types of weighted fuzzy production rules, i.e., Type 1, Type 2 and Type 3 rules. Accordingly the rules in AFSNP systems constructed in [4] are also divided into three types;

(5) Unlike Hebbian SN P system unit, weights are associated with the synapses connecting the proposition neuron and rule neuron.

(6) One of the biggest advantage of AFSNP systems in learning the weights is that it has the ability to adjust automatically. Moreover even though the architecture of the network in the framework of AFSNP systems are more complex than ASNP systems, the learning problem in this model can be turned into a learning problem in single layer network. But it has not been investigated in [4], whether the weights of the synapses connecting the rule neurons and proposition neurons can be learnt using the Widrow-Hoff learning algorithms and the learning of the weights of the synapses connecting the proposition neuron and rule neuron and rule neuron and proposition neuron can be done simultaneously. Investigation of this problem can be interesting.

In the last few years researchers concentrated more on application of the learning models. Next we discuss about the application of the learning strategies of the spiking neural P systems in solving problems in real-world applications.

Types	Network Structure	Rules	Algorithm
ASN P systems	single layer	$E/a^{\alpha} \to a^{\alpha}$	Supervised (Widrow-Hoff)
AFSN P Systems	single layer	$E/a^{\alpha} \to a^{\beta}$, $\alpha, \beta \in [0,1]$ $E/a^{\alpha} \to a^{\alpha}$ $\alpha \in [0,1]$	Supervised (Widrow-Hoff)
Hebbian SN P systems unit	single layer	$a^{k} \to (a^{n_{ij}}, S); d_{ij}$	Unsupervised (Hebbian)
SNP systems with learning functions	multilayer	$a^{p}a^{*}/a^{p} \to a; 0$ $E/a^{c} \to a; d$ $a^{s} \to \lambda$	Unsupervised (Hebbian)

Table 1: Neural-like Membrane computing models, rules and learning algorithms.

2.3 Application of Hebbian and Widrow-Hoff learning with SN P systems:

Both ASN P systems and AFSN P systems can be useful in solving real-world problems. In 2010 the learning ability of adaptive spiking neural P systems was used for solving linear adaptive filtering problem [3]. Also in 2013 a solution for the knowledge learning problem using ASN P systems was provided in [4]. In 2014, a new fault diagnosis model for power systems using AFSN P system [5] was introduced by Tu, Wang, Peng and Shi . This model has simple reasoning process followed by fast speed and learning ability. Moreover, the results of fault diagnosis using AFSN P systems are further compared with existing models in [5] and from the comparison, it is clear that this model is simple and faster and diagnosis results are efficient because of the learning ability of AFSN P systems.

NES identification: In 2018, a novel computational approach was introduced by Chen, Zhang, Weng, Shi, Wu, Zheng using spiking neural P systems to identify the nuclear targeting signals. Moreover, Hebbian learning algorithms were used to update the information of the system and for identification of NES [11].

Remarks: (1) It is important to note that the learning strategy introduced in [11] is different from the learning algorithm introduced in [2]. The SN P system with Hebbian learning strategy in [11] is divided into two module: input and predict module where input neurons receive inputs from the environment and the output neurons emit spikes to the environment.

(2) In [2], weights are associated with the firing rule and if the firing of the rules in the presynaptic neuron influences the firing of postsynaptic neuron, then the weight associated with the rule increases. Otherwise, the weight of the rule decreases and the rules with less weights are removed from the system. But in [11], the weight of a synapse can be represented by a function which will increase depending on the spikes passing along it, i.e., for each passing of spikes through a synapse, the weight of the synapse is increased by Δw. So, if at any given moment, a particular neuron has spiked t times and spikes have passed through the synapse t times, then the weight of the synapse is increased by $t\Delta w$. During computation the weight of the synapses of the input module remains fixed but the weight of the synapses in the predict module are updated depending on the number of spikes passed through the synapse.

(3) The learning algorithm introduced in [2] is used to train a single layer network containing presynaptic, i.e., input neurons and one posysynaptic, i.e. output neuron. But in [11] a multilayer network of spiking neural P systems has been trained using Hebbian learning.

(4) In input module in [11], the input received by the input is in binary and the received spike train is read by the input neurons bit by bit. The predict module contains the unique rule $aa^*/a \to a$ and the initial weight of all the synapses in the predict module has weight 1. Also, unlike the input module, the predict module has multilayer architecture containing inner, hidden and outmost layer. Again, the weights of the synapses are updated using a Hebbian learning strategy. Along with it the topological structure of the input module also remains fixed but the structure of the predict module changes because of the weights of the synapses being updated.

(5) One of the main advantages of using Hebbian learning strategy for NES identification is (i) it is simple; (ii) good results can be obtained. But it remains to be investigated whether other learning algorithms can perform better in this network and also design of more complex learning strategy for large scale networks can be considered as a future direction of research.

Types	Similarities	Differences
Hebbian SN P system	single layer	efficacy, rules with decay and delay, learning best rules, input (time unit)
SN P systems with learning functions	multilayer	Hebbian learning function, input(spike train from environment), function strengthen or weaken neural connection, weight of synapse on input module fixed, recognize module increase by one
ASN P system	single layer	firing rule without, delay, neurons contain real numbers, weight learning from data
AFSN P system	single layer	model weighted fuzzy rules, dynamic fuzzy reasoning, proposition neuron, rule neuron, learning weight of proposition neuron

Table 2: Similarities and differences

Recognize digital English letters: Recently in 2019, a new variant of spiking neural P systems with Hebbian learning function is introduced in [37]. The SN P system with learning function model has been used to recognize digital English letters and the performance has been compared with Back Propagation (BP)and probabilistic neural networks. In fact, SN P systems with learning functions have better performance than these models in the test cases with low, medium and high noise [37].

Remarks: (1) The new variant of SNP system introduced by Song, Pan, Wu, Pan, Wong, Rodríguez-Patón is called as SN P system with learning function. In this model a Hebbian learning function is associated with the SN P system. This function can strengthen or weaken the connection between the neurons during the computation. This model has some similarities and differences with SN P system with Hebbian learning strategy in [11] and the Hebbian SN P system unit in [2]:

(I) SN P systems with learning function has a multilayer architecture like the SN P system with Hebbian learning strategy in [11]. Also in this model, the input neurons receive input from the environment in the form of spike trains and the output neurons emit spikes to the environment.

(II) SN P system with Hebbian learning function contains two modules, i.e., input module and recognize module. Similarly as in [11], weight of the synapses in the input module are fixed and they do not change during computation. But whenever any neuron in the recognize module spikes, the weight of the synapse starting from the current neuron is increased by one. Also the neurons in the recognize module are arranged in three layers: innermost, middle and outermost. Moreover, the neurons in this module receive spikes from the input module and from its neighbours. The topological structure of the recognize module is complex. In recognize module, the information is transmitted from the neurons in innermost layer to the neurons in outermost layer. Moreover, the spikes in the each layer is emitted to four direction and neurons having same direction are connected with the neurons in the upper layer. But there exists no connection if the direction is opposite.

(2) In SN P systems with Hebbian learning function, the function used to strengthen, rebuild and weaken the connection between neurons are very simple in nature. So natural question arises whether complex function can have better performance in solving the problems. Also can better performance be obtained by using more complex networks in the framework of SN P systems? Moreover, investigations regarding whether these models can solve other problems in real-world application can be a topic of further research in this direction.

The learning algorithms and rules in the SNP systems have been summarized in Table 1. Table 2 summarizes similarities and differences of neural-like P system models with learning ability.

Next we give example of a network having spiking neural P systems with white holes and weights as fundamental computing units which further expands the scope of creating membrane computing models having the features of Hopfield networks and learning process of this network.

3 Example

In this section, we give an example of learning in a network constructed in the framework of spiking neural P systems. This model is called associative memory network based on spiking neural P system with white holes and weights(AMN-EWSNP). This example will further help the readers to understand learning mechanism in membrane systems. Moreover, this network of membrane systems is an attempt to incorporate the idea of Hopfield network in membrane computing framework. This model also initiates the study of constructing new networks using membrane computing models by incorporating the concepts of structure and functioning of traditional neural networks.

3.1 Structure of AMN-ESNPA

Hippocampus is a complex structure in brain which is important for forming new memories [43]. In Hippocampus, memory generation is done through different forms of expressions by neurons [45]. Moreover, in spite of advancements in the neuroscience in last few decades, the mechanism of complex interactions between neurons is still not clear. One such example is that the CA3 region in the hippocampus, as Marr proposed has a structure of recursion which can have associative memory [44] based on the biological facts and some characteristics of Hopfield networks. Also, the network based on spiking neural P system with white holes [22] and weights [23] can have associative memory. The structure of AMN-EWSNP, shown in Fig.1 is described in the following manner

$$\Pi = (O, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, in, out) \tag{1}$$

where:

- $O = \{a\}$ is the alphabet (a is called spike);
- $\sigma_1, \sigma_2, \ldots, \sigma_n$ are neurons of the form $\sigma_i = (n_i, R_i), 1 \leq i \leq n$, where (a) n_i is the initial potential in σ_i which represents the state of the pixels; (b) R is a finite set of spiking rules of form $a^*/a^{all} \rightarrow a$, where a^* is a regular expression over a, all represents the total number of spikes in the neuron. After application of this rule if the regular expression is satisfied, then all the spikes will be consumed and one spike will be send to the neurons connected to the neuron where the rule is applied;

- $syn \subseteq \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\} \times \mathbb{R}_c$ are synapses between neurons, where $i \neq j, w \neq 0$, for each $(i, j, w) \in syn$, the set of $\mathbb{R}_c \subseteq \{-1, 1\}$ and for each $(i, j) \in \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$ there exists at most one synapse $(i, j, w) \in syn$;
- $in(out) = \{\sigma_{q1}, \sigma_{q2}, \ldots, \sigma_{qn}\}$ is the set of input as well as output neurons.

The neurons $\sigma_1, \sigma_2, \ldots, \sigma_n$ can send spikes to other neurons through synapses when the condition of regular expression is met. Moreover the weights on the synapses remain fixed during the computation. Note that whenever the neurons spike, it is multiplied by the weight on the synapses and sum of these spikes is send to neurons according to the synaptic connections. Also whenever the condition in the form of regular expression is satisfied, then all spikes in σ_i are consumed and only one spike is sent. Otherwise no spikes will be sent. The above procedure is repeated until there is no change in the output or the maximum number of execution steps is reached. Then the computation is halted and compared with the original data.

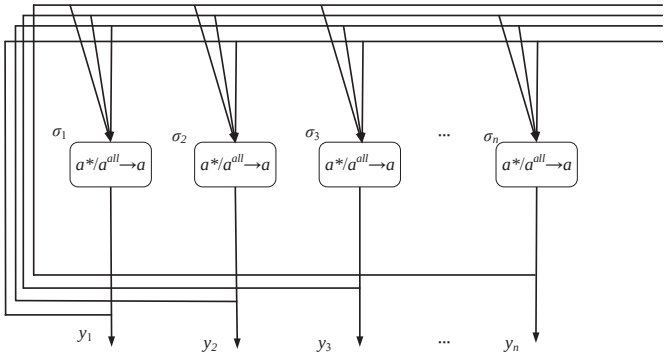


Fig. 2: An example of AMN-EWSNP structure

3.2 Weights and energy function

The weights on synapses are specially designed for the above network. In this case, the storage prescription in [53, 54] is employed to design the weights in the network. Suppose the set of states which wish to store are $x^s, s = 1, \ldots, n$. Then, the weights are updated in the following manner

$$w_{ij} = \sum_s (2x_i^s - 1)(2x_j^s - 1)(i \neq j), w_{ii} = 0. \tag{2}$$

The energy function is another important feature in the model. The role of energy function is to estimate whether the system is stable or not. If the weights are designed by using the storage prescription, the system will be stable at the end [55]. The energy function is as follows:

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} x_i x_j \tag{3}$$

where E is the value of energy function, x_i is the state of the output neuron.

3.3 Experiments and results

In this subsection, the binarized images of number one to three are encoded by $1/-1$ and the weights are devised through standard digital images in three levels of noise. Then AMN- EWSNP is constructed to associate memory with the image of the number.

Weight design by standard number image: Each number is represented by a black-white image in the size of 10×10 pixels. The representation of "two", for example, is shown in Fig.3. Each black and white image can be represented by a 10×10 array, where the value of black block is 1, the other block is -1.

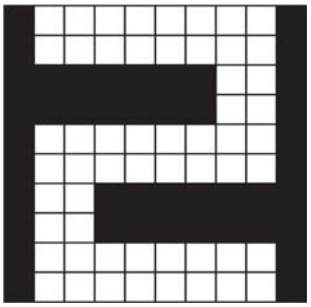
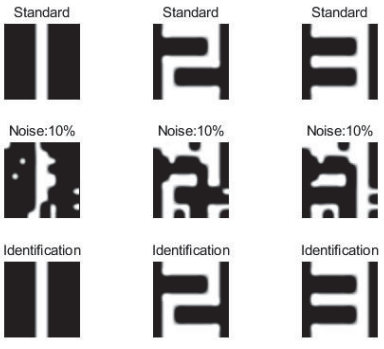


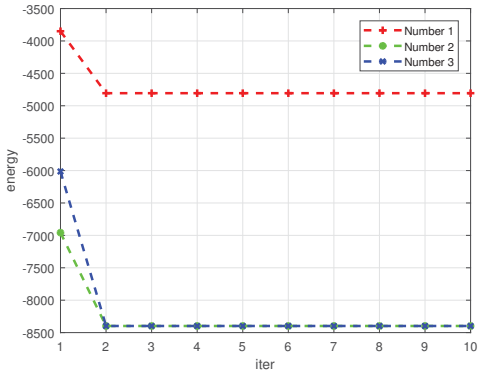
Fig. 3: Representing number "two" by 10×10 pixels array

After acquiring the standard image of number from one to three, the weight matrix is obtained by Hebb learning rule. Once a weight matrix is fixed, then the weight matrix is used to associate memory with the image with noise.

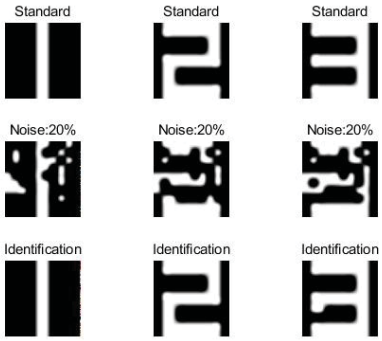
The result of AMN-EWSNP: The number image with noise can be acquired by generating random numbers to determine the need to modify the position of lattice [27]. In this paper, the distortion probability with $10\%, 20\%, 30\%$ are employed to test the ability of AMN-EWSNP and the maximum number of execution steps is 10. The result of the simulation and the energy function in this paper are shown in Fig.3. It is not difficult to see that this network can identify the set of numbers effectively.



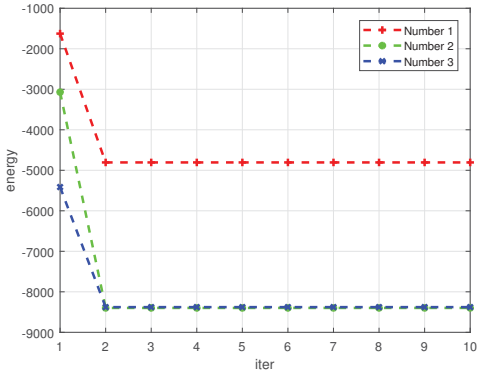
(a) Fig.3(a)



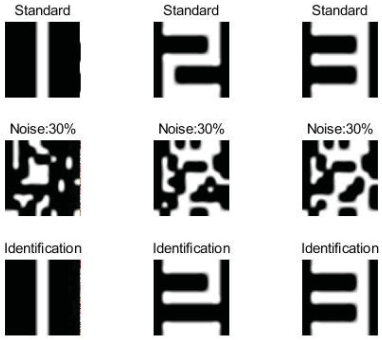
(b) Fig.3(b)



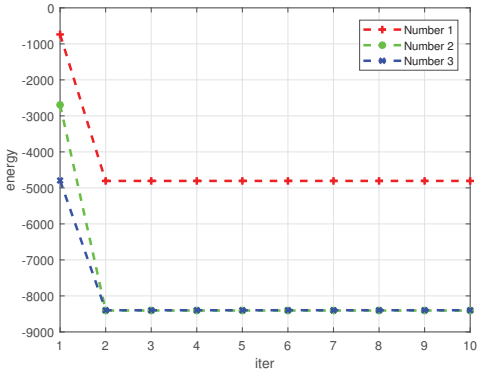
(c) Fig.3(c)



(d) Fig.3(d)



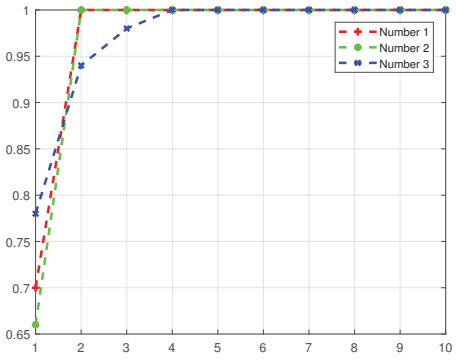
(e) Fig.3(e)



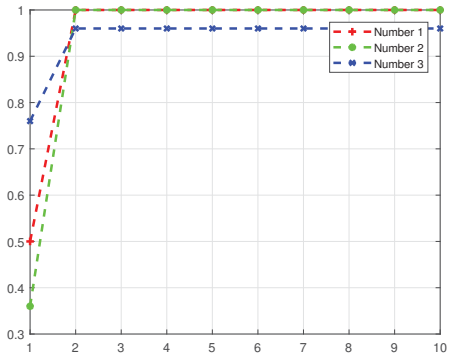
(f) Fig.3(f)

Fig.3.(a),Fig.3.(c),Fig.3.(e) are the result of simulation with noise 10%, 20%, 30% respectively. Fig.3.(b),Fig.3.(d),Fig.3.(f) are the energy function with noise 10%, 20%, 30% respectively.

The difference between the simulation result and the standard image is represented by the cosine of the angle between the vectors. The rate of the similarity during the iterative process is shown in the following figures.

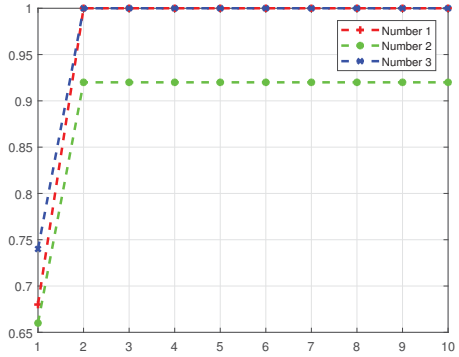


(g) Fig.4

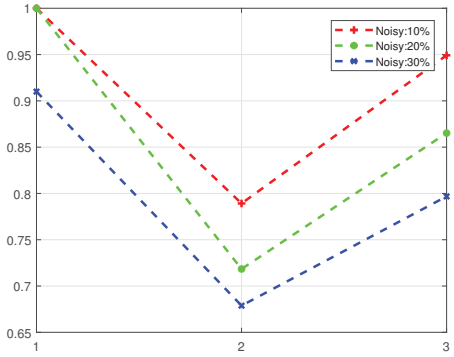


(h) Fig.5

Figs. 4, 5 and 6 represent the rate of the similarity during iteration with noisy of 10%, 20% and 30% respectively. After running the program large number of times, it is easy to find out that if the difference is small or there is no difference, the value of cosine will be greater than 0.95. Otherwise it will be less



(i) Fig.6



(j) Fig.7: average accuracy rate of the simulation

than 0.95. So the threshold of similarity rate is set as 0.95. It means that if the rate of the similarity is greater than 0.95, then the result is believed to be recognized correctly, otherwise it is deemed to be misidentified. In order to avoid the scenario, repeat the above simulation 100 times. The result of the simulation is shown in Fig.7.

4 Conclusion and future research direction

In recent years, deep learning, i.e., research regarding the deep neural networks [47] has gained huge interest from scientists and engineers because of its outstanding performance in machine learning and artificial intelligence and different areas of applications. Deep neural networks(DNNs) are relatively new variant of artificial neural networks with multiple layers between the input and the output layer which is inspired from the hierarchies in cortical visual information processing. Also the availability of powerful computing hardware, large amount of sample data and efficient learning algorithms have fostered the success of deep neural networks. Moreover, brain inspired deep neural networks have achieved human level performance in classifying images with natural scenes [51]. Also in many fields deep SNNs have been employed to build new state of art technologies where the deep neural networks have even surpassed the accuracy of the humans [52]. Following the success of DNNs in which the neurons have non-spiking behaviours, some researchers have tried to construct more powerful computing device which can mimic the working of the human brain more closely by incorporating the idea of multilayer architecture of deep neural networks and information processing between the neurons as in spiking neural networks. These models are known as deep spiking neural networks (deep SNNs) [48–50]. Moreover, in recent years one of the challenging direction of research has been to investigate the comparison of the performance of spiking deep networks and deep neural networks. Also training of deep SNNs is a challenging topic of research.

In this article we have discussed all the neural-like membrane computing models with learning ability. We have already mentioned that the learning models in the framework of spiking neural P systems considered until now have very simple structure. Most of these models have either single layer neural network or simple multilayer architecture and simple Hebbian learning algorithms in multilayer architecture have been used to train multilayer networks in the SN P systems framework to solve NES identification and digital English letter identification problems. But until now no supervised learning algorithm has been introduced to train multilayer networks in the SN P systems framework.

Single layer neural networks are not capable of solving complex problems and only can identify linearly separable data, the extension of the study of construction of new supervised and unsupervised learning algorithms for training of multilayer networks in the SN P systems framework can be a topic of future investigations. Moreover, multilayer perceptrons use a well-known supervised learning technique known as backpropagation for training of the network. Since SN P systems have formal language theory framework, it will be very interesting to construct the training algorithms like backpropagation algorithms for multilayer SN P systems using the concepts of formal language theory. Now we discuss about another important aspect of learning in the neurons, i.e., structural plasticity.

Neural plasticity is an important biological feature of SNNs. It mainly deals with the modifications happening in biological neurons. Neural plasticity can be divided into two aspects: structural plasticity

and functional plasticity. Until now we have mainly focused on the functional plasticity for learning the weights where the weight of the connecting synapses increases or decreases depending on the arrival of the spikes from the presynaptic neurons. But in case of structural plasticity, the connectivity between the neurons is modified by synapse deletion and creation.

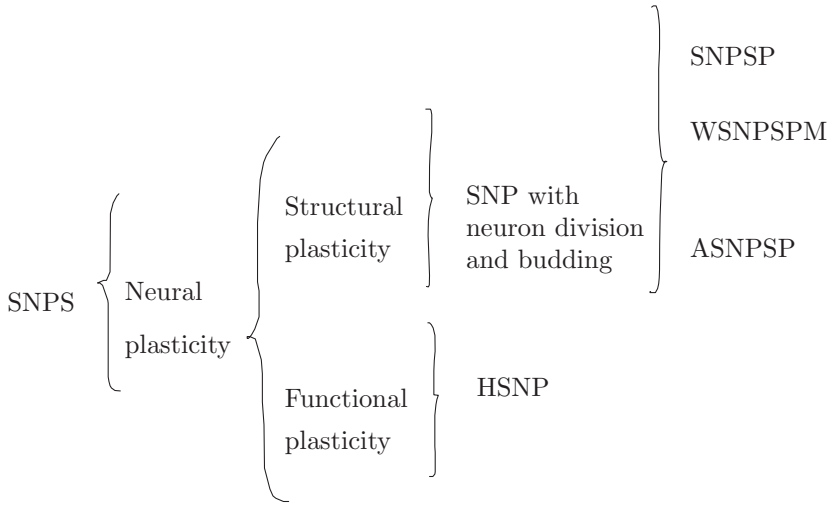


Fig. 4: SN P systems with structural plasticity and its variants

In SNNs, huge number of models have been constructed and these models have been used for learning of the synaptic weight. But until now a little investigation has been towards the effect of structural plasticity in the learning of the SNNs. Moreover, how the structural plasticity and STDP (spike-time-dependent-plasticity) work together is also not has been investigated extensively. In [13], a large scale model has been created to learn input encoding along with relation inputs and guessing the missing inputs using leaky integrate and fire neurons, STDP, homeostasis, recurrent connections and structural plasticity. Furthermore, in [13] Spiess and George have investigated the error and the amount of noise occurring in the network's responses with and without having structural plasticity. Also investigated the impact of the structural plasticity in the learning speed of the network. The use of structural plasticity in the learning process has been turned out to be useful. Moreover, the noise of the response can be reduced significantly using structural plasticity. In fact, it prevents spikes with high error rates and the time to learn the synaptic weights can be reduced by using structural plasticity with pruning. These advantages of structural plasticity in spiking neural networks give us the motivation to investigate the learning process in other computing models such as spiking neural P systems with structural plasticity.

The idea of structural plasticity for membrane computing models (SN P systems) was introduced in [6]. Subsequently, many variants such as ASNPSP, WSNPSPM have been introduced and their computational power have been investigated [7–10, 12, 42]. But until now no investigation has been initiated towards the direction of learning ability of these models. So, learning ability of SN P systems with structural plasticity and application of these models can be a direction of future research.

The process of synapse creation and deletion improves the learning speed in SNN models. The synapse creation-deletion between the neurons in SN P systems with structural plasticity is controlled by the plasticity rules in these models. So, it will be interesting to investigate, how synapse creation-deletion mechanism using the plasticity rules affects the learning process of the weights associated with rules and synapses in spiking neural P systems with structural plasticity. This investigation also can be further extended for all the variants of spiking neural P systems with structural plasticity.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (61972324, 61672437, 61702428), the New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0043) and the Sichuan Science and Technology Program (2018GZ0185, 2018GZ0086).

References

1. Gh. Păun, G. Rozenberg, A. Salomaa. The Oxford Handbook of Membrane Computing, 2010, ISBN 0199556679, 9780199556670, Oxford University Press, Inc., New York, NY, USA.
2. M.A. Gutierrez-Naranjo, M.J. Pérez-Jiménez: Hebbian learning from Spiking Neural P systems view, WMC9, LNCS 5391, 217–230, 2008.
3. H. Peng, J. Wang: Adaptive spiking neural P systems, Sixth International Conference on Natural Computation, ICNC 2010, 3008–3011, 2010.
4. H. Peng, J. Wang: Adaptive fuzzy spiking neural P systems for fuzzy inference and learning, International Journal of Computer Mathematics, 90, 857–868,2013.
5. M. Tu, J. Wang, H. Peng, P. Shi: Application of adaptive fuzzy spiking neural P systems in fault diagnosis of power systems, Chinese Journal of Electronics, 23(1), 87–92,2014.
6. F.G.C. Cabarle, H. N. Adorna, M. J. Pérez-Jiménez, T. Song: Spiking neural P systems with structural plasticity, Neural Computing and Applications, 26, 1905–1917, 2015.
7. T. Song, L. Pan: A normal form of spiking neural P systems with structural plasticity, International Journal of Swarm Intelligence, 1(4), 344–356,2015.
8. F.C.G. Cabarle, H. N. Adorna, M. J. Pérez-Jiménez: Asynchronous spiking neural P systems with structural plasticity, Unconventional Computation and Natural Computation, UCNC 2015, LNCS 9252, 132–143, 2015.
9. F.C.G. Cabarle, R. T. A. D.L Cruz, X. Zhang, M. Jiang, X. Liu, X. Zeng: On string languages generated by spiking neural P systems with structural plasticity, IEEE Transactions On Nanobioscience, 17(4), 560–566, 2018.
10. M. Sun, J. Qu: Weighted spiking neural P systems with structural plasticity working in sequential mode based on maximal spike numbers, AIP Conf. Proc., 1890, 040049-1-4, 2017.
11. Z. Chen, P. Zhang, X. Weng, X. Shi, T. Wu, P. Zheng: A computational approach for nuclear export signals identification using spiking neural P systems, Neural Computing and Applications, 29, 695–705, 2018.
12. F.G.C. Cabarle, H. N. Adorna, M. J. Pérez-Jiménez: Sequential spiking neural P systems with structural plasticity based on max/min spike number, Neural computing and applications, 27, 1337–1347 ,2016.
13. R. Spiess, R. George, M. Cook, P. U. Diehl: Structural plasticity denoises responses and improves learning speed, Frontiers in Computational Neuroscience, 10(98), 2016.
14. M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems, Fundamenta Informaticae, 71, 279–308, 2006.
15. Gh. Păun, Tracing some open problems in membrane computing, Romanian Journal of Information Science and Technology, 10 (4), 303–314, 2007.
16. Gh. Păun, Twenty six research topics about spiking neural P systems, in: Proc. Fifth Brainstorming Week on Membrane Computing, January, 263–280, 2007.
17. Gh. Păun, Computing with membranes: Journal of Computer and System Sciences 61 (1), 108–143, 2000.
18. Pico Caroni, Flavio Donato, Dominique Muller, Structural plasticity upon learning: regulation and functions, Nature Reviews Neuroscience, 13, 478–490, 2012.
19. D.O. Hebb: The Organization of Behavior. Wiley, New York ,1949.
20. T. M. Mitchell: Machine Learning. McGraw-Hill, New York, 1997.
21. M. Cavaliere, O. Egecioglu, O.H. Ibarra, S. Woodworth, M. Ionescu, Gh. Păun: Asynchronous spiking neural P systems: decidability and undecidability, in: M.H. Garzon, H. Yan (Eds.), Proc. 13th Int. Meeting on DNA Computing, Memphis, USA, LNCS 4848, Springer, Berlin, 246–255, 2008.
22. L. Pan, J. Wang, H. Hoogeboom: Spiking neural P systems with astrocytes, Neural Computation, 24 (3), 805–825, 2012.
23. L. Pan, Gh. Păun : Spiking neural P systems with anti-spikes, International Journal of Computer, Communication and Control, 4, 273–282, 2009.
24. T. Song, L. Pan, Gh. Păun: Spiking neural P systems with rules on synapses. Theoretical Computer Science, 529, 82–95, 2014, https://doi.org/10.1016/j.tcs.2014.01.001.
25. F. Cabarle, H. Adorna, M. J. Pérez-Jiménez: Sequential spiking neural P systems with structural plasticity based on max/min spike number, Neural Computing and Applications, 27 (5), 1337–1347, 2016.
26. T. Song, L. Pan, Spiking neural P systems with request rules, Neurocomputing, 193, 193–200, 2016.
27. T. Song, F. Gong, X. Liu, Y. Zhao, X. Zhang : Spiking neural P systems with white hole neurons, IEEE Transactions on NanoBioscience, 15(7), 1-1, 2016.

28. F. G. C. Cabarle, H. N. Adorna, M. Jiang, X. Zeng: Spiking Neural P Systems With Scheduled Synapses, IEEE Transactions on NanoBioscience, 16 (8), 2017.
29. L. Pan, G. Păun, M. J. Pérez-jiménez: Spiking neural P systems with neuron division and budding, Science China Information Sciences, 54 (8), 1596–1607, 2011.
30. A. Leporati, G. Mauri, C. Zandron, Gh. Păun, M. J. Pérez-Jiménez: Uniform solutions to SAT and Subset Sum by spiking neural P systems, Natural Computing, 8 (4), 681–702, 2009.
31. T.-O. Ishdorj, A. Leporati, L. Pan, X. Zeng, X. Zhang: Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources, Theoretical Computer Science, 411 (25), 2345–2358, 2010.
32. H. Peng, J. Wang, M.J. Pérez-Jiménez, H. Wang, J. Shao, T. Wang: Fuzzy reasoning spiking neural P system for fault diagnosis, Information Sciences, 235, 106–116, 2013.
33. T. Wang, G. Zhang, H. Rong, M.J. Pérez-Jiménez: Application of fuzzy reasoning spiking neural P systems to fault diagnosis, International Journal of Computer, Communication and Control, 9 (6), 786–799, 2014.
34. T. Wang, G. Zhang, J. Zhao, Z. He, J. Wang, M. Pérez-Jiménez: Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems, IEEE Transactions On Power Systems, 30 (3) , 1182–1194, 2015.
35. G. Zhang, H. Rong, F. Neri, M.J. Pérez-Jiménez: An optimization spiking neural P system for approximately solving combinatorial optimization problems, International Journal of Neural Systems, 24 (5) , 1440006, 2014.
36. G. Zhang, M. J. Pérez-Jiémnez, , Gheorghe, M.: Real-life Applications with Membrane Computing. Emergence, Complexity and Computation, Springer, 2017.
37. T. Song , L. Pan, T. Wu , Z. Pan, D. Wong, A. Rodríguez-Patón : Spiking neural P systems with learning functions, IEEE Transactions On Nanobioscience, 2019.
38. T. Song, X. Zeng, P. Zheng, M. Jiang, A. Rodríguez-Patón: A Parallel Workflow Pattern Modelling Using Spiking Neural P Systems with Colored Spikes, IEEE Transactions On Nanobioscience, 17(4), 474–484, 2018.
39. T. Song, A. Rodríguez-Patón, P. Zheng, X. Zeng: Spiking Neural P Systems with Colored Spikes, IEEE Transactions on Cognitive and Developmental Systems, 10(4), 1106–1115, 2018. .
40. X. Zeng, L. Pan, M. J. Pérez-Jiménez: Small universal simple spiking neural P systems with weights, Science China Information Sciences, 57(9), 1–11, 2014.
41. R. Freund, M. Ionescu, M. Oswald: Extended spiking neural P systems with decaying spikes and/or total spiking, International Journal of Foundations of Computer Science, 19(5), 1223–1234, 2008.
42. F. G. C. Cabarle, R. Tristan, A. D. L. Cruz, X. Zhang, M. Jiang, X. Liu, X. Zeng: On String Languages Generated by Spiking Neural P Systems With Structural Plasticity, IEEE Transactions On Nanobioscience, 17(4), 560–566, 2018.
43. R. J. Douglas: The hippocampus and behavior, Psychological bulletin, 67(6), 416, 1967.
44. D. Marr, D. Willshaw , B. McNaughton: Simple memory: a theory for archicortex. From the Retina to the Neocortex, 59–128, Birkhäuser Boston, 1991.
45. Y. Zhang : The neural mechanisms of pain-related affect and memory. Progress in natural science, 16(4), 338–345, 2006.
46. W. Maass, Networks of spiking neurons: The third generation of neural network models. Neural Networks, 10 (9), 1659–1671. doi:10.1016/S0893-6080(97)00011-7. ISSN 0893-6080, 1997.
47. J. Schmidhuber : Deep Learning in Neural Networks: An Overview. Neural Networks. 61, 85–117. arXiv:1404.7828,2015, doi:10.1016/j.neunet.2014.09.003.
48. M. Pfeiffer, T. Pfeil: Deep Learning With Spiking Neurons: Opportunities and Challenges. Frontier Neuroscience, 12(774), PMID: 30410432, 2018, doi: 10.3389/fnins.2018.00774.
49. C. Lee, P. Panda, G. Srinivasan, K. Roy: Training Deep Spiking Convolutional Neural Networks With STDP-Based Unsupervised Pre-training Followed by Supervised Fine-Tuning, Frontier Neuroscience, 12(435), PMID: 30123103, 2018, doi: 10.3389/fnins.2018.00435.
50. A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, A. Maida: Deep learning in spiking neural networks. Neural Networks, 111, 47–63, 2019, doi: org/10.1016/j.neunet.2018.12.002.
51. A. Krizhevsky, I. Sutskever, G. E Hinton : ImageNet classification with deep convolutional neural networks, In Advances in Neural Information Processing Systems, 1097–1105, 2012.
52. R. Geirhos, D. H. J. Janssen, H. H. Schútt, J. Rauber, M. Bethge, F. A. Wichmann: Comparing deep neural networks against humans: object recognition when the signal gets weaker, arXiv:1706.06969v2 [cs.CV] 11 Dec 2018.
53. L. N. Cooper: A possible organization of animal memory and learning. In: Lundqvist B, Lundqvist S (eds) Proceeding of the Nobel symposium on collective of physical systems. Academic Press, New York, 252–264,1973.
54. L. N. Cooper, F. Liberman, E. Oja: A theory for the acquisition and loss of neuron specificity in visual cortex, Biological Cybernetics, 33(9), 1979, doi: org/10.1007/BF00337414.
55. J. J. Hopfield : Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 79(8), 2554–2558, 1982.
56. S. S. Haykin, B. Widrow (Editor): Least-Mean-Square Adaptive Filters, Wiley, 2003, ISBN 0-471-21570-8.

57. W. Gerstner, W. M. Kistler: Mathematical formulations of Hebbian learning, Biological Cybernetics, 87, 404–415, 2002. doi: 10.1007/s00422-002-0353-y.
58. M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, T. Masquelier: SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks With at Most One Spike per Neuron, Frontiers in Neuroscience, 13(625), 2019, doi: 10.3389/fnins.2019.00625.

FPGA Implementation of Robot Obstacle Avoidance Controller based on Enzymatic Numerical P Systems

Zeyi Shang[1,2], Sergey Verlan[2], Gexiang Zhang[1⋆], and Ignacio Pérez-Hurtado[3]

[1] School of Electrical Engineering, Southwest Jiaotong University, Chengdu, Sichuan, China; zeyi.shang@lacl.fr, zhgxdylan@126.com
[2] Laboratoire d'Algorithmique, Complexité et Logique, Université Paris Est Créteil, Créteil, France; verlan@u-pec.fr
[3] Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, University of Seville, Seville, Spain; perezh@us.es

Abstract. It is a long-cherished wish to implement numerical P systems (NPS) on a parallel architecture so that its large scale parallelism can be exploited to speedup computation tremendously. FPGA is a reconfigurable hardware in which operations are triggered so synchronized by edge or level of activating signals, making it an eligible platform to implement NPS and its variant, enzymatic numerical P system (ENPS). In this article, a NPS and a ENPS designed as robot controllers are implemented in FPGA, achieving a speedup of 10^5 comparing to software simulation. FPGA hardened NPS in this research can be regarded as a heterogeneous multicore processor since membranes inside work as processing units which possess different functions. FPGA hardened NPS is imparted universal asynchronous receiver/transmitter (UART) communication ability to push it closer to real-life application. FPGA hardened ENPS consume less hardware resources and power for less complicate membrane structures and processes.

Keywords: Membrane Computing, numerical P system, enzymatic numerical P system, Field Programmable Gate Array (FPGA), robot membrane controller, universal asynchronous receiver/transmitter (UART)

1 Introduction

As a new branch of nature computing, the research of membrane computing practical application can not keep pace with its fruitful achievements in theoretical aspect. This situation was taken seriously by membrane community and some scholars have engaged in applications of P systems ever since a long time ago. Keeping in mind the biological background of membrane computing, using P systems as modeling framework for biological processes and ecosystems were

⋆ Corresponding author

the early explorations to apply membrane computing, referring [11][22] [23]. As a newly research progress in ecosystem modeling for P system application, the Catalan Pyrenees bearded vulture ecosystem is modeled with P systems which were simulated by CUDA-enabled GPUs [9]. By resorting to P systems as modeling framework, Giant panda population dynamics modeling is another target under research. It is noted here that the non-determinism of P systems is a valuable nature for biological and ecological system modeling. Nevertheless, for engineering applications, non-determinism is the property trying to avert. Beginning from 2015, several variants of fuzzy spiking neural P systems (FSNP) have been used in power system fault diagnosis [27] [19] [5] [31], setting a new direction for P system applications. By setting one rule per membrane, the non-determinism of FSNP is removed. The large scale parallelism of P systems turned out to be quite favorable for real life applications.

A special cell-like P system whose objects are not symbols but real number variables, numerical P system (NPS), was initiated in 2006 [21], aiming at using P systems to the potential applications in economics. In a NPS, if every membrane contains only 1 program, then the computing process and its result are deterministic, for there are no any other possibilities of programs to use. In contrary, if one membrane contains multiple programs, one has to be chose randomly to use, bringing in the non-determinism to NPS. While for most real life applications, the output of a computing device should be deterministic, so that it can be used as a key parameter to exert influences on industry processes or actuators. At least under this circumstance, the non-determinism of P systems should be avoided. In order to improving the space efficiency, simplifying the membrane structures while keeping the determinism, enzymatic numerical P system (ENPS) was introduced in 2010 [16] on the basis of NPS. The catalytic function of enzyme variables allow multiple programs contained and executed in parallel in one membrane. This is done by judging the usability of each program by comparing the quantity of every variant to the quantity of enzyme involved in the same program, namely, if the amount of enzyme is lager than the minimum value of all variables located in the left-hand-side of a program, then this program is active and can take place. Otherwise it is inactive and will not carry out. For the filtering function of enzyme variables, the computation process of ENPS is deterministic, so as the results. For performing the same function, ENPS has much simpler membrane structures than NPS, making ENPS more practical to modeling complex functional components.

Researchers began to investigate the computational power and special attributes of NPS and ENPS after these two models were presented. NPSs with migrating variables were studied in [34] while NPSs with production thresholds were analyzed in [13]. Universal ENPSs with small number of enzymatic variables are discussed in [35]. String languages generated by sequential NPSs were deliberated in [33]. In [14], NPSs with production thresholds was considered. In [12], four recent research topics on numerical P systems were summarized. In [26], universality of ENPSs was examined. Enzymatic numerical P systems using elementary arithmetic operations were investigated in [6]. The computa-

tional power of ENPSs working in the sequential mode was studied in[32]. A parallel bio-inspired framework for numerical calculations using ENPS with an enzymatic environment was constructed in [15]. ENPSs for basic operations and sorting were designed in [8]. The ways of how to improving the universality results of ENPS were researched in [25]. The pole balancing problem with ENPS was discussed in[7].

Since 2011, adopting NPS and ENPS to model autonomous mobile robots controller has been another research highlight of P system applications. As the first case putting NPS to real life application, in [4], three NPSs are developed as the controllers for Khepera III and e-puck robots to perform obstacle avoidance, wall following and following leader behaviors. These three NPSs are simulated by a software called SNUPS, which is designed as Java servlet. When running the robot, it invokes the SNUPS engine. After computing, the results will be returned to robot to control motors' speeds, performing specific behaviors. Both experiments on simulated robots and real robots were conducted to verify the control effect of NPS. The first ENPS robot controller performing obstacle avoidance behavior was proposed in [17]. It is a general controller not targeting particular robots. While no experiments were conducted by the ENPS software simulator eSNUPS [16], which is an extended version of SNUPS. The portability of NPS and ENPS robot controllers was validated in [24] by adapting the control law and the number and placement of sensors, and the dimension parameters of robots. ENPSs with different functionality were developed later on, expanding the utilizing range of ENPS besides robot motion control. For instance, an ENPS do the robot localization was presented in [18]. Robot trajectory tracking ENPS was designed in [28]. Particle swarm optimization based robot path planning ENPS was introduced in [29]. ENPSs performing image edge detecting were proposed and validated in GPU [30].

The notion of P systems' *simulation* and *implementation* should be clarified explicitly in the context of this paper, so that what had been done can be better expounded. When speaking of software *simulation* of P systems, the *simulation* means that although expected results are obtained, the way how to compute is different from the exactly theoretical procedures of P systems. Specifically, employing the NPS software designed by high level programming language, like C++ or Java, to compute a result, when the software running, the codes inside are executed line by line. This operating mode lost the large scale parallelism for all the applicable rules written in high level programming language should actuate concurrently, according to the parallelism of P systems. In short, it is formally correct to simulation P systems with software on CPU. Keeping in mind that digital integrated circuits in CPU perform the operations in reality, designing and manufacturing parallel circuits, then running P systems on these circuits is the best way to *simulate* P systems. Therefore, running P systems on parallel circuits so that these circuits perform parallelism just as P systems do is defined as *implementation* of P systems. From a hardware perspective, the software simulation of P systems is CPU simulation of P systems actually,

for it tries to simulate P systems with partial parallel circuits (multi/many-core architecture CPU/GPU).

The purpose of this paper is to implement robot obstacle avoidance membrane controllers based on (enzymatic) numerical P systems in FPGA, providing the procedure & methodology of FPGA implementation of (E)NPS. Then add universal asynchronous receiver/transmitter (UART) communication to target NPS so it can be used to replace on-board computer to control target robot in the future. The experiments together with performance comparison between CPU simulation (software simulation) and FPGA implementation are given in details. Contents are organized as follows: Section 2 presents definition of (E)NPS. Section 3 and Section 4 describes the design of target NPS and ENPS RTL model respectively. Section 5 introduces FPGA implementation flow and calculation speedup achieved by FPGA hardened (E)NPS. Section 6 compares several quality attributes of FPGA hardened target NPS and ENPS. Section 7 expounds how to impart UART communication ability to target FPGA NPS. Conclusions are drawn in Section 8.

2 Numerical P system and enzymatic numerical P system

The formal definition of a numerical P system was presented in [21] and paraphrased here for a better understanding of NPS. While the motivation of raising NPS is to deal with potential applications in economic domain in which involves large scale real number variables, NPS was put to use in robot motivation control at first. The numerical P system is called NPS for short and enzymatic numerical P system is abbreviated as ENPS hereinafter.

Definition 1. *A numerical P system is the construct*

$$\Pi = (m, H, \mu, (Var_1, Pr_1, Var_1(0)), \ldots, (Var_m, Pr_m, Var_m(0)))$$

where

1. $m > 0$ *is the number of membranes (the system degree);*
2. H *is the membrane label set storing the labels of each membrane;*
3. μ *is the membrane structure;*
4. Var_i, Pr_i, $Var_i(0)$ *are variables, programs and initial values of variables in membrane* i, $1 \leq i \leq m$.

Comparing to the multiset rewriting rules in classical symbol object P systems, the programs are quite unusual with production function in the left hand side and repartition protocol in the right hand side, taking on the form

$$P_{li} : F_{li}(x_{1i}, \ldots, x_{k_i i}) \rightarrow c_{l1}|v_1 + \cdots + c_{ln_i}|v_{n_i}$$

where $x_{1i}, \ldots, x_{k_i i}$ are variables in membrane i, F_{li} is the production function, $1 \leq l \leq n$ is the number of programs in membrane i. $v_1, \ldots, v_{n_i}$ are variables in

membrane i or in the upper immediate or lower immediate membrane of membrane i. $c_{l1}, \ldots, c_{ln_i}$ are the distribution coefficients. The result of production function is distributed as Formula 1.

$$\begin{cases} v_1 = \frac{c_{l1} \times F_{li}(x_{1i}, \ldots, x_{k_i i})}{\sum_{t=1}^{n_i} c_{lt}} \\ \vdots \\ v_{n_i} = \frac{c_{ln_i} \times F_{li}(x_{1i}, \ldots, x_{k_i i})}{\sum_{t=1}^{n_i} c_{lt}} \end{cases} \tag{1}$$

If a variable only appears in a production function, after the execution of current computation step, the former value is consumed and its new value is 0. For variables arising in both sides of the same program, their former values are overwritten by newly distributed values according to repartition protocols. If a variable appears merely in several repartition protocols of different programs, its new value is the sum of all distributed values plus its former value. On condition that variables arise in both sides of several programs, their new value after the computation step is the sum of all distributed values that overwrite their former values. Since every program is applicable and executes in parallel, in case of multiple programs populating in one membrane, one program is selected randomly to execute. This randomness imparts the non-determinism to NPS, bringing on a negative impact on robot motivation control for the control law is deterministic and the results of each computation step of NPS should also be definite. For the sake of avoiding this non-determinism of NPS, only one program is assigned to every membrane to eliminate the random selection process, obtaining a deterministic NPS as a result. The membrane controller in Figure 2 was designed in line with this deterministic requirement, containing one program each membrane.

Although the determinism is guaranteed by the one to one correspondence of membranes and programs, the membrane structure tends to complex and its efficiency declines. This drawback becomes more obvious for modeling sophisticated algorithms which need plenty of membranes to distribute operations. Enzymatic numerical P systems introduced in [16] can contain multiple programs in one membrane, adopting enzyme-like variables to decide the usability of programs. Specifically, for a program with enzyme to catalyze, if the value of a enzyme is greater than the minimum value of all the variables involved in the program, this program is applicable and will execute. The biological base for this criterion is that the concentration of a enzyme should large than that of a reactant. Different with the catalyst used in classical symbol object P systems, enzymes can be consumed or produced, and their amounts matter. For programs without enzymes, these programs are applicable automatically. The definition of ENPS is quoted bellow.

Definition 2. *A enzymatic numerical P system is the construct*

$$\Pi = (m, H, \mu, (Var_1, E_1, Pr_1, Var_1(0)), \ldots, (Var_m, E_m, Pr_m, Var_m(0)))$$

where

1. $m > 0$ *is the number of membranes (the system degree);*
2. H *is the membrane label set storing the labels of each membrane;*
3. μ *is the membrane structure;*
4. Var_i, Pr_i, $Var_i(0)$ *are variables, programs and initial values of variables in membrane* i, $1 \leq i \leq m$.
5. E_i, $1 \leq i \leq m$, *are the enzyme variables in membrane* i;

The form of a program with enzyme in a ENPS is little different from NPS programs as shown bellow,

$$P_{li} : F_{li}(x_{1i}, \ldots, x_{k_i i})(e_{j,i} \rightarrow)c_{l1}|v_1 + \cdots + c_{ln_i}|v_{n_i}$$

where $e_{j,i} \in E_i$ (j is the mark number), is the enzyme catalyzing this program. The computation procedures of ENPS is identical to that of the NPS, except that a set of programs, instead of a single program, are executed in a membrane. The function of enzymes greatly simplifies the membrane structures, making ENPS more practicable to model engineering applications. This assertion will be demonstrated later in this paper.

3 Register transfer level model design of target NPS

The FPGA was invented as a platform for prototype developing of large scale digital circuits [10] for its reconfiguration. The schema that clocks trigger or drive operations imparts the parallel processing ability to FPGA, because as many as operations can be triggered by one clock so executing simultaneously provided that there are enough hardware resources in the FPGA chip. With the digital circuit scale increases rapidly after the invention of transistors in Bell laboratory in 1947, drawing the schematic to design large scale circuits is infeasible. Characterizing the functionality of circuitry with hardware description language then compiling this characterization to schematic is the core task of electronic design automation (EDA). With EDA technology, designing large scale digital circuits in a relatively short period had became reality. After designing, the circuits should be tested to verify achieving required function, performance, resource expenditure, power consumption and other requirements. Then the mass production can be put on the agenda. While for scientific research, the prototype circuits are quite enough to realize novelty.

A increasingly popular but not conspicuous trend is that industrial circles are aware of the great flexibility of FPGA prototype circuits, because for some vital but individual applications, FPGA prototype circuits are the reliable and economical solutions. One essential matter should be clarified here that although circuit schematics can be generated by FPGA developing software, the prototype circuits in FPGA is not the schematics obtained. The look-up tables and flip-flops imitate the generated schematics, so that FPGA takes on the same functionality. For digital circuits, the input value of each basic gate port has two possibilities, 0 or 1. All the possible combinations of outcomes are computed and enumerated in a big table, specific results are looked up in the table according to input

values, then output the searched results. The flip-flop is a memory element, storing values computed in last clock cycle. It is used to construct sequential logic components [2].

The first obstacle avoidance NPS controller was proposed in [4], targeting e-puck robot, which equips 8 infrared sensors around the body. As stated above, the software simulation of NPS by invoking SNUPS engine in e-puck lost its parallelism. Running NPS on parallel hardware circuits is the real implementation of it. The obstacle avoidance control law depicted in Formula 2 and Formula 3 is not such complicated that it is suitable to be a start point for FPGA implementing. The target robot of FPGA implementation research is Pioneer 3 DX with 16 sonar sensors arranged in 2 arrays, whose placements are shown in Figure 1.

The obstacle avoidance NPS to be implemented is adapted on the basis of the first NPS in [4] according to the number of sensors of Pioneer 3 DX robot, since both the infrared sensors in e-puck and sonar sensors in Pioneer 3 DX return the distances between the robot and obstacles. But the property of Pioneer 3 DX sonar sensors make this adaption not so straightforwardly. Particularly, the difference of two distances sampled by these 2 robots lays in that e-puck's infrared sensors return value 0 when there are no obstacles and the reading of sensors increase as the distances decrease, while Pioneer 3 DX's sonar sensors return the distances of the robot to obstacles, the reading of sensors decrease as the distances decrease. The maximum detection range of Pioneer 3 DX is 5000 mm. The control law was designed in line with the e-puck sensors' nature. The transformation given in Formula 4 was done to the readings of Pioneer 3 DX sensors so that the same control law can be adopted. The revised NPS to be implemented is illustrated in Figure 2.

$$lw = CruiseSpeedLeft + \sum_{i=1}^{16} s_i * weithtLeft_i \tag{2}$$

$$rw = CruiseSpeedRight + \sum_{i=1}^{16} s_i * weithtRight_i \tag{3}$$

$$s_i = -x_i + M \tag{4}$$

In Formula 2, *lw* is the speed value accepted by left motor as the required speed to follow. *CruiseSpeedLeft* is the cruise speed, the speed when no obstacles are detected. s_i is the *transformed* sensor reading in line with 4. $weithtLeft_i$ are the weight values of sensors placed in the left-hand-side of robot. Variables in Formula 3 are the counterparts of those in Formula 2. x_i is the original readings of sensors, while M is a constant having value of 1000. i takes values from 0 to 15, corresponding to the numbers of sensors.

Digital systems consist of sequential logic components which contain registers, clocks and their control mechanisms. Flip-flops comprise a register which can perform elementary operations including load, count and shift operations. These operations are executed concurrently in system. *Register transfer operations* refer to operations aiming at data stored in registers. If a digital system

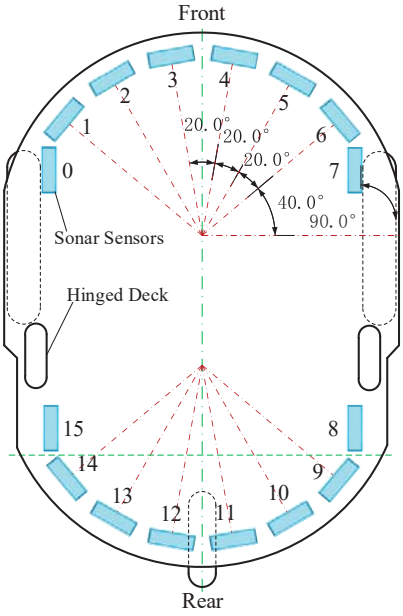


Fig. 1. This is a plan view of Pioneer 3 robot which is covered by a hinged deck on the top. The 16 rectangles in light blue are the sonar sensors surrounding the robot, just beneath the hinged deck. Sensors are arranged in two arrays in the front and in the rear. The layout of sensors in two arrays is identical, as shown explicitly.

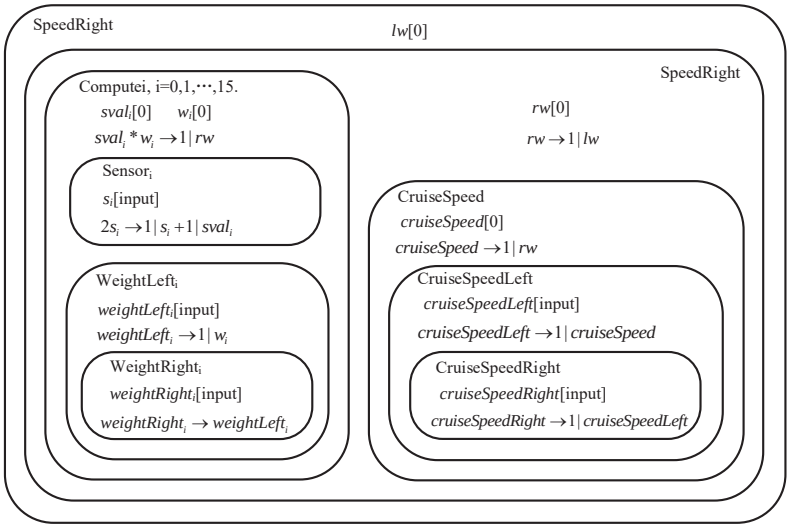


Fig. 2. This numerical P system derives from *Figure 7: Membrane controller for the obstacle avoidance behaviour* in which there are 8 infrared sensors around the target robot in [4]. The adaption here is in a straightforward way by incrementing the number of sensor to 16. Their values should be transformed in line with Formula 4. This numerical P system is called NPS1 below.

is designed by registers, involving register transfer operations and control procedures to these operations, then the digital system is illustrated at *register transfer level* and a register transfer level model is obtained consequently. Register transfer level has a higher abstraction than gate level, which specifies models in the form of schematics. As the scale of digital circuits increase dramatically, obeying the Moore's laws loosely in the past decades, register transfer level abstraction became a potent instrument to cope with super large-scale integration with super complicated functions.

The impractical of designing large scale circuits with schematics does not mean that schematics are not important. If fact, schematics of circuits are the ultimate goal of design specification phase. After the verification of circuits, the schematics will be used to manufacture hardware integrated circuits in silicon wafers. A compilation process from register transfer level to schematic level is responsible for the automatically drawing of schematics. A jargon called *synthesis* is assigned to this compilation process in electronic design automation (EDA) field. RTL models are reliably synthesizable inputs for synthesis tools supplied by a variety of venders. Elaborating the corresponding RTL model of NPS in Figure 2 is the start point and the RTL model will be the objective for downstream operations in FPGA implementation.

RTL models are specified by hardware description language (HDL) such as VHDL and Verilog. This research employs Verilog as the HDL to design the

RTL model. Verilog, initiated by Phil Moorby in 1985, is a IEEE standard HDL since 1995. The basic functional unit in Verilog is *module*. The functionality of a system is distributed among nested modules, for a module can instantiate others modules to incorporate them [3]. A module is a functional block with input and output ports through which it receiving data and sending outcomes to other modules. Intuitively, a membrane can be represented as a module since that a membrane can be regarded as a functional unit in P systems. As a matter of fact, this is the perspective conceived to fabricate the RTL model of target NPS.

The most appealing part of FPGA implementation of P systems is how to deal with membranes, which designate the unique character of membrane computing. Indeed, there are no compartments in FPGA to accommodate variable and programs. To overcome this challenge, we should come back to membrane computing theory that reveal the essence of membranes. As stated in [20], membranes do not have internal structures and substance concentrations. Their shapes and sizes are unimportant. The primary function of membranes is to build distributed space so that rules inside can take place simultaneously. So, if all the rules/programs in the RTL models are manipulated in accordance with the computation steps of P systems, how to treat membranes is not important. In this research, membranes are represented implicitly by synchronizing the execution of programs in different membranes. In fact, different rules/programs will be mapped to different hardware resources in FPGA automatically so that the distributive nature of P systems is achieved without membrane structures.

For the sake of easily distinguishing variables, membranes and modules described bellow, italics denote variables, boldfaced italics indicate membranes and boldface signifies modules. In Figure 2, variables $weightLeft_i$ and $weightRight_i$ $(i = 1, \ldots, 16)$ are sensors' weights whose values reflect the influences of sensors on different positions to the speed of left and right wheel. Supposing a obstacle is detected on the left side, the speed of left wheel should be larger than the speed of right wheel so that robot can turn right to avoid this obstacle. Based on this assumption, taking Formula 3 in account, the weight values of sensor 0, 1, 2, 3, 15 should impose negative effects to right wheel speed in order that right wheel speed is diminished. On the contrary, sensor 0, 1, 2, 3, 15 should impose positive effects to left wheel speed in order to raise its speed. Similarly, sensor 4, 5, 6, 7, 8 impose negative effect to left wheel but positive effect to right wheel. Sensors located in the rear part of the robot are unhelpful for detecting obstacles in front, so their weight values are set to zero.

Consequently, the weight values of 16 sensors, namely, variables $weightLeft_i$ and $weightRight_i$ have inverse values to manifest the positive and negative effects. Pioneer 3 should be calibrated to determine the values of $weightLeft_i$ and $weightRight_i$. The general calibration process can be stated as follows: use control law given in Formula 2 to 4, assigning some initial value to $weightLeft_i$ and $weightRight_i$ then running Pioneer 3. If it bumps into obstacles, altering initial values of these two arrays to some extent in line with the collision severity until it no longer rams any obstacles. The corresponding values of these two variable arrays are shown in Table 1.

Table 1. The calibrated values of $weightLeft_i$ and $weightRight_i$, along with a set of sampled sensors reading data which will be utilized to verify the correctness of RTL model.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$weightLeft_i$	0.1	0.4	0.6	0.8	-0.8	-0.6	-0.4	-0.1	-0.1	0	0	0	0	0	0	0.1
$weightRight_i$	-0.1	-0.4	-0.6	-0.8	0.8	0.6	0.4	0.1	0.1	0	0	0	0	0	0	-0.1
s_i	277	0	0	0	0	0	17	208	190	576	704	745	733	659	451	296

Membranes of NPS1 can be classified into two types according to their functions: delivery membranes and computing membranes. $\boldsymbol{WeightRight_i}$, $\boldsymbol{WeightLeft_i}$, $\boldsymbol{Sensor_i}$ $(i = 1, \ldots, 16)$, ***CruiseSpeedRight***, ***CruiseSpeedLeft*** and ***SpeedRight*** are delivery membranes whose duties are transmitting value of variable to another. For instance, values of variables $weightRight_i$ are sent to variables $weightLeft_i$, which are transferred to variables w_i that used in membrane $\boldsymbol{Compute_i}$ furthermore. Membranes $\boldsymbol{Compute_i}$ $(i = 1, \ldots, 16)$ and ***CruiseSpeed*** are computing membranes to calculate new value of variable rw. Computations performed by computing membranes should be synchronized to reflect the parallelism of NPS. Keep in mind that NPS1 should compute three steps to get results in such a cycle: after the first step finished, $rw = 0$, $lw = 0$; for the second step, rw obtained the left wheel speed value which will be assigned to lw in step 3 and $lw = 0$; in the next step rw acquires the expected right wheel speed value and lw attains the second step value of rw. This process repeats if computing proceeds. The root cause of this cycle stems from the delivery membranes, specifically, the value transfer process.

In digital electronics, a clock is a signal oscillating between high and low electrical level. In low-cost electronic product like an ordinary micro-controller, clock signal, commonly called as clock, is generated by a resistance capacitance oscillator (the well known RC resonator). While for high-end IC products like an FPGA, clock starts off with a quartz crystal resonator, which is a small slice of quartz crystal combined with integral amplifier circuits. The oscillating frequency of a quartz crystal is determined by the shape and size of the crystal slice. The main advantage of this quartz crystal is its tolerability to temperature variation, outputting more stable frequency when temperature highs and lows comparing to RC oscillator.

The importance of clocks is that it is the signal employed as a kind of metronome to trigger so to synchronize operations in circuits. There are two types of trigger modes in digital circuits: edge trigger and level trigger. Edge means the transition from high level to low level and vice versa, corresponding to falling edge and rising edge respectively. Level trigger is more intelligible that operations are executed when the voltage is high level or low level. The time interval between two rising (or falling) edge is called clock cycle, which is a time constant numbered by the previous rising edge's sequence number. Then the clock cycle adjacent before and after nth rising edge are the (n-1)th and nth time cycle. Variables keep their values during a clock cycle. It is emphasized here

that all the operations in RTL model are synchronized (triggered) by the rising edge of clock.

To coordinate the value transfer process in delivery membranes, counters are adopted aiming at this action. Taking the value transfer in $\boldsymbol{WeightRight_i}$ and $\boldsymbol{WeightLeft_i}$ as an example, the initial values of variables $weightRight_i$, $weightLeft_i$ and *counter* are zeros. *counter* counts in a loop from 0 to 2, to correspond computing step 1 to step 3. At the first rising edge of clock, assign the values in Table 1 to $weightRight_i$ and $weightLeft_i$. At the second rising edge of clock, the values of $weightRight_i$ are back to zeros and keep these values until the end of the third clock cycle since their values are consumed by production functions and they do not appear in any repartition protocols. The values of $weightRight_i$ loop in accordance with *counter* value loop. However $weightRight_i$ should transfer their values to $weightLeft_i$ in the second cycle according to programs in membrane $\boldsymbol{WeightLeft_i}$. During a loop of *counter*, $weightRight_i$ equal zero from the second cycle, so $weightLeft_i$ also have values zeros in the third cycle. Whether this arrangement is correct or not can be deduced from timing diagram in the form of waveform of variables, which will be detailed as follows.

Membranes $\boldsymbol{Sensor_i}$ are omitted in RTL model by substituting programs $sval_i * w_i \to 1|rw$ in membrane $\boldsymbol{Compute_i}$ with $s_i * weightLeft_i \to 1|rw$. Because the effect of program $2s_i \to 1|s_i + sval_i$ in $\boldsymbol{Sensor_i}$ is to assign sensors' readings to $sval_i$ and program $weightLeft_i \to 1|w_i$ in $\boldsymbol{WeightLeft_i}$ is to transfer the valus of $weightLeft_i$ to w_i, executing programs in $\boldsymbol{Sensor_i}$ will cost 1 more clock cycle which can be reduced by performing $s_i * weightLeft_i \to 1|rw$. Assuming that variables s_i have the sonar sensors' readings as initial values, computing membranes ($\boldsymbol{Compute_i}$ and $\boldsymbol{CruiseSpeed}$) are triggered to compute rw at rising edge of clock. In the first clock cycle, $rw = 0$ for the initial values of $weightLeft_i$ are zeros. In the second cycle, rw obtains the speed of left wheel because $weightLeft_i$ got their exact values during second cycle. In the third cycle, rw acquires the speed of right wheel because $weightLeft_i$ got the valus of $weightLeft_i$ in the first cycle, which are transferred during the second cycle. Hence the arrangement of transferring and computing can achieve the computation procedures of NPS1 accurately which computes 3 steps to get results. The whole processes of transfer and computing should perform as the timing diagram depicted in Figure 3.

Two types of Verilog HDL modules are designed to carry out value transfer and computing operations described above. **WeightRight** assigns right weight values (in Table 1) of sensors to variables $weightRight_i$, while **WeightLeft** assigns left weight values of sensors and transfers the values of $weightRight_i$ to $weightLeft_i$. Analogously, module **CruiseSpeedRight** and **CruiseSpeedRight** do the same thing to variables *cruiseSpeedRight* and *cruiseSpeedLeft*. **CruiseSpeed** transfers the value of *cruiseSpeedLeft* to *cruiseSpeed*. Module **Compute** is designed to conduct parallel computations originated from $\boldsymbol{Compute_i}$. **SpeedLeft** passes the value of rw to lw. No module corresponds to membrane ***SpeedRight*** for there are no programs whithin it. The value loop of *counter* to-

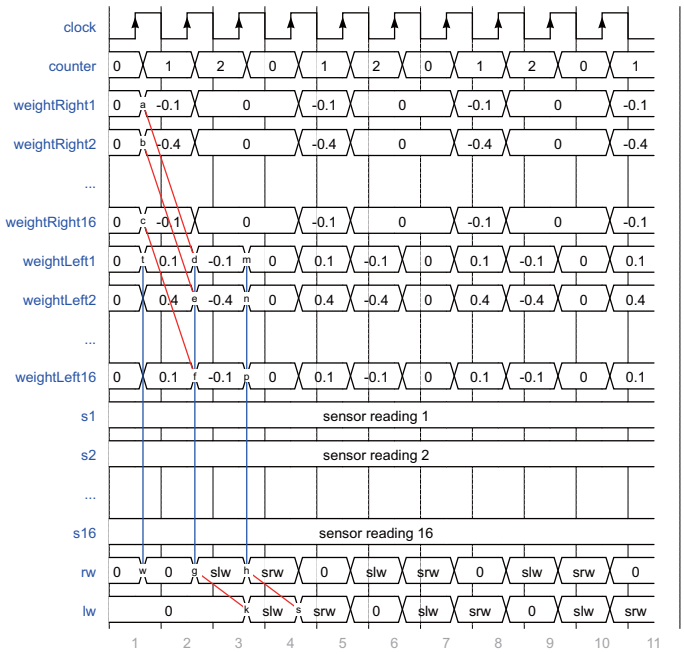


Fig. 3. This is the desired timing diagram of RTL model of NPS1. At the first rising edge of clock, transfer and computation operations are triggered simultaneously. After the rising edge (and during the first clock cycle), $weightRight_i$ and $weightLeft_i$ are delivered values given in Table 1. The outcomes obtained are 0s for the initial values of $weightLeft_i$ are 0s, although sensor variables have readings. After the second rising edge, rw gets the speed of left wheel (denoted by "slw") for $weightLeft_i$ is delivered left wheel weight values. The same reason why rw attains the speed of right wheel (denoted by "srw") after the third rising edge. Red lines indicate value transfers from $weightRight_i$ to $weightLeft_i$ and from rw to lw. Blue lines signifies parallel computing of programs in associated membranes.

gether with rising edge guaranteed the whole processes are controlled accurately. Modules are connected to be an entirety according to signals' input-output relationships, for example, the output of **WeightRight** is the input of **WeightLeft**.

From Figure 2, it is obvious that membranes are organized in nested structure. Nevertheless, modules in RTL model are not nested but are independent with each other. Another notable feature of the RTL model is that the function of modules does not conform to the function of membranes. There are no one to one correspondence between membranes and modules. In spite of these differences, the behaviors of RTL model and NPS1 are identical: at each computation step, the value of each variable and computing outcome are the same. The identity of computation steps between NPS and its RTL model reflects the validity and rationality of the RLT model. The RLT model composed of models is shown in Figure 4 in which the input-out ports and their connection relationships of modules are clarified as well.

RTL model of NPS1 should be verified after its design. In Verilog, a special module named *testbench* is designed to validate RTL models by instantiating RTL models and imposing specific input signals and then analyzing outcomes to determine the accuracy of designed RTL models. The instantiation of a RTL model is accomplished by declaring the name of RTL model and connecting ports of RTL model to some variables (signals). Then assign initial values to input variables of RTL model and define the clock cycle. The number of running clock cycle should also be declared. It is remarked that testbench just perform software simulation of RTL models, which means the execution of RTL model is conducted by CPU of host computer rather than the adopted FPGA. Testbench simulations are also called behavioral simulations. This simulation neglects the latency when signals pass through logic gates so that results are obtained at the trigger time, i.e., at the rising edge of clock. It is not the case when variables are processed by real digital circuits in FPGA where operations are triggered by rising (or falling) edges and complete after some time intervals.

As can be seen in Table 1, variables are real numbers. Unfortunately, real numbers cannot be represented in digital circuits directly. In fact, real numbers are represented in some forms of integers–fixed point number or float point number. In this research, real numbers are transformed to fixed point numbers which are easy to deal with. To be specific, each variable is assigned a 24-bit register. Allocate the first 11 bits to integer part and the following 13 bits to fraction part of a variable value. This bits' manipulation creates a range of $[-(2^{11}-1),(2^{10}-1)]$ ($[-2047,1023]$ in decimal) which includes the value range of sensors' reading [0,1000]. Each real number should be transformed to fixed point number before running the RTL model so that results obtained are also in fixed point representation. Consequently, an inverse transformation is necessary to get decimal results.

NPS and ENPS can be simulated by a software named *PeP* which is invented by Florea and Buiu. Software simulation results of NPS1 are the benchmarks of its hardware implementation which provide fair reference results. *PeP* can also offer elapsed time (in seconds) used to compute some predefined steps. From a

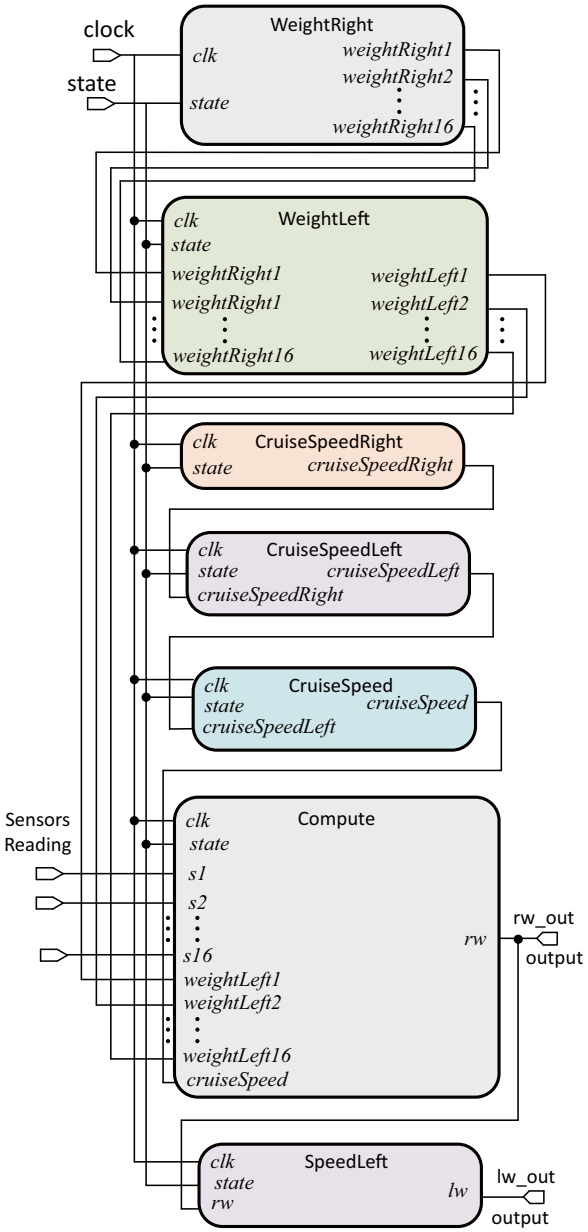


Fig. 4. RTL model of NPS1 consists of 7 modules, although NPS1 has 69 membranes. There is a one to one one-to-one correspondence between membranes and programs. This correspondence transforms the implementing of a membrane in a NPS to implementing a program inside. Programs can be synchronized in one module with parallel constructs of Verilog. This is the reason why the number of modules can be reduced substantially. Add a *state* port to NPS1 so that it possesses idle and busy state.

hardware point of view, the software simulation is a CPU implementation of a algorithm. As a result, this returned time reflects the performance of the CPU in host computer. Further more, this CPU implementation time is indispensable to compute the speedup of FPGA implementation of NPS.

The host computer is a Dell *Latitude* equipped with a Intel Core i7-7820HQ and 16 GB RAM. Target FPGA of this research is Xilinx Artix-7 xc7a35t-1cpg236c which is the core part of BASYS 3 FPGA developing board, a product of Digilent company. FPGA developing software employed is Xilinx Vivado 2018.2 which is a new generation software dedicated to develop 7 series and UltraScale FPGAs. The testbench of NPS1 RTL model is designed to verify whether it performs as what NPS1 should do. As can be seen in Figure 5 which shows the behavioral simulation waveform obtained in Vivado 2018.2, in the first three cycles, *rw-out* which corresponds *rw* holds value 0, 310.6953125, 289.3046875, behaving exactly as the computing process of NPS1 that computes three steps to get results.

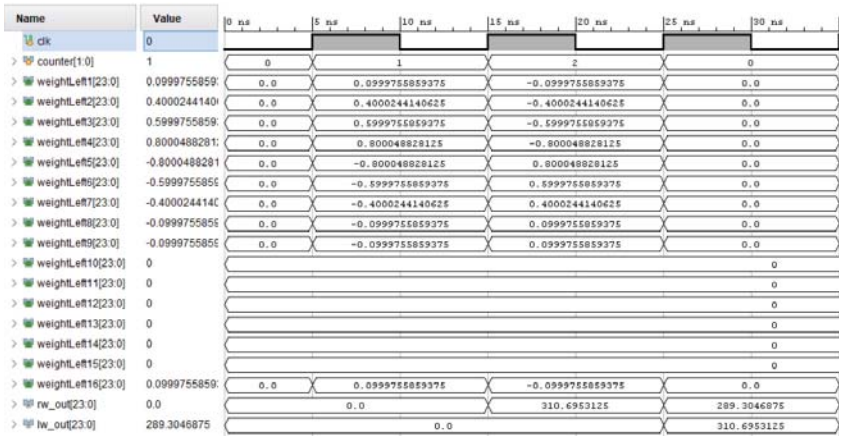


Fig. 5. Waveform of behavioral simulation of NPS1. The values of $weightLeft_i$ alternate as expected when *counter* loops its value. Sensors' readings s_i take the value in the last row of Table 1 which are abridged from waveform for the sake of taking a screenshot including computing results *rw-out* and *lw-out*.

Simulation results of RTL model are real numbers with long fractional tails. This appearance of results stems from the treatment of real number representation in FPGA, the fixed point representation. There are deviations when some constant bits are assigned to fractions. This inaccuracy can be acceptable if deviations are small enough to meet some error criterion. To validate the rightness of RTL model, input NPS1 to *PeP* and compute three steps, Figure 6 shows the outcomes. It costs 0.011703 seconds to compute 3 steps of NPS1, outputing $rw = 289.3$ and $lw = 310.7$. The errors between software simulation and

FPGA implementation is given in 5. Errors' order of magnitude are 10^{-5}, which is miniature for engineering applications like robot control. Data accuracy can be improved by assigning more bits to fractional part of a real number variable.

$$\begin{cases} e_{rw} = |\frac{289.3-289.3046875}{289.3}| = 1.6203 \times 10^{-5} \\ e_{lw} = \frac{310.7-310.6953125}{310.7} = 1.5087 \times 10^{-5} \end{cases} \tag{5}$$

```
WARNING:Maximum number of simulation steps exceeded; Simulation stopped
INFO:Simulation finished succesfully after 3 steps and 0.011703 seconds; End state below:
num_ps = {
  SpeedLeft:
    var = { lw: 310.70, }
    E = {}
  SpeedRight:
    var = { rw: 289.30, }
    E = {}
```

Fig. 6. Results of *PeP* simulation of NPS1. *PeP* runs in Windows *Command Prompt* for it does not have a GUI. Computation results together with computing steps and time costs are printed on screen to show users. (E)NPS should be described in a particular format that meets the requirements of *PeP* before running simulation.

In the testbench of NPS1 RTL model, clock cycle is set to 10 nanoseconds, but it is too early to assert that harden NPS1 costs 30 nanoseconds to complete 3 steps of computation. Because the harden NPS1 is a digital circuit which takes a period of time to calculate results after one feed of 16 sensors' reading. This period of time should be smaller than the cycle of clock so that one calculation finishes before the beginning of next clock cycle, otherwise the operations of harden NPS1 is disordered and it is impossible to get right results. This period of time can be measured inaccurately during post implementation timing simulation which will be expounded in section 5.

4 Register transfer level model design of target ENPS

ENPS allows multiple programs which will be executed concurrently contained in one membrane, imparting a feature that can simplify membrane structures greatly. ENPS1 illustrated in Figure 7 has the same function as NPS1 illustrated in 2, but composed of much less membranes. 17 membranes (69 membrans in NPS1). More importantly, ENPS1 computes only one step to get result, improving performance by three times comparing to NPS1 which calculates three steps. Performance improvement is achieved by getting rid of delivery membranes and the speed of left wheel and right wheel are calculated at the same time, not in sequential as what NPS1 does.

Each programs involves a conditional statement and its consequential judgment determines the enforceability of every program in ENPS1. Whereas, the

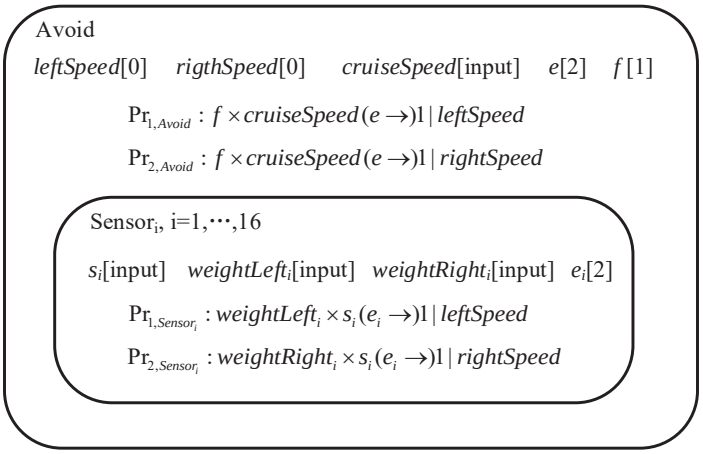


Fig. 7. Target enzymatic numerical P system ENPS1. The value of enzyme e is larger than that of f in membrane ***Avoid***, so these two programs can take place. Enzymes e_i have greater values than s_i so the 16 programs in membrane $\boldsymbol{Sensor_i}$ can execute in parallel.

values of associated enzymes are tuned so that all the programs can carry out simultaneously. Accordingly, conditional statements are ignored in the RTL model of ENPS1 to save hardware resources such as registers and logic gates. ENPS1 RTL model contains one module named **Enps** to perform the behavior of ENPS1. The behavioral simulation waveform of **Enps** is depicted in 8. *PeP* simulation results of ENPS1 is given in Figure 9.

5 FPGA implementation flow

If the outcomes are satisfactory after the behavioral simulation of RTL models, there are a serial of procedures ahead to proceed to accomplish FPGA implementation of (E)NPSs. FPGA implementation flow of Vivado 2018.2 is sketched in this section to present a whole implementation process.

Synthesis. RTL models are portrayed in HDL while the corresponding digital circuits are required for *Place* & *Route* in later procedures, so a compilation process is need to transform RTL models to digital circuits. The terminology *synthesis* is referred to this compilation. After Synthesis, a circuit possessing the same functionality of RTL model is generated. Be noted that this circuit is not the one to be placed and routed on FPGA during this step, for it is built from theoretical perspective without consideration of hardware resources of target FPGA.

Setting up physical constraints and timing constraints are the two subsequent procedures after synthesizing RTL models. Physical constraints assign ports in

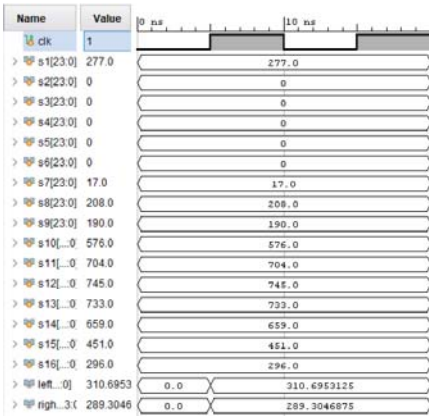


Fig. 8. Waveform of behavioral simulation of ENPS1. Left and right wheel speed variables gain their expected values after the first rising edge.

```
WARNING:Maximum number of simulation steps exceeded; Simulation stopped
INFO:Simulation finished succesfully after 1 steps and 0.002993 seconds; End state below:
num_ps = {
  Avoid:
    var = { Spl: 310.70,  Spr: 289.30,  cruiseSpeedLeft: 0.00,  cruiseSpeedRight: 0.00,  f: 0.00, }
    E = { e: 2.00, }
```

Fig. 9. *PeP* simulation results of ENPS1. Performance is improved for the reduction of computing steps by assigning sensors' weight values and readings to corresponding variables directly and calculating concurrently.

RTL model to FPGA pins so that signal input & output course can carry out on real circuits. Timing constraints set the period of clock source (a RC oscillator or a quartz crystal resonator), and signal input & output delay so that a real clock with specified cycle will be produced. The package view of target FPGA which list all the available pins can be opened under *I/O Planning* view of Vivado. The assignment of pins is illustrated in Figure 10 in which gray circles (normal input-output pins) and light blue hexagons (clock capable pins) with orange bars inside are pins mapped to RTL model ports. For vector ports (more than one bit), each bit should be assigned a pin. For instance, the output of NPS1 RTL model, *rw*, is a 24-bit variable used as one output port, so 24 pins are allocated to it. Clock signal is a 1-bit scalar which demands one pin.

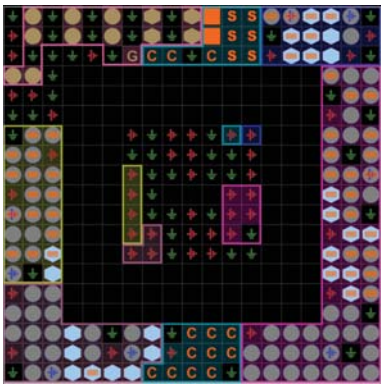


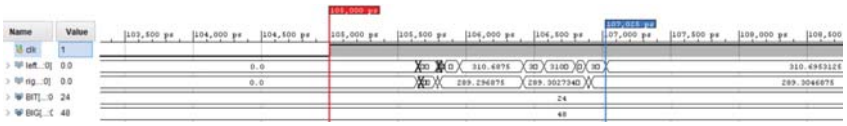
Fig. 10. The assignment of package pins of NPS1. RTL model of NPS1 has one 1-bit input port *clock* and two 24-bit output ports *rw* and *rw*, so totally 49 pins are distributed to these three ports.

Implementation. Theoretical schematic compiled in *synthesis* step is fabricated in *Implementation* step according to hardware resources in FPGA to generate a more fine-grained schematic. A important simulation after implementing is post implementation timing simulation which reflects the real elapsed time of physical circuit running since the time delay of logic elements and data paths are taken into account to build the fine-grained schematic. The underlying cause is that clock cycle is not the running time of circuits but a special time container with trigger pip to accommodate running time so that the running order is not chaotic. Static timing analysis finding the critical path would be impossible for large scale circuits, then post implementation timing simulation is a reasonable and convenient method to estimate circuit running time. The margin between clock cycle and running time is a significant parameter related to performance and should be evaluated prudently. It might also be noted that only Verilog HDL is supported for post implementation timing simulation.

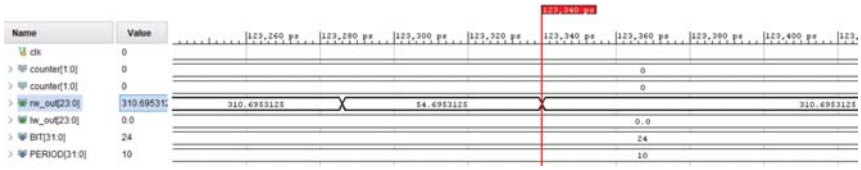
Post implementation timing simulation waveform of NPS1 and ENPS1 are shown in Figure 11. In Figure 11(a), a rising edge appears at 105 ns and the steady value of left wheel speed advents at 107.025 ns, while right wheel steady speed value arises even earlier. This implies that the computing results can be obtained within 5 ns, so the clock period of ENPS1 RTL model can be reduced to 5 ns to hold the running time of harden ENPS1. In Figure 11(b), for a rising edge at 115 ns, steady value of right wheel speed arises at 123.34 ns. The period time of clock for NPS1 can not be reduced by half for the running time of hardened NPS1 is larger than half clock cycle, but it is enough to hold the running time. Clock period instead of running time is used to calculate speedup because circuits hold after running time to wait for another clock trigger edge. Software simulation time for NPS1 and ENPS1 are 0.011703 s and 0.00293 s, which is 1.1703×10^7 ns and 2.993×10^6 respectively. The speedup of FPGA implementation of NPS1 and ENPS1 can be computed intuitively as given in Fomular 6 and 7, in which exhibits a speedup as high as a remarkable order of magnitude of 10^6. It's pointed out that the computing speed of FPGA hardened NPS1 and ENPS1 is 10^8 step per second because clock period is 10 ns, or its frequency is 100 MHz.

$$\frac{1.1703 \times 10^7}{30} = 3.901 \times 10^5 \tag{6}$$

$$\frac{2.993 \times 10^6}{5} = 5.986 \times 10^5 \tag{7}$$



(a) Post implementation timing simulation waveform of ENPS1.



(b) Post implementation timing simulation waveform of NPS1.

Fig. 11. The outcomes settle out gradually. Running time of ENPS1 is shorter than that of NPS1, indicating that the performance of harden ENPS1 is better than its counterpart of NPS1.

Place & Route planning is carried through in this step as well. In Vivado *Device* view, structures and interconnections of hardened RTL model can be

observed, as shown in Figure 12. This planning will be written into bitstream file which is produced in next step.

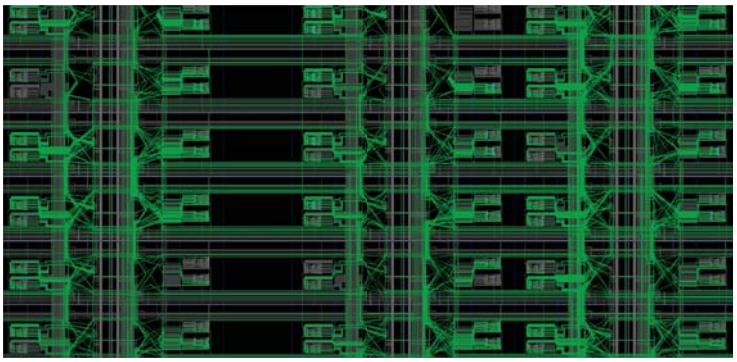


Fig. 12. Part of *Place & Route* planning of NPS1 RTL model, in which green lines connect luminous gates to construct interconnections which will be established in FPGA.

Program and debug. If post implementation timing simulation meets design requirements, FPGA configuration file, bitstream file, can be generated in *Program and debug* procedure of Vivado. Bitstream file contains all the design contents and will be downloaded to FPGA to *program* it, i.e., performing physical *Place & Route* operation and setting up constraints. After programing of target FPGA, FPGA implementation flow terminates and yields a real hardened NPS.

To examine the results of FPGA is not straightforward but involving a technical procedure, hardware debug. Indeed, not matter how precise results post implementation simulation can provide, it is just a simulation. On the contrary, hardware debug procedure can penetrate the results stored in registers of FPGA. If a signal needs to be debugged, it should be marked after synthesis. Vivado will probe these marked signals and show their values via integrated logical analyzer, a virtual software logical analyzer. By performing hardware debug, the real outcomes of design can be verified. Results of FPGA hardened ENPS1 is shown in the integrated logical analyzer window in Figure 13 in which values are in hexadecimal. Hexadecimal 26d640 and 2429c0 are decimal 310.6953125 and 289.3046875 which are expected values of left wheel and right wheel speed.

6 Comparisons of FPGA hardened NPS and ENPS

There are three quality attributes related to FPGA implementation of NPS: performance, hardware resources utilization and power consumption. The performance of FPGA ENPS1 is 6 times faster than that of FPGA NPS1 since the clock period of ENPS1 is 5 ns and it costs one cycle to compute results while

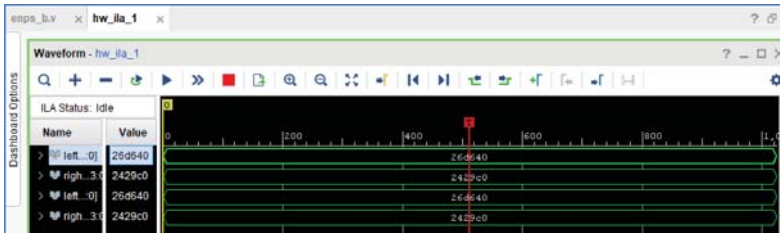
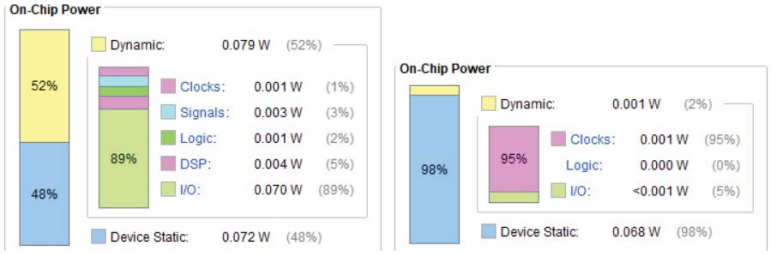


Fig. 13. Hardware debug results of ENPS1 shown in integrated logical analyzer. Values of variables are in hexadecimal instead of decimal. The reason why using hexadecimal will be revealed in Section 7.

NPS1 clock period is 10 ns and takes three cycles (30 ns) to get results. Vivado utilization report provides hardware resources costs of FPGA NPS1 and ENPS1 which are listed in Table 2 and 3. Contrasting these two tables, ENPS1 dose not use any *LUT as Logic* and *LUT Flip Flop Pairs*, and the cost of *Slice* only takes up less than one quarter than that of NPS1.

Power consumption is an increasingly important attribute with the popularity of mobile devices. Vivado evaluate power consumption of FPGA NPS1 and ENPS1, shown in Figure 14. As can be seen, power consumption of FPGA NPS1 is twice as larger as that of the FPGA ENPS1.



(a) Power consumption of FPGA NPS1 is 0.151 w.

(b) Power consumption of FPGA ENPS1 is 0.069 w.

Fig. 14. Total power consumption is the sum of device static power and dynamic power.

7 UART communication of NPS

Substituting the on-board computer of Pioneer 3 DX with FPGA NPS to control this robot is the ultimate objective, but before achieving this objective, how to replace should be considered. Pioneer 3 is controlled directly by dedicated robot motion controller which is a microcontroller [1] while computing operations are

Table 2. Hardware resources utilization of NPS1

Site Type	Used	Available	Utilized %
Slice	67	8150	0.82
SLICEL	27		
SLICEM	40		
LUT as Logic	179	20800	0.86
using O5 output only	0		
using O6 output only	107		
using O5 and O6	72		
LUT Flip Flop Pairs	32	20800	0.15
fully used LUT-FF pairs	2		
LUT-FF pairs with one unused LUT output	27		
LUT-FF pairs with one unused Flip Flop	28		
DSPs	16	90	17.78
DSP48E1 only	16		
Bonded IOB	49	106	46.23
IOB Master Pads	24		
IOB Slave Pads	24		
BUFGCTRL	1	32	3.13

Table 3. Hardware resources utilization of ENPS1

Site Type	Used	Available	Utilized %
Slice	15	8150	0.82
SLICEL	15		
SLICEM	0		
LUT as Logic	0	20800	0
LUT Flip Flop Pairs	0	20800	0
DSPs	0	90	0
Bonded IOB	49	106	46.23
IOB Master Pads	25		
IOB Slave Pads	23		
BUFGCTRL	1	32	3.13

executed by on-board computer. The microcontroller samples reading of sensors and sends them to on-board computer in line with RS232 communication electrical standard protocol. After computation finished, results are transmitted back to microcontroller to control wheel motors. Thereupon, FPGA NPS should have receiver and transmitter device to receive sensors reading and to send computation results. Universal asynchronous receiver/transmitter (UART) is such a device meeting RS232 protocol which is employed to design communication devices of FPGA NPS.

The data transferring speed of Pioneer 3 is 115200 baud (nearly 115200 Hz), which should be the working rate of UART receiver and transmitter. While the clock frequency of BASYS 3 FPGA developing board is 100 MHz, a module named **frequency splitting** is designed to generate required clock frequency according to Formula 8, where f_o is desired frequency, f_r is source clock frequency, K is frequency control word, N is the number of bits assigned to a counter inside this module. To ensure transmission accuracy, it is better to sample data at the midst of bit width because that it is in the most steady state. For this purpose, generate another clock frequency which is 16 times (115200×16 Hz) faster than UART frequency. A sampling counter counts the cycle number of the 16 times frequency. When sampling counter increments to 7 or 8, module **receiver** sample the sensors reading from microcontroller, at the frequency of 115200 Hz for it just sample at the same value of sampling counter.

$$K = \frac{f_o \times 2^N}{f_r} \tag{8}$$

UART receiver can only receive 8-bit data while a sensor reading is a 24-bit data. Therefore it is a triple receiving operation to receive a sensor reading. A register is assigned to store received 8-bit data so that 48 registers are needed to store sixteen 24-bit data. Values in three adjacent register are concatenated to form a 24-bit binary value which is a input data to NPS1. NPS1 has two operating state: idle and busy state. Before NPS1 receiving all the sensor reading, it is in idle otherwise it is in busy after fed to full. The storing rate is $\frac{1}{10}$ of receiving frequency for 10 bits are sent sequentially while computing frequency should be $\frac{1}{30}$ of receiving frequency since it performs 3 times of receiving operation to get a entire sensor reading. The working frequency of UART transmitter equals to the storing rate for it sends 10 serial bits to microcontroller which works in 115200 Hz. The RTL model of UART NPS1 is illustrated in Figure 15.

According to the computational process of NPS1, the first output of *rw* is zero and the latter two outputs are expected values. UART transmitter should skip zero and begin to transmit when the first valid value arise. So transmitter stays in idle before *rw* is nonzero. Wheel speed values to be transmitted are 24-bit data, but each time the transmitter can send only 8 bits. Hence the transmitter sends six times successively to transmit 48 bits in total. UART communication experiment between host computer and BASYS 3 is conducted to verify proposed design method. In this experiment, host computer sends sixteen 24-bit sensor readings to BASYS 3 via integrated UART transmitting port within it and

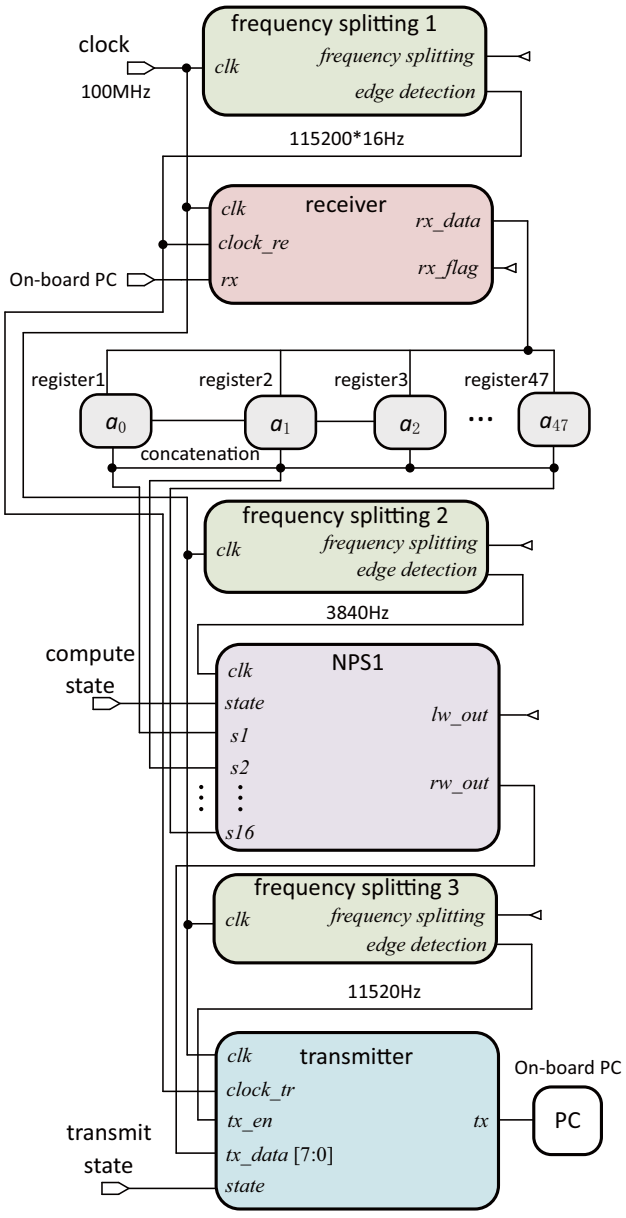


Fig. 15. RTL model of UART NPS1. Three **frequency splitting** modules are utilized to generate three different clock frequencies. The working frequencies of UART receiver and transmitter are unequal due to the opposite data flow.

BASYS 3 receives these data into forty eight 8-bit registers. Every three 8-bit registers are concatenated to construct a 24-bit register so that data inside can be 24-bit. After computation of FPGA NPS1, BASYS 3 transmits results to host computer and these values can be observed on the screen. Realterm is the serial debugging software adopted to configure ports setting and display data from BASYS 3. It is obvious that FPGA NPS1 computes and transmits correctly according to Figure 16 which shows the experiment result.

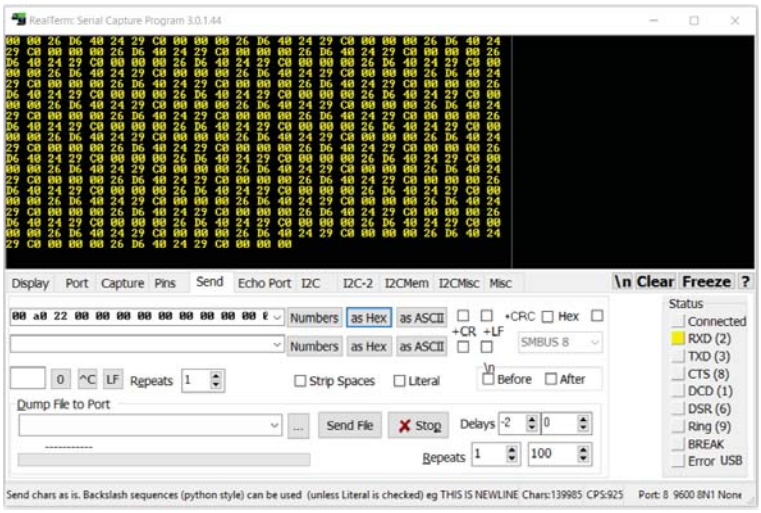


Fig. 16. Numbers are in hexadecimal for real numbers are represented in fixed point binary numbers. These data repeats 00 00 00, 26 D6 40 and 24 29 C0. Two digits of hexadecimal correspond to 8 bits in binary so three dual hexadecimal pairs comprise a 24-bit binary number. These three repeated data strings are 0, 310.6953125 and 289.3046875 in decimal, which are the desired three outcomes of *rw* of NPS1.

A notable phenomenon can be found in Figure 16 is that UART transmitter does not stop transmitting so data strings repeat. The transmitting counter will enter infinite loop if clock does not stop. Indeed, the clock never stop once beginning to work. In effect, any counter will loop forever unless it is related to some limited signals. Input signals are limited because they will stop at a time. Transmitting counter should come into play to count from 0 only after all sensor readings are received and stop counting when it counts to 5. By this way, transmitting counter will not loop and transmitter sends six 8-bit binary data. The effectiveness of this strategy is demonstrated by the simulation waveform of UART-NPS1 RTL model in Figure 17.

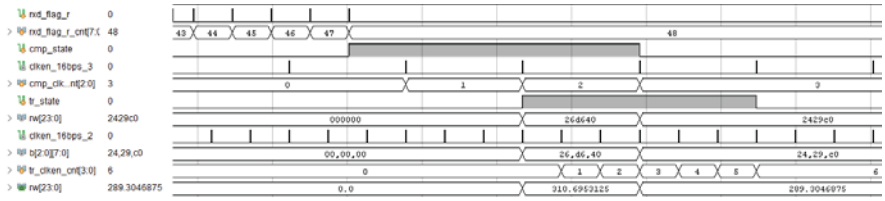


Fig. 17. When input counter counts from 0 to 47, sensor readings are received into NPS1. Computing signal *cmp-state* converts from 0 to 1 to trigger computing. When left wheel speed are outputted, transmitting signal *tr-state* shifts from 0 to 1 to trigger transmitting. At the time when transmitting counter signal *tr-clken-cnt* has value of 5, *tr-state* switches its value to terminate transmitting.

8 Conclusions

Based on the research results of this article, comparing to software simulation of P systems, FPGA implementation of P systems can achieve a remarkable speedup which can be as high as 10^6 in the case of NPS and ENPS in which non-determinism does not exist. A NPS and an ENPS which are used as robot controllers are implemented in FPGA. NPS with UART communication ability which has the potential to substitute on-board computer of Pioneer 3 robot is designed and implemented as well. To the best of authors' knowledge, it is the first time that (E)NPSs are hardened in FPGA. The large scale parallelism of (E)NPS is gained and has the potential to be exploited to control autonomous robots.

Differentiating from software simulation, FPGA implementation of (E)NPS fabricates hardware circuits inside which work as a parallel architecture to execute associated processes. The development process of FPGA implementation is much more sophisticated than software simulation since behavioral simulations of RTL model never guarantee the correctness of real circuits. It costs a lot of labor to do post implementation simulation and hardware debug to verify RTL models. FPGA ENPS1 is in the lead comparing FPGA NPS1 for its better performance, lower hardware resources and power consumption.

An FPGA hardened (E)NPS work as a controller/processor to control robots or other engineering objects, for instance a drone or a tool machine. In the perspective of processor, FPGA (E)NPS is a multicore processor in which a set of programs in a membrane play the role of a core. FPGA NPS1 implemented in this research is a heterogeneous multicore processor for cores have two types of function: transferring and computing. This research is the first step to manufacture a high performance (E)NPS chip because its prototype circuits are fabricated in FPGA.

There are two important questions should be figure out before replacing on-board computer of Pioneer 3. Firstly, how to extract sensors' reading from data package sampled by microcontroller. Secondly, the data format of sensors' reading should be ascertained. This will cause the serious modification of RTL

models. These information is not mentioned in operation manual so only experiments can help. The spectacular performance of FPGA (E)NPS is ideal for real-time processing like robot path planning, image and video processing, which will be the future research directions.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61972324, 61672437, 61702428), the New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0044), the Sichuan Science and Technology Program (2018GZ0185, 2018GZ0086)and Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06).

References

1. O. Adept. *Pioneer 3 Operations Manual*. Omron Adept MobileRobots LLC., 10 Columbia Dr. Amherst, NH, 03031 USA, revision 6.5 edition, Apr. 2017.
2. P. M. Aiken Pang. *Beginning FPGA: Programming Metal*. Apress, 2017.
3. J. Bhasker. *A Verilog HDL Primer*. Star Galaxy Publishing, 1058 Treeline Drive, Allentawn, PA 18103, second edition, 1999.
4. C. Buiu, C. I. Vasile, and O. Arsene. Development of membrane controllers for mobile robots. *Inf. Sci.*, 187:33–51, 2012.
5. K. Huang, G. Zhang, X. Wei, H. Rong, Y. He, and T. Wang. Fault classification of power transmission lines using fuzzy reasoning spiking neural p systems. *Bio-inspired Computing Theories and Applications*, Jan. 2016.
6. A. Leporati, G. Mauri, A. E. Porreca, and C. Zandron. Enzymatic numerical P systems using elementary arithmetic operations. In A. Alhazov, S. Cojocaru, M. Gheorghe, Y. Rogozhin, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing - 14th International Conference, CMC 2013, Chişinău, Republic of Moldova, August 20-23, 2013, Revised Selected Papers*, volume 8340 of *Lecture Notes in Computer Science*, pages 249–264. Springer, 2013.
7. D. Llorente-Rivera and M. A. Gutiérrez-Naranjo. The pole balancing problem with enzymatic numerical p systems. In *Proceedings of the Thirteenth Brainstorming Week on Membrane Computing*, pages 195–206, Feb. 2015.
8. S. Maeda and A. Fujiwara. Enzymatic numerical P systems for basic operations and sorting. In *2014 Joint 7th International Conference on Soft Computing and Intelligent Systems (SCIS) and 15th International Symposium on Advanced Intelligent Systems (ISIS), Kita-Kyushu, Japan, December 3-6, 2014*, pages 1333–1338. IEEE, 2014.
9. M. A. Martínez-del-Amor, L. F. Macías-Ramos, L. Valencia-Cabrera, and M. J. Pérez-Jiménez. Parallel simulation of population dynamics P systems: updates and roadmap. *Natural Computing*, 15(4):565–573, 2016.
10. C. Maxfield, editor. *FPGAs World Class Designs*. Elsevier, 2009.
11. T. Y. Nishida. A membrane computing model of photosynthesis. In G. Ciobanu, M. J. Pérez-Jiménez, and G. Paun, editors, *Applications of Membrane Computing*, Natural Computing Series, pages 181–202. Springer, 2006.

12. L. Pan, G. Păun, T. Wu, and Z. Zhang. Four recent research topics on numerical and spiking neural p systems. *Romanian Journal of Information Science and Technology*, 19(1-2):5–16, 2016.
13. L. Pan, Z. Zhang, T. Wu, and J. Xu. Numerical P systems with production thresholds. *Theor. Comput. Sci.*, 673:30–41, 2017.
14. L. Pan, Z. Zhang, T. Wu, and J. Xu. Numerical P systems with production thresholds. *Theor. Comput. Sci.*, 673:30–41, 2017.
15. S. Pang, T. Ding, A. Rodríguez-Patón, T. Song, and Z. Phen. A parallel bioinspired framework for numerical calculations using enzymatic P system with an enzymatic environment. *IEEE Access*, 6:65548–65556, 2018.
16. A. Pavel, O. Arsene, and C. Buiu. Enzymatic numerical P systems - a new class of membrane computing systems. In *Fifth International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010, University of Hunan, Liverpool Hope University, Liverpool, United Kingdom / Changsha, China, September 8-10 and September 23-26, 2010*, pages 1331–1336. IEEE, 2010.
17. A. B. Pavel and C. Buiu. Using enzymatic numerical P systems for modeling mobile robot controllers. *Natural Computing*, 11(3):387–393, 2012.
18. A. B. Pavel, C. I. Vasile, and I. Dumitrache. Robot localization implemented with enzymatic numerical P systems. In T. J. Prescott, N. F. Lepora, A. Mura, and P. F. M. J. Verschure, editors, *Biomimetic and Biohybrid Systems - First International Conference, Living Machines 2012, Barcelona, Spain, July 9-12, 2012. Proceedings*, volume 7375 of *Lecture Notes in Computer Science*, pages 204–215. Springer, 2012.
19. H. Peng, J. Wang, J. Ming, P. Shi, M. J. Prez-Jimnez, W. Yu, and C. Tao. Fault diagnosis of power systems using intuitionistic fuzzy spiking neural p systems. *IEEE Transactions on Smart Grid*, 9(5):4777–4784, Sept. 2018.
20. G. Păun. *Membrane Computing: An Introduction*. Springer, 2002.
21. G. Păun and R. A. Păun. Membrane computing and economics: Numerical P systems. *Fundam. Inform.*, 73(1-2):213–227, 2006.
22. Y. Suzuki, Y. Fujiwara, J. Takabayashi, and H. Tanaka. Artificial life applications of a class of P systems: Abstract rewriting systems on multisets. In C. Calude, G. Paun, G. Rozenberg, and A. Salomaa, editors, *Multiset Processing, Mathematical, Computer Science, and Molecular Computing Points of View [Workshop on Multiset Processing, WMP 2000, Curtea de Arges, Romania, August 21-25, 2000]*, volume 2235 of *Lecture Notes in Computer Science*, pages 299–346. Springer, 2000.
23. Y. Suzuki and H. Tanaka. Modeling p53 signaling pathways by using multiset processing. In G. Ciobanu, M. J. Pérez-Jiménez, and G. Paun, editors, *Applications of Membrane Computing*, Natural Computing Series, pages 203–214. Springer, 2006.
24. C. Vasile, A. Pavel, I. Dumitrache, and et al. Implementing obstacle avoidance and follower behaviors on koala robots using numerical p systems, 2012.
25. C. I. Vasile, A. B. Pavel, and I. Dumitrache. Improving the universality results of enzymatic numerical p systems. In *Proceedings of the Tenth Brainstorming Week on Membrane Computing*, volume 2, pages 207–214, Feb. 2012.
26. C. I. Vasile, A. B. Pavel, and I. Dumitrache. Universality of enzymatic numerical P systems. *Int. J. Comput. Math.*, 90(4):869–879, 2013.
27. T. Wang, G. Zhang, J. Zhao, Z. He, J. Wang, and M. J. Prez-Jimnez. Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural p systems. *IEEE Transactions on Power Systems*, 30(3):1182–1194, May 2015.
28. X. Wang, G. Zhang, F. Neri, T. Jiang, J. Zhao, M. Gheorghe, F. Ipate, and R. Lefticaru. Design and implementation of membrane controllers for trajectory tracking of nonholonomic wheeled mobile robots. *Integrated Computer-Aided Engineering*, 23(1):15–30, 2016.

29. X. Wang, G. Zhang, J. Zhao, H. Rong, F. Ipate, and R. Lefticaru. a modified membrane inspired algorithm based on particle swarm optimization for mobile robot path planning. *International Journal of Computers*, 6:732–745, Oct. 2015.
30. J. Yuan, D. Guo, G. Zhang, P. Paul, M. Zhu, and Q. Yang. A resolution-free parallel algorithm for image edge detection within the framework of enzymatic numerical p systems. *Molecules*, Mar. 2019.
31. G. Zhang, M. J. Prez-Jimnez, and M. Gheorghe. Electric power system fault diagnosis with membrane systems. *Real-life Applications with Membrane Computing*, Jan. 2017.
32. Z. Zhang, Y. Su, and L. Pan. The computational power of enzymatic numerical P systems working in the sequential mode. *Theor. Comput. Sci.*, 724:3–12, 2018.
33. Z. Zhang, T. Wu, and L. Pan. On string languages generated by sequential numerical P systems. *Fundam. Inform.*, 145(4):485–509, 2016.
34. Z. Zhang, T. Wu, A. Paun, and L. Pan. Numerical P systems with migrating variables. *Theor. Comput. Sci.*, 641:85–108, 2016.
35. Z. Zhang, T. Wu, A. Paun, and L. Pan. Universal enzymatic numerical P systems with small number of enzymatic variables. *SCIENCE CHINA Information Sciences*, 61(9):092103:1–092103:12, 2018.

Research on an evaluation method of rice processing loss

Falin Jiang[1], Kang Zhou[1] *, Jian Xie[2], Jian Zhou[3], and Haocheng Fang[1]

[1] School of Math and Computer,
Development Strategy Institute of reserve of food and material,
Wuhan Polytechnic University, Wuhan 430023, China
[2] China Grain Wuhan Scientfic Reserch & Design Institute, Co.ltd,
Wuhan 430023, China
[3] School of Food Science and Engineering,
Wuhan Polytechnic University, Wuhan 430023, China
{jiang93214750@gmail.com,zhoukang_wh@163.com}

Abstract. The decision of rice processing enterprises on the type and quantity of processed rice is mainly determined by the nature of raw materials and market conditions, but it is rarely considered whether the processing technology of the enterprise is suitable for the production of the rice model. In this paper, from the perspective of rice processing loss, a method of evaluating the processing loss of rice produced by enterprises is proposed by means of data analysis. The specific content of the method is to build a fitting simulation model of unit power consumption and rice yield based on production data of different scales, equipment and processes in various types of enterprises across the country, thereby establishing an evaluation system for rice processing loss; Based on the production data of the enterprises that need to be evaluated, an enterprise data optimization model of unit power consumption and rice yield rate is constructed; and the results of evaluation and diagnosis of the processing loss of the rice processing enterprise are combined. The application of this evaluation method to the grain industry can provide practical guidance for rice processing enterprises in energy conservation and consumption reduction in the processing sector.

Keywords: rice processing loss,fitting simulation model, enterprise data optimization model,evaluation system,evaluation method

1 Introduction

China is not only a big rice producing country, but also a big rice processing and consuming country. Although there is a large number of rice processing enterprises , the processing loss of rice is still very obvious. The rice yield is still at a low level for a long time and the unit energy consumption is still high [1–5]. That greatly threatens the national food security and sustainable development

* Corresponding author.

strategy. With the rapid development of network technology, data storage technology and Internet technology, as well as the popularity of computer hardware and software, this paper considers the use of data analysis methods for national rice processing enterprises. A method for evaluating the processing loss of rice produced by an enterprise is proposed. Determine whether the processing loss of rice processing enterprises exceeds the standard, so as to objectively evaluate and diagnose rice processing enterprises [6, 7].

Rice processing enterprises are traditional enterprises, and most small and medium-sized processing enterprises have not yet formed a certain scale of data informatization [8–10]. However, large and medium-sized rice processing enterprises account for only 15% of all sizes of processing enterprises in the country, which means that more than 85% of rice processing enterprises in the country do not have complete records of processing data [11-13]. This has led most rice processing enterprises to fail to correctly and effectively understand the actual data of certain processing links of the enterprise, so it is impossible to find out the problems that the processing enterprises may have in the production process, and eventually lead to the emergence of rice processing enterprises in the production situation. Low rice yields and excessive power consumption per unit cannot solve the problem fundamentally. Obviously, this kind of thing is contrary to the food recycling economy mentioned in the national thirteenth five-year plan for resource-conserving society and food industry construction, which promotes energy conservation and emission reduction and food reduction. Food industry. How to solve this problem correctly and effectively is also very urgent [14-16].

In view of the production status of China's rice processing enterprises, this paper through the relevant data sets provided by China Grain Wuhan Scientfic Reserch & Design Institute, Co.ltd and the discussion and investigation of some rice processing enterprises of different scales. In-depth analysis of the current status of China's rice processing losses and the deep-seated causes, combined with big data, data analysis and data processing to propose a method to evaluate the processing loss of rice produced by enterprises [17], for the processing of rice processing enterprises Practical guidance on energy saving and consumption reduction.

2 Evaluation system for rice processing loss

2.1 Basic concepts

The production costs of rice processing enterprises mainly come from energy consumption and raw material consumption. The energy consumption is mainly reflected in the industrial electricity consumption of process equipment such as rice machine, rice milling machine, rice processing and other coarse grain separators. Raw material consumption refers to the consumption of rice raw materials. In rice processing enterprises, energy consumption and raw material consumption can be described by unit electricity consumption and rice production. Therefore, rice yield and unit electricity consumption are the most effective data to assess

the losses of processing enterprises. An important basis for measuring the technical level of rice processing enterprises. The relevant definitions of rice processing losses are as follows:

Let W be the total electricity consumption of the rice processing in a certain period of time, A is the total weight of the rice produced by the rice processing enterprise during this period, and R is the total rice use of the enterprise in the rice processing during this period.

The unit power consumption of rice during processing is defined as follows:

$$p = \frac{W}{\sum_{j=1}^{n} A_j} \tag{1}$$

The unit power consumption of the j model rice during processing is defined as follows:

$$p_j = \frac{\sum_{i=1}^{s} W_{ij}}{\sum_{i=1}^{s} A_{ij}} \tag{2}$$

The rice yield of rice processed into rice is defined as follows:

$$q = \frac{\sum_{j=1}^{n} A_j}{R} \times 100\% \tag{3}$$

The rice yield of rice processed into j model rice is defined as follows:

$$q_j = \frac{\sum_{i=1}^{s} A_{ij}}{\sum_{i=1}^{s} R_{ij}} \times 100\% \tag{4}$$

Where n is the sum of the number of rice varieties processed and produced by the rice companies in the period, i is the number of rice processing enterprises at that time, and s is the number of rice processing enterprises in the period.

2.2 Construction of evaluation system for rice processing loss

Rice unit power consumption and rice yield are important factors reflecting the production efficiency of rice processing enterprises. Therefore, the unit power consumption and rice yield of rice processing enterprises are selected as important indicators for constructing the evaluation system of rice processing enterprises. The evaluation system for rice processing loss is a system for evaluating whether the products of rice processing enterprises are suitable for processing. The evaluation system for rice processing loss consists of two parts: the evaluation index and the evaluation standard value. At present, the evaluation indicators are determined as follows: According to the two dimensions of “processing

scale and rice type" of rice processing enterprises, two types of indicators: rice unit electricity consumption and rice yield are set. Our team plans to set evaluation metrics from more dimensions as the sample collection of the database increases. For example, the evaluation indicators can be extended to seven dimensions "processing scale, rice model, processing equipment, region, processing technology and the origin and quality of rice". The evaluation standard value is the social average production level of the indicator on the same type of rice produced by rice processing enterprises of the same scale.

The construction of the rice processing loss assessment system is mainly completed in five steps:

Step 1: According to the statistical principle, a sample survey was conducted on the whole society of rice processing enterprises to establish a database. General production data for rice processing companies of all scale include: the weight of paddy use , production weight of various rice, and total electricity consumption for rice processing..

Step 2:According to the important factors affecting the production cost of rice processing enterprises and the data characteristics in the sample database, determine the evaluation indicators and the fitted simulation model used to establish the evaluation criteria values.

Step 3: After pre-processing the data, classify the data in the sample database according to the scale dimension.

Step 4: For each type of data, the evaluation criteria value is calculated by fitting a simulation model that evaluates the standard values.

Step 5: According to the two dimensions of "processing scale, rice type", the evaluation index and the evaluation standard value are combined to obtain a rice processing loss evaluation system.

The flow chart of the rice processing loss evaluation system is shown in Fig. 1.

2.3 Application process of evaluation method for rice processing loss

The evaluation system of rice processing loss reflects the average level of loss produced by rice processing enterprises of the same scale in the same type of rice. The loss of rice produced by enterprises refers to the electricity consumption of rice produced in the unit and the rice yield. For a rice processing enterprise, its production data can be designed to calculate the unit power consumption and rice yield of different type of rice. Thus, the unit power consumption and the rice yield of different type of rice in rice processing enterprises can be compared with rice. Comparing the corresponding evaluation standard values of the processing loss evaluation system, it is analyzed whether the processing of these varieties of rice is reasonable, and thus, the production status of the rice processing enterprises is evaluated and diagnosed. Therefore, based on the evaluation system of rice processing loss, the evaluation method of rice processing loss was designed. The specific detailed steps of the method are as follows:

Step 1: Construct a rice processing loss assessment system.

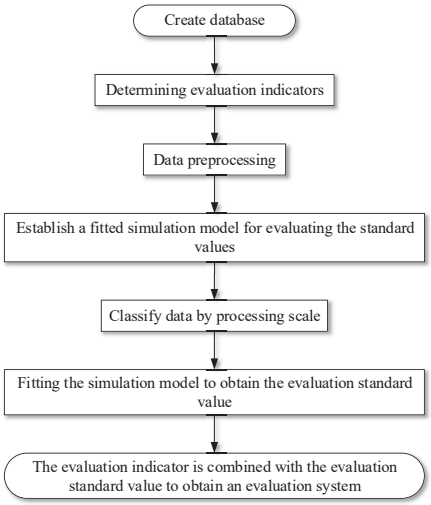


Fig. 1. The flow chart of the rice processing loss evaluation system

Step 2: Collect data generated during the production process of the rice processing enterprise to be evaluated, and establish a database. These production data include: paddy use weight, The weight of each type of rice, and total electricity consumption.

Step 3: Pre-processing data in the database and establishing an enterprise optimization model for the data.

Step 4: Optimize the processing data of the company

Step 5: After pre-processing the data in the database, according to the rice variety processing dimension, the data processing simulation model of the rice processing enterprise is used to calculate the rice processing loss of the rice processing enterprise, that is, the unit power consumption and the production rate of each variety of rice produced by rice processing enterprises.

Step 6: According to the processing dimension and the rice type, the calculation result of the rice processing enterprise is compared with the corresponding evaluation standard value of the rice processing loss evaluation system, and the deviation value of the unit power consumption is calculated. Rice production by rice processing companies.

Step 7: According to the symbol and size of the deviation between the unit power consumption and the rice yield of the rice processing enterprise, the production status evaluation of the rice processing enterprise is completed.

Step 8: According to the evaluation results, contact the actual situation, conduct in-depth investigations into the enterprise, and then diagnose the production status of the rice processing enterprises to find out the problems and solutions of the processing enterprises.

The flow chart of the evaluation method of the enterprise to be evaluated is shown in Fig. 2.

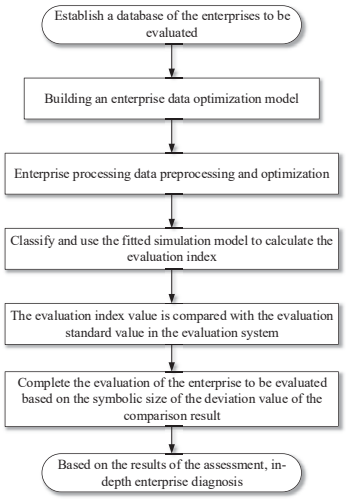


Fig. 2. The flow chart of the rice processing loss evaluation system

3 Model analysis and calculation of rice processing loss evaluation method

Four studies were crucial when studying rice processing loss assessment methods. Data preprocessing in the database, evaluation of the fitted simulation model of the standard values, design of the enterprise data optimization model, and algorithm design of the rice processing loss. The following is a separate introduction.

3.1 Sample data set and data preprocessing

The data set comes from the questionnaire data of the whole society of the rice processing industry, so the data set inevitably has the characteristics of low quality, large size, no noise and inconsistency. Therefore, data preprocessing of this data set is an important task. The data preprocessing used in this article mainly includes three methods: data cleaning, data integration and data simplification. The order of the three processing operations is not sequential, and some preprocessing methods can be used multiple times. The following is a brief introduction: data cleansing is mainly to deal with lost and dirty data in the data set; data integration mainly integrates and processes data in multiple data sources, and resolves semantic ambiguity into a consistent data set with the

same attributes and consistent names; Reduce the main identification of data sets that need to be mined, remove irrelevant dimensions, reduce the amount of data and reduce the scope of processing.

The data preprocessing of the data set is as follows:

Step 1: Import the two data sets into the SPSS software, analyze the correlation of the attribute indicators, and obtain the correlation coefficient between the indicators and other indicators.

Step 2: Discuss and study with data source enterprises and rice processing enterprises of different scales, and based on the SPSS correlation coefficient table, screen out the attribute indicators related to the rice processing link, thereby deleting irrelevant attributes and reducing the data dimension.

Step 3: Use Microsoft Excel to detect similar duplicate objects caused by multiple data sources and delete similar duplicate data items.

Step 4: For the sample data with data loss in the data set, find the original questionnaire according to the number. If the data is still missing, just delete the data.

Step 5: Design the enterprise data optimization model using linear regression ideas, and delete the abnormal data items that do not meet the rules. The enterprise data optimization model is described in detail below.

3.2 Evaluation of the establishment of diagnostic models

The evaluation diagnostic model consists of two sets of models: a fitted simulation model for evaluating standard values and an enterprise data optimization model. The two sets of models are described in detail below.

Fitting simulation model for evaluating standard values: The fitting simulation model of the evaluation standard value is to process the production data of the whole society rice processing enterprises, and obtain the social average level of unit electricity consumption and rice yield of different types of rice under the same scale.

Therefore, the production data of the whole society rice processing enterprises are first classified according to the scale of the enterprises. We divide the scale into eight categories: 45, 75, 120, 150, 180, 225, 300 and 450 tons / day. In this paper, two types of rice are studied: japonica and indica rice; for each type of rice, the rice varieties produced by the enterprise are divided into six categories: High quality first grade rice, high quality second grade rice, high quality third grade rice, first grade rice, second grade rice and third grade rice.

Let W be the total electricity consumption of the rice processing enterprises in a certain period of time, A represents the total rice output of the enterprise rice processing during the period, R represents the rice processing volume during the rice processing, p represents the unit power consumption of processing rice during this period, q represents the rice yield of the rice processed by the enterprise during this period, i represents the rice processing enterprise number, and j represents the rice type of the rice, That is, A_{ij} represents the j-type rice

production of the rice processing company in the period, p_j represents the unit power consumption of the j-type rice produced by the rice processing company during the period, and q_j represents the rice yield of the j-type rice produced by the rice processing enterprise during the period, W_i represents the total electricity consumption of rice processing companies during this period.

The general idea of constructing a fitted simulation model for evaluating the standard values is to treat the production data of the enterprise as a whole and establish different rice unit power consumption based on the idea of linear least squares fitting. Establish the unit power consumption and the production rate of the different types of rice for the calculation. The fitting simulation model of the evaluation standard value is established respectively for the unit power consumption and the rice yield of different types of rice.

(1)Fitting simulation model of unit power consumption

The unitized power consumption model calculates the value and the actual power consumption to achieve the minimum squared error squared as the target, Based on the calculated values of the unit power consumption model and the actual power consumption, the optimal fitting simulation model for the power consumption of different types of rice with different scales of japonica and indica rice was established. The yield and total electricity consumption of rice are known, and the evaluation standard type of the power consumption per unit of rice is obtained.

The objective functions of different models of different types of rice unit power consumption simulation models are as follows:

$$\min \sum_{i=1}^{s} (\sum_{j=1}^{6} A_{ij} \cdot p_j - W_i)^2 \tag{5}$$

The constraint is:

$$p_j > 0 \tag{6}$$

(2)Fitting simulation model of the rice yield

The combined rice productivity model is used to calculate the actual rice yield, aiming at the square of the least square error. According to the rice productivity model and the actual rice yield, the optimal fitting simulation model is obtained. rice yields of different types of rice from different japonica and indica rice varieties were established. The yield of rice and the amount of rice used are known, and the type of evaluation standard for the rice yield is obtained.

The objective functions of different models of different types of the rice yield simulation models are as follows:

$$\min \sum_{i=1}^{s} (\sum_{j=1}^{6} (\frac{A_{ij}}{q_j} - R_i))^2 \tag{7}$$

The constraint is:

$$q_j > 0 \tag{8}$$

Among them, s is the number of rice processing enterprises under different scales, and the fitting simulation model of the calculation standard values of japonica and indica rice is the same. In order to reduce the complexity of the calculation space, this paper calculates $\frac{1}{q_j}$, when Q_j is involved in the standard value of rice evaluation.

Enterprise Data Optimization Model: The enterprise data optimization model is to process or modify the abnormal data items that do not conform to the regularity after the data processing of the plurality of production data of the rice processing of the enterprise to be evaluated, and obtain the rice processing data of the high-quality enterprise to be evaluated.

Let W be the total electricity consumption of the rice processing enterprises in a certain period of time, A represents the total rice output of the enterprise rice processing during the period, R represents the rice processing volume during the rice processing, p represents the unit power consumption of processing rice during this period, q represents the rice yield of the rice processed by the enterprise during this period, i represents the rice processing enterprise number, and j represents the rice type of the rice , That is, A_{ij} represents the j-type rice production of the rice processing company in the period, p_j represents the unit power consumption of the j-type rice produced by the rice processing company during the period, and q_j represents the rice yield of the j-type rice produced by the rice processing enterprise during the period, W_i represents the total electricity consumption of rice processing companies during this period.W_{ij} indicates the power consumption of the i-type rice processing enterprise in producing j-type rice during this period.

The overall idea of constructing the enterprise optimization model is to treat the production data of the enterprise to be evaluated as a whole, based on the idea of linear regression and combined with the production rules of rice processing enterprises, and optimize the data to obtain high-quality production data. The enterprise optimization model is established separately for the unit power consumption and the metering rate of the enterprise to be evaluated.

(1)Enterprise data optimization model of unit power consumption

Taking the evaluation standard value of unit power consumption and the actual power consumption to achieve the squared error of the minimum error as the goal, the enterprise data optimization model of the unit power consumption of different types of rice is established. It is known that the total amount of electricity used by rice processing enterprises, the data of different types of rice production enterprises, and the evaluation standard values of unit consumption of the same scale, calculate the electricity consumption of different types of rice.

The objective functions of different types of rice unit power consumption enterprise optimization models are as follows:

$$\min \sum_{j=1}^{6} \left(\frac{W_{ij}}{A_{ij}} - p_j\right)^2 \tag{9}$$

The constraint is:

$$\sum_{i=1}^{6} w_{ij} = W_i \tag{10}$$

(2)Enterprise data optimization model for rice yield

Taking the evaluation standard value of rice yield and the actual rice yield to achieve the squared error of the minimum error as the goal, the enterprise data optimization model of the rice yield of different types of rice is established. It is known that the total rice used by rice processing companies, the data of different types of rice production enterprises, and the evaluation standard values of unit consumption of the same scale, Calculate the amount of rice used by different types of rice.

The objective functions of different types of rice yield enterprise optimization models are as follows:

$$\min \sum_{j=1}^{6} (r_{ij} \cdot q_j - A_{ij})^2 \tag{11}$$

The constraint is:

$$\sum_{i=1}^{6} r_{ij} = R_i \tag{12}$$

The evaluation standard value of the rice processing loss calculated by the formulas (5) and (7) is taken as the center of the space coordinate system, and the unit power consumption and rice yield of the different types of rice of the enterprise calculated by the formulas (9) and (11) are calculated according to the formula. (1) and formula (3) are converted into unit power consumption and rice yield as coordinate points. Find the coordinate points far from the center of the coordinate system, and modify or directly delete the enterprise data items corresponding to these coordinate points to complete the final data optimization.

3.3 Algorithm design of rice processing loss

The rice processing loss evaluation system requires a reasonable evaluation method, an effective evaluation and diagnosis model, and an accurate calculation of the evaluation standard value and the evaluation index of the enterprise to be evaluated. In this paper, the evaluation standard value is the same as the calculation idea of the enterprise evaluation index to be evaluated. Therefore, the calculation evaluation standard value is used as a detailed example. The specific operation steps are as follows:

Step 1: using Microsoft Excel 2010 for data processing of data reduction and data quality for japonica and indica rice processing enterprises;

Step 2: using 1stOpt 7.0, Lingo and IBM SPSS Statistics 21 software programming, the pre-processed data is used to calculate the average rice yield and unit power consumption as the reference value;

Step 3: analyze the correlation coefficient of the three software valuation evaluation standard values, and finally select 1stOpt 7.0 with low mean square error and high fitting precision;

Step 4: using the various algorithms in 1stOpt 7.0 to calculate the pre-processed data using the fitted simulation model, and finally select the Macquart Algorithm with the shortest calculation time;

Step 5: using the enterprise data optimization model to complete the data optimization for the processed data sets of japonica and indica rice under the data pre-treatment;

Step 6: the data set optimized by the data is calculated by the McQuart algorithm in 1stOpt 7.0 using the fitted simulation model, and the average of the rice yield and unit power consumption is taken as the evaluation standard value.

4 Experimental results and analysis

4.1 Experimental data source and experimental environment

The database is the industry statistics provided by China Grain Wuhan Scientfic Reserch & Design Institute, Co.ltd, including data on rice processing by rice processing enterprises of different scales in the country. Among them, there are 5851 samples of japonica and 6653 samples of indica rice. The sample distribution of japonica and indica rice under common scale is shown in Table 1.

Table 1. Number of samples of Japonica and Indica Rice in the database at common scale

Scale (tons/day)	45	75	120	150	180	225	300	450
Number of japonica samples	436	638	500	781	397	404	405	157
Number of indica rice samples	479	750	576	874	432	455	455	190

Because the database of rice processing enterprises is small in most scales, this paper only analyzes the data of several common processing scales in rice processing enterprises. Table 1 shows the number of samples of rice processing enterprises with several common processing scales under japonica and indica rice in the database.

The experimental environment is:

Processor: Intel Core i7-7700 @ 3.60GHz quad core;

Memory: 16 GB;

Operating system: Windows 7 Ultimate 64-bit;

Software: Microsoft Excel 2010, 1stOpt 7.0, Lingo, and IBM SPSS Statistics 21.

4.2 Evaluation system for calculating rice processing loss

According to the statistical principle, sample the whole society rice processing enterprises and establish a database. The common production data of rice processing enterprises in all walks of life include: rice use, Production weight of various types of rice, and total electricity consumption of rice processing. Table 2 shows some data after pretreatment of the processing data of japonica in the database of rice processing enterprises.

Table 2. Partial data display after pretreatment of japonica processing data in the database of rice processing enterprises

Company number	Scale (daily production capacity)	High quality first grade rice (kg)	High quality secondary rice (kg)	High quality tertiary rice (kg)	First grade rice (kg)	Second grade rice (kg)	Third grade rice (kg)	Rice use (kg)	Power consumption (Kw · h)
698	4185	601211	21441	0	0	0	0	929331	28510000
5096	3750	31984	0	0	283235	79356	0	672947	31541025
1261	3000	275611	0	0	0	0	0	418000	15430000
7956	2700	0	0	0	0	0	200880	324000	12052800
2352	2550	0	0	0	122011	82890	21175	334630	18481640
7722	2260.5	0	0	0	1439	0	0	2248	71850
3234	2250	53100	98998	0	0	66902	38400	415298	13289536

Table 3. Evaluation index and evaluation standard value of the unit power consumption of japonica in the rice processing loss assessment system

Processing scale	Original sample size	Sample size after data processing	High quality first grade rice	High quality second grade rice	High quality third grade rice	First grade rice	Second grade rice	Third grade rice	Fitting accuracy (R2)
45	436	358	45.31	42.93	36.65	49.45	48.23	47.91	0.79
75	638	406	52.99	52.59	52.15	49.88	47.62	44.3	0.75
120	500	398	58.7	58.47	58.23	53.3	46.85	33.39	0.8
150	781	660	49.29	47.57	46.49	56.47	53.48	40.89	0.66
180	397	257	55.42	53.43	51.48	53.75	52.3	43.98	0.82
225	404	251	59.47	54.18	50.41	57.66	52.35	45.91	0.8
300	405	367	53.86	51.7	38.82	62.15	55.69	38.16	0.8
450	157	157	55.03	53.81	51.22	56.94	52.47	41.61	0.8

After pre-processing the data in the database, calculate the rice processing loss of the rice processing enterprise, that is, calculate the evaluation standard values of different types of japonica and indica rice by formula (5) and formula (7). The calculation results of the evaluation index and the evaluation standard value in the rice processing loss evaluation system are shown in the following table. Table 3 shows the evaluation index and evaluation standard value of the unit power consumption of japonica in the rice processing loss assessment system, Table 4 shows the evaluation index and evaluation standard value of the rice yield of indica rice in the rice processing loss assessment system.

After optimizing the processing data of japonica, the calculation result using the formula (5) of the McQuat algorithm in 1stOpt 7.0 is shown in Table 3. It can be seen that the fitting accuracy of the evaluation standard value in the evaluation system is high, indicating the evaluation index calculated by the

fitting simulation model fits well with the social average level of the unit power consumption of different types of rice under the same scale.

After optimizing the processing data of indica rice, the calculation results using the formula (7) using the McQuat method in 1stOpt 7.0 software are shown in Table 4. From Table 4, it can be seen that the rice in the evaluation system of rice processing loss is calculated in the rice. The fitting accuracy of the evaluation index and the evaluation standard value is very high, and it is close to 1, indicating that the evaluation index calculated by the fitting simulation model and the social average level of the rice yield of different types of rice under the same scale are planned. The degree of integration is very high.

By analyzing the fitting precision of the evaluation standard values calculated by the formulas (5) and (7), it is found that the fitting precision of the formula (5) is significantly smaller than the formula (7). It shows that the unit power consumption of different rice processing enterprises at the same scale has a larger difference than the rice yield. That is, the energy saving and consumption reduction of the processing enterprises in the processing of rice should focus on the unit electricity consumption of rice processing.

When comparing the horizontal and vertical analysis of the evaluation standard values, we found that there are certain regularities in the evaluation standard values of japonica and indica rice. The intuitive regular changes are shown in Fig. 3 and Fig. 4, Fig. 3 shows the evaluation standard value of unit electricity consumption of 225 japonica in the rice processing loss evaluation system,Fig. 4 is a standard evaluation value of rice productivity of high quality tertiary indica rice in the rice processing loss evaluation system.

Table 4. Evaluation index and evaluation standard value of the rice yield of indica rice in the rice processing loss assessment system

Processing scale	Original sample size	Sample size after data processing	High quality first grade rice	High quality second grade rice	High quality third grade rice	First grade rice	Second grade rice	Third grade rice	Fitting accuracy (R2)
45	479	427	66.50%	66.55%	66.99%	64.97%	64.99%	65.13%	0.96
75	750	634	65.31%	65.34%	66.81%	64.02%	64.39%	64.43%	0.92
120	576	460	64.38%	64.84%	65.30%	64.21%	64.26%	64.52%	0.97
150	874	794	65.52%	65.68%	65.83%	64.89%	64.99%	65.57%	0.98
180	432	360	64.95%	65.15%	65.65%	64.08%	64.30%	64.32%	0.99
225	455	344	63.93%	65.40%	65.86%	61.45%	61.88%	61.94%	0.97
300	455	422	63.97%	64.13%	64.20%	63.82%	63.86%	64.07%	0.99
450	190	143	63.01%	63.09%	63.91%	63.04%	63.05%	63.71%	0.97

It can be seen from Fig. 3 and Fig. 4 that the unit power consumption of high-quality rice and ordinary rice of japonica and indica rice increases with the improvement of rice quality, and the rice yield decreases with the increase of processing scale. Tables 2 and 3 use the fitted simulation model and the enterprise data optimization model to calculate the evaluation method. The mean square error is smaller and the fitting precision coefficient is higher. It is indicated that the evaluation index and evaluation standard value of the rice processing loss evaluation system are feasible.

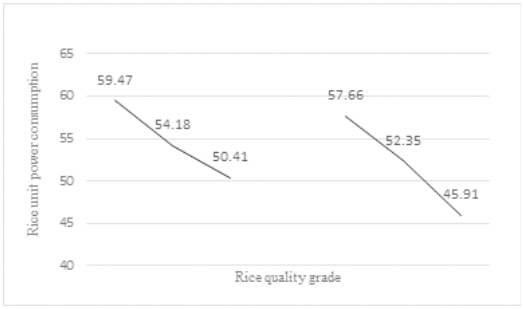


Fig. 3. Evaluation standard value of unit electricity consumption of 225 japonica in the rice processing loss evaluation system

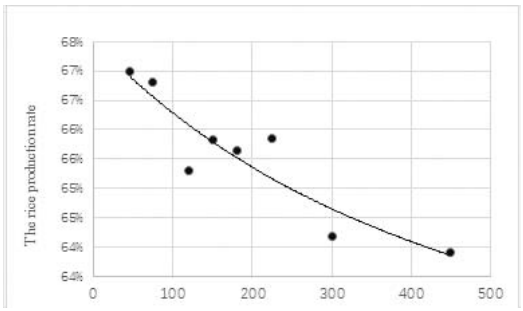


Fig. 4. Evaluation value of rice productivity of high quality tertiary indica rice in the rice processing loss evaluation system

4.3 Application of evaluation method for rice processing loss of enterprises to be evaluated

Through the fitting simulation of data pre-processing, evaluation index and enterprise data optimization of some rice processing data in a certain period of time, the electricity consumption of different types of rice units power consumption and rice yield. The company's calculations are then compared to evaluation criteria of the same size. The comparison results are shown in the table below. Table 5 is the evaluation standard value and unit power consumption table calculated by the enterprise to be evaluated, and Table 6 is the evaluation standard value and rice yield calculated by the enterprise to be evaluated.

Tables 5 and 6 show that in the comparison between the calculated unit power consumption and the rice yield and the evaluation standard value of the rice processing loss evaluation system, only the high-quality tertiary rice and the ordinary tertiary rice are available. There are significant d ifferences with the evaluation indicators. The electricity consumption per unit of the assessed enterprise was significantly lower than the evaluation standard value of the same scale, and there was no significant difference between the rice yield and the evaluation index of the same scale.

Table 5. Evaluation standard value and unit power consumption table calculated by the enterprise to be evaluated

	High quality first grade rice	High quality secondary rice	High quality tertiary rice	First grade rice	Second grade rice	Third grade rice
Evaluation standard value	59.47	54.18	50.41	57.66	52.35	45.91
Enterprise calculated value	59.20	54.83	53.69	57.58	52.01	50.36

Table 6. Evaluation standard value and rice yield calculated by the enterprise to be evaluated

	High quality first grade rice	High quality secondary rice	High quality tertiary rice	First grade rice	Second grade rice	Third grade rice
Evaluation standard value	64.82%	66.34%	67.08%	64.73%	64.84%	65.13%
Enterprise calculated value	64.74%	66.50%	66.32%	65.09%	65.13%	64.93%

Based on the results of the comparison, we contacted the actual situation and we conducted a purposeful investigation into the company. Therefore, the production status of the rice processing enterprise is diagnosed, and then the problems and solutions existing in the processing enterprise are found, and the following conclusions are obtained:

(1) There is a problem in the production status of the rice processing enterprise, mainly because the company has problems in the production of high-quality tertiary rice and tertiary rice.

(2) The sales of high-quality third-grade rice and third-grade rice in the market are not ideal, and the expenses incurred by enterprises in upgrading equipment are not proportional to the benefits of selling high-quality three-grade rice and third-grade rice.

(3) The company did not update the processing equipment of high-quality third-grade rice and third-grade rice in time, resulting in lower unit power consumption of high-quality three-grade rice and third-grade rice than the evaluation standard value.

(4) It is recommended that the rice processing enterprise reduce the processing and production of high-quality three-grade rice and three-grade rice.

In summary, the evaluation method can provide practical guidance for rice processing enterprises in energy saving and consumption reduction in the processing section in the rice processing loss evaluation system.

5 Conclusions and prospects

From the perspective of rice processing loss, this paper proposes a method to evaluate the processing loss of rice produced by enterprises by means of data analysis, and establishes an evaluation system for rice processing loss. Through the model analysis and experimental results of the evaluation method, it can be seen that the evaluation method has the following advantages: (1) The evaluation index and evaluation standard value of the rice processing loss evaluation system

can truly and effectively reflect the production efficiency of the rice processing enterprises; (2) The evaluation method of rice processing loss can quickly and effectively evaluate and diagnose the processing loss of the rice processing enterprises; (3)The evaluation method of rice processing loss can provide practical guidance for rice processing enterprises in energy saving and reducing consumption reduction in processing.

By assessing the processing loss of the enterprise rice processing enterprises, the problems of the processing links can be discovered in time and effectively managed, so as to achieve efficient processing efficiency and optimal allocation of resources. In the future, we will further improve the evaluation system by collecting more dimensions of rice processing enterprises processing data, such as scale, rice model, processing equipment, region, processing technology and the origin and quality of rice more dimensions of rice processing enterprises of relevant information, continue to expand and optimize this evaluation system, then build a conservation-minded society in our country and promote the ”13th Five-Year” development plan for the grain industry.

Acknowledgments. This work is partially supported by The Subproject of the National Key Research and Development Program of China (Grant No. 2017YFD0401102-02).

References

1. Li Weiqiang. Automatic Monitoring System for Rice Processing[J]. Grain Processing, 2018, 43(06): 28-29.
2. Liu Jingxue, Cai Yuanpei, Li Changle, Shi Haihui. Discussion on the key process control technology of rice processing [J]. Research on food issues, 2018 (06): 24-28.
3. Li Yinan. Zoomlion's overall solution for building grain processing intelligent engineering [J]. Agricultural Machinery, 2018 (11): 132.
4. Wang Hongmei, Li Yinan. Zoomlion's overall solution for building grain processing intelligent engineering [J]. Agricultural Machinery Quality and Supervision, 2018 (10): 41.
5. Li Wenjing, Liu Wenliang, Xia Yongxing, Zhang Yuan, Chen Wei, Lin Tao, Sun Jianchao, Li Guangwei, Cao Haijun, Shu Jiao, Yu Wenhai.Discussion on friction loss in rice processing factory[J]. Food and Food Industry,2018 , 25 (05): 5-7 + 12.
6. Wu Yichen, Fan Zhe. Capital Negative Distortion and the Increase of Profit Rate of Grain Processing Enterprises——Taking Rice Processing Industry as an Example[J]. Food Economics Research, 2018, 4(01): 31-42.
7. Chen Yandong. Cost Management Problems and Countermeasures of Grain Processing Enterprises [J]. Finance and Accounting, 2018 (18): 123+125.
8. Grain Processing Industry [J]. Hunan Agriculture, 2018 (06): 6.
9. Chen Zhe, Deng Yi, Zhang Shunmi, Hu Lingyan, Wang Xinhua. Research on the Status Quo and Countermeasures of Loss and Waste in Grain Processing in China[J]. Grain Science and Technology, 2018, 43(05): 96-99.
10. Wu Xianting. On the Quality Control of Rice Processing Process [J]. Grain Processing, 2018, 43 (02): 23-25.

11. Zhu Jianwu, Lei Yanqing, Yan Jun. Research on Improving the Whitening Accuracy of Rice Milling Machine to Reduce Broken Rice Rate[J]. Grain Science and Technology,2018,43(03):83-85.
12. Qian Chaocheng. Analysis of energy-saving and grain-saving strategies of grain processing enterprises [J]. Southern Agricultural Machinery, 2018, 49 (05): 46.
13. Zhu Xihui. Over-processing behavior of rice processing enterprises and its influencing factors——Taking Jiangsu as an example[J]. Food Science and Technology,2018,43(02):105-107+118.
14. QiangANG, Agricultural products processing characteristics and quality evaluation, Journal of Integrative Agriculture, Volume 17, Issue 5, 2018, Pages 975-976.
15. Ye-lu ZHENG, Qi-yun HE, Ping QIAN, ZeLI, Construction of the Ontology-Based Agricultural Knowledge Management System, Journal of Integrative Agriculture, Volume 11, Issue 5, 2012, Pages 700-709.
16. Ying Xu, Biao Zhang, Lingxian Zhang, A technical efficiency evaluation system for vegetable production in China, Information Processing in Agriculture, Volume 5, Issue 3, 2018, Pages 345-353.
17. Wuhan University of Light Industry. Data analysis system and method for grain processing loss: China, CN201810408482.2[P].2018-09-25.

Formal Approach to cP System Verification

Yezhou Liu ✉, Radu Nicolescu, and Jing Sun

The University of Auckland, Auckland, New Zealand
yliu442@aucklanduni.ac.nz, r.nicolescu@auckland.ac.nz,
jing.sun@auckland.ac.nz

Abstract. As a recently proposed membrane computing model, cP systems are capable of solving computational hard and distributed problems. Although several membrane systems were formally verified in previous research, none of their approaches was applicable to cP systems. To formally verify the safety and liveness properties of cP systems, we solve a famous NPC problem – the subset sum problem in cP systems and use the PAT3 and ProB model checkers to verify the solution. Our cP solution outperforms previous work in time complexity and uses fewer rules. To perform model checking in cP systems, we define several mapping rules from cP notation to formal verification languages CSP# and B. This work is the first study on formal verification of cP systems, our study showed that cP solutions can be effectively transformed into model checking problems and verified automatically.

Keywords: Formal Verification · Model Checking · cP systems · P systems · Subset Sum Problem

1 Introduction

Since membrane computing was proposed in 1998 [1], many variants of membrane systems (P systems) have been published, which include P systems with active membranes [2], tissue P systems [3], Spiking Neural P systems [4], kernel P systems [5] and P systems with complex objects (cP systems) [6]. Major P system variants share two fundamental features: computational completeness and computational efficiency. P systems have equivalent computational power to Turing machines and can create exponential workspace like cell division in real life [7]. By using P systems, many computational hard problems can be solved in polynomial or linear time/steps.

To verify P systems, formal approaches including model checking and theorem proving are used. One of the earliest P verification work simulated P systems by communicating X-machines [8]. Later, many model checking tools were used to verify P systems including Omega [9], SPIN [9–13], NuSMV [13–15], ProB [16], UPPAAL [17], PRISM [15] and kPWorkbench [13, 15]. Besides model checking, theorem proving was also used to formally verify P system variants [9, 14, 18–20].

By combining features from both cell-like P systems and tissue P systems, cP systems are capable of solving computational hard and distributed problems.

Many NPC or NP-hard problems were solved in cP systems over the past few years, which include the Hamiltonian cycle problem [21], travelling salesman problem [21] and the propositional satisfiability problem [6]. With the development of cP systems, it is important to simulate cP systems and formally prove their safety and liveness properties.

In this study, we propose a subset sum solution in cP systems with a ruleset of five rules, which can solve the problem in at most $n+2$ steps. To verify the cP solution, we define two mapping guidelines from cP system specifications into CSP# and B notation. We model check the properties of our cP solution by using PAT3 [22] and ProB [23] and illustrate how to design cP rules to improve the model checking efficiency.

As an extensible framework, PAT3 is suitable for verifying distributed and concurrent systems. Compare to other model checking tools, PAT3 can perform competitively in terms of both running time and extensibility [22]. When modelling cP systems in PAT3, both communication among multiple top-cells and manipulations of sub-cells can be simulated by using events and processes. PAT3 also can simulate non-deterministic and probabilistic choices in cP systems. Another model checker we choose in this study is ProB, which was successfully used in a previous P system research [16]. ProB supports state-space visualization and a rich set of data operations, which is especially suitable for simulating cP sub-cell manipulations.

The paper is organized as follows. In Section 2, we include the background of cP systems, the existing P solution to the subset sum problem and the introduction of PAT3 and ProB. In Section 3, we introduce the cP notation and the solution to the subset sum problem. We illustrate how to transform cP system descriptions into formal verification problems in Section 4. A case study and evaluation are shown in Section 5. Section 6 concludes the work with future directions.

2 Background

Inspired by traditional P systems (tree-based, like a single cell) and tissue P systems (graph-based, multiple cells communicate through protein channels [3]), cP systems consist of networks of top-cells, where each top-cell can contain multiple nested sub-cells. Only top-cells in cP systems have high-level multiset rewriting rules, sub-cells are often treated as local datasets. Compare to other P system variants, cP systems often can effciently solve a computational difficult problem or represent a proper algorithm with a small number of high-level rules.

The fundamental design of a cP system includes Prolog style compound terms, high-level multiset rewriting rules and optional constraints – promoters and inhibitors. cP rules can be extended by communication constructs inspired from CSP and actor models [24]. In this study, to solve the subset sum problem, we use a simple version of cP systems, which only has one top-cell with multiple sub-cells.

In the rest of Section 2, we review the previous works on solving the subset sum problem by using P system variants and introduce the model checkers PAT3 and ProB.

2.1 P solutions to the subset sum problem

As a special case of Knapsack problem, the subset sum problem is NP complete. Many P system variants were used to solve the problem and compared to each other. An early work was published in 2004, where P systems with active membranes were used to solve the subset sum problem [25] and the Knapsack problem [26]. In the P solution, solving the problem could be linear, while the pre-computing required polynomial time [25]. Later, another linear solution to the subset sum problem was presented, which used P systems with membrane creation rules to solve the problem in at most $2n + 2k + 12$ steps, where n is the number of elements in the initial set and k is the constant to be reached [27]. Besides traditional P systems, other P system variants were also used to solve the problem in different research works (Table 1).

2.2 PAT3 and ProB

The first version of PAT (Process Analysis Toolkit) model checker was proposed in 2008 [34], which aimed to analyze event-based compositional systems. An extended version of communicating sequential processes (CSP) language which called CSP# was proposed in PAT to describe system specifications, which can help refinement checking, LTL(Linear Temporal Logic) checking and system simulation. A fully automatic approach on checking fairness properties using PAT was presented later by Sun et al. [35]. The second main release of PAT was focused on handling system analysis under fairness [36]. The latest version of PAT is PAT3, which has a four-layer structure: modeling layer, abstraction layer, intermediate representation layer and analysis layer. The four layers contain domain specific components, abstraction and reduction techniques, semantic models and model checking algorithms, respectively [22]. PAT3 can be extended with custom modeling languages, model checking algorithms and reduction techniques.

Two ways of system analysis are supported in PAT3: simulation and model checking. In the PAT3 simulator, we can trace how cP rules were used nondeterministically in different processes and track global state variables of the system. The model checker of PAT3 has several verification options, such as deadlock-freeness, nonterminating, divergence-freeness, deterministic, reachability, refinement, safety-LTL properties and liveness properties checking.

Similar to PAT3, ProB also contains a model checker and a refinement checker [37]. Two main proof activities in ProB are consistency checking and refinement checking. The consistency checking monitors if the invariants are preserved during different operations, the refinement checking is used to check if a machine is a refinement of another [23].

P system variants	**Number of rules**	**Number of Steps**	**Comments**
P systems with active membranes [25]	$5n + 5k + 18$	$2n + 2k + 2$	polynomial pre-computing time
P systems with membrane creation [27]	$12n + 4k + 49$	$2n + 2k + 12$	
P systems with active membranes [28]	$2n + 48$	$3n + 2 * min(k, \omega(A)) + 19$	ω is the weight function
P systems with active membranes and dissolution rules [29]	$6n + 3k + 14$	$O(n)$	semi-uniform
P systems with active membranes, weak division, dissolution and without polarization [30]	$O(n^2 + log(k))$	$2n+2*log(k)+20$	
Spiking Neural P systems by exploiting nondeterminism [31]	$2n + 2$	$O(1)$	semi-uniform, exponential time to build the SN P systems
Spiking Neural P systems using maximum parallelism, exploiting nondeterminism [32]	$2n + 2$	$O(1)$	an improved version of [31], the initializing time of the SN P systems was reduced to polynomial.
Tissue P systems with cell division [33]	$n*log(k+1)+5n+2*log(k+1)+3*log(n)+26$	$n + log(n) + log(k + 1) + 12$	
cP systems (This work)	5	$n + 2$	

Table 1: P solutions to the subset sum problem. In the figure, a semi-uniform setting means that for each instance of the subset sum problem, a specific P system is built to solve that instance.

ProB can work as both an animator and a model checker of cP systems. When simulating a cP model, the whole state-space of the system can be visualized and the shortest trace could be tracked. B language has built-in operations on logical predicates, numbers, sets, sequences, relations and functions, which is helpful on modelling cP sub-cells. In addition to B language, ProB also supports other modeling languages including Event-B, Z, TLA+ and CSP. Deadlock-freeness, invariant violations, reachability, LTL/CTL assertions and other errors can be checked in ProB.

3 Methodology

This section includes two parts: the introduction to the cP notation and our cP solution to the subset sum problem. We first introduce the cP syntaxes by using a couple of examples; then explain our solution to the problem and discuss how to expand cP rules to reduce the state-space for model checking.

3.1 cP system notation

cP notation includes compound terms and multiset rewriting rules, we only introduce a part of syntaxes related to our solution (Fig. 1), other examples of cP syntaxes can be found in previous papers [21, 38, 24, 39]. cP notation supports one-way first-order syntactic unification, which is similar to pattern matching in functional programming languages.

$< simple-term > ::= < atom > | < variable >$
$< compound-term > ::= < functor > (< argument >)$
$< functor > ::= < atom >$
$< argument > ::= < term > ...$
$< rule > ::= < lhs > \rightarrow_{\alpha} < rhs >< promoters >$
$< lhs > ::= < state >< term > ...$
$< rhs > ::= < state >< term > ...$
$< promoters > ::= (| < term-or-eq >)...$

Fig. 1: cP system grammar (lhs = left-hand-side, rhs = right-hand-side)

Simple cP terms consist of atoms and variables. Similar to Prolog, lower-case letters are used to present atoms and upper-case letters are used to declare variables. Ground terms in cP systems include $a(\lambda)$, $b(1)$, $c(11)$, $d(1^3)$, $e(b)$ and $f(b^2c^3)$; terms containing variables include $a(X)$, $b(X1)$, $c(XY)$ and $d(X\_)$. In cP terms, $a(\lambda) \equiv a(0)$, 1 is the unity symbol, $a(1^3) \equiv a(111) \equiv a(3)$ and _ denotes anonymous variable. By using pattern matching, the term $a(X)$ can be matched to $a(\lambda)$, $a(1)$, $a(11)$, $a(1^3)$ or any other terms.

Compound terms in cP systems are like first-order terms in Prolog, that are recursively built from simple terms. Compound terms include $a(b(X))$, $a(b(c)d(Y))$ and $a(b(X)fY)$. When a compound term is not grounded, its variables can be matched to different terms. For example, by matching $a(XY1) = a(111)$, we could get $(X, Y = 1, 1)$, $(X, Y = \lambda, 11)$ or $(X, Y = 11, \lambda)$.

Rules in cP systems are used to describe the rewriting of multisets. By applying a cP rule, the lhs is consumed and the rhs is produced. The α in cP rules represents the application model. There are two application models in cP systems, which are exactly-once (1) and max-parallel (+). Suppose we have three

terms $a(1^2)$, $a(1^3)$, $a(1^3)$ and a rewriting rule $S_1\ a(1X) \rightarrow_1 S_2\ a(X)$, the rule can be unified in different ways to each terms including $S_1\ a(11) \rightarrow_1 S_2\ a(1)$ and $S_1\ a(11^2) \rightarrow_1 S_2\ a(1^2)$. Since the application model here is exactly-once, the rule would be non-deterministically applied to exact one term – $a(1^2)$, $a(1^3)$ or $a(1^3)$, we could finally get a computation result of $a(1)$, $a(1^3)$, $a(1^3)$ or $a(1^2)$, $a(1^2)$, $a(1^3)$. If a similar rule is running in the max-parallel model, which is $S_1\ a(1X) \rightarrow_+ S_2\ a(X)$, the system would apply the rule to $a(1^2)$, $a(1^3)$ and $a(1^3)$ in a maximum parallel manner, the computation result will be $a(1)$, $a(1^2)$, $a(1^2)$.

cP rules can represent mathematic operations and comparisons. For example, $x := y + z \equiv\ y(Y)\ z(Z) \rightarrow_1 x(YZ)$, $x := y - z \equiv \rightarrow_1 x(X) \mid y(XZ)\ z(Z)$ and $x \leq y \equiv x(X)\ y(XY)$. The explanations of these rules are straight forward. The rule $y(Y)\ z(Z) \rightarrow_1 x(YZ)$ means "the value of y is Y, the value of z is Z, the system consumes one copy of term y and one copy of term z to produce one copy of term x, which value is $Y + Z$, written as $x(YZ)$", which is a destructive version of $x = y + z$. The term $x(X)\ y(XY)$ can be explained as "term x contains X, y contains X and some Y, where in cP systems Y is non-negative, so we have $x \leq y$". The subtraction rule has a promoter, which is $\mid y(XZ)\ z(Z)$. To apply a rule with a promoter, the promoter must exist in the system, but will not be consumed during the computation. Following the subtraction rule, nothing would be consumed in the system, since the lhs of the rule is empty; term $x(X)$ will be produced when term $y(XZ)$ and $z(Z)$ exist in the system. This subtraction rule is non-destructive, after producing a copy of term $x(X)$, terms $y(XZ)$ and $z(Z)$ would not be consumed.

3.2 cP system solution to the subset sum problem

By using the multiset rewriting rules, we build our cP system solution to the subset sum problem. The subset sum problem is defined as follow.
$INSTANCE$: a set $S = \{i_1, i_2, ..., i_n\}$ of positive integers and a target integer T.
$QUESTION$: is there a subset $A \subseteq S$ such that $\sum_{x \in A} x = T$?

We solve the subset problem using five rules (Fig. 2), the algorithm is a layer-by-layer search over all subsets of S, finding out if there exists a subset A satisfies $\sum_{x \in A} x = T$. In the cP rules, $m(M)$ denotes the original set S, $t(T)$ denotes the target integer T, $o(O)$ is the final output of the system and $p(P)$ refers to a path of the layer-by-layer search, which stores the used/visited elements $u(U)$, unused/unvisited elements $n(N)$ and the sum of used elements $s(S)$. The terms $\mid m(M)$, $\mid p(u(X)\_s(T))\ t(T)$, $\mid p(u(X)n(m(Y)Z)s(S))$ and $\mid p(\_n(\lambda))$ are promoters.

There are four possible cP states in the system, which is different from states in the state-space of model checkers. The initial cP state of the system is S_0, by applying rule (1) once, it creates a path term from the original set S, which contains $u(\lambda)$, $n(M)$ and $s(\lambda)$ – all elements in S are marked as unused, the sum of the path/subset is 0. After rule (1) being used, the cP state of the system will be changed to S_1, then the rest of rules will be available.

Rule (2), (3), (4) and (5) start in S_1, the cP system would choose them following a weak priority order. Since rule (2) is written before rule (3), (4) and (5), in state S_1, the cP system would always consider rule (2) as the first option.

S_0	$\rightarrow_1$	S_1	$p(u(\lambda)\ n(M)\ s(\lambda)) \mid m(M)$	(1)
S_1	$\rightarrow_1$	S_2	$o(X) \mid p(u(X)\ \_\ s(T))\ t(T)$	(2)
S_1	$\rightarrow_1$	S_3	$\mid p(\_\ n(\lambda))$	(3)
S_1	$\rightarrow_+$	S_1	$p(u(Xm(Y))\ n(Z)\ s(SY)) \mid p(u(X)\ n(m(Y)Z)\ s(S))$	(4)
$S_1\ p(\_)$	$\rightarrow_+$	S_1		(5)

Fig. 2: cP ruleset to solve the subset sum problem

Once rule (2) was tried and the system state is still in S_1, the system would move to rule (3), and so on. In the ruleset, rule (1), (2) and (3) run in exactly-once mode and rule (4) and (5) run in max-parallel mode.

Rule (2) describes a termination of the cP system – if the system could find a path/subset which sum of elements equals to T, the cP system would change its state to S_2 and output the subset elements. In S_2, none of these rules is applicable, so the cP system halts. After checked all subsets of S, if the cP system could not find any subset A satisfies $\sum_{x \in A} x = T$, it would reach another terminating state S_3 after applying rule (3) exactly-once.

In the definition of cP systems, both rule (4) and (5) commit to change the cP state to S_1, thus they can be used in one cP step following the weak priority order. By using rule (4), the system creates new paths by move one element from unused set $u(U)$ to used set $n(N)$ and recompute the sum $s(S)$. The products created by cP rules are not available in the same step – we can consider the products are sent to a "product membrane" temporarily, which will only be activated in the next step.

After rule (4) being used, in the same cP step, rule (5) will clean all path terms except new generated ones. Rule (5) can save a huge amount of memory in the real-life implementation of cP systems, although theoretically cP systems have unlimited workspace and can trade memory to time.

The cP solution solves the subset sum problem in at most $n+2$ steps – when $\sum_{x \in S} x = T$ or $\nexists A, A \subseteq S, \sum_{x \in A} x = T$. By removing rule (5) from the ruleset, we can get a theoretically shorter cP solution with only four rules, but it has a severe memory issue in real-life simulations, which may cause the model checkers cannot finish checking the cP system in a reasonable time. Without rule (5), the system would not delete old path terms after produced new path terms by using rule (4). Since the old path terms are still in the system, they can apply rule (4) again and again and keep generating same path terms until the system halts.

To further improve the state-space for model checking, it is necessary to remove duplicate path terms. In the cP solution, multiple copies of same paths could be generated. For example, when $S = \{1, 2, 3, 4\}$, $T = 10$, three copies of $p(u(m(1)m(2)m(3))\ n(m(4))\ s(6))$ can be generated from three different path terms $p(u(m(1)m(2))\ n(m(3)m(4))\ s(3))$, $p(u(m(1)m(3))\ n(m(2)m(4))\ s(4))$ and $p(u(m(2)m(3))\ n(m(1)m(4))\ s(5))$ by applying rule (4).

Rule (b) in Fig. 3 needs logarithmic time to remove duplicate term copies, which can be used in our solution to help the model checkers getting rid of memory explosion.

$$S_4\ p(X)\ p(X) \rightarrow_+ \quad S_4 \quad p(X) \qquad \text{(b)}$$

Fig. 3: cP ruleset to remove duplicate paths $O(log(N))$

A modified cP solution is shown in Fig. 4. By adding rules (a) (b) and (c) to the solution, we introduce a new cP state S_4, which is used to remove the duplicate term copies. If multiple copies of $p(X)$ exist in the system, the system state would be changed from S_1 to S_4 (rule (a)); then the cP system will keep removing these copies (rule (b)), until there is no duplicate $p(X)$ anymore; then the system state will change back to S_1 (rule (c)) and the rest of rules can be applied.

$$\begin{array}{lll}
S_0 \rightarrow_1 S_1 & p(u(\lambda)\ n(M)\ s(\lambda)) \mid m(M) & (1)\\
S_1 \rightarrow_1 S_4 & \mid p(X)\ p(X) & (a)\\
S_4\ p(X)\ p(X) \rightarrow_+ S_4 & p(X) & (b)\\
S_4 \rightarrow_1 S_1 & & (c)\\
S_1 \rightarrow_1 S_2 & o(X) \mid p(u(X)\ _\ s(T))\ t(T) & (2)\\
S_1 \rightarrow_1 S_3 & \mid p(_\ n(\lambda)) & (3)\\
S_1 \rightarrow_+ S_1 & p(u(Xm(Y))\ n(Z)\ s(SY)) \mid p(u(X)\ n(m(Y)Z)\ s(S)) & (4)\\
S_1\ p(_) \rightarrow_+ S_1 & & (5)
\end{array}$$

Fig. 4: cP solution with duplicate paths removing rules

4 Transformation of cP descriptions into CSP# and B

To verify our cP solution to the subset sum problem and to check if the cP system meets its requirements, we need to model the cP system in formal languages. Instead of manually translate the cP system to the target formal model individually, we propose a set of mapping guidelines from cP rules to CSP# and B, which can be used as a set of translation rules for future formal verification studies in cP systems. Ideally, a translation tool could be developed to automate the transformation process.

4.1 Translating cP systems to CSP#

In CSP#, most of cP system terms need to be modeled into processes. Multisets and sets in cP systems could be modelled as array structures in CSP#. The rewriting can be modelled as expression in events and processes. cP states can be modelled as global variables. cP terms can be modelled as global variables including constants, containers and integers. Promoters could be modelled as conditions. The set of mapping guidelines is shown in Table 2.

cP component	**cP notation**	**CSP# Expression**	**Example**
grounded term	$t(10)$	marco	#define t 10;
variable term	$a(X)$	variable	var a;
multiset, set	$a(1,1,2,3)$	array	var a = [1,1,2,3];
cP state	S_1	global variable	var state = 1;
promoter	$\mid x(X)\ y(X)$	condition	if(x == y)...
rewriting	$\rightarrow_1$ $x(YZ) \mid y(Y)\ z(Z)$	statement	x = y + z;

Table 2: Mapping guidelines for transforming cP systems into CSP#

Following the guidelines, we can translate cP rules in Fig. 2 to CSP# (Fig. 5). In the translation, we use fixed-size array to model cP multisets, an alternative is to use C# lists in the CSP#, which have variable-length and are more flexible than fixed-size arrays. Rule (5) does not need to be translated, because CSP# uses events to simulate cP systems, after an event was performed, its process would automatically move to execute the next event, the previous path terms would not exist in the system anymore. If we need to simulate the cP system without the consuming rule, we can achieve it by generating path term processes in each computation round. The system output is the solution subset of the problem or [-1,-1,-1,-1].

On transforming cP systems to CSP#, integer arrays are used to represent multisets $m(M)$, $u(U)$ and $n(N)$, integer variables are used to model singleton sets $s(S)$ and the cP state. Both state change and rewriting are translated to statements inside of events. One challenge here is to model the nested cP multisets $p(P)$, there is no suitable data structure in CSP# can achieve it. In this case, we implicitly model it by using a group of arrays and variables together. Compare to rule (4), the translations of rule (1), (2) and (3) are quite straight forward. In the second line of the rule (4) translation, the system creates processes by using CSP# notation $[]i : 0..(N-1)@rule4...$, which is a syntax sugar of $P(1)[]P(2)...[]P(N-1)$, where [] is the choice operator. The translation means either $P(1)$ or $P(2)...$ or $P(N-1)$ may execute, which simulates the nondeterministical generation of path terms in the cP system.

Variable declarations #define N 4; var set = [1,2,3,4]; var used[N]; var not_used[N]; var output[N]; var sum; var size; var state = 0;	
cP rules	**CSP# translation**
$S_0 \rightarrow_1 S_1\ p(u(\lambda)\ n(M)\ s(\lambda)) \mid m(M)$	S0() = rule1{not_used = set; sum = 0; size = 0; state = 1;}-> S1_1();
$S_1 \rightarrow_1 S_2\ o(X) \mid p(u(X)\ \_\ s(T))\ t(T)$	S1_1() = check_r2{if(goal){state = 2;}}-> if(state == 2){S2()} S2() = rule2{output=used;} -> Skip;
$S_1 \rightarrow_1 S_3 \mid p(\_\ n(\lambda))$	S1_1() = check_r3{if(!goal && size == N){state = 3;}} -> if(state == 3){S3()} S3() = rule3{output=[-1,-1,-1,-1];} -> Skip;

$S_1 \rightarrow_+ S_1\ p(u(X\ m(Y))\ n(Z)\ s(SY)) \mid p(u(X)\ n(m(Y)\ Z)\ s(S))$	S1_1() = if(state == 1){S1_2()}
	S1_2() = []i:{0..(N-1)}@ rule4{
	state = 1;
	if(not_used[i] != 0){
	used[size] = not_used[i];
	not_used[i] = 0;
	sum = sum + used[size];
	size++;}
	} -> S1_1();

$S_1\ p(\_) \rightarrow_+ S_1$	//consuming rule

Fig. 5 mapping cP subset sum solution to CSP#

4.2 Translating cP systems to B

Compare to CSP#, B language has a rich set of built-in set and sequence operators, which could be used to model cP sub-cells. Our mapping guidelines for translating cP systems to B is shown in Table 3.

cP component	**cP notation**	**B expression**	**Example**
grounded term	$t(10)$	constant	CONSTANT t
variable term	$a(X)$	variable	VARIABLES a
multiset	$a(1,1,2,3)$	sequence	a := [1,1,2,3];
set	$a(1,2,3,4)$	sequence	a := {1,2,3,4};
cP state	S_1	global variable	state := 1;
promoter	$\mid x(X)\ y(X)$	predicate	PRE x = y THEN
rewriting	$\rightarrow_1$ $x(YZ) \mid y(Y)\ z(Z)$	statement	x := y + z;

Table 3: Mapping guidelines for transforming cP systems into B

When modelling cP sets in B, we use set operators. For example, we have two sets $a(X)$ and $b(Y)$ and we want to add them together to get a new set $c(XY)$, we should use the set union operator $c := a \backslash / b$ instead of the summation operator $c := a + b$. Similarly, to compute the difference of cP sets $a(XY)$ and $b(Y)$, we could translate the cP rule to $c := a - b$, where $-$ is the set difference operator. Since sets in B cannot contain duplicate elements, cP terms including $a(1, 1^2, 1^2)$ and $a(b(1)b(1)c(1^2)d(1^3))$ need to be translated to sequences in B.

Following the guidelines, we can translate cP rules in Fig. 2 to B (Fig. 6). Similar to CSP#, the consuming rule can be handled by ProB automatically. In

the translation example, if no solution could be found, the system simply output an [−1] to the environment.

Variable declarations CONSTANTS set,target_num PROPERTIES set = {1,2,3,4} & target_num = 10 VARIABLES state, sum, used, not_used INVARIANT state: 0..3 & used: seq(set) & not_used: POW(set) & sum >= 0	
cP rules	**B translation**
$S_0 \rightarrow_1 S_1\ p(u(\lambda)\ n(M)\ s(\lambda)) \mid m(M)$	rule1 = PRE state = 0 THEN not_used := set; used :=[]; sum := 0; state := 1 END;
$S_1 \rightarrow_1 S_2\ o(X) \mid p(u(X)\ \_\ s(T))\ t(T)$	out2 <-- rule2 = PRE state = 1 & sum = target_num THEN out2:= used; state := 2 END;
$S_1 \rightarrow_1 S_3 \mid p(\_\ n(\lambda))$	out3 <-- rule3 = PRE state = 1 & sum /= target_num & card(not_used) = 0 THEN out3 := [-1]; state := 3 END
$S_1 \rightarrow_+ S_1\ p(u(X\ m(Y))\ n(Z)\ s(SY)) \mid p(u(X)\ n(m(Y)\ Z)\ s(S))$	rule4(x) = PRE state = 1 & x:not_used THEN used := x -> used; not_used := not_used - {x}; sum := sum + x; state := 1 END;
$S_1\ p(\_) \rightarrow_+ S_1$	//consuming rule

Fig. 6 Mapping cP subset sum solution to B

In the translation, B uses sets and sequences to simulate cP multisets and use constants and integer variables to model cP singleton sets. cP outputs are modelled by operation outputs, promoters and inhibitors are translated to predicates. The rewriting and state change are transformed into operation statements. Rule (4), the path generation rule, is different from other rules – to model the non-deterministic generation of path terms, a parameter x is used. ProB will check all possible x values satisfying $x \in n(N)$, which simulates the max-parallel mode of cP systems.

4.3 Limitations of simulating cP systems using CSP# and B

To completely simulate cP systems in model checkers is difficult. As a membrane computing model, all available cP rules can be appiled at same time (in the same cP step), which can create an exponential state-space in a small number of steps. Model checkers work in a different way, they explore the state-space state by state to check system properties. To deal with the memory explosion issue, some model checkers including PAT3 can generate their workspace on the fly, dynamically check the system properties.

Some operations in cP systems are hard to be simulated in CSP# and B. For example, in the destructive summation rule: $S_1\ x(X)y(Y) \rightarrow_1 S_2\ z(Z)$, except translating the main rewriting logic $x + y = z$ to CSP#, we also need to simulate the consumption of x and y. After this rule being applied, x and y will not exist in the system anymore. To release the memory of x and y is an option, which currently cannot be achieved in CSP# and B.

Another challenging of translating cP systems into programming languages is: multiple copies of term $a(X)$ can exist in a cP system, while in programming languages including CSP# and B, the identifier "a", which is used to represent

this term, only can have one value at a time. To declare multiple variables with the same identifier is also not supported.

There is no built-in variable-length array in CSP#, when modelling a cP system, we need to allocate a size for each multiset manually or use external C# lists, which will make the automatic translation from cP systems to PAT3 languages more complex. As mentioned, CSP# does not support generic containers, only integer arrays or integers can be used in modelling cP terms – when modelling nested multisets, we need to create multiple integer arrays and variables to store information and put them together to represent the nested multiset, to track the relationship among these variables could be challenging, which will also make the translation hard to read and easy to make mistakes.

When using B sequences to model cP multiset terms, adding two multisets together can be achieved by iteratively using the sequence functions "front(s)" and "tail(s)", but to compute the difference of two sequences (cP multisets) is complex and inefficient in B. To improve the efficiency on modelling cP multisets in B, a support library with a number of functions need to be implemented.

Although there are some challenges on simulating cP systems in formal languages, most of system properties can be verified by model checkers. Without generating the full state-space at once, model checkers still can check all system states and try to find counterexamples for given specifications.

5 Evaluation

In this section, we explain our cP system solution in detail by using an example and introduce our formal verification results.

5.1 Simulation and translation of the cP solution

Suppose we need to solve a subset sum problem where $S = \{1, 2, 3, 4\}$ and $T = 10$, recall the ruleset in Fig. 2, the cP systems starts in S_0, with a target number $t(10)$ and elements in set S – $m(m(1)m(2)m(3)m(4))$. By applying rule (1) once, we could get a path term $p(u(\lambda)\ n(m(1)m(2)m(3)m(4))\ s(\lambda))$ and the system state would be changed to S_1.

In the next step, the path term $p(u(\lambda)\ n(m(1)m(2)m(3)m(4))\ s(\lambda))$ is available, and the system state is S_1. Rule(2) and (3) cannot be used, because their promoters do not exist in the system. The system will run rule (4) – using $p(u(\lambda)\ n(m(1)m(2)m(3)m(4))\ s(\lambda))$ to create new path terms in a maximum parallel manner without changing the system state. After applying rule (4), in the same step, rule (5) will clean all path terms in the system except new generated ones.

The cP system will keep checking and computing the rules step by step, until a final state – S_2 or S_3 is reached. A manual simulation can be seen in Fig. 7. We mark unavailable terms using grey colour in the table. As mentioned in Section 3, some path variables may have multiple copies.

State	Terms in the cP system	Step	Rule applied
S_0	*t(10), m(m(1)m(2)m(3)m(4)).*	*0*	
S_1	*t(10), m(m(1)m(2)m(3)m(4)), p(u(λ)n(m(1)m(2)m(3)m(4))s(λ)).*	*1*	*(1)*
S_1	*t(10), m(m(1)m(2)m(3)m(4)), p(u(λ)n(m(1)m(2)m(3)m(4))s(λ)),* *p(u(m(1))n(m(2)m(3)m(4))s(1)), p(u(m(2))n(m(1)m(3)m(4))s(2)),* *p(u(m(3))n(m(1)m(2)m(4))s(3)), p(u(m(4))n(m(1)m(2)m(3))s(4)).*	*2*	*(4)*
S_1	*t(10), m(m(1)m(2)m(3)m(4)),* *p(u(m(1))n(m(2)m(3)m(4))s(1)), p(u(m(2))n(m(1)m(3)m(4))s(2)),* *p(u(m(3))n(m(1)m(2)m(4))s(3)), p(u(m(4))n(m(1)m(2)m(3))s(4)).*	*2*	*(5)*
S_1	*t(10), m(m(1)m(2)m(3)m(4)),* *p(u(m(1))n(m(2)m(3)m(4))s(1)), p(u(m(2))n(m(1)m(3)m(4))s(2)),* *p(u(m(3))n(m(1)m(2)m(4))s(3)), p(u(m(4))n(m(1)m(2)m(3))s(4)),* *p(u(m(1)m(2))n(m(3)m(4))s(3)), p(u(m(1)m(3))n(m(2)m(4))s(4)),* *p(u(m(1)m(4))n(m(2)m(3))s(5)), p(u(m(2)m(3))n(m(1)m(4))s(5)),* *p(u(m(2)m(4))n(m(1)m(3))s(6)), p(u(m(3)m(4))n(m(1)m(2))s(7)).*	*3*	*(4)*
S_1	*t(10), m(m(1)m(2)m(3)m(4)),* *p(u(m(1)m(2))n(m(3)m(4))s(3)), p(u(m(1)m(3))n(m(2)m(4))s(4)),* *p(u(m(1)m(4))n(m(2)m(3))s(5)), p(u(m(2)m(3))n(m(1)m(4))s(5)),* *p(u(m(2)m(4))n(m(1)m(3))s(6)), p(u(m(3)m(4))n(m(1)m(2))s(7)).*	*3*	*(5)*
S_1	*t(10), m(m(1)m(2)m(3)m(4)),* *p(u(m(1)m(2))n(m(3)m(4))s(3)), p(u(m(1)m(3))n(m(2)m(4))s(4)),* *p(u(m(1)m(4))n(m(2)m(3))s(5)), p(u(m(2)m(3))n(m(1)m(4))s(5)),* *p(u(m(2)m(4))n(m(1)m(3))s(6)), p(u(m(3)m(4))n(m(1)m(2))s(7)),* *p(u(m(1)m(2)m(3))n(m(4))s(6)), p(u(m(1)m(2)m(4))n(m(3))s(7)),* *p(u(m(1)m(3)m(4))n(m(2))s(8)), p(u(m(2)m(3)m(4))n(m(1))s(9)).*	*4*	*(4)*
S_1	*t(10), m(m(1)m(2)m(3)m(4)),* *p(u(m(1)m(2)m(3))n(m(4))s(6)), p(u(m(1)m(2)m(4))n(m(3))s(7)),* *p(u(m(1)m(3)m(4))n(m(2))s(8)), p(u(m(2)m(3)m(4))n(m(1))s(9)).*	*4*	*(5)*
S_1	*t(10), m(m(1)m(2)m(3)m(4)),* *p(u(m(1)m(2)m(3))n(m(4))s(6)), p(u(m(1)m(2)m(4))n(m(3))s(7)),* *p(u(m(1)m(3)m(4))n(m(2))s(8)), p(u(m(2)m(3)m(4))n(m(1))s(9)),* *p(u(m(1)m(2)m(3)m(4))n(λ)s(10)).*	*5*	*(4)*
S_1	*t(10), m(m(1)m(2)m(3)m(4)), p(u(m(1)m(2)m(3)m(4))n(λ)s(10)).*	*5*	*(5)*
S_2	*t(10), m(m(1)m(2)m(3)m(4)), p(u(m(1)m(2)m(3)m(4))n(λ)s(10)),* *o(m(1)m(2)m(3)m(4)).*	*6*	*(2)*

Fig. 7 A step by step manual simulation of the subset sum cP solution

One solution subset was found in step 6, which is $\{1, 2, 3, 4\}$. If multiple solutions exist, the system state will be changed to S_2 right after the first solution is found, then the cP system will stop, since none of the rules has a lhs with state S_2. If there is no solution satisfy the problem, for instance $T = 11$ and $S = \{1, 2, 3, 4\}$, the system will terminate in state S_3 with empty output (Fig. 8).

State	Terms in the cP system	Step	Rule applied
S_3	*t(10), m(m(1)m(2)m(3)m(4)), p(u(m(1)m(2)m(3)m(4))n(λ)s(10)).*	*6*	*(3)*

Fig. 8 A snapshot of the final state – solution not found

By performing the table simulation, we can observe that in our solution, the worst-case running steps is $n + 2$, when $\sum_{x \in S} x = T$ or $\nexists A, A \subseteq S, \sum_{x \in A} x = T$. To verify this, we modelled the cP system in CSP# and B following the mapping rules defined in Section 4 and checked its features by using PAT3 and ProB.

5.2 Model checking result from PAT3 and B

To check the effectiveness of the cP solution, we chose a medium size problem, where $S = \{1, 2, 4, 55, 56, 57, 119\}$, and checked the cP rules with different T values (Fig. 9). Both model checkers verified that our cP system can find solutions to the subset sum problem. On choosing different heuristics to traverse the state-space, sometimes different solution subsets could be found by the model checkers.

	Expected Answer	PAT3			ProB		
		Goal Found	*Output*	*cP Steps*	*Goal Found*	*Output*	*cP Steps*
T = 294	Y	Y	[119, 57, 56, 55, 4, 2, 1]	9	Y	[57,1,119,56,55,4,2]	9
T = 62	Y	Y	[57, 4, 1]	5	Y	[2,4,56] [1,4,57]	5
T = 7	Y	Y	[4,2,1]	5	Y	[1,2,4]	5
T = 295	N	N	[-1,-1,-1,-1]	9	N	[-1]	9
T = 70	N	N	[-1,-1,-1,-1]	9	N	[-1]	9
T = 8	N	N	[-1,-1,-1,-1]	9	N	[-1]	9

Fig. 9 Verification of the cP solution when n = 7

The feature checking of the cP system is shown in Fig. 10. Features including deadlockfree (safety, weak-liveness), terminating (safety), divergencefree (safety), invariant violation (safety), reachability (liveness) and LTL properties (liveness) are verified. Some of the property checking results depend on S and T. For instance, state S_2 only can be reached when the answer to the subset sum problem is "yes", meanwhile, S_3 could not be reached. From the results we can find that the general running time of PAT3 is less than ProB, especially when the problem size is large. Deadlock-freeness and divergence-freeness checking are often slower than checking other properties, since more states and transitions need to be checked. The subset sum problem is NP-complete, the state-space grows significantly with the increase of the problem size. ProB's state-space contains 18743 states when $n = 7$ and 13492904 states when $n = 10$.

When checking deadlocks in ProB, "pseudo deadlocks" need to be distinguished from deadlocks. Any state which has no outgoing edge is a sink, ProB would treat a sink as a deadlock. In the cP solution, after applied rule (2) or rule (3), the system's cP state would be changed to S_2 or S_3, which may cause sinks. Since no rule can be used in S_2 or S_3, ProB would treat them as deadlock states. In our experiments, sinks in cP systems are not counted as deadlocks.

Checking the cP solution with a large-size problem is time consuming. To check all features, when $n = 10$, the running time of PAT3 is around $2300s$ and ProB is more that $3000s$. It is still acceptable, but it is better to make some changes in the cP rules to speed up the model checking. In this example, we could add duplicate path terms removing rules to the solution following Fig. 4 – although it is probably unnecessary in a theoretical cP system, it could be very helpful in practice. We re-modelled the system using C# set in PAT3 and set

Subset Sum Problem	*n = 4* *S = {1,2,3,4}* *T = 10*				*n = 7* *S = {1,2,4,55,56,57,119}* *T = 295*				*n = 10* *S = {1,2,3,4,55,56,57,119,235,244}* *T = 777*			
Expected Answer	**Yes**				**No**				**No**			
Feature	**PAT3**		**ProB**		**PAT3**		**ProB**		**PAT3**		**ProB**	
	True/ False	Time (s)	True/ False	Time (s)	True/ False	Time (s)	True/ False	Time (s)	True/ False	Time (s)	True/ False	Time (s)
Goal reaches	T	0.001 0.001 0.001	T	< 0.5 < 0.5 < 0.5	F	0.420 0.429 0.408	F	5 4 4	F	568.2 561.4 587.5	F	3090 3202 3080
Deadlock-freeness	T	0.002 0.002 0.003	T		T	0.410 0.422 0.408	T		T	480.3 452.6 474.1	T	
Invariant violation (ProB)	-		F		-		F		-		F	
New errors (ProB)			F				F				F	
Terminating (PAT3)	T	0.002 0.002 0.001	-		T	0.002 0.002 0.002	-		T	0.003 0.003 0.003	-	
Divergence-freeness (PAT3)	T	0.003 0.002 0.003			T	0.652 0.662 0.713			T	851.4 861.9 862.5		
Non-deterministic (PAT3)	T	0.001 0.001 0.001			T	0.002 0.002 0.002			T	0.002 0.002 0.003		
S_2 reached	T	0.002 0.001 0.002	T	-	F	0.401 0.422 0.417	F	-	F	499.8 488.7 491.1	F	-
S_3 reached	F	0.001 0.001 0.001	F		T	0.001 0.001 0.001	T		T	0.005 0.005 0.004	T	

Fig. 10 Model checking results – cP solution without removing duplicate terms

in ProB to remove duplicate path terms. By doing so, the state-space of $n = 7$ decreased to 132 and $n = 10$ decreased to 1028 in ProB. The checking results are shown in Fig. 11, which we obtained same property checking results of the system with less running time.

Subset Sum Problem	***n = 4*** ***S = {1,2,3,4}*** ***T = 10***				***n = 7*** ***S = {1,2,4,55,56,57,119}*** ***T = 295***				***n = 10*** ***S = {1,2,3,4,55,56,57,119,235,244}*** ***T = 777***			
Expected answer to the problem	**Yes**				**No**				**No**			
Feature	**PAT3**		**ProB**		**PAT3**		**ProB**		**PAT3**		**ProB**	
	True/ False	Time (s)	True/ False	Time (s)	True/ False	Time (s)	True/ False	Time (s)	True/ False	Time (s)	True/ False	Time (s)
Goal reaches	T	0.001 0.001 0.001	T	< 0.5 < 0.5 < 0.5	F	0.027 0.025 0.028	F	< 0.5 < 0.5 < 0.5	T	0.691 0.703 0.685	T	< 0.5 < 0.5 < 0.5
Deadlock-freeness	T	0.002 0.002 0.002	T		T	0.029 0.029 0.028	T		T	0.489 0.491 0.485	T	
Invariant violation (ProB)	-		F		-		F		-		F	
New errors (ProB)			F				F				F	
Terminating (PAT3)	T	0.001 0.001 0.001	-		T	0.002 0.002 0.001	-		T	0.002 0.002 0.002	-	
Divergence-freeness (PAT3)	T	0.003 0.003 0.003			T	0.053 0.047 0.052			T	0.850 0.846 0.839		
Non-deterministic (PAT3)	T	0.001 0.001 0.001			T	0.002 0.002 0.002			T	0.003 0.002 0.002		
S_2 reached	T	0.001 0.001 0.001	T	-	F	0.025 0.028 0.025	F	-	F	0.434 0.428 0.433	F	-
S_3 reached	F	0.001 0.001 0.001	F		T	0.001 0.001 0.002	T		T	0.004 0.003 0.004	T	

Fig. 11 Model checking results – cP solution with rules to remove duplicate terms

5.3 Discussion

We showed that it is possible to verify a cP system's features effectively by using model checking. In addition to feature checking, model checking also can help finding design errors in cP rules. Considering the rules in Fig. 12 which has a critical error, after adding an element $m(Y)$ from the unused term $n(m(Y)Z)$ to the used term $u(Xm(Y))$, the system does not remove $m(Y)$ from unused term $n(m(Y)Z)$, which means one element in S can be used multiple times when constructing a subset. By model this ruleset in the model checkers with the problem that $S = \{1, 2, 3, 4\}$ and $T = 16$, both PAT3 and ProB can easily find a solution subset $\{4, 4, 4, 4\}$, which violates the definition of the subset sum problem.

$$
\begin{array}{llll}
S_0 & \to_1 & S_1 & p(u(\lambda)\ n(M)\ s(\lambda)) \mid m(M) \quad (1)\\
S_1 & \to_1 & S_2 & o(X) \mid p(u(X)\ \_\ s(T))\ t(T) \quad (2)\\
S_1 & \to_1 & S_3 & \mid p(\_\ n(\lambda)) \quad (3)\\
S_1 & \to_+ & S_1 & p(u(Xm(Y))\ n(m(Y)Z)\ s(SY)) \mid p(u(X)\ n(m(Y)Z)\ s(S)) \quad (4)\\
S_1\ p(\_) & \to_+ & S_1 & \quad (5)
\end{array}
$$

Fig. 12: cP ruleset with a design error

PAT3 with CSP# does not have an actual state-space, since it is event-based. To visualize the simulation, PAT3 can display an event transition diagram. Compare to PAT3, ProB can show its state-space after checking a system model. In a cP system, if its state-space can be controlled well by designing cP rules or choosing a suitable problem size, ProB is a great option to perform model checking, since its visualization tool is especially useful in finding algorithm errors of the cP rules. If the problem size is large or the communication of cP top-cells is involved, PAT3 could be a better choice, since its performance is better and CSP# is designed for modelling distributed, concurrent systems. For verifying async cP systems, we would also recommend PAT3, which supports real-time system modelling. Finally, to design and develop a cP system verifier, PAT3 and ProB can be combined in a back-end, a user interface can be implemented on top of it, which accepts cP system models as inputs. Some example cP verification results of PAT3 and ProB can be found in the appendix.

6 Conclusion

Since cP systems become increasingly important on solving computational hard problems and distribute problems, it is meaningful to formally verify cP systems. In this study, we solved the subset sum problem in cP systems and verified the solution by using model checking tools PAT3 and ProB. Our cP solution only contains one top-cell with five rules, which can solve the problem in at most $n + 2$ steps. We checked safety and liveness properties of the system, illustrated how to find out design errors in cP systems by using model checking. By verified different cP rulesets to the same problem, we demonstrated it is possible to improve model checking efficiency with additional cP rules. This work will have a significant impact on formal verification of cP systems and can assist the practice use of cP solutions.

As the first formal verification work in cP systems, we proposed two mapping guidelines to transform cP systems to verification problems in CSP# and B. Our future cP system formal verification work includes three main directions: 1) to formally verify multi-cell cP systems which solve sync/async distributed computational hard problems; 2) to implement cP systems in theorem provers and to prove cP theorem and lemmas; 3) to design a formal programming language for cP systems, which supports automatically translation from cP language to different verification languages, thus all the cP models can be checked automatically.

References

1. G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108 – 143, 2000.
2. A. Păun, "On P systems with active membranes," in *Unconventional Models of Computation, UMC2K*, pp. 187–201, Springer, 2001.

3. C. Martín-Vide, G. Păun, J. Pazos, and A. Rodríguez-Patón, "Tissue P systems," *Theoretical Computer Science*, vol. 296, no. 2, pp. 295–326, 2003.
4. M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta informaticae*, vol. 71, no. 2, 3, pp. 279–308, 2006.
5. M. Gheorgue, F. Ipate, C. Dragomir, L. Mierla, L. Valencia Cabrera, M. García Quismondo, and M. d. J. Pérez Jiménez, "Kernel P systems-version 1," *Proceedings of the Eleventh Brainstorming Week on Membrane Computing, 97-124. Sevilla, ETS de Ingeniería Informática, 4-8 de Febrero, 2013,*, 2013.
6. R. Nicolescu, F. Ipate, and H. Wu, "Programming P systems with complex objects," in *International Conference on Membrane Computing*, pp. 280–300, Springer, 2013.
7. G. Păun, "Introduction to membrane computing," in *Applications of Membrane Computing*, pp. 1–42, Springer, 2006.
8. J. Aguado, T. Balanescu, T. Cowling, M. Gheorghe, M. Holcombe, and F. Ipate, "P systems with replicated rewriting and stream X-machines (Eilenberg machines)," *Fundamenta Informaticae*, vol. 49, no. 1-3, pp. 17–33, 2002.
9. Z. Dang, O. H. Ibarra, C. Li, and G. Xie, "On the decidability of model-checking for P systems," *Journal of Automata, Languages and Combinatorics*, vol. 11, no. 3, pp. 279–298, 2006.
10. F. Ipate, R. Lefticaru, and C. Tudose, "Formal verification of P systems using Spin," *International Journal of Foundations of Computer Science*, vol. 22, no. 01, pp. 133–142, 2011.
11. R. Lefticaru, C. Tudose, and F. Ipate, "Towards automated verification of P systems using Spin," *International Journal of Natural Computing Research (IJNCR)*, vol. 2, no. 3, pp. 1–12, 2011.
12. M. Gheorghe, F. Ipate, R. Lefticaru, M. J. Pérez-Jiménez, A. Ţurcanu, L. Valencia Cabrera, M. García-Quismondo, and L. Mierlă, "3-col problem modelling using simple kernel P systems," *International Journal of Computer Mathematics*, vol. 90, no. 4, pp. 816–830, 2013.
13. M. Gheorghe, R. Ceterchi, F. Ipate, S. Konur, and R. Lefticaru, "Kernel P systems: from modelling to verification and testing," *Theoretical Computer Science*, vol. 724, pp. 45–60, 2018.
14. F. Ipate, M. Gheorghe, and R. Lefticaru, "Test generation from P systems using model checking," *The Journal of Logic and Algebraic Programming*, vol. 79, no. 6, pp. 350–362, 2010.
15. M. Gheorghe, S. Konur, and F. Ipate, "Kernel P systems and stochastic P systems for modelling and formal verification of genetic logic gates," in *Advances in Unconventional Computing*, pp. 661–675, Springer, 2017.
16. F. Ipate and A. Turcanu, "Modeling, verification and testing of P systems using Rodin and ProB," *Proceedings of the Ninth Brainstorming Week on Membrane Computing, 209-219. Sevilla, ETS de Ingeniería Informática, 31 de enero-4 de febrero, 2011*, 2011.
17. B. Aman and G. Ciobanu, "Modelling and verification of weighted spiking neural systems," *Theoretical Computer Science*, vol. 623, pp. 92–102, 2016.
18. M. A. Martínez-del Amor, I. Pérez-Hurtado, M. J. Pérez-Jiménez, A. Riscos-Núñez, and F. Sancho-Caparrini, "A simulation algorithm for multienvironment probabilistic P systems: A formal verification," *International Journal of Foundations of Computer Science*, vol. 22, no. 01, pp. 107–118, 2011.
19. B. Aman and G. Ciobanu, "Verification of membrane systems with delays via Petri nets with delays," *Theoretical Computer Science*, vol. 598, pp. 87–101, 2015.

20. W. Yuan, G. Zhang, M. J. Pérez-Jiménez, T. Wang, and Z. Huang, "P systems based computing polynomials: design and formal verification," *Natural Computing*, vol. 15, no. 4, pp. 591–596, 2016.
21. J. Cooper and R. Nicolescu, "The Hamiltonian cycle and travelling salesman problems in cP systems," *Fundamenta Informaticae*, vol. 164, no. 2-3, pp. 157–180, 2019.
22. Y. Liu, J. Sun, and J. S. Dong, "PAT 3: An extensible architecture for building multi-domain model checkers," in *2011 IEEE 22nd International Symposium on Software Reliability Engineering*, pp. 190–199, IEEE, 2011.
23. M. Leuschel and M. Butler, "Prob: A model checker for B," in *International Symposium of Formal Methods Europe*, pp. 855–874, Springer, 2003.
24. A. Henderson and R. Nicolescu, "Actor-like cP systems," in *International Conference on Membrane Computing*, pp. 160–187, Springer, 2018.
25. M. J. P. Jiménez and A. R. Núñez, "Solving the subset-sum problem by P systems with active membranes," *New Generation Computing*, vol. 23, no. 4, pp. 339–356, 2005.
26. M. J. Pérez-Jiménez and A. Riscos-Núnez, "A linear-time solution to the knapsack problem using P systems with active membranes," in *International Workshop on Membrane Computing*, pp. 250–268, Springer, 2003.
27. M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and F. J. Romero-Campero, "A linear solution of subset sum problem by using membrane creation," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pp. 258–267, Springer, 2005.
28. C. Graciani-Díaz and A. Riscos-Núñez, "Looking for simple common schemes to design recognizer P systems with active membranes that solve numerical decision problems," in *International Conference on Unconventional Computation*, pp. 94–104, Springer, 2005.
29. M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A. Riscos-Núñez, and F. J. Romero-Campero, "On the power of dissolution in P systems with active membranes," in *International Workshop on Membrane Computing*, pp. 224–240, Springer, 2005.
30. D. Díaz-Pernil, M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "A logarithmic bound for solving subset sum with P systems," in *International Workshop on Membrane Computing*, pp. 257–270, Springer, 2007.
31. A. Leporati, C. Zandron, C. Ferretti, and G. Mauri, "On the computational power of spiking neural P systems," *Proceedings of the Fifth Brainstorming Week on Membrane Computing, 227-245. Sevilla, ETS de Ingeniería Informática, 29 de Enero-2 de Febrero, 2007*, 2007.
32. A. Leporati, C. Zandron, C. Ferretti, and G. Mauri, "Solving numerical NP-complete problems with spiking neural P systems," in *International Workshop on Membrane Computing*, pp. 336–352, Springer, 2007.
33. D. Díaz-Pernil, M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "Solving subset sum in linear time by using tissue P systems with cell division," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pp. 170–179, Springer, 2007.
34. J. Sun, Y. Liu, and J. S. Dong, "Model checking CSP revisited: Introducing a process analysis toolkit," in *International symposium on leveraging applications of formal methods, verification and validation*, pp. 307–322, Springer, 2008.
35. J. Sun, Y. Liu, A. Roychoudhury, S. Liu, and J. S. Dong, "Fair model checking with process counter abstraction," in *International Symposium on Formal Methods*, pp. 123–139, Springer, 2009.

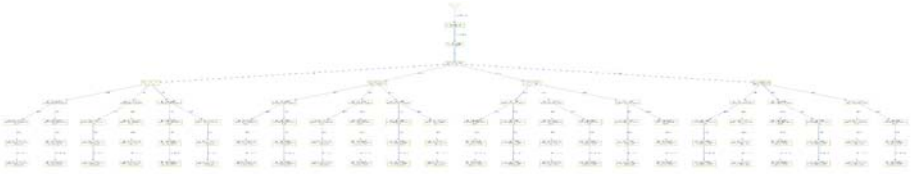
36. J. Sun, Y. Liu, J. S. Dong, and J. Pang, "PAT: Towards flexible verification under fairness," in *International Conference on Computer Aided Verification*, pp. 709–714, Springer, 2009.
37. M. Leuschel and M. Butler, "ProB: an automated analysis toolset for the B method," *International Journal on Software Tools for Technology Transfer*, vol. 10, no. 2, pp. 185–203, 2008.
38. R. Nicolescu and A. Henderson, "An introduction to cP systems," in *Enjoying Natural Computing*, pp. 204–227, Springer, 2018.
39. R. Nicolescu, "Most common words–a cP systems solution," in *International Conference on Membrane Computing*, pp. 214–229, Springer, 2017.

A Verification results of the model checking tools

A.1 An example of PAT3 event transition diagram

PAT3 simulator tracks all transitions, all the events including if-else condition, rule checking and state checking are tracked. PAT3 does not visualize the search state-space of the cP solution to the subset sum problem.

A.2 A state-space comparison between removing and do not removing duplicate terms

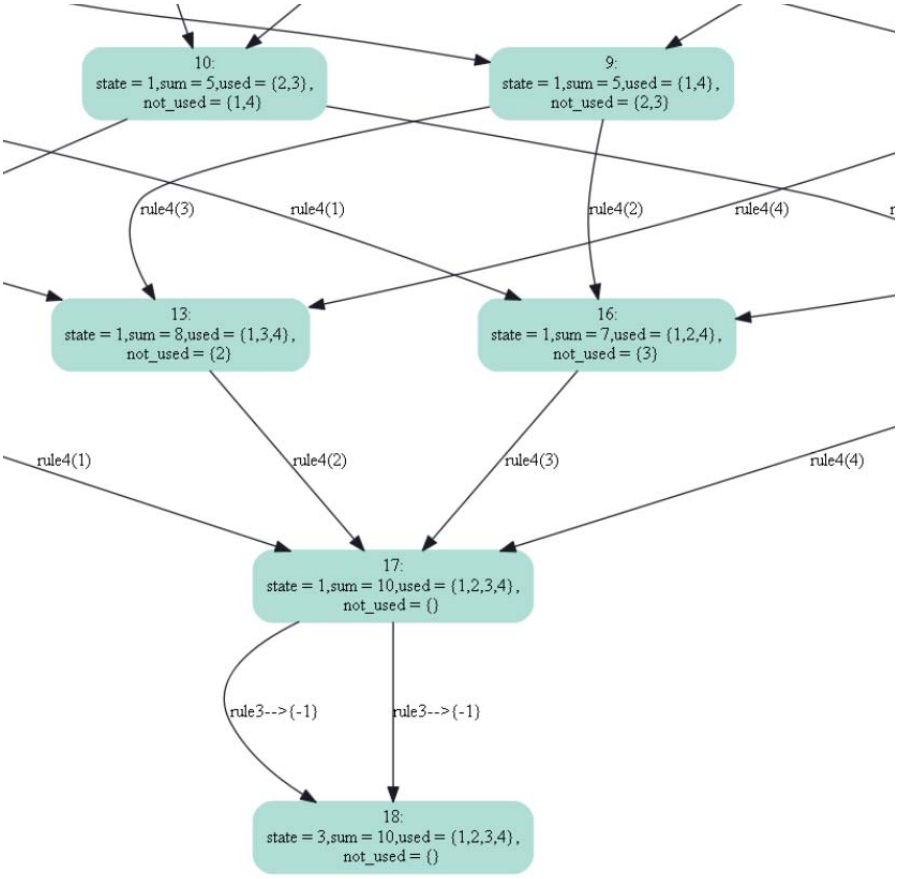


State-space in ProB: S = {1,2,3,4}, T = 11, without removing duplicate terms, 90 states in total

State-space in ProB: S = {1,2,3,4}, T = 11, after removed duplicate terms, 20 states in total

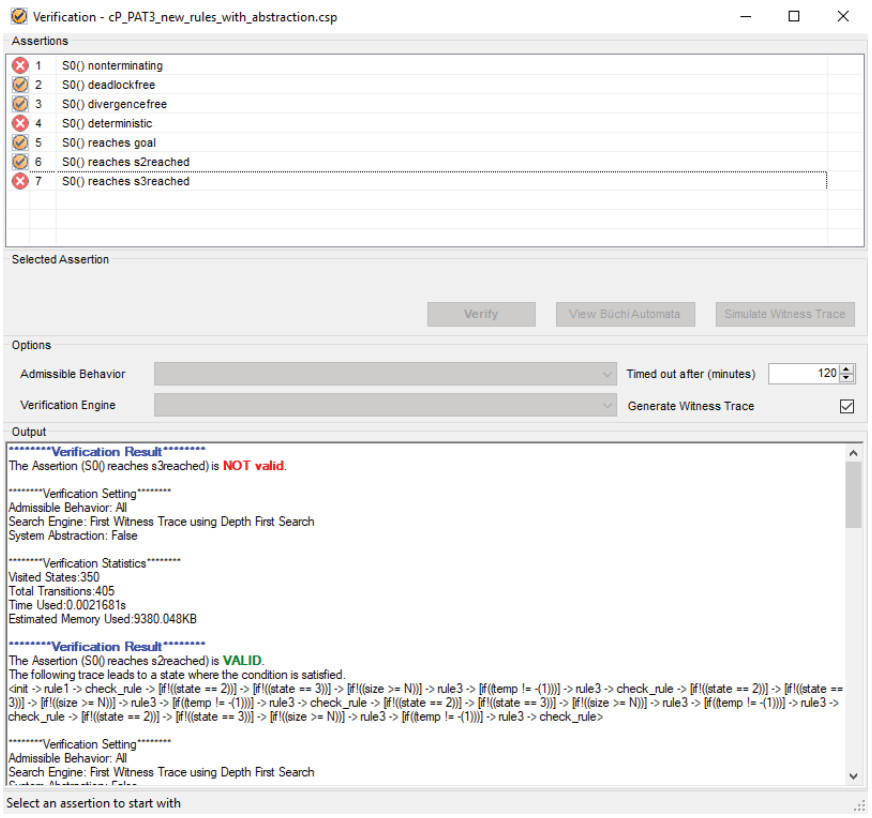
In the cP solution, adding cP rules to remove duplicate path terms can significantly reduce the model checking state-space. In this example, $n = 4$, by removing duplicate path term copies, the number of states in the ProB state-space decreased from 90 to 20.

A.3 A closer view of ProB state-space



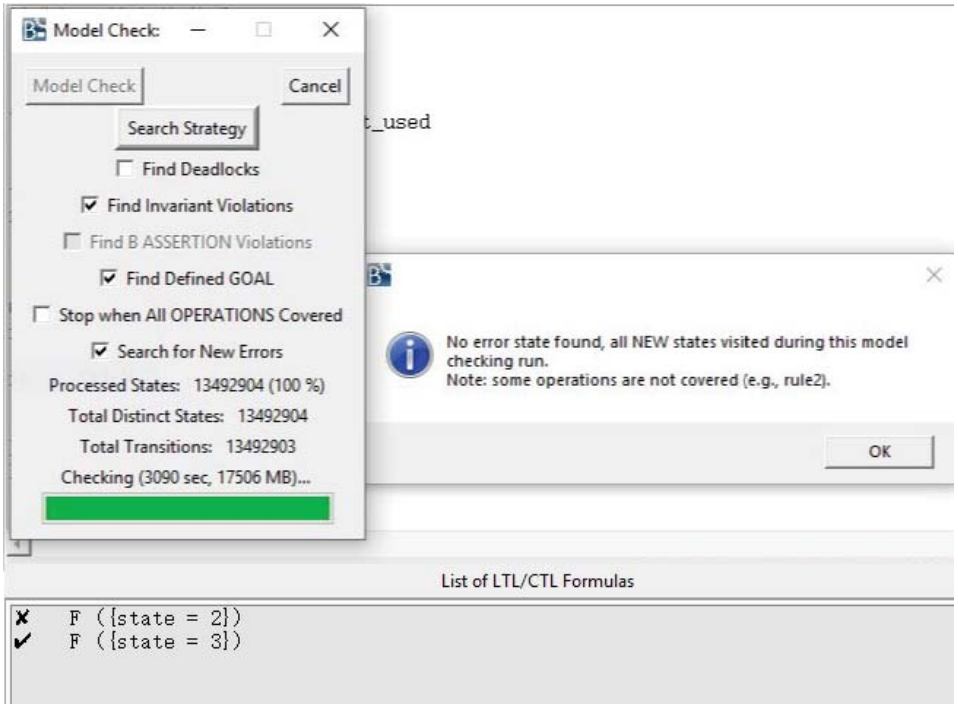
By visualizing the state-space in ProB, all the state variables and operations can be tracked, which is useful on checking algorithm and design errors in cP systems.

A.4 An example of PAT3 model checking result



Here is an example of the PAT3 verification result, when $S = \{1, 2, 3, 4\}$, $T = 10$. Detailed information including trace, searching heuristics, visited states, running time and memory used could be displayed.

A.5 An example of ProB model checking result



Here is an example of ProB model checking result, when $n = 10$ and the answer to the subset sum problem is "no". Compare to PAT3, less build-in features can be checked in ProB, but LTL/CTL features could be used to describe custom features.

Notes on Improved Normal Forms of Spiking Neural P Systems and Variants

Ivan Cedric H. Macababayao[1], Francis George C. Cabarle[1,2⋆], Ren Tristan A. de la Cruz[1], Henry N. Adorna[1], Xiangxiang Zeng[3]

[1]Algorithms & Complexity
Dept. of Computer Science, University of the Philippines Diliman
Diliman 1101 Quezon City, Philippines.
[2]Shenzhen Research Institute of Xiamen University
Xiamen University, Shenzhen 518000, Guangdong, China.
[3]School of Information Science and Engineering
Hunan university 410082, Changsha, China.

Abstract. Spiking Neural P Systems (SNP) are membrane computing systems that abstracts the function and communications between biological neurons. One variant of SNP, called Spiking Neural P Systems with Structural Plasticity (SNPSP) add the concept of dynamic synapses, that is, dynamic connections between the neurons. For both SNP and SNPSP, synapses (or edges) are used for sending and receiving spikes between neurons, while rules govern the sending of these spikes, and in the case of SNPSP, the creation and/or deletion of synapses. This paper investigates on normal forms for SNP and SNPSP, by adding and modifying restrictions set in both systems. In particular, this paper (1) fixes a programming bug in a previous SNP normal form, (2) reduces the types of regular expressions in both SNP and SNPSP normal forms, and (3) reduces the number of rules per neuron in the SNPSP normal form.

Key words: Membrane computing, Spiking neural P systems, Structural plasticity

1 Introduction

Membrane computing is a branch of natural computing that is inspired by the massive parallelism of living cells [11,1]. The general goal of membrane computing is to design computing models that abstract concepts from biology – from cells or groups of cells, to more complex organs like the brain. These computing models are called P systems, and have three main types, namely: (1) cell-like P systems, (2) tissue-like P systems, and (3) neural-like P systems. Cell-like P systems take advantage of the hierarchical structure of biological cells, tissue-like P systems take advantage of the way cells communicate and connect to each other (usually represented as nodes in graphs), while neural P systems take advantage

⋆ corresponding author fccabarle@up.edu.ph

of the structure and functions of brain cells. This paper will focus on a specific kind of neural P system, which is the Spiking Neural P System, and its variant Spiking Neural P Systems with Structural Plasticity.

Spiking Neural P Systems (SNP for short) were first introduced in [6]. SNP abstract neurons. Specifically, SNP imitate the communication process between neurons, in which instructions or information are being derived by sending and receiving *spikes* between each other through the use of *synapses*. This paper will also work with a variant of SNP, which is the Spiking Neural P System with Structural Plasticity (SNPSP, for short), first introduced in [4], which introduces synapse creation and deletion as opposed to static synapses.

SNPSP work in almost the same way as SNP, with one notable difference: SNPSP allow *synaptogenesis* and *synapse deletion*, which means that neurons are able to create and/or delete synapses by themselves. This added feature opens up a lot of possibilities – most of which are still unexplored. Other variants of SNP with (implied) dynamism for synapses or edges between neurons exist, e.g. the generalised eSNP systems in [2], SNP that can create new neurons in [9], and more recently adding schedules in synapses in [3]. Works on SNPSP, along with [3], are focused in computations that explicitly involve dynamism with synapses.

Both SNP and SNPSP systems have been shown to be Turing complete. In addition, there is an active line of research that involves proving that SNP are still complete even with additional restrictions that seek to simplify the system. These are called *normal forms*. Results with regards to SNP normal forms include [5], [13], and [8]. Examples of restrictions in SNP are not using delays and/or forgetting rules, reducing the number of rules per neuron, and many others. Normal forms are also being investigated with respect to SNPSP, with largely the same goal. Examples of SNPSP normal forms can be found in [12], [7].

This work improves on results presented in [8], as well as answer an open problem formulated there. Specifically, this work investigates on whether the number of regular expressions in an SNP normal form be reduced from three, while keeping the other restrictions in the original normal form (e.g. each neuron can have a maximum of two rules). Note that in [8] the authors state that they believe it is not the case that there is a universal SNP, without forgetting rules and delays, that uses less than three regular expressions. This work also improves on the results in [7] by adding more restrictions and making some of its restrictions even stricter. In particular, we try to limit the SNPSP to only one rule per neuron, and also restrict the number of regular expressions that can be used. This is done without violating the prior restrictions introduced in [7].

The approach to these open problems will be by simulating register machines using SNP and SNPSP systems that follow the restrictions stated above. In particular, we provide modules of SNPSP configurations to simulate the ADD, SUB, and HALT instructions of register machines.

In sections 2 we briefly introduce the concepts of SNP and its variant SNPSP. In section 3 we show updates and improvements on the normal forms for both

SNP and SNPSP. This section shows the main results in this paper. In section 4 we give the final remarks.

2 SNP and SNPSP

2.1 SNP

Spiking neural P systems (SNP) [6] have the following construct:

$$\Pi = (O, \sigma_1, ..., \sigma_m, syn, in, out)$$

where:

1. $O = \{a\}$ is the singleton alphabet, and a is called the spike
2. $\sigma_1, ..., \sigma_m$ are neurons of the form $\sigma_i = (n_i, R_i)$ with $1 \leq i \leq m$, where
   (a) n_i indicates the initial number of spikes in σ_i
   (b) R_i is the finite set of rules of σ_i, having the form: $E/a^c \rightarrow a$, where E is a regular expression over O, $c \geq 1$
3. $syn \subseteq \{1, .., m\} \times \{1, .., m\}$ with $(i, i) \notin syn$ for $1 \leq i \leq m$, is the set of synapses between neurons
4. $in, out \in \{1, .., m\}$ indicate the input and output neurons

2.2 SNPSP

An SNPSP is a P system with the following construct [4]:

$$\Pi = (O, \sigma_1, ..., \sigma_m, syn, in, out)$$

where:

1. $O = \{a\}$ is the singleton alphabet, and a is called the spike
2. $\sigma_1, ..., \sigma_m$ are neurons of the form $\sigma_i = (n_i, R_i)$ with $1 \leq i \leq m$, where
   (a) n_i indicates the initial number of spikes in σ_i
   (b) R_i is the finite set of rules of σ_i, and each rule can have either of the following forms:
      i. Spiking Rule: $E/a^c \rightarrow a$, where E is a regular expression over O, $c \geq 1$
      ii. Plasticity Rule: $E/a^c \rightarrow \alpha(\beta, K, N_j)$, where E is a regular expression over O, $c \geq 1$, $\alpha \in \{+, -, \pm, \mp\}$, $\beta \in \{a, \lambda\}$, $K \geq 0$, $N_j \subseteq \{1, ..., m\}$
3. $syn \subseteq \{1, .., m\} \times \{1, .., m\}$ with $(i, i) \notin syn$ for $1 \leq i \leq m$, is the set of synapses between neurons
4. $in, out \in \{1, .., m\}$ indicate the input and output neurons

For any neuron σ_i, we denote the set of neurons which has σ_i as their presynaptic neuron as $pres(i)$, so that $pres(i) = \{\sigma_j | (i, j) \in syn\}$. Similarly, we denote the set of neurons which has σ_i as their postsynaptic neuron as $pos(i)$, so that $pos(i) = \{\sigma_j | (j, i) \in syn\}$.

Spiking Rules, having the form $E/a^c \rightarrow a$, are applied as follows: If the neuron σ_i contains k spikes and $a^k \in L(E)$ and $k \geq c$, then the rule $E/a^c \rightarrow a$ is applied. This means that the neuron σ_i *fires*, consuming c spikes from σ_i such that only $k-c$ spikes remain, while also emitting (sending) c spikes to all of σ_i's presynaptic neurons. For cases where the rule is $E/a^c \rightarrow a$ and $E = a^c$, then the rules is usually simplified to the form $a^c \rightarrow a$ (For example, $a^2/a^2 \rightarrow a$ is simplified to $a^2 \rightarrow a$).

On the other hand, Plasticity Rules, of the form $E/a^c \rightarrow \alpha(\beta, K, N_j)$ are applied as follows: If the neuron σ_i contains k spikes and $a^k \in L(E)$ and $k \geq c$, then the rule $E/a^c \rightarrow \alpha(\beta, K, N_j)$ is applied. The set N_j is a collection of neurons that σ_i can either connect to or disconnect from, depending on the configuration of the rule. Like spiking rules, plasticity rules will also consume c spikes such that only $k-c$ spikes remain. At the same time, one of the following cases will apply, depending on the values of α:

1. $\alpha = +$: neuron σ_i creates synapses to at most K neurons from N_j. If $|N_j - pres(i)| \leq K$, deterministically create a synapse to all available neurons in $N_j - pres(i)$. If $|N_j - pres(i)| > K$, then nondeterministically select K neurons from $N_j - pres(i)$ and create synapses to these selected neurons.
2. $\alpha = -$: neuron σ_i deletes synapses from at most K neurons from N_j. If $|pres(i)| \leq K$, deterministically delete synapses from all available neurons in $pres(i)$. If $|pres(i)| > K$, then nondeterministically select K neurons from $pres(i)$ and delete the synapses from these selected neurons.
3. $\alpha = \pm$: at time t, neuron σ_i first attempts to create synapses to at most K neurons from N_j, following the procedure for when $\alpha = +$. Then, at time $t+1$, neuron σ_i deletes K synapses.
4. $\alpha = \mp$: at time t, neuron σ_i first attempts to delete synapses from at most K neurons from N_j, following the procedure for when $\alpha = -$. Then, at time $t+1$, neuron σ_i creates K synapses.

Note that for $\alpha \in \{\pm, \mp\}$, only the priority between creation or deletion is changed, but the application is similar to $\alpha \in \{+, -\}$. Hence, synapses that were created (or deleted) at time t will not necessarily be deleted (or created) at time $t+1$.

The result of a computation is denoted to be the difference between the first two firings of the neuron σ_{out}. This difference, $t_2 - t_1$, is said to be the number that is computed by Π. The set of all numbers computed by Π using this method is denoted as $N_2(\Pi)$.

Whenever a neuron creates a synapse (when $\alpha \in \{+, \pm, \mp\}$), an embedded spike is always sent to the receiver of the synapse. This embedded spike is sent at the time the synapse is created. Thus, when a plasticity rule with $\alpha \in \{+, \pm, \mp\}$ is applied, a destination neuron receives one spike at the step when a synapse is created.

Both SNP and SNPSP are largely similar in syntax and function. Notable differences are: (1) SNPSP has the capability to create and delete synapses, and (2) SNPSP does not allow the use of delays and forgetting rules.

3 Updates on Normal Forms of SNP and SNPSP

An improved normal form for SNP was given in [8], which restricted the SNP to a maximum of two rules per neuron and used at most three regular expressions $a(aa)^*$, $a(aaa)^*$, and $a^2 \cup a$.

An improved normal form for SNPSP was given in [7]. This normal form contained the following restrictions: (1) only $a \in \{\pm\}$ is used for all plasticity rules, (2) only plasticity rules are used, (3) there are no synapses both in the initial and final configuration of the SNPSP. This normal form used a maximum of three rules per neuron.

In this section we do two things. First we update the improved SNP normal form from [8] to fix an interference issue in its SUB module. Next we further improve the SNPSP normal form from [7], by adding the following additional restrictions: (1) all neurons will only have one rule each, and (2) it only uses three types of regular expressions.

We construct SNP and SNPSP systems that simulate a register machine. A register machine is a construct $M = (m, I, l_0, l_h, R)$, where m is the number of registers, I is the set of instruction labels, l_0 is the start label, l_h is the halt label, and R is the set of instructions, where every label in I only labels one instruction in R. The instructions are ADD, SUB, and $HALT$, where:

1. $l_i : (ADD(r), l_j, l_k)$: the value in register r is increased by 1, then nondeterministically go to either instruction l_j or l_k
2. $l_i : (SUB(r), l_j, l_k)$: if the value in register r is nonzero, then subtract 1 from r and go to instruction l_j, otherwise do not modify the value in r then go to instruction l_k
3. $l_h : HALT$: the halt instruction

It is possible for ADD and SUB modules to modify the value of the same register. It is also possible for two or more ADD's or two or more SUB's to modify the value of a single register. However, note that the neuron 1 was never involved in any SUB module [6]. Hence the register associated with the HALT module will not contain the rule of neuron r from the SUB module.

3.1 SNP Improved Normal Form from [8]

The results in [8] showed that there exists a normal form for SNP where:

1. there is a maximum of two rules per neuron, and
2. the system only uses at most three kinds of regular expressions in its rules.

We will be using these results later in showing a similar normal form for SNPSP.

There is, however, an interference issue in the SUB module from [8]. In particular, we look at the case when at least two SUB modules are operating at some point (not necessarily at the same time) on the same register r. Looking at Figure 6 (see appendix) will show the following. Suppose that there are at least two SUB modules that are connected to the neuron r, and that the neuron r

initially contains more than two spikes (and hence, the subtraction will succeed). Then assume that the neuron l_i receives a single spike at time t. Notice that at time $t+2$, not only will the neuron r send a spike to the neuron l_{i_3} belonging to the current SUB module being simulated, but it will also send spikes to all other neurons l_{i_3} belonging to the other SUB modules that are not being simulated at the moment. This means that at time $t+5$, at least one module will fire "spontaneously" due to the stray spike.

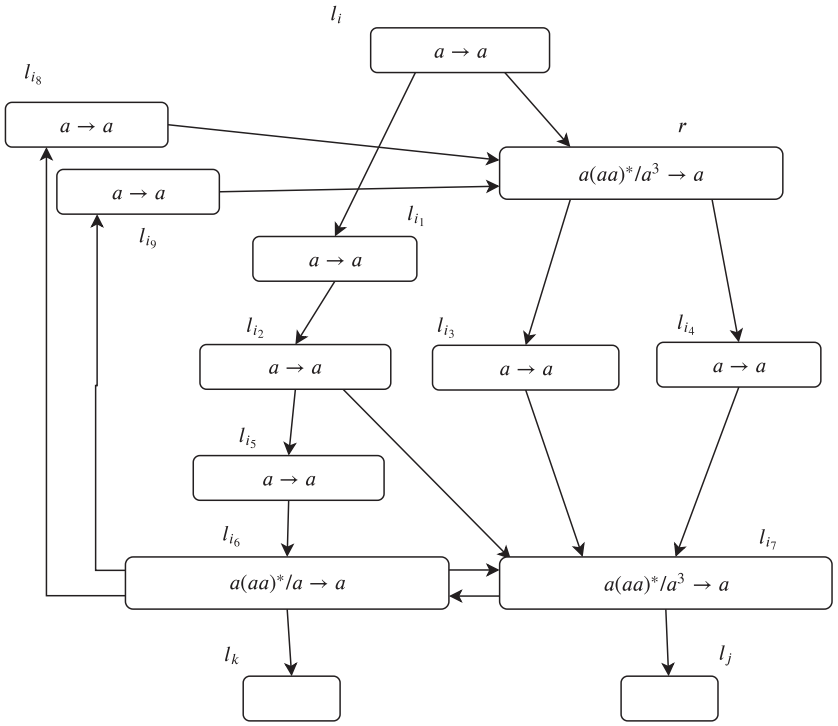


Fig. 1. Theorem 1; SNP: Edited Module SUB simulating $l_i : (SUB(r), l_j, l_k)$

We provide a minor fix to this issue. The new SUB module is seen in Figure 1. Notice that: (1) no new regular expressions were added, and that (2) all the neurons introduced contains only a single rule. Thus the restrictions of the normal form from [8] are not violated. Moreover, the regular expression $a(aaa)*$ from the original configuration is no longer used in this system. And, since this regular expression is also not used in the ADD and FIN (HALT) modules presented in [8] (see Figures 5 and 7 in the appendix for the ADD and FIN(HALT) from [8]), the number of regular expressions used is effectively reduced to 2. These regular expressions are $a \cup a^2$ and $a(aa)^*$.

Let us denote by $N_2SNP(rule_k^e)$ the families of all sets $N_2(\Pi)$ computed by SNP systems, where each neuron in the system has at most k rules, and all rules have at most e distinct regular expressions [8].

Theorem 1. $N_2SNP(rule_2^2) = NRE$.

Proof. The edited SUB module (Figure 1) runs as follows. Assume that the register r is connected to at least two SUB modules, and that the register r contains more than two spikes. Further assume that the neuron l_i receives a single spike at time t. Then at time $t+1$, the neuron l_i will send a spike to the neuron r, as well as to another neuron l_{i_1}. Following the trail of spikes up until l_{i_3} will show that the "correct" l_{i_7} will receive three spikes (thereby allowing it to fire in the next step), while the l_{i_7}'s of other SUB modules will only receive two spikes (preventing them from firing in the next step). Also notice that number of spikes in the l_{i_7}'s of other SUB modules are not offset, since it will stay an even number. In the same way, the neuron l_{i_6} will also not fire since it will receive exactly two spikes (one from i_5 and another from l_{i_7}, both at the same time). Thus only l_j will fire as the next instruction.

In the event that the SUB fails (i.e. the neuron r does not contain any spikes to subtract from), then the following will happen: At time $t+1$ the neuron l_i will send a spike to both neurons r and l_{i_1}. At this point, r will only have a single spike, and thus will not have enough spikes to consume to be able to fire. The neuron l_{i_1} will fire as usual. Following the spike from l_{i_1}, notice that it will "go through" neurons l_{i_2}, l_{i_5}, and l_{i_6} until finally arriving at l_k, from which the next module starts. Notice also that the neuron l_{i_6} also sends spikes to l_{i_8} and l_{i_9}, which will in turn send one spike each to r. This will cause r to fire and "flush out" the spikes it received in this SUB module (recall that it originally had none). The spikes from r fired in this way will eventually be swallowed by l_{i_7}, which will not fire in the next step due to having an even number of spikes.

The modules ADD and HALT which are part of the proof remain unchanged from [8], and are shown in the Appendix, Figures 5 and 7. □

3.2 An improvement to the SNPSP normal form

The improved normal form for SNPSP from [7] used the following restrictions:

1. $a \in \{\pm\}$ for all plasticity rules.
2. The set of synapses is empty in the initial configuration and final configuration of the system
3. Spiking rules will not be used in any of the neurons
4. Each neuron can have at most three rules

The system shown in [7] used five regular expressions, namely: a, a^2, a^3, $a(a^2)^+$, and $a^3(a^2)^+$. Thus the SNPSP normal form from [7] showed that:

Theorem 2. $N_2SNPSP(\{\pm\}, rule_3^5, syninit_\emptyset, synhalt_\emptyset, pplas) = NRE$.

We aim to further improve this normal form, borrowing techniques used in [8]. We add the following restrictions:

1. Each neuron can have only one rule
2. Only two types of regular expressions can be used in the system

Note that the first restriction stated is actually an improvement of an already existing three-rule restriction.

Theorem 3. $N_2SNPSP(\{\pm\}, rule_1^2, syninit_\emptyset, synhalt_\emptyset, pplas) = NRE.$

Proof. To prove Theorem 3, we only need to prove that $NRE \subseteq N_2SNPSP(\{\pm\}, rule_1^3, syninit_\emptyset, synhalt_\emptyset, pplas)$, since the other direction of the inclusion is already shown in [10]. Thus, we construct an SNPSP system Π that simulates a given register machine M and Π satisfies the conditions of the theorem. For each register r in M, there is an associated neuron in Π, which is denoted by σ_r. This neuron σ_r contains $2n$ spikes, where n is the value stored in register r. Simulating an instruction $l_i : (OP(r) : l_j, l_k)$ in M means that its equivalent in Π, σ_{l_i} has one spike and is activated to perform $OP \in \{ADD, SUB\}$ and that afterwards it sends one spike to either σ_{l_j} or σ_{l_k} to begin simulating the next instruction. When M executes l_h, which is the HALT instruction, then Π terminates the computation of M. For Π, this means that σ_{out} fires twice, with the time step difference between the two firings corresponds to the number stored in register 1 in M.

Module ADD simulating $l_i : (ADD(r), l_j, l_k)$ is shown in Figure 2.

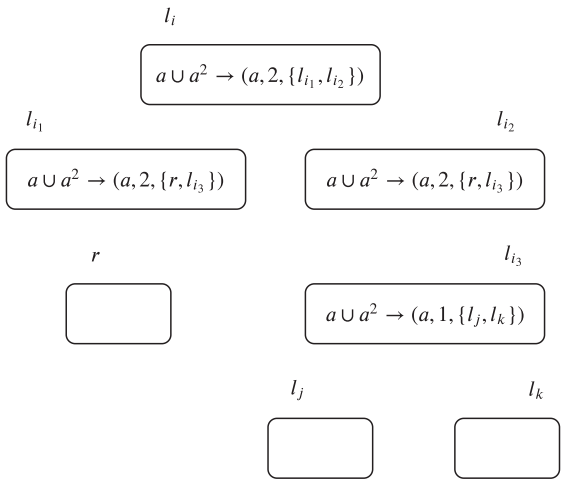


Fig. 2. Theorem 3; SNPSP: Module ADD simulating $l_i : (ADD(r), l_j, l_k)$

Module ADD works as follows. Assume that at time t, σ_{l_i} has one spike and no other neuron in the module has any spike, except for the neuron σ_r which represents the register r. Then at time t, neuron l_i applies its only rule $a \cup a^2 \to (a, 2, \{l_i^1, l_i^2\})$, which creates synapses to neurons l_i^1 and l_i^2 (and therefore

also sends one embedded spike each). At time $t+1$, neuron l_i deletes its synapses to l_i^1 and l_i^2. At the same time, neurons l_i^1 and l_i^2, both having one spike, apply their rules. These rules will create synapses to neurons r and l_i^3, thus sending two embedded spikes to neurons r and l_i^3. At this point, σ_r now has $2n+2$ spikes. At time $t+2$, neurons l_i^1 and l_i^2 delete their synapses from r and l_i^3. At the same time, the neuron l_i^3, now having 2 spikes, will apply its rule. This will cause neuron l_i^3 to create a synapse to either l_j or l_k, with the choice being done nondeterministically. This ends the simulation of module ADD, and at time $t+3$, either σ_{l_j} or σ_{l_k} will begin its instruction.

Module SUB simulating $l_i : (SUB(r), l_j, l_k)$ is shown in Figure 3.

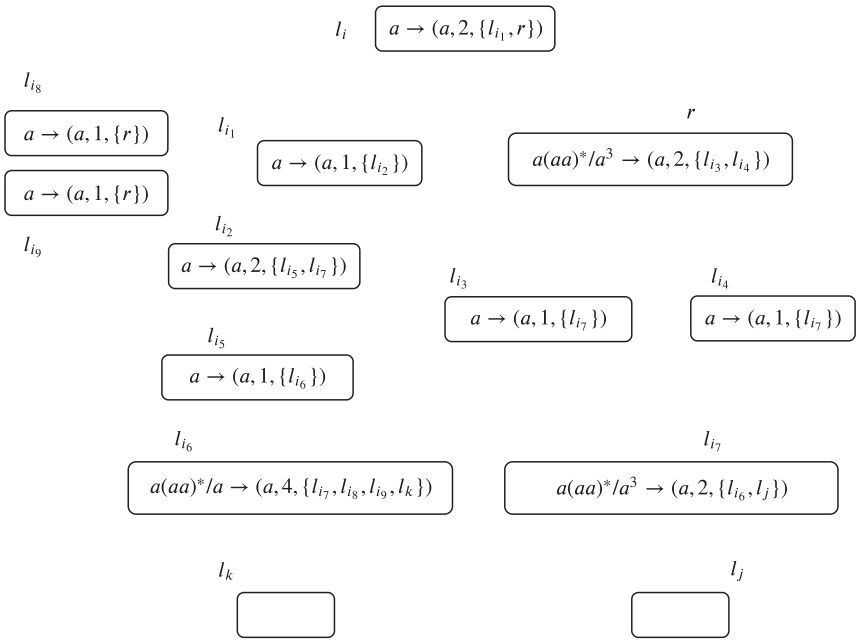


Fig. 3. Theorem 3; SNPSP: Module SUB simulating $l_i : (SUB(r), l_j, l_k)$

First, notice that the structure of Figure 3 is very similar to that of Figure 1. They run in the same fashion. The only big differences are that instead of the synapses in Figure 1, the SNPSP version uses plasticity rules in their place, and that the SNPSP version has less neurons to work with. Secondly, note that following the example from [8] and also for the sake of brevity, the regular expression $a \cup a^2$ is abbreviated to just a in this module. Inspecting Figure 3 will show that all neurons with this type of rule will only ever receive one spike at a time. This abbreviation also applies in the HALT module.

Module SUB works as follows. This module will start to work when a single spike is received by the neuron l_i. If l_i receives the spike at time t, then at $t+1$

it will create synapses and send a spike to neurons r and l_{i_1}. In the event that r contains 2 spikes or more, then it should now have $2n+1$ spikes. It will follow that r sends spikes to l_{i_3} and l_{i_4}, which will in turn send spikes to l_{i_7}. Neuron l_{i_7} will also receive a third spike from l_{i_2}, which will make sure that no interference occurs. At time $t+4$, neuron l_j will fire. In the event that the neuron r contains no spikes, then the following will happen. Neuron r will receive a spike but will not do anything. Meanwhile, l_{i_1} will receive a spike and then sends it to l_{i_2}. The spike will continue down from l_{i_2} to l_{i_5}, and at this point l_{i_7} will also receive a spike but will not do anything. The neuron l_{i_6} will send spikes to the following neurons: l_{i_7} to keep its spike count to an even number, l_{i_8} and l_{i_9} which will keep the neuron r to its original spike count, and finally to l_k to execute the next instruction. Thus at time $t+5$ the next instruction will run. In the same time step, l_{i_8} and l_{i_9} sends a spike each to the neuron r. In the next two time steps, the neuron r will start a trail of spikes up to l_{i_7}, from which the spike count will be even.

Module HALT is shown in Figure 4

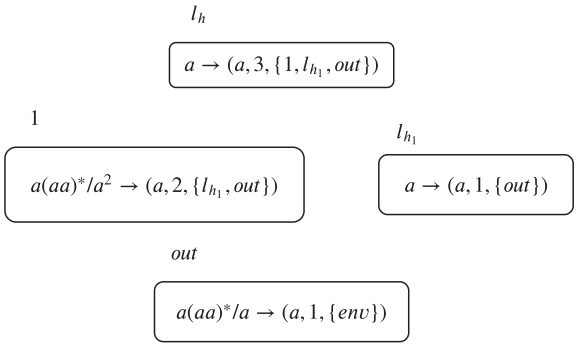


Fig. 4. Theorem 3; SNPSP: Module HALT

Module HALT works as follows. It starts when the neuron l_h receives a single spike, say at time t. At time $t+1$, l_h will send create synapses to neurons 1, l_{h_1}, and *out*. At time $t+2$, the first spike to the environment is fired. Also, starting from time $t+2$ up until neuron 1 is left with only one spike, the following will happen: both 1 and l_{h_1} will create a synapse (and send an embedded spike) to *out*, and neuron 1 also creates one to l_{h_1}. The synapses created are then deleted in the next step. Thus, neuron *out* receives 2 spikes every two steps, and neuron 1 loses two spikes every two steps as well. Eventually, neuron 1 will be only left with a single spike. At this step, only neuron l_{h_1} will be able to fire. The neuron *out* receives only one spike, thereby leaving it with odd-numbered spikes. In the next step, neuron *out* fires into the environment for the second time. The time difference between the first and second spikes to the environment is $2n$.

By the constructions of the modules ADD, SUB, and HALT above, it can be seen that Π correctly simulates the register machine M. Therefore, $N(M) \subseteq N(\Pi)$. □

4 Final remarks

In this work we fixed an interference issue in the improved normal form for SNP presented in [8], by modifying the SUB module from that system. The fix was done without breaking the restriction set in the normal form. In this fix, the number of regular expressions used is also reduced to two. Hence, the conjecture in [8], that there will be no universal SNP without forgetting rules and delays that also uses less than three regular expressions, is answered in the negative. This result was then used to further improve the improved normal form for SNPSP presented in [7]. The main improvement is that (1) we were able to reduce the maximum number of rules per neuron to one (previously three), and also we were able to limit the types of regular expressions to only two. These regular expressions are $a \cup a^2$ and $a(aa)^*$.

This gives answer to open problems formulated in [8], and [7].

It remains an open problem whether the number of regular expressions of the same type of SNP can further be reduced. That is, is it possible to have a system that only uses one type of regular expression? In the event that it is, can the one-rule-per-neuron restriction be maintained?

Acknowledgements

I.C.H. Macababayao and R.T.A. de la Cruz are supported by graduate scholarships from the DOST-ERDT project. F.G.C. Cabarle thanks DOST-ERDT project research grant; the Dean Ruben A. Garcia PCA AY2018–2019, an RLC AY2018–2019 grant, and Project No. 191904 ORG (2019–2020) of the OVCRD in UP Diliman. H.N. Adorna is supported by the Semirara Mining Corp Professorial Chair for Computer Science, RLC grant from UPD OVCRD, and DOST-ERDT research grant. The work was supported by the National Natural Science Foundation of China (Grant Nos. 61472333, 61772441, 61472335, 61672033, 61425002, 61872309, 61771331), Project of marine economic innovation and development in Xiamen (No. 16PFW034SF02), Natural Science Foundation of the Higher Education Institutions of Fujian Province (No. JZ160400), Natural Science Foundation of Fujian Province(No. 2017J01099), Basic Research Program of Science and Technology of Shenzhen (JCYJ20180306172637807).

References

1. The P systems web page, `http://ppage.psystems.eu/`.

2. Alhazov, A., Freund, R., Oswald, M., Slavkovik, M.: Extended spiking neural P systems. In: Hoogeboom, H.J., Păun, G., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing LNCS vol 4361, pp. 123–134. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
3. Cabarle, F.G.C., Adorna, H.N., Jiang, M., Zeng, X.: Spiking neural P systems with scheduled synapses. IEEE Transactions on Nanobioscience 16(8), 792–801 (2017)
4. Cabarle, F.G.C., Adorna, H.N., Pérez-Jiménez, M.J., Song, T.: Spiking neural P systems with structural plasticity. Neural Computing and Applications 26(8), 1905–1917 (2015)
5. Ibarra, O.H., Păun, A., Păun, G., Rodríguez-Patón, A., Sosík, P., Woodworth, S.: Normal forms for spiking neural P systems. Theoretical Computer Science 372(2-3), 196–217 (2007)
6. Ionescu, M., Păun, Gh., Yokomori, T.: Spiking Neural P Systems. Fundam. Inf. 71(2,3), 279–308 (2006)
7. Macababayao, I.C.H., Cabarle, F.G.C., de la Cruz, R.T.A., Adorna, H.N., Xiangxiang, Z.: An improved normal form for spiking neural p systems with structural plasticity. In: Păun, G. (ed.) Proceedings of the 20th International Conference on Membrane Computing, CMC20. pp. 429–438. Bibliostar (2019)
8. Pan, L., Păun, G.: Spiking neural P systems: an improved normal form. Theoretical Computer Science 411(6), 906–918 (2010)
9. Pan, L., Păun, G., Pérez-Jiménez, M.J.: Spiking neural P systems with neuron division and budding. Science China Information Sciences 54(8), 1596 (2011)
10. Păun, A., Păun, G.: Small universal spiking neural p systems. BioSystems 90(1), 48–60 (2007)
11. Păun, G.: Membrane Computing: An Introduction. Springer Berlin Heidelberg (2002)
12. Song, T., Pan, L.: A normal form of spiking neural P systems with structural plasticity. International Journal of Swarm Intelligence 1(4), 344–357 (2015)
13. Song, T., Pan, L., Jiang, K., Song, B., Chen, W.: Normal forms for some classes of sequential spiking neural P systems. IEEE Transactions on Nanobioscience 12(3), 255–264 (2013)

A Appendix

The following are the ADD module, SUB module, and FIN (HALT) module of the improved SNP normal form presented in [8].

Module ADD simulating $l_i : (ADD(r), l_j, l_k)$ is shown in Figure 5.
Module SUB simulating $l_i : (SUB(r), l_j, l_k)$ is shown in Figure 6.
Module HALT is shown in Figure 7

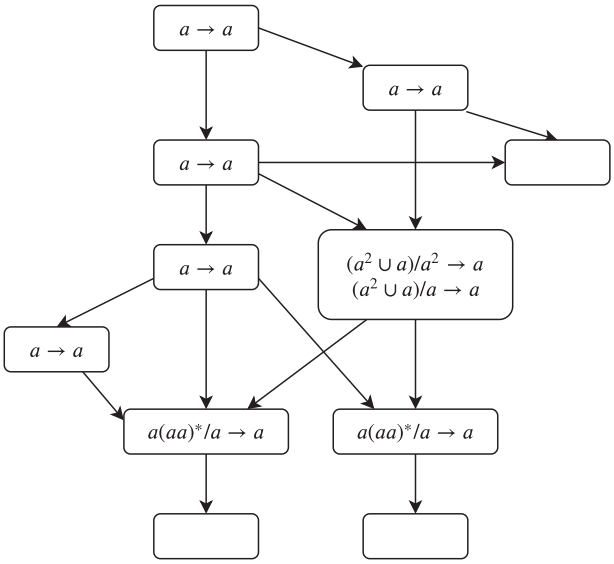


Fig. 5. SNP: Module ADD simulating $l_i : (ADD(r), l_j, l_k)$

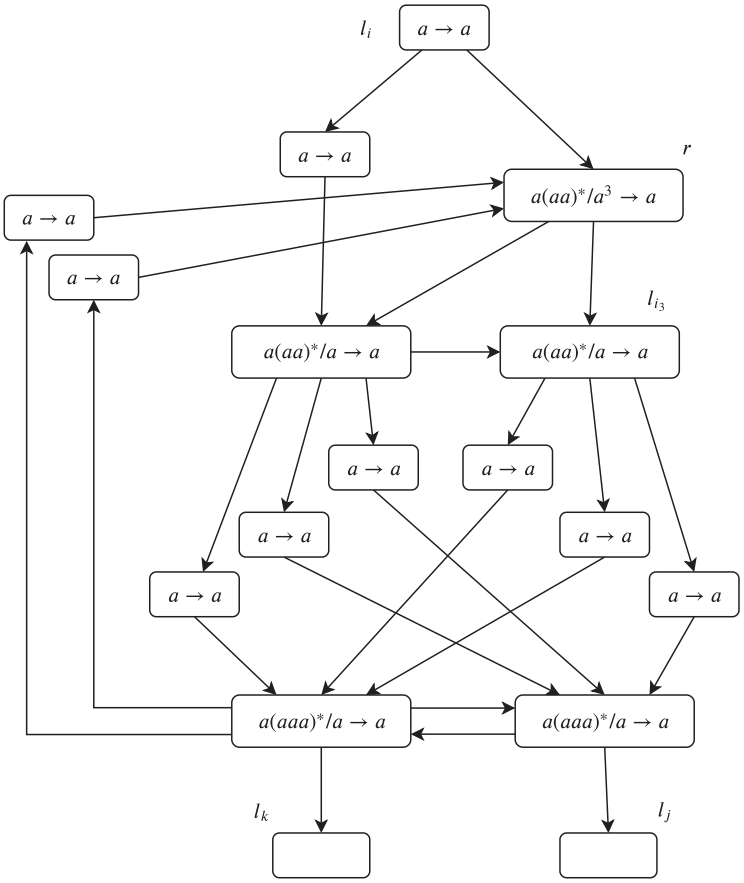


Fig. 6. SNP: Module SUB simulating $l_i : (SUB(r), l_j, l_k)$

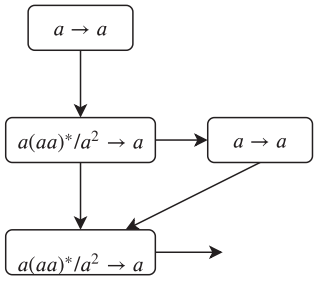


Fig. 7. SNPSP: Module HALT

A Framework for Evolving Spiking Neural P Systems

Lovely Joy Casauay[1], Ivan Cedric H. Macababayao[1], Francis George C. Cabarle[1,2⋆], Ren Tristan A. de la Cruz[1], Henry N. Adorna[1], Xiangxiang Zeng[3], Miguel Ángel Martínez-del-Amor[4]

[1]Algorithms & Complexity
Dept. of Computer Science, University of the Philippines Diliman
Diliman 1101 Quezon City, Philippines.
[2]Shenzhen Research Institute of Xiamen University
Xiamen University, Shenzhen 518000, Guangdong, China.
[3]School of Information Science and Engineering
Hunan university 410082, Changsha, China.
[4]Research Group on Natural Computing, Dept. Computer Science and AI,
University of Seville, Seville, Spain

Abstract. In current literature, there is a lack of research on the optimization of *spiking neural P systems* (SN P systems) and, consequently, also a lack of automation to do this process of optimization. We address this gap by designing a genetic algorithm (GA) framework that transforms an initial SN P system Π_{init}, designed to approximate a function $f(w, x, y, \ldots) = z$, into a smaller or more precise system Π_{final} that also approximates the output z given the same input/s $w, x, y, \ldots$.
The design of the GA framework is constrained by evolving Π_{init} only through its topology. The rules inside the neurons must stay constant, while the synapses and neurons may vary.
The results of the experiments conducted show that evolving the topology of a designed Π_{init} using genetic algorithms does not only lessen its number of neurons and synapses, but also helps it achieve a higher precision. The GA framework is especially effective on Π_{init}'s containing the subgraph of an already better SN P system that computes f.

Keywords: Spiking neural P system · Membrane computing · Neural computing · Genetic algorithm · Evolutionary computing

1 Introduction

Nature is a great influence on the way we perceive computation, and this perception influences the manner in which we address problems with natural computing.

We begin with the following quote from Gross [1998]:

⋆ corresponding author fccabarle@up.edu.ph

> "Life is computation. Every single living cell reads information from a memory, rewrites it, receives data input (information about the state of its environment), processes the data and acts according to the results of all this computation. Globally, the zillions of cells populating the biosphere certainly perform more computation steps per unit of time than all man made computers put together." [11]

In this study, three branches of natural computing are considered.

The first one is **neural computing**, which is inspired by the way the human brain computes. Neural computing has shown a lot of advances in recent years, specifically with the use of *artificial neural networks* (ANNs) in the field of artificial intelligence (AI). The application of ANNs has been successful in a wide array of modern-day applications including, but not limited to, the fields of medicine, transportation, robotics, and music.

The second relevant branch of natural computing was introduced by Gheorghe Păun in [21], and he called this **membrane computing**. The new computing device Păun designed, called *P system*, is based on the notion of a membrane structure. Although membrane computing is a relatively new field of study, it is, nonetheless, very interesting because of its Turing-completeness and inherent parallelism. Within 6 years of its introduction, membrane computing had known applications in computer security, NP-complete optimization problems (and other computationally hard problems), computer graphics, biology, and linguistics as seen in [3].

The last branch we will discuss is **evolutionary computing**, a research area inspired by the process of natural evolution based on Darwinian principles. The general idea behind evolutionary computing is that given an environment filled with a population of candidate solutions, the quality of these solutions (*i.e.* how well they can solve the problem) determines the rate in which they will survive within the environment and be kept for constructing further candidate solutions [6].

Combining some of the different branches of natural computing has become a common approach in literature for solving problems. A particularly interesting and popular research direction is the application of evolutionary computing algorithms to neural computing and membrane computing, which resulted in neuroevolution (NE), membrane-inspired evolutionary algorithms (MIEA), and automated design of membrane computing models (ADMCM) [8] [28].

In this study, we investigate the application of one of the main paradigms of evolutionary computing called *genetic algorithm* (GA) on a recently developed computing model called *spiking neural P system*, an interplay between the concepts of spiking neural networks and P systems. Ideas from this study were first given in a presentation during BWMC2018 in `https://www.gcn.us.es/files/bwmc2018-evolsnp-present.pdf`.

2 Preliminaries

2.1 Spiking Neural P Systems

In this section, we discuss a relatively new kind of P system called *Spiking Neural P systems*, *a.k.a.* SN P systems, introduced in [15].

Before the introduction of SN P systems, neural-like P systems were introduced in [18]. This kind of P system was inspired by neurobiology and incorporated ideas such as: the replication of impulses in the case of multiple synapses, linking neurons to several neighboring neurons, the state of a neuron, among others. The fact that most neural impulses are almost identical, however, was not incorporated. This is the part where Ionescu et al. [15] took inspiration from *spiking neural networks* (SNNs), a.k.a. the third generation of neural networks.

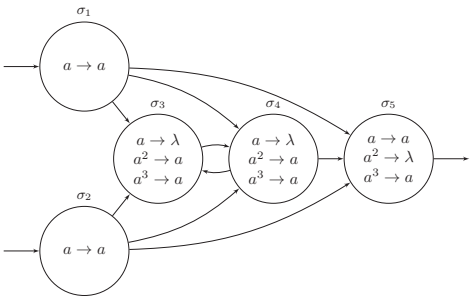


Fig. 1: An example of an SN P system. It simulates the *binary addition* function. σ_1 and σ_2 are the input neurons, while σ_5 is the output neuron. See table 1 for a sample run.

Table 1: A sample run of the SN P system presented in figure 1. This system simulates the binary addition function: a represents 1 and λ represents 0. The input spike trains in_1 and in_2 correspond to what is received by the input neurons σ_1 and σ_2, respectively. In this example, $in_1 = 111$ and $in_2 = 101$, thus the output spike train must be $out = 1100$.

time	in_1	in_2	σ_1	σ_2	σ_3	σ_4	σ_5	out
0	aaa	$a\lambda a$	-	-	-	-	-	λ
1	aa	$a\lambda$	a	a	-	-	-	$\lambda\lambda$
2	a	a	a	λ	a^2	a^2	a^2	$\lambda\lambda\lambda$
3	-	-	a	a	a^2	a^2	a^2	$\lambda\ \lambda\lambda\lambda$
4	-	-	-	-	a^3	a^3	a^3	$\lambda\lambda\ \lambda\lambda\lambda$
5	-	-	-	-	a	a	a	$a\lambda\lambda\ \lambda\lambda\lambda$
6	-	-	-	-	-	-	-	$aa\lambda\lambda\ \lambda\lambda\lambda$

Ionescu et al. [15] took the fundamental characteristic of SNNs, which is the use of *pulse encoding*, and described their framework as such: "...in this framework, time is (also) a data support; it is not (only) a computing resource as in usual complexity theory, but a way to encode information". This means that instead of having the information encoded in a sequence of different symbols, it is encoded in the *sequence of moments* in which a *spike* occurs, represented by a unique symbol.

In the same paper, Ionescu et al. showed that SN P systems are Turing-complete both in accepting and generative mode. In accepting mode (spikes *can* be received from the environment), deterministic SN P systems are sufficient, while in generative mode (spikes *cannot* be received from the environment), SN P systems of only one neuron behaving non-deterministically are sufficient. Additionally, non-deterministic systems have been demonstrated to solve NP-complete problems, such as SUBSET SUM and 3SAT, in constant time [17]. This display of computational power makes SN P systems one of the most intriguing and promising types of membrane systems.

Refer to [22] for the formal definition of SN P systems in a general form and in the extended (*i.e.* the rules are able to produce more than one spike) computing (*i.e.* the system is able to take an input and provide an output) version.

2.2 Genetic Algorithms

Genetic algorithm (GA) was introduced by John Holland in his book called *Adaptation and Natural Systems* [14]. His work became one of the major foundations of later studies and applications of what we now know as *evolutionary computing*. Although optimization was not the main focus of Holland's work on adaptive systems, the work of his graduate students in [5] and [9] explored the application of GA in this area of study, and later served as a strong basis and motivation for future work. To present day, the use of GAs for optimization continues to be popular and is frequently successful in real applications [23].

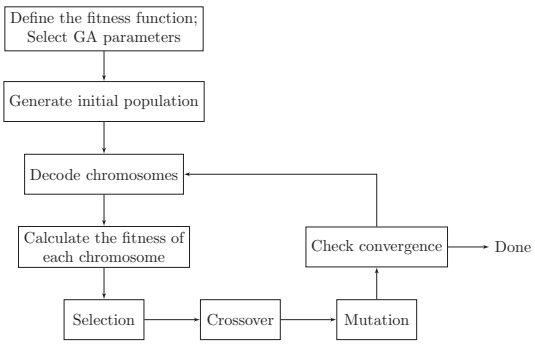


Fig. 2: Canonical genetic algorithm flowchart. This is a slightly modified version of the binary genetic algorithm flowchart in [13], wherein we replaced "cost", "select mates", and "mating" with "fitness", "selection, and "crossover", respectively.

In figure 2, we see the process of how a canonical (*i.e.* binary) genetic algorithm works. We start with designing the GA by defining the fitness function that will guide the *selection* operator later, as well as selecting the GA parameters. The evolution starts by generating the initial population of binary strings [27], also known as *genotypes* [14] or *chromosomes* [24]. For our purposes, we will use the latter terminology.

Next, we decode the chromosomes. In canonical GA where the chromosomes are binary strings and the fitness function, in most cases, computes with continuous (i.e. real-valued) inputs, the chromosomes must be decoded into continuous values for calculating their fitness[1] [13].

[1] Although the terms *evaluation* and *fitness* can be used interchangeably in literature, there is still a slight distinction between them as explained in [27]. Evaluation functions compute the output for each chromosome independently from other chromosomes, whereas fitness functions compute the output for each chromosome with respect to the other chromosomes in the current population. This is important to note since an evaluation function is used in this study to implement the design of the genetic algorithm function, which will be discussed in section 5. Moreover, the term "precision" shall be used to denote the output of an evaluation function in succeeding parts of this paper, while we will continue to use the term "fitness" for the output of a fitness function.

A fitness function may be a mathematical function, an experiment, or a game. The goal is to find the chromosome representing an appropriate solution in the population to get a desired output from the fitness function. The fitness of a chromosome is calculated by getting the distance between the desired and the actual output value of the fitness function—the smaller this distance is, the fitter the chromosome [13].

Below are some of the ways a fitness function can be defined [27]:

1. In canonical GA, fitness is equal to $f_i/\bar{f}$ where f_i is the fitness of chromosome i and $\bar{f}$ is the average fitness of all the chromosomes in the current population;
2. In [1] and [26], they calculated the fitness based on the rank of a chromosome in the current population, and;
3. In [10], they employed tournament selection as a sampling method.

Once the fitness is calculated, the *intermediate population* is formed through *selection*. The probability that a chromosome will be copied and placed in the intermediate population is proportional to its fitness. One of the ways selection can be done is by using *remainder stochastic sampling*, wherein a chromosome of fitness $f_i/\bar{f} = 1.48$ places 1 copy in the intermediate population and has a 0.48 chance to place a second copy, and a chromosome of fitness $f_i/\bar{f} = 0.17$ will have a 0.17 chance to place a copy [27].

After the selection, *crossover* and *mutation* operators are applied to form the *next population*. During the execution of the GA, a *generation* starts from the current population up to when the next population is formed.

Crossover is applied to random pairs of chromosomes, which are also referred to as *parents*. Crossover is regarded as the main feature of GA that distinguishes it from other evolutionary algorithms, as it uses the concept of genetic recombination wherein *child* chromosomes are formed using a combination of their parents' traits, represented by segments of bits [20] [23]. These child chromosomes are then placed in the next population. Mutation is done canonically by randomly flipping the bits of a chromosome, thus introducing new genetic material into the population to get out of the local minimum.

After fitness calculation, selection, crossover, and mutation are done, we repeat the process of evolution until a predefined condition for convergence is met or the maximum number of generations is reached.

3 Chromosome Representation

As mentioned in section 2.2, evolution in genetic algorithms starts by generating an initial population of chromosomes. We use chromosomes to represent a solution, i.e. an SN P system, within a population. An SN P system Π can be encoded into a chromosome, formally defined below. See [22] for the formal definition of SN P systems which is used throughout this section.

Definition 1 (Chromosome)

A chromosome is a construct of the form

$$chrom = (num\_nrns, spikes, rules, input\_nrns, output\_nrns, syn\_matrix, presicion, fitness, active\_nrns, connected\_nrns, connected\_to\_input\_nrns, connected\_to\_output\_nrns),$$

where:

1. $num\_nrns = m + 1$, *where* m *is the degree, i.e. number of neurons, of* Π. *The environment is counted as a neuron in a chromosome, hence we refer to* nrn_{m+1} *as the environment neuron;*
2. $spikes = \{n'_1, \ldots, n'_{m+1}\}$, *where* $n'_i \geq 0$ *is the number of initial spikes contained in* σ_i, *where* σ_i *is a neuron in* Π;
3. $rules = \{R'_1, \ldots, R'_{m+1}\}$, *where* R'_i *is a finite set of rules contained in* σ_i *of the same form as* R_i:
4. $input\_nrns = \{index_1, \ldots, index_{|in|}\}$, *where* $index_j = i$ *and* nrn_i *is an input neuron,* $i \in in$;
5. $output\_nrns = \{m + 1\}$, *where* nrn_{m+1} *is the environment neuron;*
6. $syn\_matrix = \{syn\_row_1, \ldots, syn\_row_{m+1}\}$, *where* $syn\_row_i = \{syn\_col_1, \ldots, syn\_col_{m+1}\}$ *and* $syn\_col_j \in \{0, 1\}$. $syn\_matrix[i][j] = 0$ *and* $syn\_matrix[i][j] = 1$ *signify the absence and presence of a synapse from* nrn_i *to neuron* j, *respectively;*
7. $precision = p$, *where* $0 \leq p \leq 1$;
8. $fitness = f$, *where* $0 \leq f \leq 1$;
9. $active\_nrns = \{index_1, \ldots, index_k\}$ *with* $k \geq 0$, *where* $index_j = i$ *and* nrn_i *is an active neuron;*
10. $connected\_nrns = \{index_1, \ldots, index_k\}$, *with* $k \geq 0$, *where* $index_j = i$ *and* nrn_i *is a connected neuron;*
11. $connected\_to\_input\_nrns = \{index_1, \ldots, index_k\}$, *with* $k \geq 0$, *where* $index_j = i$ *and there is a path from at least one of the input neurons to* nrn_i;
12. $connected\_to\_output\_nrns = \{index_1, \ldots, index_k\}$, *with* $k \geq 0$, *where* $index_j = i$ *and there is a path from* nrn_i *to the environment neuron;*

The concept of *active* and *connected* neurons is discussed in section 5.5. Below is the formal definition of the SN P system that simulates the *binary addition* function in figure 1. Again, this based on the definition from [22]:

$$\Pi_{bin\_add} = (O, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, syn, in, out),$$

where:

1. $O = \{a\}$
2. $\sigma_1 = (0, \{a \to a\})$
   $\sigma_2 = (0, \{a \to a\})$
   $\sigma_3 = (0, \{a \to \lambda, a^2 \to a, a^3 \to a\})$
   $\sigma_4 = (0, \{a \to \lambda, a^2 \to a, a^3 \to a\})$
   $\sigma_5 = (0, \{a \to a, a^2 \to \lambda, a^3 \to a\})$
3. $\{(1,3), (1,4), (1,5), (2,3), (2,4), (2,5), (3,4), (4,3), (4,5)\}$
4. $in = \{1, 2\}$
   $out = \{5\}$

A chromosome is implemented as a Python object with each set implemented as a list. Π_{bin_add} can be encoded into a chromosome $chrom_{bin\_add}$ with the following attribute values:

1. $num\_nrns = 6$
2. $spikes = [0, 0, 0, 0, 0, 0]$
3. $rules = [$
   $[a \to a]$,
   $[a \to a]$,
   $[a \to \lambda, a^2 \to a, a^3 \to a]$,
   $[a \to \lambda, a^2 \to a, a^3 \to a]$,
   $[a \to a, a^2 \to \lambda, a^3 \to a]$,
   $[]$
   $]$

4. $input\_nrns = [1, 2]$
5. $output\_nrns = [6]$
6. $syn\_matrix = [$
   $[0, 0, 1, 1, 1, 0],$
   $[0, 0, 1, 1, 1, 0],$
   $[0, 0, 0, 1, 0, 0],$
   $[0, 0, 1, 0, 1, 0],$
   $[0, 0, 0, 0, 0, 1],$
   $[0, 0, 0, 0, 0, 0],$
   $]$
7. $precision = 1$
8. $fitness = 1$
9. $active\_nrns = [1, 2, 3, 4, 5, 6]$
10. $connected\_nrns = [1, 2, 3, 4, 5, 6]$
11. $connected\_to\_input\_nrns = [1, 2, 3, 4, 5, 6]$
12. $connected\_to\_output\_nrns = [1, 2, 3, 4, 5, 6]$

The precision of this SN P system is 100% according to the experiments conducted. Fitness is computed by dividing the precision of a chromosome by the average precision in the population, thus its fitness is also 100% given that it's the only chromosome in a population.

4 Genetic Algorithm Framework

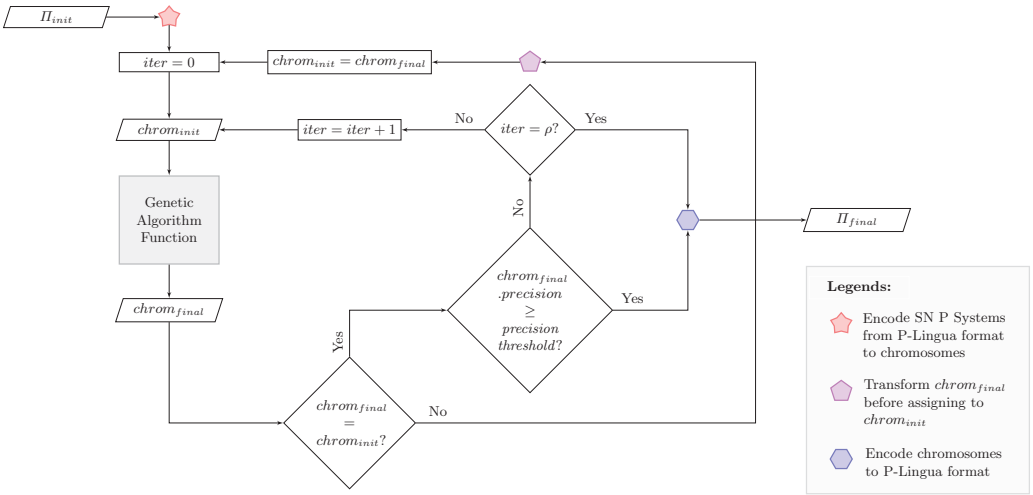


Fig. 3: Genetic algorithm framework. ρ is the maximum number of inner loops performed within an outer loop. This is a parameter set by the user. All chromosomes are implemented as Python objects that have a *precision* attribute, see section 3. The *precision threshold* is a parameter set by the user which halts the genetic algorithm framework when it is reached.

In this section, we discuss the main output of this study: a genetic algorithm framework that transforms an initial SN P system Π_{init}, designed to approximate a function $f(w, x, y, \ldots) = z$, into a smaller or more precise system Π_{final} that also approximates the output z given the same input/s $a, b, c, \ldots$.

Figure 3 provides a visual representation of the GA framework. As denoted by the red square, we begin by encoding the initial SN P system that we want to

evolve, called Π_{init}, from its *P-Lingua*[23] format into a chromosome $chrom_{init}$ that can be processed by the framework. After encoding Π_{init}, the variable *iter* is set to zero and $chrom_{init}$ is inputted to the *Genetic Algorithm Function*. As can be seen in the figure, the GA framework contains two loops: an *outer* and an *inner* loop.

The **outer loop** checks whether the $chrom_{final}$ outputted by the GA function is the same as $chrom_{init}$ or not. If it is not, $chrom_{final}$ is transformed before being set as $chrom_{init}$ for the next loop. Note that if $chrom_{final}$ is not the same as $chrom_{init}$, it is sure to be a better SN P System because of how the GA function's *Selection* operator is implemented. This is discussed in section 5.2.

We use outer loops in our implementation of the GA framework for a number of reasons. First, they reduce the supervision needed during execution as evolution continues as long as a better solution, i.e. SN P System, is achieved. They automate the need for a user to execute the framework again using a previous output as input. Second, they act as checkpoints that provide a clear insight on the state of an experiment by giving the user an ability to compare the $chrom_{final}$'s of each outer loop. With this, it is possible to see what is removed or added in an SN P System that made it better from its previous form, and by how much it has improved. Last, and most importantly, they allow the transformation of $chrom_{final}$'s before setting them as the new $chrom_{init}$'s to achieve better results in future evolutions. This transformation is denoted by the purple square.

In our case, we can delete either inactive or disconnected neurons from the $chrom_{final}$'s to prevent them from being reconnected back later on. See sections 5.5, 5.6, and 6.2 for more information.

The **inner loop** is dependent on whether the variable *iter* has reached its maximum value ρ. The constant ρ is a parameter set before executing the framework and signifies the maximum number of *evolutions* the framework can make within one *outer loop*.

The inner loop is an important and necessary part of the framework as it addresses one of the key characteristics of genetic algorithms. GA is a heuristic that depends on the quality of its initial population of solutions. The probability of converging to a better final solution increases as the quality of the initial population increases. As such, if the GA function has failed to produce a $chrom_{final}$ better than the current $chrom_{init}$ to start a new outer loop, and $chrom_{final}$ has not yet reached the *precision threshold* set by the user, *iter* is incremented by 1 and the GA function is called once again to pseudo-randomly generate another initial population. Once the framework has reached the maximum number of evolutions ρ in one outer loop, the GA framework halts. The current $chrom_{init}$ is encoded into P-Lingua format, as denoted by the blue square. It is then outputted as Π_{final}.

[2] `http://www.p-lingua.org/wiki/index.php/Main_Page/`

[3] **P-Lingua** is a programming language which aims to be a standard in defining P systems. We use its current latest version: *version 4.0*. Figure 7 shows the P-Lingua format used for the SN P system in figure 1.

5 Genetic Algorithm Function

Now that we are done discussing the overall design of the genetic algorithm framework, in this section, we will talk about the design and implementation of the major components of the genetic algorithm function used by the framework. Let us look at figure 4 for the visual representation of the genetic algorithm function.

After receiving $chrom_{init}$ as input, the initial population of chromosomes, $chrom\_pop_0 = \{chrom_{0,1}, chrom_{0,2}, \ldots, chrom_{0,r}\}$, is formed with the mutations of $chrom_{init}$. Once the initial population is generated, $chrom\_pop_0$ is sent to the SN P system simulator, along with a_i from $S = \{(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)\}$, where $k \geq 1$ is the number of fitness cases, a_i is a set of input spike trains $\{\alpha_{i,1}, \ldots \alpha_{i,j}\}$, $j = |in|$, and b_i is the *ideal* output spike train with respect to a_i. The algorithms presented in [2] guided the simulator's implementation.

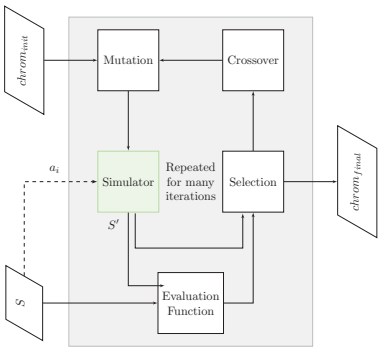


Fig. 4: Design of the genetic algorithm function. a_i is from $S = \{(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)\}$, where $k \geq 1$ is the number of fitness cases, a_i is a set of input spike trains $\{\alpha_{i,1}, \ldots \alpha_{i,j}\}$, $j = |in|$, and b_i is the *ideal* output spike train with respect to a_i. S' contains a_i and b'_i, wherein the latter is the set of *actual* output spike trains. See figure 5 for the SN P System Simulator, denoted by the green square.

In figure 5, we see that the SN P simulator first translates each of the chromosomes in $chrom\_pop_0$ into their corresponding P-lingua formats. The SN P systems are simulated in this format to produce $S' = \{S'_1, S'_2, \ldots, S'_r\}$, where $r = |chrom\_pop_0|$, $S'_u = \{(a_1, b'_1), (a_2, b'_2), \ldots, (a_k, b'_k)\}$, and $b'_{i,u}$ is the *actual* output spike train of the mutated chromosome $chrom_{0,u}$. After the simulation, S and S' are both sent to the evaluation function to compute the precision of all $chrom_{0,u}$'s. This is done by comparing the value of each actual output spike train $b'_{i,u}$ with its corresponding ideal output spike train b_i.

The set containing the precision of each chromosome $prec_0$ is sent to the selection operator, along with $chrom\_pop_0$. The parent chromosomes are selected for the crossover operator. Chromosomes resulting from the crossover undergo mutation, forming the new population $chrom\_pop_1$.

The cycle of evolution continues until a predefined halting condition is met and the fittest chromosome $chrom_{final}$ is selected.

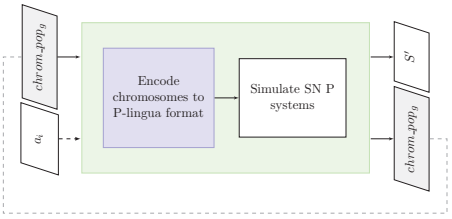


Fig. 5: SN P system simulator. The simulator translates the chromosomes to their P-lingua format, and outputs the same set of chromosomes that were inputted. b'_i's which will form the S' output are generated with the a_i input. $chrom\_pop_g$ is the population of chromosomes at generation g.

5.1 Mutation

In all succeeding generations after the first, each chromosome in the population has a chance to be mutated according to the mutation rate parameter, which is set before the execution of the GA framework. Once a chromosome is tagged for mutation, the function pseudo-randomly chooses between the following methods of mutation[4], *i.e.* all three have equal chances of getting chosen:

1. *disc_nrn():* This method is used for pseudo-randomly choosing any neuron within a chromosome to be disconnected, aside from input neurons, the environment neuron[5], or already disconnected neurons. The chosen neuron is disconnected by first creating synapses from all pre-synaptic neurons of its in-going synapses to all post-synaptic neurons of its outgoing synapses, and then deleting all of its in- and outgoing synapses.
2. *del_syn():* This method is used for pseudo-randomly choosing a synapse to delete within a chromosome.
3. *add_syn():* This method is used for pseudo-randomly choosing a synapse to be added, with the condition that the environment neuron cannot be used as a pre-synaptic nor post-synaptic neuron. Note that the environment neuron cannot supply spikes to any other neuron. Previously disconnected neurons may be reconnected using this method.

5.2 Selection

In our implementation, the selection operator not only selects the chromosomes that will be placed in the next generations, but also checks for convergence at the end of every generation. The following details its implementation:

1. *Check for convergence every generation.* At the start of evolution, $chrom_{init}$ is set as $chrom_{best}$. Evolution is immediately halted at generation g if the population contains a chromosome $chrom_i$ that fulfills one of the following conditions:
   (a) $chrom_i$ is more precise than $chrom_{init}$, or;
   (b) $chrom_i$ is as precise as $chrom_{init}$ and has fewer neurons than $chrom_{init}$.
   If such a $chrom_i$ does exist, the GA function outputs it as $chrom_{final}$. Otherwise, a chromosome $chrom_j$ is set as the new $chrom_{best}$ if it is as precise as the current $chrom_{best}$ and has fewer neurons or synapses. Upon reaching the maximum number of generations, the GA function outputs $chrom_{best}$ as $chrom_{final}$. Note that the GA function only outputs either a better chromosome than $chrom_{init}$ or $chrom_{init}$ itself.

[4] Pseudo-random selections of one element within a defined set are done using the method *randint* of a Python library called *random*

[5] The environment is counted as a neuron in the chromosome definition of an SN P system. See section 3.

2. *Select chromosomes for the next generation.* If the GA function did not converge nor reach the maximum number of generations, the *Selection* operator will then rank the population according to their fitness and get a number of highest ranking chromosomes, as in [4]. The amount of chromosomes selected are dependent on the selection rate parameter set by the user. The **fitness** of a chromosome is calculated by dividing its precision with the average precision in the population. The precision of a chromosome is outputted by the *evaluation function.*

5.3 Crossover

The crossover operator is inspired by the one used in [19].

Crossover is done by pseudo-randomly selecting two chromosomes, $chrom_1$ and $chrom_2$ in the population as parents and a neuron nrn_i contained by both chromosomes. If the set of post-synaptic neurons connected to the outgoing synapses of $chrom_1$ and $chrom_2$ are different, these sets are swapped. This is implemented by interchanging the i^{th} row of the corresponding synapse matrices of the chosen pair of chromosomes.

If the current population size is less than what is set as parameter[6], all of the chromosomes are automatically added to the next population. A pair chromosomes are then pseudo-randomly selected as parents and crossed-over to produce two new chromosomes. This is done until the next population is full.

Otherwise, if the current population size is equal to what is required, each chosen pair of chromosome in the population has a chance to be crossed-over depending on the crossover rate parameter. Chromosomes that were not selected for crossover, i.e. *not* parents, are added to the next population as is. Those that were successfully crossed-over are discarded, and the resulting chromosomes are placed in the next population, instead. In other words, successful parents are never carried over to the next population.

5.4 Evaluation Function

The evaluation function used to compute precision of each chromosome is implemented by finding the longest common substring (LCS) between the ideal and actual output spike trains. This is the chosen method to eliminate error in computation that is caused by the unnecessary padding in actual output spike trains. For example, if the ideal output spike train is "101100" and the actual output spike train is "1110110000", using LCS for the evaluation function gives us a precision of 100% because the latter contains all of 6 characters of the ideal spike train: "~~11~~**101100**00". Directly comparing the two binary strings would only give us a precision of $1/6 = 16.67\%$ as their second characters do not match

[6] This happens when the selection rate is less than 100%, since the *Selection* operator would only be placing the highest-ranking chromosomes from the last population to the current one.

anymore. Note that the padding in the actual output spike train varies depending on the rules and topology of an SN P system.

Algorithm 1 describes the implementation of the evaluation function in pseudocode form. Although not used here, it is important to note that according to Gusfield [12], the longest common substring between two strings can be found in linear time by using a generalized suffix tree. We leave this optimization as part of our future work.

5.5 Inactive vs. Disconnected Neurons

To properly understand the validations used in the GA function discussed later in this section and the experiment setup discussed in section 6, we must first talk about the difference between an *inactive* vs. *disconnected* neuron.

A neuron is **inactive** if it is connected by at least one synapse to the rest of the SN P System, but renders no effect on the resulting spike train. During validation, it is checked if there is a path from a neuron to the environment. If yes, then it is checked if there is a path from one of the input neurons to the neuron in question. If there is, then the neuron is active. Otherwise, it is checked if the neuron has any initial spikes and it has a corresponding rule to use. If yes, then the neuron is active, otherwise it is inactive.

A neuron is **disconnected** if it doesn't have a path from any of the input neurons nor to the environment. For example, a neuron that has does not have any in-going or outgoing synapses (other than a self-loop) is considered disconnected. A group of neurons that are connected to each other but are not connected to the rest of the system are also considered disconnected.

The set of disconnected neurons in an SN P System is a subset of the set of inactive neurons. Likewise, the set of active neurons in an SN P System is a subset of the set of connected neurons.

5.6 Validations Used in the GA Function

There are a total of 6 validations used in the GA function. The following four (4) remain consistent for all the GA function variants:

1. Only *active neurons* must be counted during the comparison of the chromosomes.
2. Every time either mutation or crossover is performed, the resulting chromosome must always have a path from at least one of the input neurons to the environment. If it does not, the output spike train produced by the corresponding SN P System would not be dependent on any of the input spike trains, making it an invalid solution. The untransformed chromosome is kept.
3. If a resulting chromosome from either mutation or crossover is the same as the input to the current outer loop, then it is invalid and the untransformed chromosome is kept for the next generation.

4. The resulting SN P System Π_{final} from the GA framework must never have a self-loop. Before outputting the SN P System, a method named *remove_self_loops()* is called first on the corresponding chromosome. Section 5.7 details the logic behind and implementation of the method.

The following two are changed in the other variants of the GA function in section 6.2.

1. The formation of self-loops is not allowed. Self-loops could result from the following methods:
   (a) *add_syn()* - by picking the same neuron as both the pre-synaptic and post-synaptic neuron, or;
   (b) *disc_nrn()* - by disconnecting a neuron that has synapses to and from the same neuron.
2. *Inactive neurons* must be deleted after the end of each outer loop. The result of an outer loop is always better than its input, otherwise, the evolution stops. With this, we can delete all inactive neurons, making it impossible for them to get reconnected in the next outer loop via *add_syn()*.

5.7 Removing Self-Loops

The following procedure was observed to yield derived chromosome without self-loops that produces the same results as the chromosome containing self-loops it is derived from. We will refer to these as $chrom_{new}$ and $chrom_{old}$, respectively.

This procedure is repeated for all neurons nrn_i that has a self-synapse, i.e. $syn\_matrix[i][i] = 1$.

1. A neuron that has a self-synapse is duplicated. We will refer to this neuron and its duplicate as nrn_{orig} and nrn_{dupl}, respectively.
2. The self-synapse of nrn_{orig} is removed from the synapse matrix, i.e. $syn\_matrix[orig][orig] = 0$.
3. All in-going synapses to nrn_{orig} must be replicated to nrn_{dupl} such that $\forall nrn_i$, $syn\_matrix[i][dupl] = 1$ if $syn\_matrix[i][orig] = 1$, where $i = 1, 2, 3...m$ and m is the number of neurons in the SN P System.
4. Lastly, there must be a synapse from nrn_{orig} to nrn_{dupl} and vice versa, i.e. $syn\_matrix[orig][dupl] = 1$ and $syn\_matrix[dupl][orig] = 1$.

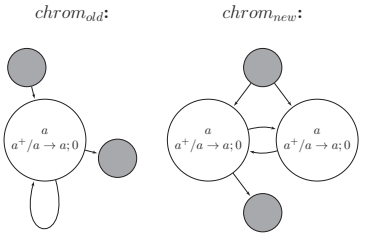


Fig. 6: Removing a self-loop. In this figure, $chrom_{old}$ is transformed into $chrom_{new}$ by using the procedure described in section 5.7. In $chrom_{old}$, nrn_{orig} is used as an auxiliary neuron that constantly consumes and produces one spike regardless of whether its pre-synaptic neuron spikes or not. $chrom_{new}$ utilizes both nrn_{orig} and nrn_{dupl} in the same way. A similar mechanism can be found in one of Păun's SN P Systems in [22]. The system is for "Computing a Boolean function of three variables".

The procedure works by ensuring that both nrn_{orig} and nrn_{dupl} have the same amount of spikes at all times, and the nature of the self-loop is preserved by adding a two-way connection between the neurons. Figure 6 shows a simple application of this procedure.

6 Experiments and Results

In this section, we discuss the experiment setup and results of this study.

6.1 Input Design

As discussed in section 4, the genetic algorithm framework is designed to transform an initial SN P System Π_{init}. The GA framework can only accept deterministic SN P systems, thus 7 in total were designed to approximate the *binary addition* and *binary subtraction* functions, and will serve as Π_{init}'s.

There are three categories of Π_{init}'s: *baseline*, *original*, and *adversarial*. See table 2 for the general description of each Π_{init}. Figures 8a to 10b provide visual representations for each, and are direct outputs from the Graphviz library[7] used with Python. Note that figures 1 and 8a both represent the same SN P system.

Table 2: Characteristics of the initial SN P Systems, Π_{init}'s, used in the experiments.

	Binary Addition			Binary Subtraction			
	Baseline	Original	Adversarial	Baseline	Original	Original v2	Adversarial
Precision	100%	75.138375%	31.05496%	100%	38.60768%	26.04025%	60.45469%
Number of Neurons	5	11	8	8	32	33	16
Number of Synapses	10	14	19	15	38	41	59

6.1.1 Baseline category The Π_{init}'s under this category are the smallest designs with the closest approximations. Figures 8a and 8b show the baseline Π_{init}'s for binary addition and binary subtraction, respectively.

6.1.2 Original category The Π_{init}'s under this category were the first to be designed in this study, and are significantly larger and less precise than the ones under the baseline category. The set of neurons for the binary addition Π_{init} in this category is a superset of its baseline counterpart.

For binary subtraction, there are two Π_{init} versions in this category. After designing version 1, which we will refer to as *original v1*, we noticed that while it's set of neurons is larger than its baseline counterpart, it still lacked one. That is, given the set of neurons for *original v1* $O_1 = \{\sigma_{0,O_1}, ..., \sigma_{x,O_1}\}$ and for baseline $B = \{\sigma_{0,B}, ..., \sigma_{y,B}\}$, where $x > y$, we have: $B - B_n \subset O_1$ and $B_n \not\subset O_1$, where $B_n = \{\sigma_{n,O_1}\}$.

With this, a new version was designed, which we will refer to as *original v2*. In this version, the missing neuron is added to the *original v1* Π_{init} by directly adding synapses from all input neurons and a synapse to the output neuron, making it an *active*[8] neuron. The hypothesis is that the average precision of

[7] `https://pypi.org/project/graphviz/`
[8] See section 5.5

original v2 would be higher than *original v1*'s because the former contains all the neurons from the baseline, which already has a 100% precision as seen in table 2.

Figures 8c show the baseline Π_{init} for binary addition. Figures 9a and 9b show the baseline Π_{init}'s of versions 1 and 2 for binary subtraction, respectively.

6.1.3 Adversarial category The Π_{init}'s under this category follow the designs of their baseline counterparts but with m additional neurons, where m is the degree of the baseline Π_{init}[9], making the degree of the adversarial Π_{init} equal to $2m$. These added neurons, which contain pseudo-randomly generated rules and initial spikes, are connected pseudo-randomly.

The added neurons were allowed to have at least one up to five of the following rules: $a \to a$, $a^2/a \to a$, $a^3/a \to a$, $a^4/a \to a$, $a^5/a \to a$, $a/a \to \lambda$, $a^2/a \to \lambda$, $a^3/a \to \lambda$, $a^4/a \to \lambda$, and $a^5/a \to \lambda$. Note that by definition, if a firing rule is applicable, then no forgetting rule is applicable, and vice versa. Thus, in the implementation, it was made sure that if a firing rule with E_1 and a forgetting rule with E_2 were chosen to be in the same neuron, and $L(E_1) \cap L(E_2) \neq \emptyset$, only the firing rule is tagged as applicable and can be used by the neuron.

Figures 10a and 10b show the baseline Π_{init}'s for binary addition and binary subtraction, respectively.

6.2 Variants of the Genetic Algorithm Function

In section 5.6, we discussed the validations used by the GA function. We recall the two validations below which are used in *variant 1* and are changed in the succeeding variants:

1. The formation of self-loops is not allowed.
2. Inactive neurons must be deleted after the end of each outer loop.

Table 3: Variants of the Genetic Algorithm function The "self-loops" mentioned in this table refer only to the self-loops during evolution. Moreover, regardless of whether a GA function variant incorporates self-loops during evolution or not, its output will never have a self-loop. This is explained further in section 5.6. The deletion of inactive and disconnected neurons referenced in this table is only with regards to the transformation of the resulting chromosome after the current outer loop and before the beginning of a new one.

	Self-loops not allowed	Self-loops allowed	Inactive neurons are deleted	Disconnected neurons are deleted
Variant 1	✓		✓	
Variant 2	✓			✓
Variant 3		✓	✓	
Variant 4		✓		✓

Table 3 details the differences between the four variants of the GA function. There are two hypotheses:

1. Allowing self-loops during evolution would improve the precision of the resulting Π_{final}'s by allowing the addition of duplicate neurons that may be used to escape a local maximum. Thus, variants that allow self-loops, i.e. 3 and 4, would have higher average precision values than the other variants.[10]

[9] Refer to [22] for the formal definition of an SN P system.

[10] We leave the study on the correlation between the amount of self-loops within an SN P system and its precision for future work.

2. Deleting *disconnected neurons* instead of *inactive neurons* would improve the precision of the resulting Π_{final}'s as it would prevent the GA framework from deleting possibly important neurons that are inactive but still connected from the former $chrom_{final}$ which would be the new $chrom_{init}$.

6.3 Experiment Setup

We generated the set of fitness cases $S, |S| = 100$ for the binary addition and subtraction functions. The set of fitness cases used is the same for all of the Π_{init}'s for each function.

Table 4: Population size set for the initial SN P Systems, Π_{init}'s, used in the experiments.

Binary Addition			Binary Subtraction			
Baseline	Original	Adversarial	Baseline	Original	Original v2	Adversarial
15	40	40	20	80	80	40

The GA framework is run five times for each of the 7 Π_{init}'s that were designed (3 binary addition + 4 binary subtraction) to get an insight on the average behavior of each variant of the GA function. Since we have 4 variants of the GA function, that gives us a total of 140 experiments.

The following is a list of the default parameters:

1. Maximum number of inner loops, ρ: 10
2. Maximum number of generations per inner loop: 75
3. Mutation rate: 50%
4. Crossover rate: 30%
5. Selection rate: 60%
6. Population size: 80
7. Precision threshold: 100%

All of the default parameters were used consistently for all of the runs in the experiment, except for the **population size** parameter. This was set differently for each Π_{init} depending on its number of neurons, as can be seen in table 4.

6.4 Results and Discussion

After running the experiments described in section 6.3, we summarized the results for the binary addition function in table 5, the binary subtraction function in table 6, and the general results in table 7.

Below is the definition for each descriptor used in the results tables:

1. *Average Precision.* This is the average precision of the 5 Π_{final} that were outputted by the GA framework.
2. *Average Number of Neurons.* This is the average number neurons of the 5 Π_{final} that were outputted by the GA framework.
3. *Average Number of Synapses.* This is the average number synapses of the 5 Π_{final} that were outputted by the GA framework.
4. *Average Number of Outer Loops.* This is the average number of outer loops done by each of the 5 GA framework runs.
5. *Average Number of Inner Loops.* This is the average number of inner loops loops executed in each outer loop, done by each of the 5 GA framework runs.
6. *Average Number of Generations.* This is the average number of generations executed in each inner loop, executed in each outer loop, done by each of the 5 GA framework runs.

7. *Average Runtime (s).* This is the average runtime in seconds of each inner loop, executed in each outer loop, done by each of the 5 GA framework runs.

Comparing the data between tables 2 and 5, it can be seen that the GA framework introduced in section 4 was able to successfully produce a Π_{final}'s of similar structure with the *baseline* Π_{init} for binary addition, with 5 neurons and 10 synapses. It was also able to maintain the initial precision of 100%.

As for the *baseline* Π_{init} for binary subtraction, we turn to table 6 and see that the GA framework was also able to maintain the initial precision of 100%. It was able to produce Π_{final}'s with an unexpectedly better topology, containing an average of 7.15 neurons and 14.15 synapses, as opposed to the 8 neurons and 15 synapses of the input.

The GA framework significantly improved the *adversarial* Π_{init}'s for both binary addition and subtraction, having an average final precision of 100% from the initial precision of 31.05496% and 60.45469%, respectively. It also reduced the initial number of neurons from 8 to an average of 5.1 for binary addition, and from 16 to an average of 7.15 for binary subtraction. Lastly, it also reduced the initial number of synapses from 19 to an average of 10.4 for binary addition, and from 59 to an average of 14.45 for binary subtraction.

For the *original* category, the GA framework did have better results, although not as significant as the two previous categories.

The precision for the binary addition went up from 75.13838% to an average of 87.57182%. The number of neurons went down from 11 to an average of 5.2, and the number of synapses from 14 to an average of 9.85.

For binary subtraction, *original v1* had an initial precision of 38.60768% which went up to an average of 62.54856%, while *original v2's* precision went from 26.04025% up to an average of 57.77093%. The number of neurons and synapses for *original v1* went down from 32 to an average of 13.2, and from 38 to an average of 16.65, respectively. As for *original v2*, its number of neurons and synapses went down from 33 to an average of 6.75, and from 41 to an average of 9.5, respectively.

The average precision of *original v2* is lower than that of *original v1's*, the size of its topology was significantly reduced. This goes against the hypothesis that *original v2* would have a higher average precision (i.e. closer to *baseline*'s) because it contains all of the neurons from the *baseline* category, which has a 100% precision. Given that the average size of the resulting topologies from *original v2* is significantly smaller than that of *original v1*'s, the theory thus far is that the former kept on getting trapped in one of its local maximum in terms of precision. Since the precision can no longer be increased and the only other criteria for evolution is the size of the topology, the GA framework started shrinking it down while maintaining the same precision. Due to the time constraints of this study, however, as well as our scope's limitations, we cannot test this theory and know why the GA framework is possibly being trapped at a local maximum by replicating the experiments.

The results from tables 5 and 6 show that all variants of the GA function were able to significantly improve the average precision and size of the topology for each input Π_{init}.

The summarized results in table 7 show that *variant 3* yielded the highest average precision of 89.39876%, with an average of 6.65714 neurons and 11.88571 synapses. It is a close second to *variant 2* in terms of topology size, having only a difference of 0.17143 and 0.45714 in average neurons and synapses, respectively. Furthermore, only *variant 3* was able to evolve Π_{init} from the *original v1* category of the binary subtraction function to get a 100% precision in its fifth run. Figure 11 provides a visual representation for this SN P system.

Given the above information, as well as the fact that *variant 3* significantly produced more precise SN P systems compared to *variant 2* with a difference of 4.71486% in average precision, the former is considered to produce the best results.

Variant 3 allows self-loops during evolution along with *variant 4*, and these variants produced higher average precision values compared to *variant 1* and *variants 2*, respectively. The former variants were also able to achieve 100% precision for the *original* category of the *binary addition* function. In other words they increased the precision with average of 24.86163%, while the latter variants only increased the precision with an average of 0.00526%. These results support the hypothesis that allowing self-loops during evolution can improve precision by allowing the addition of duplicate neurons that may be used to escape a local maximum of the system.

Considering that *variants 1 and 3* of the GA function are set to delete inactive neurons, the results show that this configuration yielded higher average precision values compared to *variants 2 and 4*, respectively. This doesn't support the hypothesis that deleting disconnected neurons instead would result to a higher average precision, as it would prevent the GA framework from deleting possibly important neurons that are inactive but still connected in the Π_{init} used in the current outer loop. This finding is subject to future experimentation since it is not part of the scope and limitations of this study.

The results are inconclusive as to how allowing self-loops during evolution, deleting inactive neurons, or deleting disconnected neurons affect the average size of the resulting topologies.

7 Final Remarks

The use of genetic algorithms is an effective way to transform an initial spiking neural P system (SN P system), Π_{init}. Evolving the topology of Π_{init} does not only lessen the number of its neurons and synapses, but also helps it achieve a higher precision.

The results also show that allowing self-loops helped in evolving the SN P systems, as it acts as a way to add duplicate neurons that might be used by the system to escape one of its local maximum for precision. The algorithm introduced in section 5.7 transformed the SN P systems back to a valid format

successfully, i.e. without self-loops, after incurring self-loops during evolution. It was also shown that deleting inactive neurons yielded higher average precision values over deleting disconnected neurons.

Out of the four variants of the GA function design, *variant 3*, which was configured to allow self-loops during evolution and delete inactive neurons, produced the overall best results.

The results are inconclusive as to how allowing self-loops during evolution, deleting inactive neurons, or deleting disconnected neurons affect the average size of the resulting topologies.

The experiments done in this study were mainly focused on the design of the GA framework and comparing the effects of using different validations. The following are some recommendations for future studies that may be done using the genetic algorithm framework introduced in section 4.

1. Investigate the use of different values for parameters such as the mutation, selection, and crossover rates;
2. Try other ways of implementing the selection operator. Here are a couple of recommended approaches: 1. the *Roulette Wheel* approach used in [7] and [16], and; 2. *remainder stochastic sampling* mentioned in section 2.2 [27];
3. Try other ways of implementing the mutation operator. Weights could be applied such that each of the mutation methods have different chances of being chosen. More mutation operators could also be incorporated, such as one that adds neurons;
4. Try other ways of implementing the crossover operator. An n-point crossover could be applied like 5-point crossover implemented in [7]. It can also be implemented by forming a child chromosome that contains either the intersection or union of the sets of neurons and synapses from its parents [25];
5. Create a derived GA framework from the one introduced here which does not require an initial SN P System Π_{init} to produce its initial population;
6. Include the rules in evolving SN P Systems, not just the topology;
7. Test the use of other viable algorithms for the evaluation function;
8. Figure out if and why the GA framework is getting trapped at a local maximum. This can be replicated by running one of its variants and using the *binary subtraction* function's Π_{init} in the *original v2* category as input. Use the same fitness cases, as well as the parameter values enumerated in section 6.3;
9. Improve the GA framework by adding the functionality to evaluate Π_{init} before inputting it to the GA function for evolution.

As mentioned in section 5.4, the optimisation of the GA framework by using a *generalized suffix tree* to find the *longest common substring* is left for future work. The study on the correlation between the amount of self-loops within an SN P system and its precision is left for future work, as well.

Acknowledgements

I.C.H. Macababayao, R.T.A. de la Cruz, F.G.C. Cabarle, and H.N. Adorna acknowledge support by grants from the DOST-ERDT project. F.G.C. Cabarle ac-

knowledges support from the Dean Ruben A. Garcia PCA AY2018–2019, an RLC AY2018–2019 grant, and Project No. 191904 ORG (2019–2020) of the OVCRD in UP Diliman. H.N. Adorna is supported by the Semirara Mining Corp Professorial Chair for Computer Science, and RLC grant from UPD OVCRD. M.A. Martínez-del-Amor acknowledges the support of the research project TIN2017-89842-P (MABICAP), co-financed by *Ministerio de Economía, Industria y Competitividad (MINECO)* of Spain, through the *Agencia Estatal de Investigación (AEI)*, and by *Fondo Europeo de Desarrollo Regional (FEDER)* of the European Union. The work was supported by the National Natural Science Foundation of China (Grant Nos. 61472333, 61772441, 61472335, 61672033, 61425002, 61872309, 61771331), Project of marine economic innovation and development in Xiamen (No. 16PFW034SF02), Natural Science Foundation of the Higher Education Institutions of Fujian Province (No. JZ160400), Natural Science Foundation of Fujian Province(No. 2017J01099), Basic Research Program of Science and Technology of Shenzhen (JCYJ20180306172637807).

Bibliography

[1] Baker, J.E.: Adaptive selection methods for genetic algorithms. In: Proceedings of an International Conference on Genetic Algorithms and Their Applications. pp. 101–111. Hillsdale, New Jersey (1985)

[2] Carandang, J., Villaflores, J.M.B., Cabarle, F.G.C., Adorna, H.N., Martinez-del Amor, M.A.: Cusnp: Spiking neural p systems simulators in cuda. Romanian Journal of Information Science and Technology 20(1), 57–70 (2017)

[3] Ciobanu, G., Păun, G., Pérez-Jiménez, M.J.: Applications of Membrane Computing, vol. 17. Springer (2006)

[4] David, O.E., Greental, I.: Genetic algorithms for evolving deep neural networks. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. pp. 1451–1452. ACM (2014)

[5] De Jong, K.A.: Analysis of the behavior of a class of genetic adaptive systems. dept. Computer and Communication Sciences, University of Michigan, Ann Arbor (1975)

[6] Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, vol. 53. Springer (2003)

[7] Eskandari, E., Ahmadi, A., Gomar, S., Ahmadi, M., Saif, M.: Evolving spiking neural networks of artificial creatures using genetic algorithm. In: Neural Networks (IJCNN), 2016 International Joint Conference on. pp. 411–418. IEEE (2016)

[8] Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: From architectures to learning. Evolutionary Intelligence 1(1), 47–62 (2008)

[9] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989)

[10] Goldberg, D.E.: A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. Complex Systems 4(4), 445–460 (1990)

[11] Groß, M.: Molecular computation. In: Gramß, T., Bornholdt, S., Groß, M., Mitchell, M., Pellizzari, T. (eds.) Non–Standard Computation. Wiley–VCH, Weinheim (1998)

[12] Gusfield, D.: Algorithms on Strings, Trees and Sequences: Computer Science and Computational biology. Cambridge University Press (1997)

[13] Haupt, R.L., Haupt, S.E.: Practical Genetic Algorithms, vol. 2. Wiley New York (1998)

[14] Holland, J.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Application to Biology. University of Michigan Press (1975)

[15] Ionescu, M., Păun, G., Yokomori, T.: Spiking neural p systems. Fundamenta Informaticae 71(2, 3), 279–308 (2006)

[16] Khan, Q.S.U., Li, J., Zhao, S.: Training deep autoencoder via vlc-genetic algorithm. In: International Conference on Neural Information Processing. pp. 13–22. Springer (2017)
[17] Leporati, A., Zandron, C., Ferretti, C., Mauri, G.: On the computational power of spiking neural p systems. Proceedings of the Fifth Brainstorming Week on Membrane Computing, 227-245. Sevilla, ETS de Ingeniería Informática, 29 de Enero-2 de Febrero, 2007 (2007)
[18] Martin-Vide, C., Pazos, J., Păun, G., Rodríguez-Patón, A.: A new class of symbolic abstract neural nets: Tissue p systems. In: International Computing and Combinatorics Conference. pp. 290–299. Springer (2002)
[19] Miller, G.F., Todd, P.M., Hegde, S.U.: Designing neural networks using genetic algorithms. In: ICGA. vol. 89, pp. 379–384 (1989)
[20] Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press (1999)
[21] Păun, G.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108–143 (2000)
[22] Păun, G.: Spiking neural p systems. a tutorial. Bulletin of the European Association for Theoretical Computer Science (2007)
[23] Reeves, C.: Genetic algorithms. In: Handbook of Metaheuristics, pp. 55–82. Springer (2003)
[24] Schaffer, J.D.: Some effects of selection procedures on hyperplane sampling by genetic algorithms. Genetic Algorithms and Simulated Annealing pp. 89–103 (1987)
[25] Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
[26] Whitely, D.: The genitor algorithm and selective pressure proceedings of the 3rd international conference on genetic algorithms (1989)
[27] Whitley, D.: A genetic algorithm tutorial. Statistics and Computing 4(2), 65–85 (1994)
[28] Zhang, G., Gheorghe, M., Pan, L., Pérez-Jiménez, M.J.: Evolutionary membrane computing: A comprehensive survey and new results. Information Sciences 279, 528–551 (2014)

Appendix A P-Lingua File Example

bin-add_baseline.pli

```
1:  @mu = in1, in2, 1, aux1, aux2, out
2:  @marcs = (in1, 1)
3:  @marcs += (in1, aux1)
4:  @marcs += (in1, aux2)
5:  @marcs += (in2, 1)
6:  @marcs += (in2, aux1)
7:  @marcs += (in2, aux2)
8:  @marcs += (aux1, aux2)
9:  @marcs += (aux2, aux1)
10: @marcs += (aux2, 1)
11: @marcs += (1, out)
12: @min = in1, in2
13: @mout = out
14: [a → a]'in1
15: [a → a]'in2
16: [a → a]'1
17: [a * 2 → #]'1
18: [a * 3 → a]'1
19: [a → #]'aux1
20: [a * 2 → a]'aux1
21: [a * 3 → a]'aux1
22: [a → #]'aux2
23: [a * 2 → a]'aux2
24: [a * 3 → a]'aux2
```

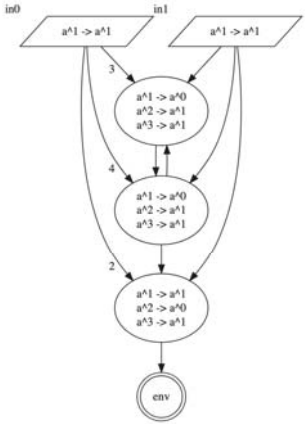
Fig. 7: P-Lingua format used for the SN P system in figure 1. The GA framework only parses the neuron, spike, synapse, and rule declarations in a P-Lingua file and ignores the rest of the lines.

Appendix B Evaluation Function Algorithm

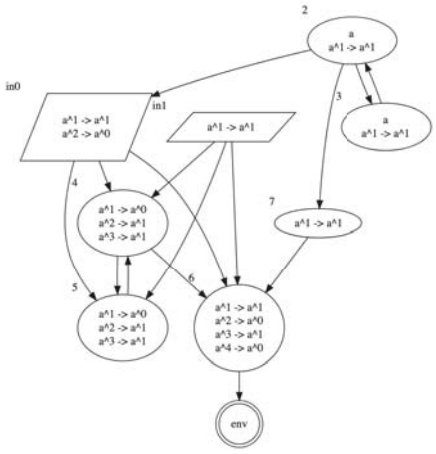
Algorithm 1 Evaluation function using LCS

```
1:  function GetLCSLength(actual_sptr, ideal_sptr)
2:      LCS ← empty string
3:      while len(actual_sptr) > len(LCS) do
4:          suffix ← actual_sptr
5:          if len(LCS) > 0 then
6:              prefix ← suffix[0 : len(LCS)]
7:              suffix ← suffix[len(LCS) :
                    len(suffix)]
8:          else
9:              prefix ← suffix[0 : 1]
10:             suffix ← suffix[1 : len(suffix)]
11:         while prefix in ideal_sptr and
                len(suffix) ≠ 0 do
12:             prefix.append(suffix[0])
13:             suffix ← suffix[1 : len(suffix)]
14:         if prefix not in ideal_sptr then
15:             prefix ← prefix[0 : len(prefix) − 1]
16:         if len(prefix) > len(LCS) then
17:             LCS ← prefix
18:         actual_sptr ← actual_sptr[1 :
                len(actual_sptr)]
19:     return len(LCS)
20: function GetPrecision(actual_sptr, ideal_sptr)
21:     LCSLength ← GetLCSLength(actual_sptr,
            ideal_sptr)
22:     precision ← LCSLength/len(ideal_sptr)
23:     return precision
```

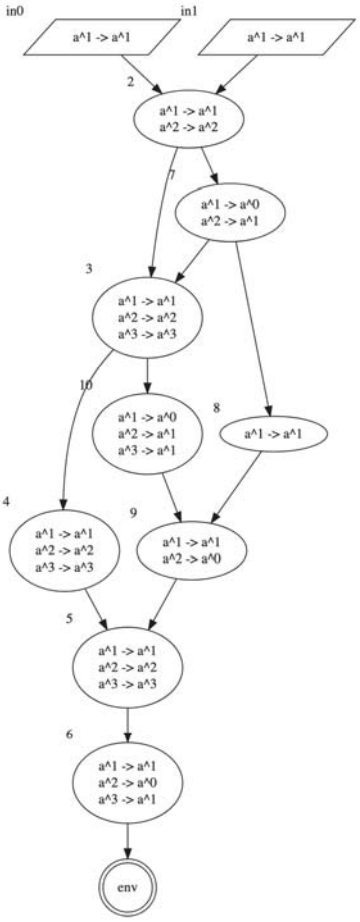
Appendix C Initial SN P Systems



(a) Π_{init} for the *binary addition* function: *baseline* category.

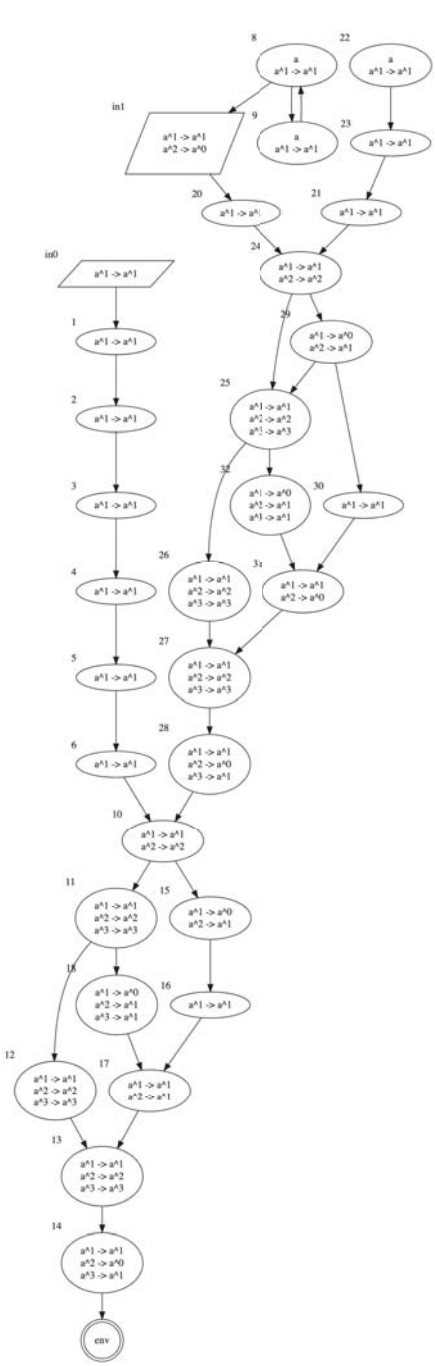


(b) Π_{init} for the *binary subtraction* function: *baseline* category.

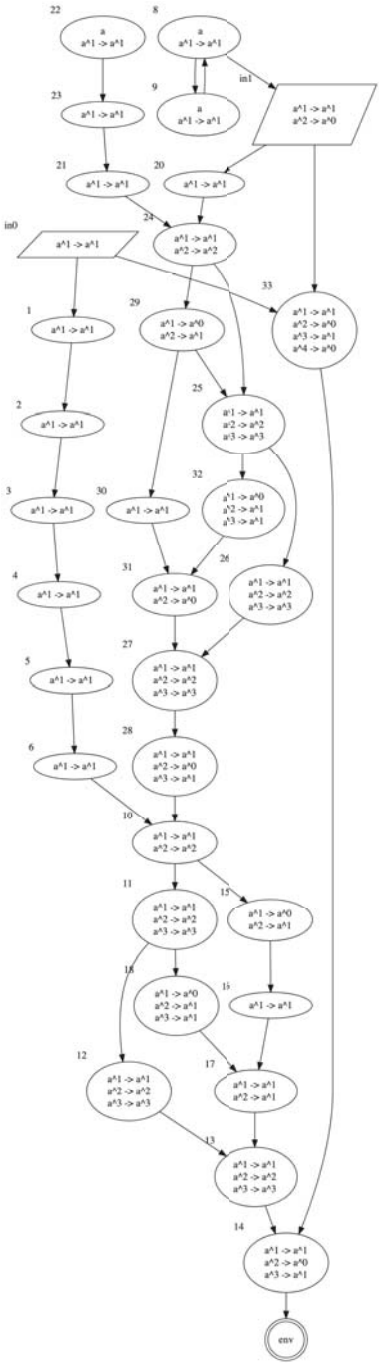


(c) Initial SN P System Π_{init} for the *binary addition* function: *original* category.

Fig. 8: Initial SN P systems Π_{init}'s under the baseline category and the *binary addition* Π_{init} under the *original* category.

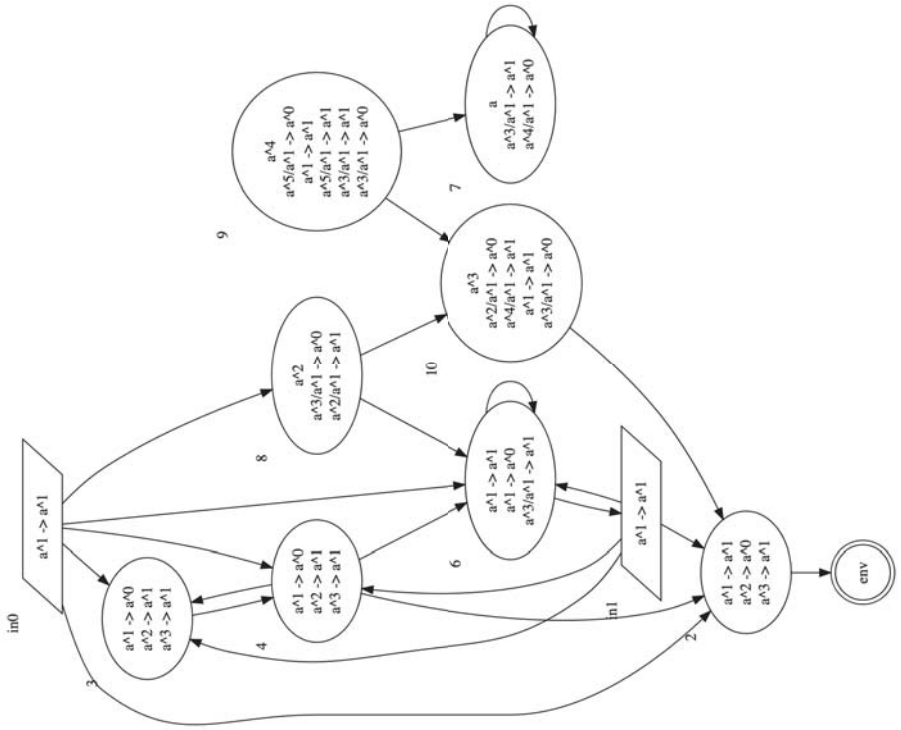


(a) Π_{init} for the *binary subtraction* function: *original, version 1* category.

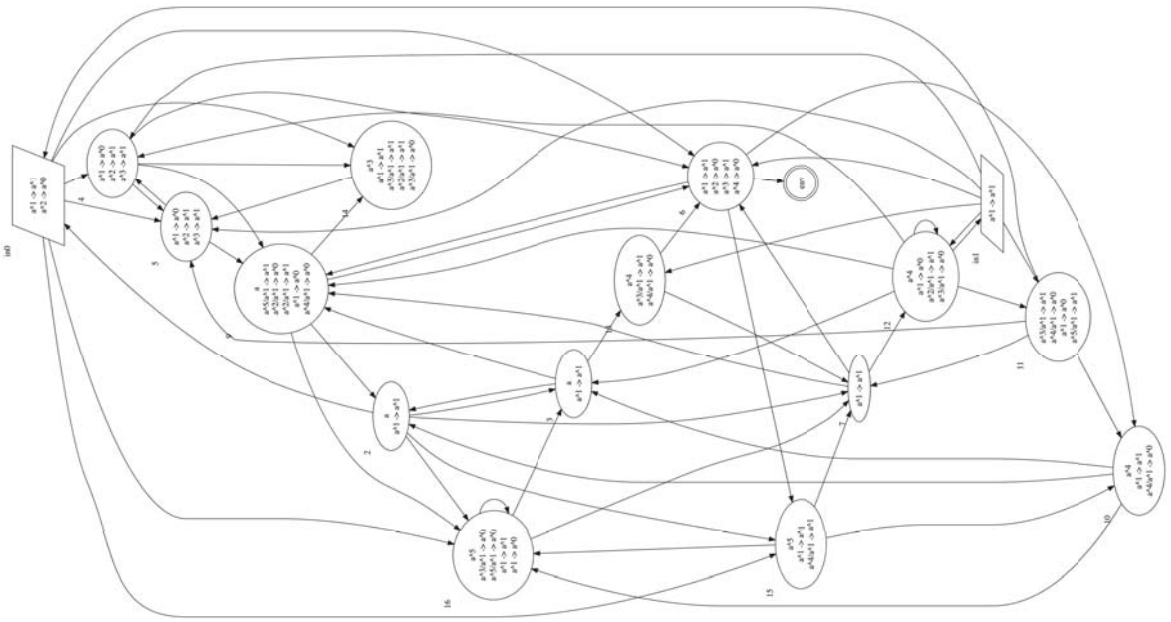


(b) Π_{init} for the *binary subtraction* function: *original, version 2* category.

Fig. 9: Two (2) versions of initial SN P systems Π_{init}'s for *binary subtraction* under the original category.



(a) Initial SN P System Π_{init} for the *binary addition* function: *adversarial* category.



(b) Initial SN P System Π_{init} for the *binary subtraction* function: *adversarial* category.

Fig. 10: Initial SN P systems Π_{init}'s under the adversarial category.

Appendix D Final SN P System Example

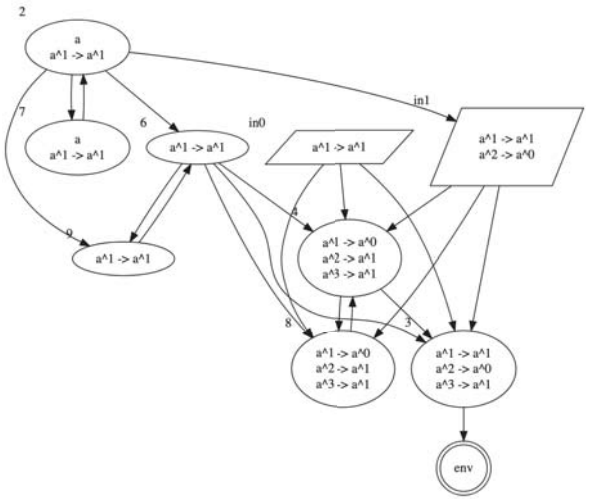


Fig. 11: Final SN P System Π_{final} outputted in the fifth run of the GA framework using variant 3 of the GA function. The Π_{init} for the *binary subtraction* function in the *original v1* category was supplied as input. This has a precision of 100%.

Appendix E Experiment Results in Tabular Form

Table 5: Binary Addition Results

		Variant 1	Variant 2	Variant 3	Variant 4
Baseline	Ave. Precision	100%	100%	100%	100%
	Ave. Number of Neurons	5	5	5	5
	Ave. Number of Synapses	10	10	10	10
	Ave. Number of Outer Loops	1	1	2	2
	Ave. Number of Inner Loops	1	1	1	1
	Ave. Number of Generations	75	75	38	38
	Ave. Runtime (s)	127.3124	133.79615	51.58424	50.17064
Original	Ave. Precision	75.13837%	75.1489%	100%	100%
	Ave. Number of Neurons	5.4	5.4	5	5
	Ave. Number of Synapses	9.8	9.6	10	10
	Ave. Number of Outer Loops	4.8	5.4	5.6	5.6
	Ave. Number of Inner Loops	3.91	3.453333	1.17714	1.19714
	Ave. Number of Generations	34.11065	28.31717	25.98381	25.02762
	Ave. Runtime (s)	182.77748	260.15345	119.30703	115.56649
Adversarial	Ave. Precision	100%	100%	100%	100%
	Ave. Number of Neurons	5.4	5	5	5
	Ave. Number of Synapses	11.6	10	10	10
	Ave. Number of Outer Loops	4	4	5	4.4
	Ave. Number of Inner Loops	1	1	1	1
	Ave. Number of Generations	34.73667	26.71	20.48	20.07
	Ave. Runtime (s)	171.00478	139.75218	84.54192	77.45007

Table 6: Binary Subtraction Results

		Variant 1	Variant 2	Variant 3	Variant 4
Baseline	Ave. Precision	100%	100%	100%	100%
	Ave. Number of Neurons	7	7	7.2	7.4
	Ave. Number of Synapses	14	14	14.2	14.4
	Ave. Number of Outer Loops	2	2	3.8	3.6
	Ave. Number of Inner Loops	1	1	1	1
	Ave. Number of Generations	38	38	20.73334	21.96667
	Ave. Runtime (s)	124.28844	150.43626	47.09733	56.51845
Original v1	Ave. Precision	62.5326%	59.6779%	68.45752%	59.5262%
	Ave. Number of Neurons	14	9	11.6	18.2
	Ave. Number of Synapses	11.8	11.8	18	25
	Ave. Number of Outer Loops	13.4	16.6	15.6	13.8
	Ave. Number of Inner Loops	1.88721	1.88285	1.66441	2.00065
	Ave. Number of Generations	23.39129	17.58746	25.68916	24.57917
	Ave. Runtime (s)	1114.59989	1129.22675	1047.03467	1684.42416
Original v2	Ave. Precision	57.56825%	57.96051%	57.33382%	58.22112%
	Ave. Number of Neurons	5	6.8	5.8	9.4
	Ave. Number of Synapses	6.4	9.6	7	15
	Ave. Number of Outer Loops	14.2	16	16.4	17.8
	Ave. Number of Inner Loops	1.69372	1.63318	1.57468	1.81486
	Ave. Number of Generations	12.49539	14.73186	9.73649	20.37585
	Ave. Runtime (s)	144.49061	299.61667	166.09787	465.59094
Adversarial	Ave. Precision	100%	100%	100%	100%
	Ave. Number of Neurons	7.4	7.2	7	7
	Ave. Number of Synapses	14.8	15	14	14
	Ave. Number of Outer Loops	11.6	11.8	11.8	12.4
	Ave. Number of Inner Loops	1	1	1	1
	Ave. Number of Generations	29.83683	30.82398	20.19793	23.5539
	Ave. Runtime (s)	323.39028	349.18024	141.60848	182.41118

Table 7: Average of all results for each GA function variant

	Variant 1	Variant 2	Variant 3	Variant 4
Ave. Precision	85.03418%	84.6839%	89.39876%	88.24962%
Ave. Number of Neurons	7.02857	6.48571	6.65714	8.14286
Ave. Number of Synapses	12.28571	11.42857	11.88571	14.05714
Ave. Number of Outer Loops	7.28571	8.11429	8.6	8.51429
Ave. Number of Inner Loops	1.64156	1.56705	1.20232	1.28752
Ave. Number of Generations	35.36726	33.02435	22.97439	24.79617
Ave. Runtime (s)	312.55199	351.73739	236.75308	376.01885

A Framework for Evolving Spiking Neural P Systems with Rules on Synapses

Celine Anne A. Moredo[1], Ryan Chester J. Supelana[1], Dionne Peter Cailipan[1], Francis George C. Cabarle[1,2], Ren Tristan A. de la Cruz[1], Henry N. Adorna[1], Xiangxiang Zeng[3], Miguel Ángel Martínez-del-Amor[4]

[1]Algorithms & Complexity, Dept. of Computer Science,
University of the Philippines Diliman
Diliman 1101 Quezon City, Philippines.
[2]Shenzhen Research Institute of Xiamen University
Xiamen University, Shenzhen 518000, Guangdong, China.
[3]School of Information Science and Engineering
Hunan university 410082, Changsha, China.
[4]Research Group on Natural Computing, Dept. Computer Science and AI,
University of Seville, Seville, Spain

Abstract. In this paper, we present a genetic algorithm framework for evolving Spiking Neural P Systems with rules on synapses (RSSNP systems, for short). Starting with an initial RSSNP system, we use the genetic algorithm framework to obtain a derived RSSNP system with fewer resources (fewer and simpler rules, fewer synapses, less initial spikes) that can still produce the expected output spike trains. Different methods in the selection of parents and in the calculation of fitness are incorporated. We also try the framework on 5 RSSNP systems that compute bitwise AND, OR, NOT, ADD, and SUB respectively to gather data on how the framework behaves. Lastly, we discuss the asymptotic complexity of the algorithm and its effectiveness in generating fitter RSSNP systems based on which methods were used.

Keywords: Membrane Computing · Spiking Neural P Systems · Genetic Algorithm.

1 Introduction

People encounter different types of problems every day and in order to progress through their lives, and they need to learn how to solve those problems. Although most everyday problems, like cooking or fixing your bed, can be solved without the means of a computer, many classic problems in computer science are continuing to be solved with new algorithms or computing models.

Computer scientists from around the world have proposed several unique solutions (or algorithms) to solve computationally hard problems such as the *traveling salesman problem*. Many of them base their algorithms using the Turing Machine, which our modern computers took inspiration from. While many of

these are elegant, it is, more often than not, difficult and time-consuming to construct them. Having a system that could design a minimized version of a candidate solution could let us easily save a significant amount of resources when solving problems.

[13] started the study of incorporating the nature of cells into computing and calls this field as **Membrane Computing**. His work catapulted numerous different studies on the field, and eventually, [7] introduced a new model of computation that is Turing-complete, the **Spiking Neural P System**. This mathematical model is interesting because it closely resembles the third generation of neural networks, Spiking Neural Networks, which is commonly used for many applications in Machine Learning. It has also been shown that a variant of this new model, specifically Spiking Neural P Systems with Neuron Division and Budding can be used to solve the SAT problem by creating an exponential workspace in linear time [12].

Many Spiking Neural P Systems have already been proposed to solve some of the problems today. As mentioned above, if it is possible to design a simpler solution (in this case, a Spiking Neural P System) to reduce the amount of resources needed to solve a problem, we may be able to save a significant amount of time when using applications that use this model.

In this work, we will first discuss the variation of the SNP systems called Spiking Neural P Systems with rules on synapses (RSSNP systems, for short). We will then discuss the genetic algorithm framework, and how we applied it to the RSSNP systems. This includes our method of initializing the population, calculating the fitness, selecting the parents, crossover methods and mutation rate, offspring validity checking, and termination. Next, we discuss our implementation and experiments. Lastly, we show the results, our analysis, and the discussion on possible future works. We note that ideas from this work were first given in a presentation during BWMC2018 in https://www.gcn.us.es/files/bwmc2018-evolsnp-present.pdf.

2 Preliminaries

2.1 Spiking Neural P Systems with Rules on Synapses

For this study, we will be dealing with a specific type of SNP system called **Spiking Neural P Systems with Rules on Synapses** (**RSSNP**) [15]. The main difference between SNP systems and RSSNP systems is that the rules are on the synapses and not on the neurons, which allows the system to use different mechanisms for the spike consumption.

Definition 1 (RSSNP system). *A Spiking Neural P System with Rules on Synapses, of degree $m \geq 1$, is a construct of the form*

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, in, out),$$

where:

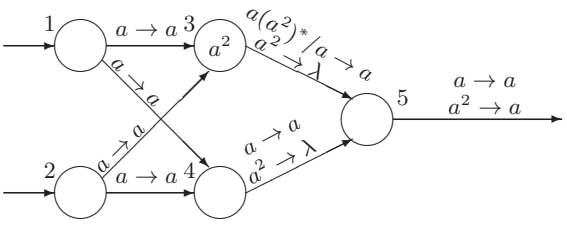


Fig. 1: An RSSNP system Π_{XOR} that simulates the XOR gate.

1. $O = \{a\}$ is the singleton alphabet that is also called spike;
2. $\sigma_1, \ldots, \sigma_m$ are the neurons of form $\sigma_i = (n_i)$, where $n_i \geq 0$ is the initial number of spikes contained in σ_i, $1 \leq i \leq m$;
3. syn, the set of synapses; each element is a pair $((i,j), R_{(i,j)})$, where:
   (a) (i,j) indicates the indices of the neurons the synapse is connecting with $1 \leq i,j \leq m$, and $i \neq j$;
   (b) $R_{(i,j)}$ is the set of rules on the synapse, which can be of the following forms:
      i. $E/a^c \to a^p; d$ where E is a regular expression over O, $c \geq p \geq 1$, $d \geq 0$;
      ii. $a^s \to \lambda$ where $s \geq 1$ and for each rule $E/a^c \to a^p; d$, of type i. from any $R_{(i,j)}$, $a^s \notin L(E)$;
4. $in, out \subseteq \{1, 2, ..., m\}$ indicate the sets of the input and output neurons, respectively.

Extended spiking rules are of the form $E/a^c \to a^p; d$ with $p \geq 1$. If $p = 1$, it is called a standard spiking rule. If $L(E) = a^c$, the rule can be written as $a^c \to a^p; d$, or if $d = 1$, the rule can be simplified by not writing d.

Spiking rules are applied as follows: a rule is enabled if a synapse is connected to a neuron σ_i that contains k spikes where $k \geq c$ and $a^k \in L(E)$. By applying the rule, c spikes are consumed, so only $(k - c)$ spikes are now contained. There are cases in which a neuron satisfies rules on different synapses, and with it, different processing mechanisms can be applied. For $E_1/a^{c_1} \to a^{p_1}$ on synapse (i,j) and $E_2/a^{c_2} \to a^{p_2}$ on synapse (i,k), assuming σ_i contains k spikes, where $a^k \in L(E_1) \cap L(E_2), p_1 \leq c_1 \leq k, p_2 \leq c_2 \leq k$:

1. If $c_1 = c_2 = c$, equal spike consumption strategy is used, so only c spikes are consumed, and p_1 and p_2 spikes are generated in (i,j) and (i,k), respectively. Note that it doesn't matter if $c_1 + c_2 > k$;
2. If $c_1 \neq c_2$, maximum spike consumption strategy is used, so $max(c_1, c_2)$ spikes are consumed, and p_1 and p_2 spikes are generated in (i,j) and (i,k), respectively.

Rules of the form $a^s \to \lambda$ with $s \geq 1$ are called forgetting rules and are applied as follows: if neuron σ_i contains exactly s spikes, and no firing rule is enabled, the rule is used and s spikes are consumed.

Similar to SNP systems, a global timer exists which marks the time for each neuron and synapse. For cases wherein a neuron satisfies two different rules on the same synapse, the system will non-deterministically choose which of the enabled rules to apply.

A configuration of a system is defined to be the number of spikes in each neuron and the number of steps until each synapse becomes open, so the initial configuration of a system is $\langle n_1, n_2, \ldots, n_m, 0, 0, \ldots, 0\rangle$ where all synapses are open and $n_1, n_2, \ldots, n_m$ is the initial number of spikes contained in neurons $\sigma_1, \sigma_2, \ldots, \sigma_m$.

A computation is a series of transitions starting from the initial configuration. A computation ends when the system reaches a configuration in which no rule can be applied on any synapse. The result of a computation can be defined in various ways —it can be the number of spikes sent to the environment, the time interval between two spikes, etc.

The system may or may not have an input neuron or output neuron defined. In doing so, we obtain a system working in the accepting or the generating mode, respectively.

Consider an RSSNP Π_{XOR} that simulates the XOR gate shown in Figure 1. We have $\sigma_1 = \sigma_2 = \sigma_4 = \sigma_5 = \{0\}$, $\sigma_3 = \{2\}$, $syn = \{((1,3),\{a \to a\}), ((1,4),\{a \to a\}), ((2,3),\{a \to a\}), ((2,4),\{a \to a\}), ((3,5),\{a(a^2)^*/a \to a, a^2 \to \lambda\}), ((4,5),\{a \to a, a^2 \to \lambda\}), ((5,env),\{a \to a, a^2 \to a\})\}$, $in = \{1,2\}$, and $out = \{5\}$.

As shown in Table 1, putting one spike each to neurons σ_1 and σ_2 at the beginning of the computation, Π_{XOR} operates as follows: At time step t_0, the rule $a^2 \to \lambda$ is activated due to the initial spikes in σ_3, but no spikes are sent to neurons nor to the environment. Due to the inputs, σ_1 and σ_2 contain 1 spike each and the rule $a \to a$ on synapses $(1,3), (1,4), (2,3), and (2,4)$ are activated at time step t_1. σ_3 and σ_4 contain 2 spikes each, which causes to activate the rule $a^2 \to \lambda$ on synapses $(3,5)$ and $(4,5)$ at time step t_2. Afterwards, the computation halts without sending any spike to the environment because there are no longer any rules that can be activated and all synapses are open.

If, however, Π_{XOR} is given one spike only to σ_1, it would operate as shown in Table 2: The rule $a^2 \to \lambda$ is activated due to the initial spikes in σ_3 and no spikes are still sent to neurons nor to the environment at time step t_0, but at time step t_1, only σ_1 will contain a spike, so the rule $a \to a$ will only be activated on synapses $(1,3)$ and $(1,4)$. At time step t_2, σ_3 and σ_4 contain one spike each, so the rule $a \to a$ on synapse $(4,5)$ and rule $a(a^2)^*/a \to a$ on on synapse $(3,5)$ will be activated. σ_5 receives one spike each from σ_3 and σ_4, so it will send 2 spikes to the environment due to the rule $a^2a^*/a^2 \to a$ on synapse $(5, env)$. The computation then halts because there are no longer any applicable rules and all synapses are open.

Upon studying the behaviour of this system, it can be observed that the spikes in σ_3 and the rule $a^2 \to \lambda$ from its outgoing synapse can together be omitted. This is to be noted as we will later on discuss simplifying RSSNP systems in the succeeding sections.

Time Step	σ_1	σ_2	σ_3	σ_4	σ_5
t_0	0	0	2	0	0
t_1	1	1	0	0	0
t_2	0	0	2	2	0

Table 1: Computation of Π_{XOR} Computing 1 XOR 1

Time Step	σ_1	σ_2	σ_3	σ_4	σ_5
t_0	0	0	2	0	0
t_1	1	0	0	0	0
t_2	0	0	1	1	0
t_3	0	0	0	0	2

Table 2: Computation of Π_{XOR} Computing 1 XOR 0

2.2 Genetic Algorithm

The Genetic Algorithm, which stems from the principles of genetics and natural selection, is a method for solving optimization and search problems. The genetic algorithm starts by generating a random population of candidate solutions (called individuals), with each iteration called a generation. Each individual has a set of properties or attributes that define it, called the chromosome. In each generation, a fitness function, defined at the start, is used to assess how fit each individual is. The population will then undergo the process of selection. Here only some of the individuals will be chosen. The probability of an individual being selected is based on its fitness score. The selected individuals will produce offsprings which will form the next generation through the process of crossover or reproduction in Biology. After the offspring have been created, there is a probability that the offspring will undergo mutation. The entire process is repeated until the stopping condition, which is defined at the start, has been met [10].

3 Problem Statement

Extensive research has already been conducted to artificial neural networks (ANN, for short) and likewise, the idea of an evolving ANN for optimizing its performance (faster execution times or higher accuracy in classification problems, etc.) is nothing new. [9] and [8] have proposed methods for encoding feed-forward neural networks, namely **direct** and **grammatical encoding**, to evolve them using genetic algorithm. Many similar studies like [6] and [16] use different methods but ultimately have the same goal.

It is a different case for RSSNP systems. The concept of performance in an RSSNP system has some differences from ANNs. The output of a deterministic RSSNP will always be the same if given the same input. Just like a mathematical function, we would prefer an RSSNP system that outputs the *expected* set of bits to the environment rather than random and unexpected bits. To quantify this

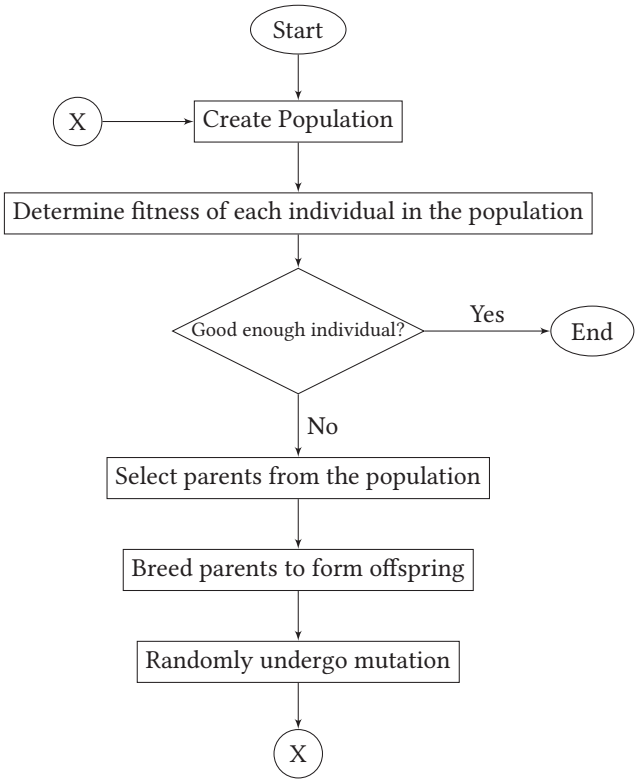


Fig. 2: General flowchart of a Genetic Algorithm

idea of performance, let there be an input and output pair (x, y). When an RSSNP accepts x and exactly produces y, then it has produced an expected output so it has good performance. If the output is not exactly y, then the RSSNP has a lower performance for each bit that is incorrect. Although RSSNP systems and ANNs are similar as both were inspired by the human brain, no formal study has been conducted to evolve RSSNP systems in a similar fashion to that of neural networks.

A possible first step to evolve RSSNP systems is by reducing the resources (neurons and the initial number of spikes in the system, rules and their parameters, synapses). An RSSNP system with fewer resources would most likely result in faster execution time since fewer steps will have to be taken before the system halts. Smaller RSSNP systems also consume less memory space which is another advantage. It might also be possible to find a desired RSSNP system that outputs the expected bitstrings within a larger RSSNP system.

In this study, we want to answer the question:

– How do we reduce a given RSSNP Π_{init} to an RSSNP with fewer resources Π'_{init}, but still consistently produces an expected output?

4 Objectives of the Study

The goal of the study is to design a **genetic algorithm** framework for reducing the number of *rules* and *synapses* of a given RSSNP system.

More specifically, the main goal is to create a framework wherein given:

1. a finite set of input-output spike train pairs S=$\{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ where each set of input spike train $a_i = \{\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ij}\}$ for $j = |in|$ and $\alpha_{ik}, b_i \in \{0, 1\}^+$ for $1 \leq k \leq j$,
2. an RSSNP system $\Pi_{init} = (O, \sigma_1, \ldots, \sigma_m, syn, in, out)$ that produces an output spike train that matches with b_i when fed with a_i with an error rate of ϵ.

We can obtain $\Pi_{final} = (O', \sigma'_1, \ldots, \sigma'_m, syn', in', out')$ with the following characteristics:

1. produces an output spike train that matches with b_i when fed with a_i with an error rate of ϵ' where $\epsilon' \leq \epsilon$.
2. $|syn'| \leq |syn|$
3. The total number of rules in Π_{final} is less than or equal to the total number of rules in Π_{init}
4. The maximum number of spikes consumed in any rule in Π_{final} is at most the maximum number of spikes consumed in any rule in Π_{init}

5 Scope and Limitations of the Study

The scope of the study will be limited to a specific type of P System called Spiking Neural P Systems with Rules on Synapses, also known as RSSNP, wherein the following restrictions would be used:

1. **Deterministic Systems**. A synapse in the system should not contain any two rules $R_1 = (E_1, c_1, p_1, d_1)$ and $R_2 = (E_2, c_2, p_2, d_2)$ such that $L(E_1) \cap L(E_2) \neq \emptyset$ and $R_1 \neq R_2$. It should also not contain any synapses $S_1 = ((i_1, j_1), R_{(i_1,j_1)})$ with rule $R_3 = (E_3, c_3, p_3, d_3)$ and $S_2 = ((i_2, j_2), R_{(i_2,j_2)})$ with rule $R_3 = (E_4, c_5, p_5, d_6)$ where $i_1 = i_2$, $j_1 \neq j_2$, $L(E_3) \cap L(E_4) \neq \emptyset$, and $c_3 \neq c_4$.
2. **Equal Spike Consumption**. The RSSNP we will be dealing with will consume equal amounts of spike when handling rules enabled on different synapses coming from the same neuron. Given $E_1/a^{c_1} \rightarrow a^{p_1}; d_1$ on synapse (i, j) and $E_2/a^{c_2} \rightarrow a^{p_2}; d_2$ on synapse (i, k), assuming σ_i contains n spikes, where $a^n \in L(E_1) \cap (E_2), p_1 \leq c_1 \leq n$, $p_2 \leq c_2 \leq n$, $d_1, d_2 \geq 0$, we will only handle cases in which $c_1 = c_2$.

3. **Constant Neurons**. The number of neurons in the system is constant such that no neuron is added or removed in the system while it is undergoing evolution.
4. **Synchronous mode**. The system operates under a global timer which marks the time for all neurons and synapses.
5. **Regular Expression**. The regular expression to be used when expressing a rule on a synapse should only be in the form of $a^i(a^j)^*$ where $i \geq 0$, $j \geq 0$ and $i + j \geq 0$. Rules of other forms will not be accepted.
6. **No delays**. The rules on the synapses in the system should not contain any delays. Given rules of the form $E/a^c \rightarrow a^p; d$ in the system, all rules should have $d = 0$.

6 Related Work

6.1 Evolving Neural Networks

[6] compares the results of using evolutionary programming to evolve neural networks against using the back-propagation algorithm. In the experiments with evolutionary programming, the networks had a fixed number of neurons and the edges connecting the neurons remained constant, but only evolved the weights of the network. His experiments included evolving multilayer perceptrons to solve the XOR problem and the Gasoline Blending problem. Evolutionary programming "solved" the XOR problem and Gasoline Blending problem before the 40th and 100th generation respectively, as compared to 240 and 400 to 500 generations (epochs) for back-propagation.

[10] suggests that it is possible to apply the genetic algorithm to evolve neural networks. One possible strategy was to apply a genetic algorithm to a fixed network and only evolving the weights as done by [11]. They represented an individual in the population as a vector containing the weights of the network when reading off from left to right and top to bottom. The fitness of an individual was determined by using its chromosomes (or properties) as the weights of the network and taking the sum of the squares of the errors after running the network on a specific training set. The crossover operation was done by selecting incoming links from parents randomly and copying those weights to the offspring. Another way to apply a genetic algorithm to evolve neural networks is done in [9]. A network was encoded as an adjacency matrix and offspring were created by selecting a random row index and swapping the corresponding rows between two parent matrices. This method for genetic algorithm allowed for neural networks with low error rate but its problems in performance are apparent in larger and more complex networks. [8] proposes another method for applying a genetic algorithm to a neural network that does not greatly affect performance for larger networks. The paper uses *grammatical encoding* wherein a set of rules, called a *grammar*, is used to generate a network.

6.2 Matrix Representation of RSSNP Systems

[2] presents a matrix representation of SNP systems in order to simulate these systems on graphics processing units (or GPU, for short) and further improved their work in [3] by implementing a computation simulation algorithm for SNP systems on GPUs.

[4] proposes an algorithm for simulating RSSNP systems. Their work includes an extension of the matrix representation mentioned previously as well as pseudocodes for simulating the RSSNP continuously during consecutive time steps. The algorithm is also able to handle non-determinism which will be useful for future research on the topic of this study.

7 Algorithm

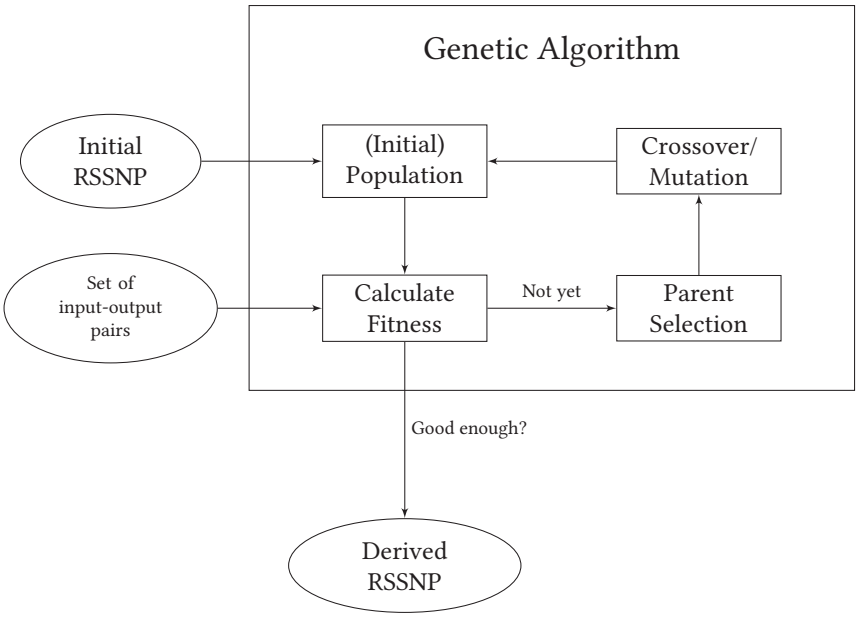


Fig. 3: Overview of the Genetic Algorithm Framework

The main framework will take an RSSNP system and a set of input-output spike train pairs as input. The genetic algorithm framework will then create an initial population of RSSNP systems that have resources less than or equal to the input RSSNP system as described in Section 7.1. The fitness of each RSSNP system in the population will be calculated by comparing the generated output of each system with the given input-output spike train pairs using the

methods in Section 7.2. A low error means high fitness. RSSNP systems will then be selected using methods described in Section 7.3 to reproduce. Selected RSSNP systems will be bread to create offspring which will replace the previous generation as stated in Section 7.4. Each offspring will then be mutated with a chance as described in Section 7.5. An overview of the framework is illustrated in Figure 3.

7.1 Initializing Population

Given an initial RSSNP $\Pi_{init} = \{O, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, in, out\}$ and population size *popsize*, we create *popsize* individuals by randomizing the resources of Π_{init} except rules connected to an input neuron or the output neuron. We can do this by applying the mutation function, which consists of the following operations, to Π_{init}:

1. **Delete rule**. For every rule in the system, there is a probability that it will be deleted.
2. **Change connected synapses**. For every synapse in the system, there is a probability that its source or destination neuron will change.
3. **Change the regular expression**. For every rule in the system, there is a probability that its regular expression will change.
4. **Decrease consumed spikes**. For every rule in the system, there is a probability that its consumed spikes will decrease.
5. **Decrease produced spikes**. For every rule in the system, there is a probability that its produced spikes will decrease.
6. **Decrease initial number of spikes**. For every neuron in the same, there is a probability that its initial number of spikes will decrease.

During this step only, each of the aforementioned operations has a 50% chance of being applied to a rule. In other words, the mutation rate is set to 50%.

For example, using the RSSNP shown in Figure 4, we can derive the following RSSNPs displayed in Figure 6. In Figure 5a, method (1) is applied and rule $a^2(a^2)^*/a^2 \to a$ on synapse $(6, 5)$ is deleted, while in Figure 5b, due to 2, the rule $a^3 \to \lambda$, which used to be on synapse $(4, 6)$, is now on synapse $(4, 7)$ because the synapse outgoing from σ_4 to σ_6 is now going outgoing from σ_4 to σ_6. Applying 3, the regular expression of the rule $a^2(a^2)^*/a^2 \to a$ on synapse $(6, 5)$ in Figure 5c changed from $a^2(a^2)^*$ to $a^2(a)^*$. Figures 5d and 6a show the same rule $a^2 \to a^2$ on synapse $(5, 4)$ being altered by methods (4) and (5), respectively. Method (6) is shown to be applied in Figure 6b where the initial number of spikes in σ_3 is 0.

Given Π_{init} has r rules, the time complexity of this step is $O(r \cdot popsize)$. The time it takes to mutate Π_{init} is $O(r)$ because applying each mutation method in 7.1 to a rule only takes $O(1)$, and it is repeated r times. Π_{init} is mutated *popsize* times to populate the population.

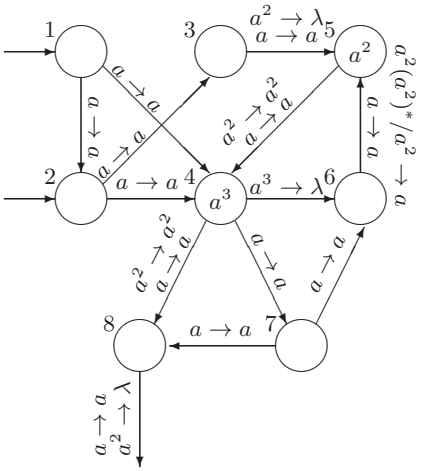


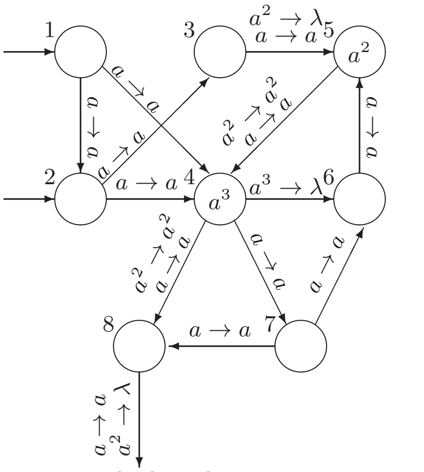
Fig. 4: RSSNP that does not compute anything

7.2 Calculating Fitness

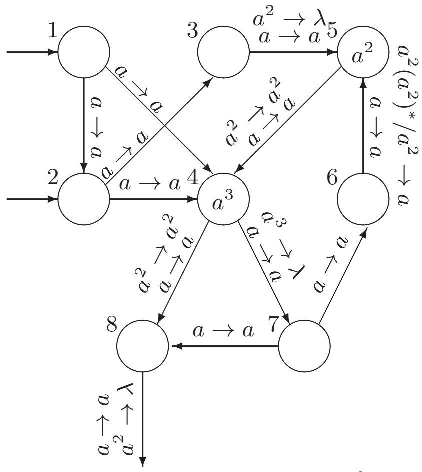
Once the population $P = \{\Pi'_1, \Pi'_2, \ldots, \Pi'_{popsize}\}$ is created and given a finite set of input-output spike train pairs $S = \{(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\}$ where each $a_i = \{\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ij}\}$ for $j = |in|$ and $\alpha_{ik}, b_i \in \{0,1\}^+$ for $1 \leq k \leq j$, we feed each a_i to every individual Π'_j and produce an output spike train b'_i, that is, we perform $\Pi'_j(a_i) = b'_i$ for every Π'_j in P. Once $|b'_i| = 3 \cdot |b_i|$[1], we stop the computation of $\Pi'_j(a_i)$. Then, we evaluate each generated output spike train b'_i produced by Π'_j. To do this, we compare b'_i with b_i using a string matching method f and the score of b'_i is the ratio of the output of f over the length of b_i or $score_i = \frac{f(b_i, b'_i)}{|b_i|}$. If the computation of $\Pi'_j(a_i)$ was halted or in other words, $b'_i \geq 3 \cdot |b_i|$, then $score_i$ is further halved. For this study, the following methods are selected as f:

1. **Longest Common Substring**. Returns the length of the longest spike train from b'_i that is a sub string of b_i. Using this method has the advantage of determining if the RSSNP is able to output the expected bits consecutively. If such RSSNP is found (can consecutively output all of the expected bits), then it will receive 100% fitness rating. Though, this method will generally give a lower fitness than Method (2) due to the problem having more constraints.
2. **Longest Common Subsequence**. Returns the length of the longest subsequence common to b'_i and b_i. Using this method allows us to see if the RSSNP can produce the expected bits in order, even if not consecutively.

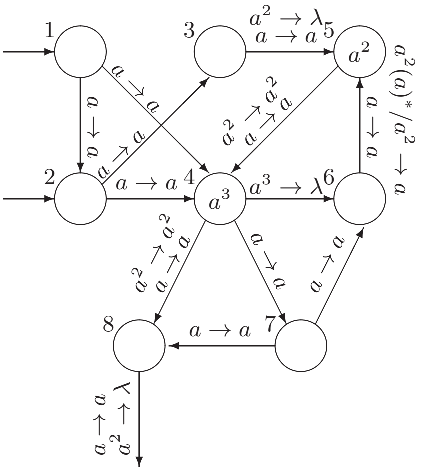
[1] It is to be noted that the upper bound $3 \cdot |b_i|$ is used for the system to be given enough time to (1) process the input spike train and (2) release the output spike train.



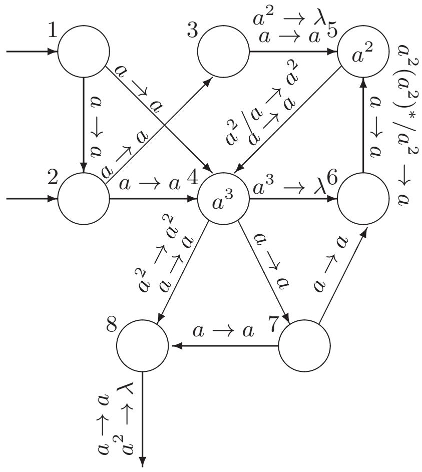
(a) Rule $a^2(a^2)^*/a^2 \to a$ on synapse $(6,5)$ is deleted



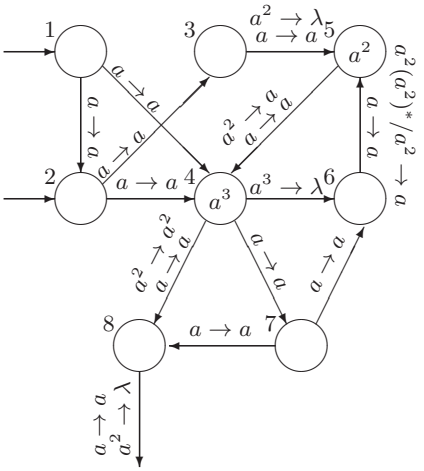
(b) Outgoing neuron of rule $a^3 \to \lambda$ changed from σ_6 to σ_7



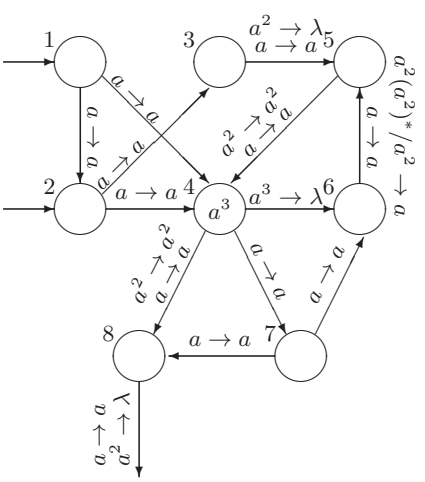
(c) Regular expression of rule $a^2(a^2)^*/a^2 \to a$ on synapse $(6,5)$ changed to $a^2(a)^*$



(d) Number of spikes consumed by rule $a^2 \to a^2$ on synapse $(5,4)$ decreased from 2 to 1



(a) Number of spikes produced by rule $a^2 \to a^2$ on synapse $(5, 4)$ decreased from 2 to 1



(b) Initial number of spikes in σ_5 decreased from 2 to 0

Fig. 6: RSSNPs derived from Figure 1 using the Delete Rule Operation in Section 1, Change Connected Synapses operation in Section 2, Change Regular Expression operation in Section 3, Decrease Consumed Spikes operation in Section 4, Decrease Produced Spikes operation in Section 5, and Decrease Initial Number of Spikes in Section 6

This can give us deeper information on the structure of the RSSNP (if there are extra neurons or synapses that delay spikes to the environment, etc.).

Once each b'_i has its corresponding $score_i$, the fitness of Π'_j is calculated as the average of all the scores of b'_i or $fitness_j = \frac{score_1 + score_2 + ... + score_n}{n}$.

A higher fitness means lower error ϵ. So we achieve Π'_j with a lower error by attaining RSSNPs with higher fitness, thus achieving the first bullet defined in Section 4.

Given $b_{max} = max(b_1, b_2, \ldots, b_n)$ and n is the number of input-output spike train pairs, the time complexity of this step is $O(\ {b_{max}}^2 \cdot n \cdot popsize)$. According to [5] and [17], Longest Common Substring and Longest Common Subsequence's time complexity is both $O(n \cdot m)$, where n and m are the length of the strings being compared. We can assume $n = m = 3 \cdot b_{max}$ because the maximum length of the strings to be compared is $3 \cdot b_{max}$, so methods (1) and (2)'s time complexity is $O(3 \cdot {b_{max}}^2)$ or $O({b_{max}}^2)$, and we repeat this n times for all input-output spike train pairs. We perform this for all Π'_j in P.

7.3 Parent Selection

After determining the fitness of every Π'_j, we sort P by descending fitness. Suitable parents are then chosen to produce offspring that will be included in the succeeding population. For this study, there are 3 methods used to select parents:

1. **Top 50% of the population**. The first half of the population are selected as parents. We use this selection method because it is simple and can easily be implemented. Also, there is always a high probability of having individuals with high fitness to become parents. A disadvantage is that this method is prone to getting stuck at an undesired solution. This is due to its nature of trying to converge to a solution at the start rather than looking for other possible solutions [14].
2. **25% of the population based on fitness**. We take the fitness of every individual Π'_j in the population and calculate their sum $\sum_{j=1}^{n} fitness_j$. The probability of each individual Π'_j to be selected as a parent is $\frac{fitness_j}{\sum_{j=1}^{n} fitness_j}$. After calculating the probability of each individual, we select 25% of the population to be parents based on their probabilities; the higher its probability, the more likely it is to become a parent. Unlike in method (1), using this method lets the framework introduce diversity of RSSNP systems. This, in turn, will allow us to see a variety of individuals.
3. **Top 25% of the population + 25% of the population based on fitness**. A hybrid of the other methods. It is similar to method (1), but only the first quarter of the population is selected. Then, we calculate the probability of being selected as a parent to the other 75% of the population using the same method in (2). After calculating the probability of each individual from the remaining 75%, we select $\frac{popsize}{4}$ individuals to become parents based on their probabilities. Combining methods (1) and (2) allows us to take advantage of the qualities of both methods.

RSSNP	Fitness
1	87.85%
2	85%
3	84.3%
4	80%
5	77.45%
6	73%
7	68.23%
8	65.05%

Table 3: Example of a population of $popsize = 8$

To give a better understanding, assume a population of $popsize = 8$ with fitness of individuals defined in Table 3. Using method (1), the probability of

each RSSNP being a parent is shown in Table 4a. The top 4 RSSNPs with the highest fitness are sure to be parents of the next generation while the rest have no chance. On the other hand, using method (2) gives all RSSNPs a chance to become a parent as shown in Table 4b. In Table 4c, it can be seen that method (3) automatically entails the top 25% of the population to be parents while the rest have a probability dependent on their fitness.

RSSNP	Fitness	Probability of Being a Parent
1	87.85%	100%
2	85%	100%
3	84.3%	100%
4	80%	100%
5	77.45%	0%
6	73%	0%
7	68.23%	0%
8	65.05%	0%

(a) Top 50% of the Population

RSSNP	Fitness	Probability of Being a Parent
1	87.85%	14.15%
2	85%	13.69%
3	84.3%	13.58%
4	80%	12.88%
5	77.45%	12.47%
6	73%	11.76%
7	68.23%	10.99%
8	65.05%	10.48%

(b) 25% of the Population Based on Fitness

RSSNP	Fitness	Probability of Being a Parent
1	87.85%	100%
2	85%	100%
3	84.3%	18.82%
4	80%	17.86%
5	77.45%	17.29%
6	73%	16.29%
7	68.23%	15.23%
8	65.05%	14.52%

(c) Top 25% of the Population + 25% of the Population Based on Fitness

Table 4: Application of selection method to Table 3

Only one selection method is used throughout the whole framework at a time. Meaning, for every generation in a run, the same selection method is employed to select the parents of the next generation.

After selecting the parents, the top 50% of the population in the current generation will be included in the population of the next generation regardless of whichever method was used to select parents. This is to ensure that the highest fitness in the population will be maintained in the succeeding generations.

Regardless of which method is selected, the time complexity for this step is $O(popsize \cdot \log popsize)$. Before any method is applied, P is sorted by descending fitness. From [1], Timsort's time complexity is $O(n \log n)$, where n is the number of elements. Since there are $popsize$ elements in P, sorting it using Timsort would take $O(popsize \cdot \log popsize)$. Method (1)'s time complexity is just $O(1)$, while methods (2) and (3)'s is $O(n)$. Adding the time complexity of the method chosen will still result to $O(popsize \cdot \log popsize)$.

7.4 Crossover

Once the parents are selected, we group the parents into pairs and perform crossover to produce offspring that share qualities with their parents.

For each pair of parents, we select a rule from each parent randomly and interchange them with one another. In other words, we swap the selected rules of the parents, thus creating two new individuals. Using this method, we can see that two offspring will be produced for each pair of parents.

For example, consider Figure 5a and Figure 5b as a pair of parents. The rules chosen are $a^3 \to \lambda$ on synapse $(4, 6)$ and $a^2(a^2)^*/a^2 \to a$ on synapse $(6, 5)$ from Figures 5b and 5c, respectively. We obtain the RSSNPs shown in Figure 7 as offspring.

We perform crossover until $\frac{popsize}{2}$ new invdividuals or offspring are created to fill up the population.

The time complexity of this step is just $O(popsize)$ since swapping the selected rules of the parents only takes $O(1)$ and it is only repeated at most $popsize$ times.

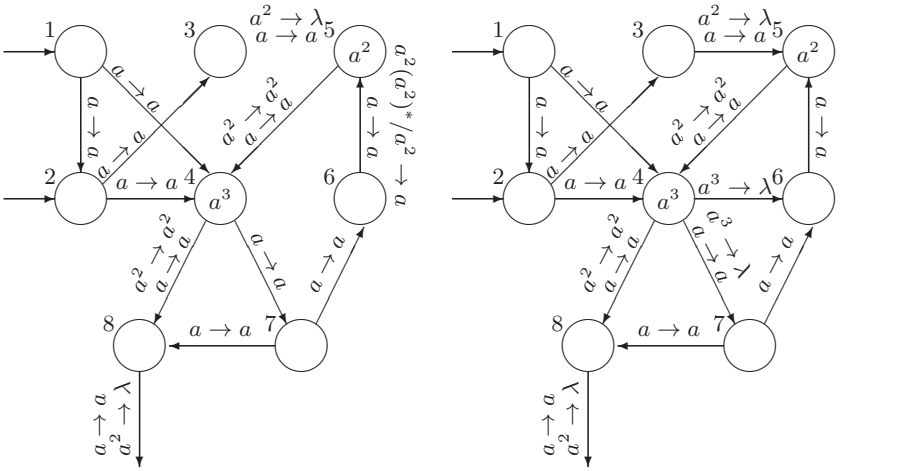
7.5 Mutation

Given a mutation rate $rate_m$ and the set of newly created offspring $P' = \{\Pi''_{\frac{popsize}{2}+1}, \ldots, \Pi''_{popsize}\}$, we randomize the resources of each Π''_j except rules connected to an input neuron or the output neuron. We can do this by applying the following operations enumerated in Section 7.1.

Similar to Section 7.1, the time complexity of this is step is also $O(r \cdot popsize)$, where r is the number of rules in Π_{init} since the same process is applied, but this time to the newly created offspring.

7.6 Checking if the offspring is valid

After creating an offspring, we need to make sure that it is valid. An RSSNP is considered valid for this study if it exhibits all of the following qualities:



(a) Rule $a^3 \to \lambda$ on synapse $(4, 6)$ is swapped with rule $a^2(a^2)^*/a^2 \to a$ on synapse $(6, 5)$

(b) Rule $a^2(a^2)^*/a^2 \to a$ on synapse $(6, 5)$ is swapped with rule $a^3 \to \lambda$ on synapse $(4, 6)$

Fig. 7: Crossover of Figure 5a and Figure 5b

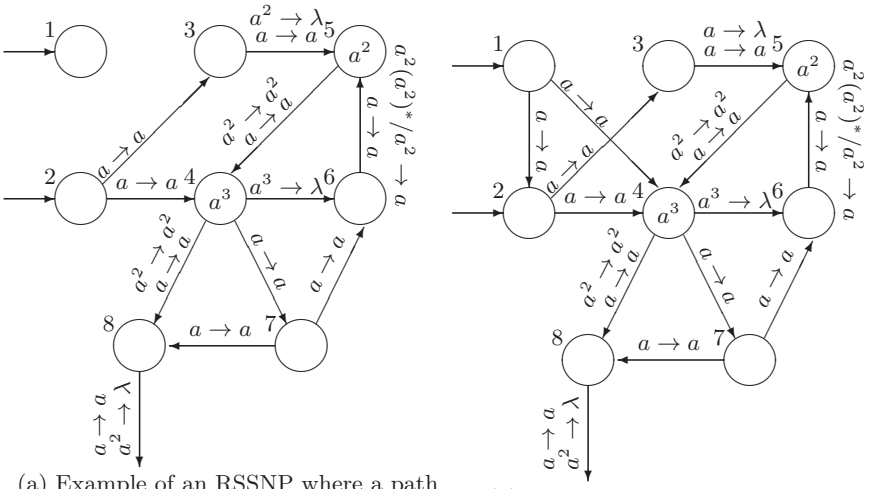
1. **All input neurons are connected to the output neuron**. There must exist a path from all of the input neurons to the output neuron.
2. **No non-determinism**. All synapses in the system must not contain two or more rules with intersecting regular expressions. Rules contained in different synapses but from the same source neuron must not have different consumed spikes.

For example, Figure 8a is an invalid RSSNP because a path from the input neuron σ_1 to the output neuron σ_8 does not exist. On the other hand, Figure 8b is also invalid because the regular expression of the rules $a \to a$ and $a \to \lambda$ on synapse $(3, 5)$ have an intersection making the system to be non-deterministic.

If at least one of the qualities above are not followed by a produced offspring, it will be deleted and new offspring will be created by going back to Section 7.4.

After the population is restored to its original size of *popsize*, the algorithm will proceed to Section 7.2 and will cycle through the algorithm again.

The time complexity for this step is $O(r^2)$ where r is the number of rules in the system. To check if all the input neurons are connected to the output neuron, the Breadth-first search, which has a time complexity of $O(|V| + |E|)$, where V is the number of vertices and E is the number of edges in the graph, is employed. Thus, method (1) has a time complexity of $O(|in| + |syn|)$. On the other hand, method (2) takes $O(r^2)$ since we compare the rules with each other. Combining both methods, the time complexity is $O(|V| + |E| + r^2)$ or $O(r^2)$.



(a) Example of an RSSNP where a path from an input neuron to the output neuron does not exist

(b) Example of a Non-Deterministic RSSNP

Fig. 8: Example of invalid RSSNPs

7.7 Termination

The algorithm halts when the specified number of generations have been reached. There are no stopping criteria to give the framework more opportunities to reduce the number of resources of the "fittest" RSSNP system.

8 Implementation

To simulate this algorithm on a computer, the programming language Python version 3 was used.

A problem arises when simulating how the output neurons will send spikes to the environment. For the purpose of this experiment, we created an additional neuron σ_{env} to represent the environment. This neuron does not have outgoing synapses and has incoming synapses from the output neurons that are supposedly connected to the environment.

Before we begin the algorithm, the input RSSNP Π_{init} and the set of input-output spike trains S must be properly represented in the selected programming language. Below is the representation of the RSSNP shown in Figure 1 which is modelled similarly to [4].

Listing 1.1: RSSNP representation

```
example_rssnp = {
    'neurons': 9,
```

```
    'synapses': 13,
    'rules': [
        [0, 1, (1, 0), 1, 1, 0],
        [0, 3, (1, 0), 1, 1, 0],
        [1, 2, (1, 0), 1, 1, 0],
        [1, 3, (1, 0), 1, 1, 0],
        [2, 4, (1, 0), 1, 1, 0],
        [2, 4, (2, 0), 2, 0, 0],
        [3, 5, (3, 0), 3, 0, 0],
        [3, 6, (1, 0), 1, 1, 0],
        [3, 7, (1, 0), 1, 1, 0],
        [3, 7, (2, 0), 2, 2, 0],
        [4, 3, (1, 0), 1, 1, 0],
        [4, 3, (2, 0), 2, 2, 0],
        [5, 4, (1, 0), 1, 1, 0],
        [5, 4, (2, 2), 2, 1, 0],
        [6, 5, (1, 0), 1, 1, 0],
        [6, 7, (1, 0), 1, 1, 0],
        [7, 8, (1, 0), 1, 1, 0],
        [7, 8, (2, 0), 2, 0, 0]
    ],
    'init_config': [0, 0, 0, 3, 2, 0, 0, 0, 0],
    'rule_status': [-1 for x in range(18)],
    'input_neurons': [0, 1],
    'environment': 8
}
```

As seen in Listing 8, to represent an RSSNP, a dictionary with the following keys is used:

1. `neurons`: Indicates the number of neurons (including the environment) in the system
2. `synapses`: Indicates the number of synapses in the system
3. `rules`: Contains the rule vector lists as defined by [4] of the system
4. `init_config`: Contains the rule status as defined by [4] of the system
5. `input_neurons`: Indicates the index of the input neurons of the system
6. `environment`: Indicates the index of the neuron which will act as the environment

To represent the input and output spike trains, an array of Python dictionaries is used. Each dictionary contains the keys `inputs` and `out`. `out` contains the expected output spike train from the RSSNP. `inputs` is an array of dictionary consisting only of two keys `index` and `input`. `index` indicates the index of the input neuron which the input spike train in `input` will be put. The input spike train is inserted from left to right, meaning the most significant bit is the first one to be added and then the least significant bit is the last. An example of this can be seen in Listing 8 where the list of input and output spike trains represents that of expected of an RSSNP that simulates the XOR gate.

Listing 1.2: Input and Output spike trains representation

```
'inout_pairs = [
    {
        'inputs': [
            {'index': 0, 'input': '0101010'},
            {'index': 1, 'input': '1010101'}
        ],
        'out': '11111111'
    },
    {
        'inputs': [
            {'index': 0, 'input': '111111111111'},
            {'index': 1, 'input': '001010001001'}
        ],
        'out': '110101110110'
    },
    ...
]
```

To be able to accommodate input spike trains and get the number of spikes produced per time step, we made a few modifications to [4]'s algorithm as seen in Algorithm 1. We defined an Input Vector $IN = [\ in_1, \ldots, in_d, \ldots, in_p]$, where p is the number of input neurons in the system, and in_d is the index of an input neuron and for each in_e in $P \setminus \{in_d\}$, $in_d \neq in_e$. An Input Spike Train Vector IS is also defined to list all the incoming spikes of the p input neurons. It is defined as $IS = [is_1, \ldots, is_d, \ldots, is_p]$, where is_d is the spike train for input neuron σ_{in_d}. A spike train is_d is represented as $is_d = [is_{d_1}, \ldots, is_{d_b}, \ldots, is_{d_s}]$, where s is the length of the input spike train, and $is_{1_b} \in \{0, 1\}$ is the input at time b. Along with the previous inputs $SS^{(0)}$ and RL, the algorithm requires IN and IS as inputs. Lines 16-18 are added so that for every input neuron in_d, its corresponding input $is_{in_{d_t}}$ at time step t is added.

Instead of returning the $UnexploredStates$, the algorithm now returns the output spike train OS. OS is defined as $OS = [os_1, \ldots, os_e, \ldots, os_q]$, where $os_e \in \{0, 1\}$ is the output of the system at time $e - 1$.

The Configuration Vector $CF^{(t)}$ is also extended to include a special neuron σ_{env} that will act as the environment. $CF^{(t)}$ is now defined as $CF^{(t)} = [cf_1^{(t)}, \ldots, cf_i^{(t)}, \ldots, cf_n^{(t)}, cf_{env}^{(t)}]$, where $cf_i^{(t)}$ for $1 \geq i \leq n$ is the number of spikes in σ_i at time t, and $cf_{env}^{(}t)$ is the output of the system at time $t - 1$, where $t \neq 0$. Line 14 shows that after computing for the next System State $SS^{(t+1,w)}$, $cf_{env}^{(t+1)}$ is appended to OS to record the output for time step (t). It is then reinitialized to 0 since we want to get the number of spikes produced per time step, not the sum of the number of spikes produced by that time step.

We also added another condition for adding $SS^{(t,w)}$ to the list of unexplored states $UnexploredStates$ as shown by line 20. If at time step t, there is still an input, regardless of $SS^{(t,w)} \in ExploredStates \cup UnexploredStates$, it is still

added to *UnexploredStates*. This is to ensure that the output spike produced by the inputs $is_{in_{d_t}}$ are still recorded.

To prevent the algorithm from running into an infinite loop, we define a maximum number of time steps $MaxStep$ for the computation. $MaxStep$ is defined to be $3 \cdot |OS'|$, where $|OS'|$ is the expected output spike train produced. We also set a counter *Step* which tracks the current time step of the computation. It can be seen in lines 25-28 that if $Step > MaxStep$, the computation halts.

Algorithm 1: Main Simulation Algorithm

Input: $SS^{(0)}$, RL, IN, $IS, MaxStep$
Output: OS

1 $UnexploredStates \leftarrow \{SS^{(0)}\}$;
2 $ExploredStates \leftarrow \{\}$;
3 $OS \leftarrow \{\}$;
4 $Step \leftarrow 0$;
5 **while** *UnexploredStates is not empty* **do**
6 Get $SS^{(t)}$ from *UnexploredStates*;
7 $CF^{(t)}, RS^{(t)} \leftarrow SS^{(t)}$;
8 $SY^{(t)} \leftarrow SynapseStatus(RS^{(t)}, RL)$;
9 $AP^{(t)} \leftarrow RuleApplicability(CF^{(t)}, SY^{(t)}, RL)$;
10 $AM^{(t)} \leftarrow ActivationVectors(AP^{(t)}, RL)$;
11 $RM'^{(t)}, LM^{(t)} \leftarrow ApplyRules(AM^{(t)}, RS^{(t)}, RL)$;
12 $CM^{(t+1)}, RM^{(t+1)} \leftarrow NextState(CF^{(t)}, RM'^{(t)}, LM^{(t)}, RL)$;
13 **for** *each* $(CF^{(t+1,w)}, RS^{(t+1,w)})$ **do**
14 Append $cf_{env}^{(t+1,w)}$ to OS;
15 $cf_{env}^{(t+1,w)} \leftarrow 0$;
16 **for** *each* in_d **do**
17 $cf_{in_d} \leftarrow cf_{in_d} + is_{in_{d_{t+1}}}$
18 **end**
19 $SS^{(t+1,w)} = (CF^{(t+1,w)}, RS^{(t+1,w)})$;
20 **if** $SS^{(t+1,w)} \notin (ExploredStates \cup UnexploredStates)$ ***or*** $Step \leq max(|is_1|, \ldots, |is_p|))$ **then**
21 Add $SS^{(t+1,w)}$ to *UnexploredStates* list;
22 **end**
23 **end**
24 Put $SS^{(t)}$ in *ExploredStates* list;
25 $Step \leftarrow Step + 1$;
26 **if** $Step \leq MaxSteps$ **then**
27 **break**;
28 **end**
29 **end**
30 **return** OS

9 Experiments

To test the algorithm, we created RSSNP systems that have extra rules and neurons for every function. We did this by adding extraneous rules, neurons, and synapses to the already existing RSSNP that performs that function. Extraneous rules are modifications of existing rules put into other synapses. Extraneous synapses were added to connect existing neurons to the extraneous ones and extraneous neurons to the existing ones. We also modified some of the original rules in the system to ensure that the system still computes the intended output. Due to limited memory space in the testing computer used, we only added 3 extra neurons for each RSSNP. Doing so, we wanted to check whether the algorithm we created is be able to delete the extraneous rules and remove the synapses connecting the extraneous neurons. We also wanted to check whether the rules we modified would return to its original form.

The RSSNP systems used are listed in Table 5 and diagrams of the RSSNP systems (generated by the graphviz library in Python) are shown in Figures 9 through 13. In each diagram, the input neurons, as defined in 5, are represented by the round, triangular-shaped neurons while the environment shown in 6 is represented by the double-circle. Each rule is represented by a string of the form $(i, j)/c- > p$ where i and j refer to the regular expression in the rule as discussed in Section 5 and c refers to the number of spikes consumed when the rule is fired and c indicates how many spikes will be produced.

Using the RSSNPs mentioned previously, we executed the experiments shown in Table 6 using the different selection methods enumerated in Section 7.3. Each experiment was run 5 times and the average of the highest fitness in each run was recorded which is shown in Figures 14 until 18.

RSSNP	Function	Fitness		Figure
		Longest Common Substring	Longest Common Subsequence	
and_adversarial	AND	59%	84%	9
or_adversarial	OR	74%	97%	10
not_adversarial	NOT	64%	95%	11
add_adversarial	ADD	48 %	72%	12
sub_adversarial	SUB	60%	79%	13

Table 5: List of RSSNP systems used

10 Analysis and Discussion

Firstly, the RSSNP systems that were created by the framework do satisfy the constraints given in Section 4. During the last generations of each run, the

Table 6: List of experiments for varying selection methods

Experiment No.	Population Size	Mutation Rate	Fitness Function	No. of Generations
1	12	1%	Longest Common Substring	20
2	12	1%	Longest Common Subsequence	20
3	12	1%	Longest Common Substring	50
4	12	1%	Longest Common Subsequence	50
5	12	2%	Longest Common Substring	20
6	12	2%	Longest Common Subsequence	20
7	24	1%	Longest Common Substring	20
8	24	1%	Longest Common Subsequence	20
9	24	2%	Longest Common Substring	20
10	24	2%	Longest Common Subsequence	20
11	12	2%	Longest Common Substring	50
12	12	2%	Longest Common Subsequence	50
13	24	1%	Longest Common Substring	50
14	24	1%	Longest Common Subsequence	50
15	24	2%	Longest Common Substring	50
16	24	2%	Longest Common Subsequence	50

RSSNP with the highest fitness always had less than or equal resources to the initial RSSNP. This is due to the nature of the algorithm that never adds resources to a system that already exists in each generation. All operations discussed in Section 7.1 only deleted or decreased. The crossover function explained in Section 7.4 had 2 parent RSSNPs swap rules, which means neither of the two RSSNPs will gain resources "for free." With that said, the GA framework is successful in creating RSSNP systems with less resources. The question now is whether or not those RSSNP systems can produce the expected output spike trains correctly.

It can be seen from Figure 14 to Figure 18 that using the selection method (1) rarely yields the highest average fitness. Table 7 shows a run of Experiment 1 using selection method (1) with add_adversarial as the initial RSSNP. In the table, it can be seen that the highest fitness of the population stopped increasing at Generation 8. Upon observing the population per generation of that run, the top 50% s with highest fitness of the population started looking similar at Generation 8. After every generation, the RSSNPs in the population started looking similar, and by Generation 13, the top 50% of the population with highest fitness are all the same RSSNPs. As a result, performing crossover became meaningless since the parents have the same configurations, so the results offspring would just end up being a copy of its parents.

These results support the evaluation of selection methods used in GA by [14]. For this GA framework, the same scenario applies wherein the search space for generating candidate solutions becomes limited at a very early generation. With this, method (1) is inefficient in solving the problem stated in 3.

On the other hand, we can see that using selection method (2) yields the highest average fitness most of the time. A possible reason for this is because instead of exploiting the qualities of the top s, this method gives every individual

Generation	Highest Fitness
1	53%
2	53%
3	57%
4	57%
5	65%
6	65%
7	65%
8	69%
9	69%
10	69%
11	69%
12	69%
13	69%
14	69%
15	69%
16	69%
17	69%
18	69%
19	69%
20	69%

Table 7: Highest fitness per generation of Run 3, Experiment 1 of add_adversarial

in the population a chance to be a parent. This gives a diverse offspring, where a better-fit can be found. Hence, we can say selection method (2) is effective in finding solutions to the problem stated in 3.

Comparing Figure 19 to Figure 23, we can see that the boxes of Longest Common Substring are shorter and relatively closer with one another in Figure 23. This means that the fitness of the resulting RSSNPs from sub_adversarial do not vary regardless of different mutation rate, selection method, number of generations, and population size. The boxes' medians can also be seen to be almost equal. This implies that whichever selection method is chosen, the fitness of the resulting RSSNP from sub_adversarial is likely to be the same if a different selection method is chosen.

Using fitness method (1), sub_adversarial has an initial fitness of 60% but still yielded the lowest resulting fitness compared to the other RSSNPs with lower initial fitness such as add_adversarial. This is shown in Figure 18 where the average fitness at the last generation of experiments that used method (1) (odd-numbered experiments) did not reach 90%.

It is worth noting that sub_adversarial is relatively large compared to the other RSSNPs in Table 5. Whereas the other RSSNPs have 6-7 neurons, 10-13 synapses, and 10-15 rules, sub_adversarial has 14 neurons, 26 synapses, and 32 rules. This may be the reason why Figure 23 is different compared to others. It is a fact that it takes longer for the algorithm to mutate larger RSSNPs, which is why it is possible that a higher number of generations is required before we see

obvious differences and for sub_adversarial to yield a higher fitness using method (1).

It can also be observed that it was easy for the framework to find an RSSNP that has 100% fitness when handling the NOT function compared to the others. This could be due to its nature of being a **unary** operation as compared to the others which are binary. This could imply that the framework is effective in giving RSSNPs that compute unary problems with high fitness.

On the other hand, the GA framework had difficulty in finding RSSNPs that computes the ADD (addition) and SUB (subtraction) functions. When using Longest Common Substring in computing fitness, it was uncommon for the GA framework to find an RSSNP with a fitness of at least 90% (4/64 instances) while handling the ADD function while no RSSNP with at least 90% fitness was found for SUB (highest fitness is 88%). Since the initial ADD RSSNP had a fitness of 48% and 81% for SUB, there were some improvements. The poor results from the aforementioned functions could be caused by the greater number of rules and existence of more complex rules (rules that consume or produce a greater number of spikes, more complex regular expressions) than the other initial RSSNPs. This suggests that the complexity of rules in the initial system can greatly affect the rate of growth of fitness.

11 Final Remarks

Overall, the use of a genetic algorithm to solve the problem stated in Section 3 is a feasible approach. In an exhaustive and deterministic algorithm, the maximum number of configurations needed to be checked is $O(2^{nmlijcp})$ where

- n = number of neurons
- m = number of rules
- l = number of synapses
- i, j = max{coefficients of the regular expressions,1}
- c = maximum number of spikes consumed in a rule
- p = max{number of spikes produced in a rule,1}

A genetic algorithm has the advantage of not needing to check every possible combination of resources and would require a maximum of $O(gen \cdot popsize)$ (where gen = the specified number of generations, and $popsize$ = the population size) configurations to be generated, which is still large but significantly lower than the exponentially massive value of the aforementioned method.

We have seen that the RSSNP systems created by the genetic algorithm framework still have a long way to go before they can produce the desired output spike trains perfectly. But we have introduced a new method for designing RSSNP systems using a genetic algorithm. Depending on which methods in the genetic algorithm framework were used, the results may differ significantly.

This work only touches the surface of what can be done to evolve RSSNP systems. For one thing, genetic algorithms should be suitable for any type of RSSNP system, but we have yet to see its potential for more difficult functions.

Another problem is that only reducing the resources might not be enough for more complex RSSNP systems. In any case, this is a good first step. The existence of a genetic algorithm framework for evolving RSSNPs may prove to be beneficial for similar developments in the future.

To check if the genetic algorithm framework can still be improved such that it can find a better solution with fewer generations, a different selection method can be used. This is to check if there is a better way of selecting parents for the next generation. A different method of crossover between parents can also be performed to see if the offsprings produced can be improved. Another fitness function that evaluates the number of resources in the RSSNP can also be incorporated into the system so that the size of the system can also be considered when evaluating. Lastly, suitable stopping criteria may be incorporated to further improve the outputs of the framework.

The genetic algorithm framework can also be extended to support RSSNPs that are non-deterministic and have delays.

Acknowledgements

I.C.H. Macababayao, R.T.A. de la Cruz, F.G.C. Cabarle, and H.N. Adorna acknowledge support by grants from the DOST-ERDT project. F.G.C. Cabarle acknowledges support from the Dean Ruben A. Garcia PCA AY2018–2019, an RLC AY2018–2019 grant, and Project No. 191904 ORG (2019–2020) of the OVCRD in UP Diliman. H.N. Adorna is supported by the Semirara Mining Corp Professorial Chair for Computer Science, and RLC grant from UPD OVCRD. M.A. Martínez-del-Amor acknowledges the support of the research project TIN2017-89842-P (MABICAP), co-financed by *Ministerio de Economía, Industria y Competitividad (MINECO)* of Spain, through the *Agencia Estatal de Investigación (AEI)*, and by *Fondo Europeo de Desarrollo Regional (FEDER)* of the European Union. The work was supported by the National Natural Science Foundation of China (Grant Nos. 61472333, 61772441, 61472335, 61672033, 61425002, 61872309, 61771331), Project of marine economic innovation and development in Xiamen (No. 16PFW034SF02), Natural Science Foundation of the Higher Education Institutions of Fujian Province (No. JZ160400), Natural Science Foundation of Fujian Province(No. 2017J01099), Basic Research Program of Science and Technology of Shenzhen (JCYJ20180306172637807).

References

1. Auger, N., Nicaud, C., Pivoteau, C.: Merge strategies: from merge sort to timsort (2015)
2. Cabarle, F., Adorna, H., Martinez-del Amor, M.A.: Simulating spiking neural p systems without delays using gpus. International Journal of Natural Computing Research (IJNCR) **2**(2), 19–31 (2011)
3. Cabarle, F.G.C., Adorna, H., Martinez-Del-Amor, M.A., Perez-Jimenez, M.J.: Improving GPU Simulations of Spiking Neural P Systems. Romanian Journal of Information Science and Technology **15**(1), 5–20 (2012)

4. Cabarle, F.G.C., de la Cruz, R.T.A., Cailipan, D.P.P., Zhang, D., Liu, X., Zeng, X.: On solutions and representations of spiking neural p systems with rules on synapses. Information Sciences (2019). https://doi.org/https://doi.org/10.1016/j.ins.2019.05.070, http://www.sciencedirect.com/science/article/pii/S0020025519304876
5. Flouri, T., Giaquinta, E., Kobert, K., Ukkonen, E.: Longest common substrings with k mismatches. Information Processing Letters **115**(6-8), 643–647 (2015)
6. Fogel, D.B., Fogel, L.J., Porto, V.: Evolving Neural Networks. Biological cybernetics **63**(6), 487–493 (1990)
7. Ionescu, M., Păun, G., Yokomori, T.: Spiking neural P systems. Fundamenta informaticae **71**(2, 3), 279–308 (2006)
8. Kitano, H.: Designing neural networks using genetic algorithms with graph generation system. Complex systems **4**(4), 461–476 (1990)
9. Miller, G.F., Todd, P.M., Hegde, S.U.: Designing Neural Networks using Genetic Algorithms. In: ICGA. vol. 89, pp. 379–384 (1989)
10. Mitchell, M.: An introduction to genetic algorithms. MIT press (1998)
11. Montana, D.J., Davis, L.: Training Feedforward Neural Networks Using Genetic Algorithms. In: IJCAI. vol. 89, pp. 762–767 (1989)
12. Pan, L., Păun, G., Pérez-Jiménez, M.J.: Spiking Neural P systems with Neuron Division and Budding. Science China Information Sciences **54**(8), 1596 (2011)
13. Păun, G.: From cells to (silicon) computers, and back. In: New computational paradigms, pp. 343–371. Springer (2008)
14. Saini, N.: Review of selection methods in genetic algorithms. International Journal Of Engineering And Computer Science **6**(12), 22261–22263 (2017)
15. Song, T., Pan, L., Păun, G.: Spiking neural P systems with rules on synapses. Theoretical Computer Science **529**, 82–95 (2014)
16. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
17. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. J. ACM **21**(1), 168–173 (Jan 1974). https://doi.org/10.1145/321796.321811, http://doi.acm.org/10.1145/321796.321811

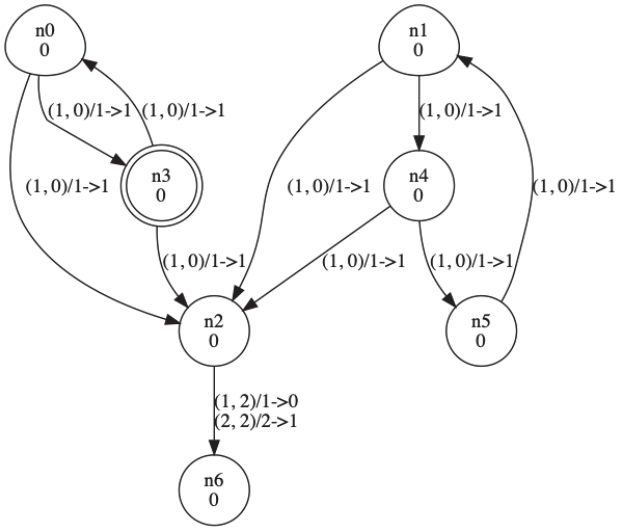


Fig. 9: and_adversarial

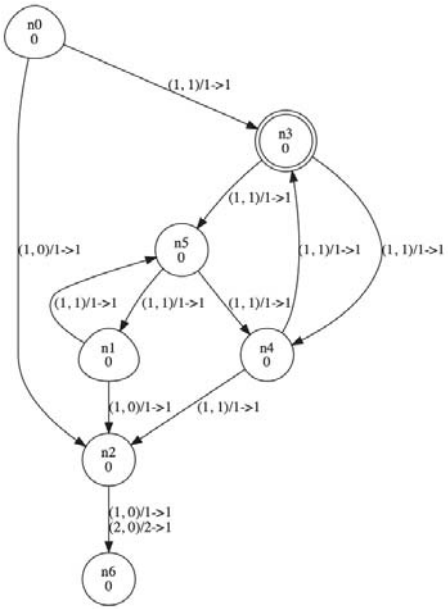


Fig. 10: or_adversarial

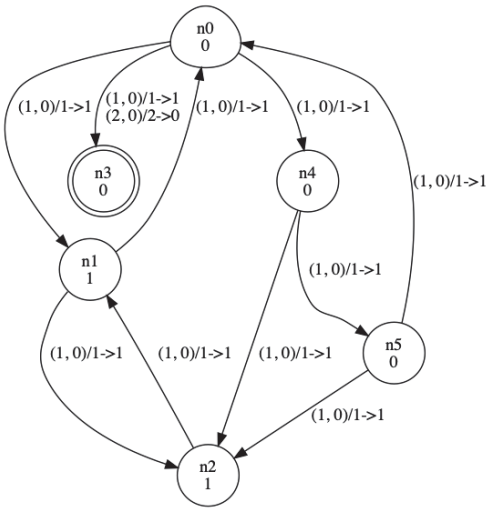


Fig. 11: not_adversarial

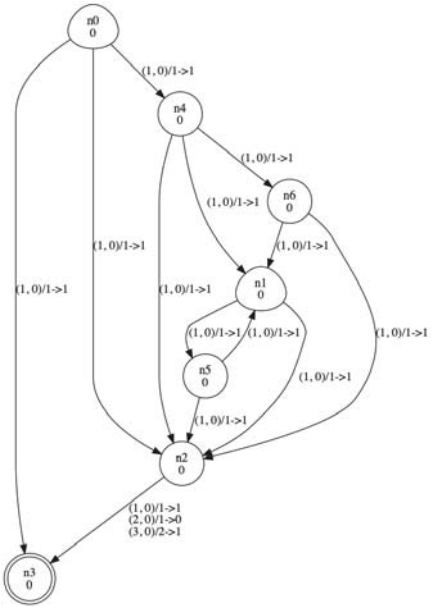


Fig. 12: add_adversarial

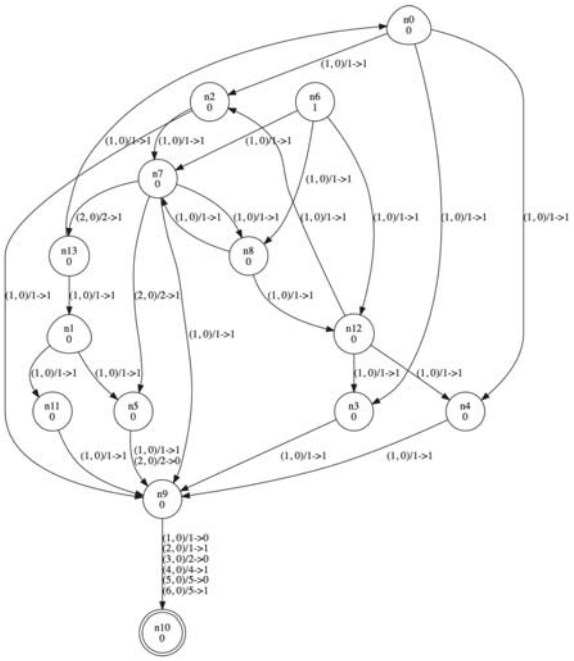


Fig. 13: sub_adversarial

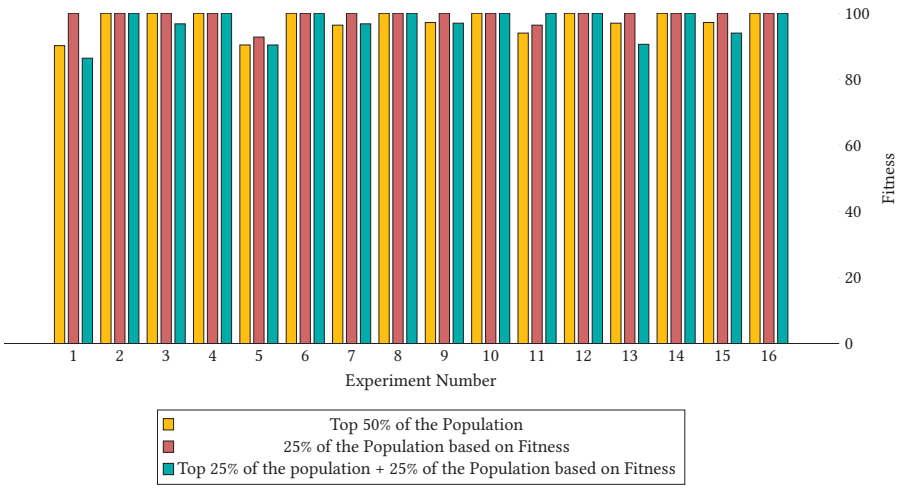


Fig. 14: Average Fitness Value of and_adversarial at the last generation

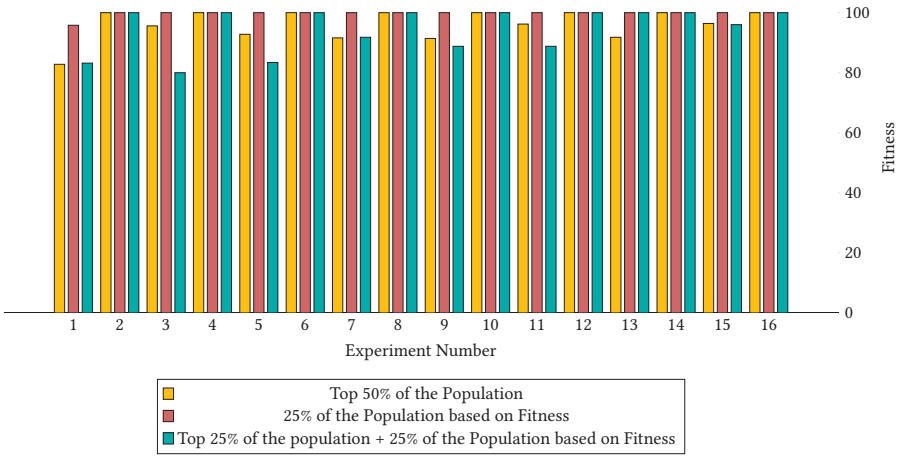


Fig. 15: Average Fitness Value of or_adversarial at the last generation

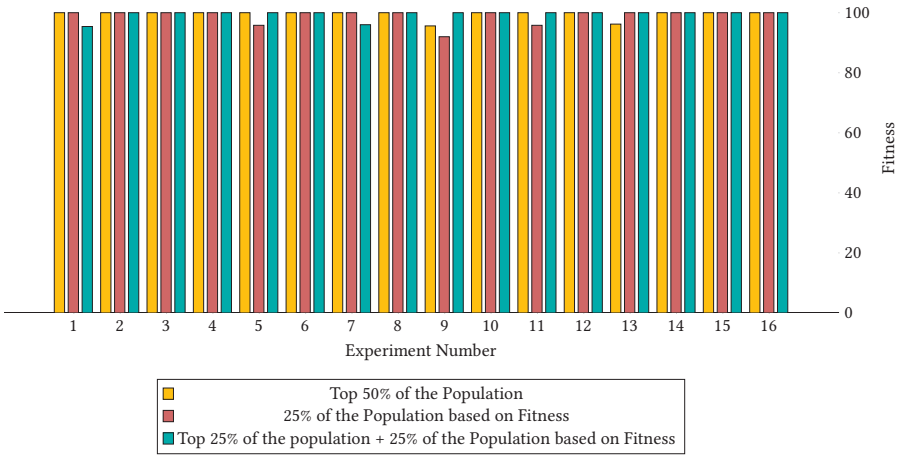


Fig. 16: Average Fitness Value of not_adversarial at the last generation

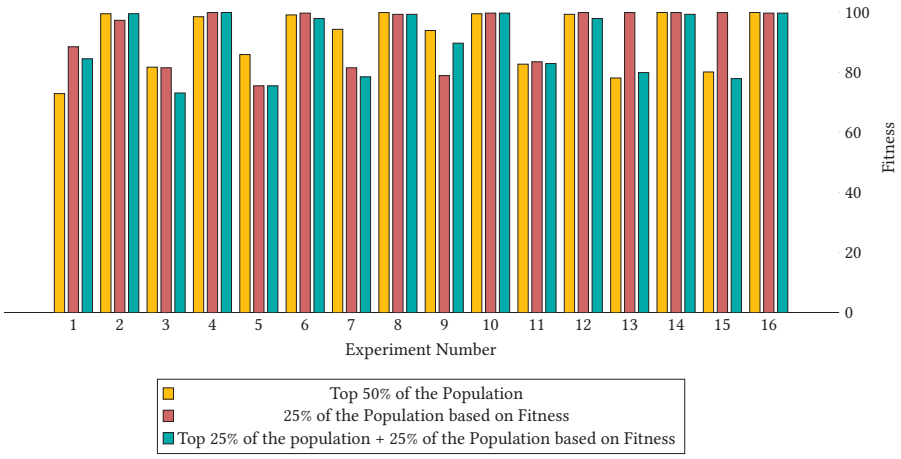


Fig. 17: Average Fitness Value of add_adversarial at the last generation

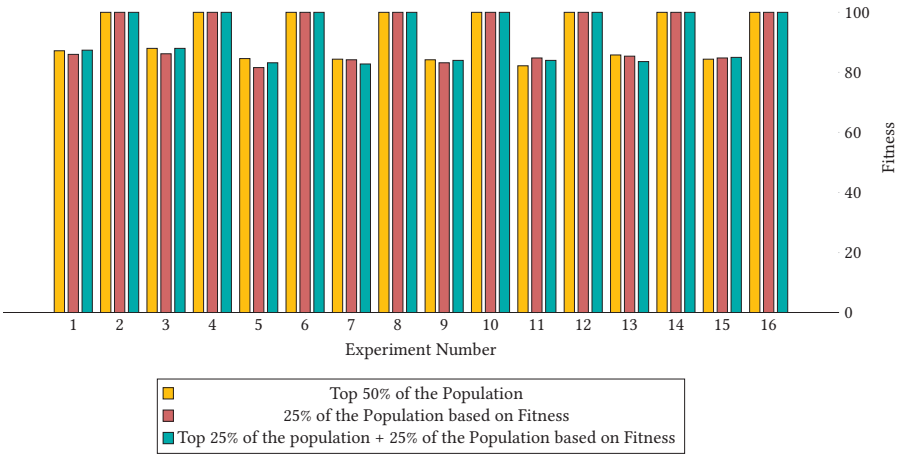


Fig. 18: Average Fitness Value of sub_adversarial at the last generation

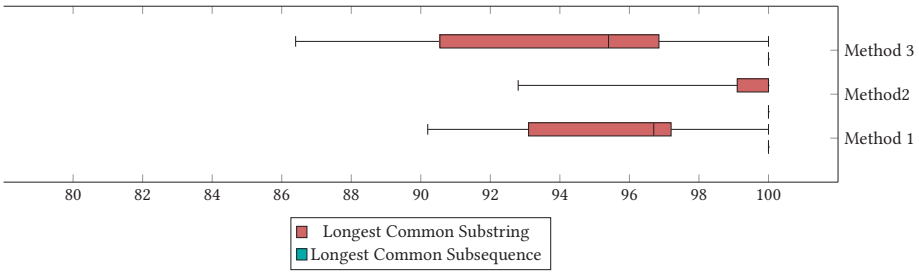


Fig. 19: Average Fitness Value of and_adversarial at the last generation according to the selection method used

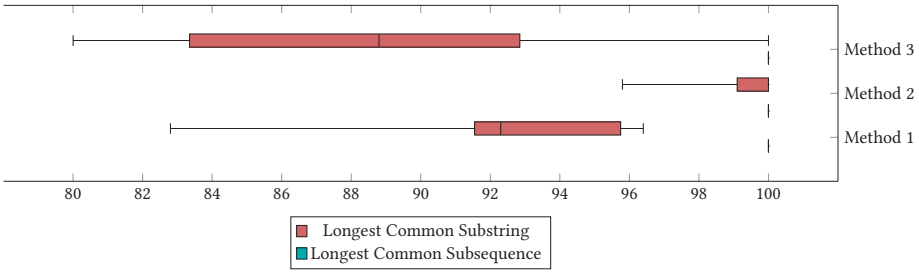


Fig. 20: Average Fitness Value of or_adversarial at the last generation according to the selection method used

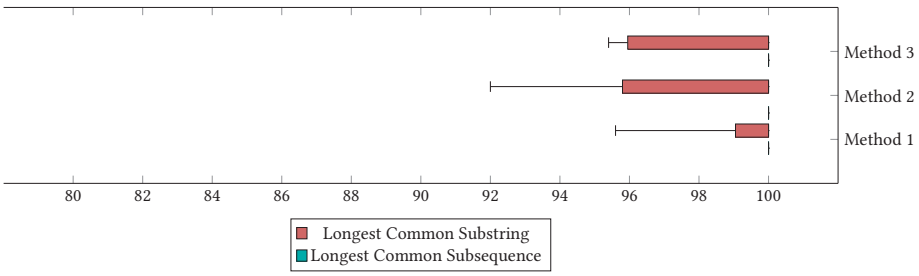


Fig. 21: Average Fitness Value of not_adversarial at the last generation according to the selection method used

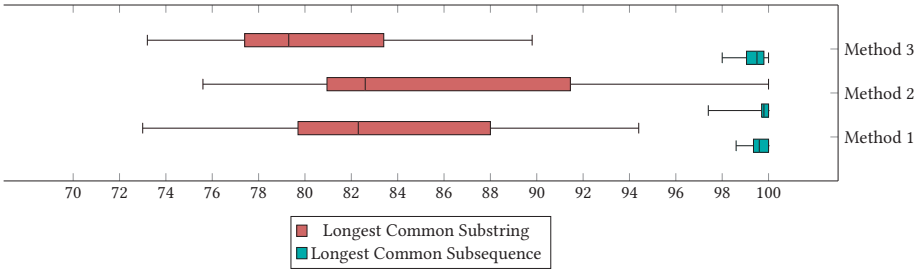


Fig. 22: Average Fitness Value of add_adversarial at the last generation according to the selection method used

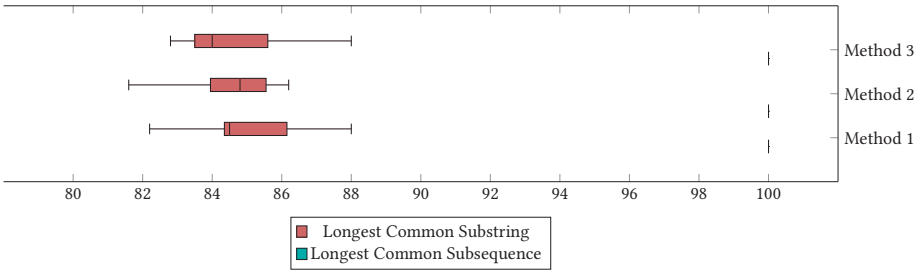


Fig. 23: Average Fitness Value of sub_adversarial at the last generation according to the selection method used

Non-intrusive Electrical Appliances Recognition Method Based on Deep Learning

Zhibin Yu[1*], Hong Chen[1], Chunxia Chen[2*], Gexiang Zhang[1], and Xiantai Gou[1 *]

[1] School of Electrical Engeneering, Southwest Jiaotong University, Chengdu, China
zbinyu@126.com

[2] School of Electrical Engeneering, Southwest Jiaotong University, Chengdu, China
240935533@qq.com

Abstract. Non-intrusive load monitoring (NILM) has become a focused issue because of its low cost, high reliability and easy implementation and maintenance in the on-line status-monitor of electrical appliances and energy management. In this paper, a NILM method, which is based on the voltage−current ($V-I$) trajectory features screening and deep transfer learning, is proposed to improve the recognition accuracy of appliances. Aiming at the problem that a large number $V-I$ trajectory feature are not available due to noise, a preprocessing algorithm is proposed to improve the quality of V¨CI trajectory features. And when $V-I$ trajectory features sample data of certain appliances is very small or new appliances are appear in status-monitor of appliances, the traditional method is difficult to recognize the appliances. Thus, based on deep transfer learning, an algorithm is presented to solve the "$small-sample$"problem of data in status-monitor of appliances. Finally, we tested the proposed method on practical testing data. The average recognition rate of household appliances by using the method reaches up to %99. The numerical results demonstrate that the proposed method has higher accuracy than the traditional method.

Keywords: Non-intrusive load identification; voltage¨Ccurrent trajectory; data preprocessing, deep network; transfer learning.

* Corresponding author.

1 Introduction

Energy is vital to human survival, life and development of the main foundation. However, energy consumption leads to increasingly serious energy shortage and environmental pollution problems [1]. Improving energy efficiency is an important means to solve the current energy and environmental problems. At present, electricity has become the main form of energy consumption in people's daily life. Power saving is the most effective and reliable way to save energy and protect the environment. Currently, residential houses consume approximately 15 percent of electrical energy in China. Studies have shown that if residential consumers are provided with a real time feedback of how much energy their home consumes, 10-15% of energy can be saved [2]. And that demand side management is a major development nowadays to study how consumers can manage energy usage and increase energy efficiency at the same time. On the other hand, the consumer wants to directly control their domestic appliances to change the shape of the system load in order to reduce their power load. They will need to manage the process of load reduction and load recovery to achieve the desired effects. Hence, the information of residential energy consumption has a wide variety of applications for the consumer, electrical power companies and governments. Currently, the information can be obtained by intrusive appliance load monitoring (IALM) which is based on separate system by using sensors that are attached to each appliance in the house. However, it is inconvenient and uneconomical to set up many sensors for each appliance. Besides, there are still some disadvantages such as multiple sensor configuration and huge installation efforts in the practical application.

To overcome these disadvantages of IALM, non-intrusive Load Monitoring (NILM) was introduced by George W. Hart of the Massachusetts Institute of Technology. As opposed to IALM, the non-intrusive load monitoring only needs to install a smart meter at the $user's$ power supply entrance, which integrates data acquisition, data processing and load identification functions. By collecting and analyzing the total current and terminal voltage electricity used by residential users, the smart meter can achieve the purpose of monitoring the electrical information of individual household appliances. Such a non-intrusive load monitoring system can carry out energy monitoring, fault monitoring, fault analysis and other types of power quality control analysis without installing a large number of monitoring equipment [3]. Therefore, NILM has received extensive at-

tention from researcher, power companies and major research institutions [4–8] since it was first proposed.

The load identification is the basic and key issue in non-intrusive load monitoring technology [9]. It can help residents understand the electricity consumption habits, also help to consciously reduce consumption and stop loss, and that can realize fault diagnosis and reduce the loss caused by failure and so on. The NILM algorithm uses different appliances load features to identify appliances. Load features are the electrical behavior of an individual appliance when it is in operation. The quality of the features extracted directly relates to the accuracy and reliability of the load recognition system.

In traditional non-intrusive load monitoring, the electrical parameters (such as variation of power [7], current harmonics [10], transient voltage [11], active/ Disaggregated power [12]), are as features to identify electrical appliances in a simple environment. Based on the feature parameters, and high appliance recognition accuracy can be obtained. However, the accuracy will be greatly reduced for appliances with similar power consumptions by using NILM that is based on the features related to power; and the NILM method, which is based on the features related to transient parameters, are used to recognize the appliances will lead to an increase cost of hardware equipment due to the high sampling frequency and large data storage capacity for extracting transient feature.

In recent years, the voltage-current (V-I) trajectory, which is plotted based on the steady-state voltage and current, is studied to express appliances¡¯ electrical characteristics [13–15]. Apparently, it is easy to get the trajectory features by using the measured data of voltage and current of appliances in steady-state operation. Based on the trajectory features, not only higher recognition accuracy of appliance can be got, but also the cost of hardware equipment be reduced. In addition, the advantages and physical meanings of the trajectory features have been illustrated in [13–15]. And how to quantify the trajectory features has been studied in [15]. Especially, in order to get better appliances identification results than the artificial selection trajectory characteristics approach, the deep learning network is introduced to extract the characteristics of voltage-current trajectory based on the powerful self-learning abilityof deep learning [16, 17]. In [16], the authors applied the deep convolutional neural network to extract automatically the key trajectory characteristics for appliance classification. In these approaches, the voltage-current trajectory images were used as sample data to train and

test the build feature extraction model. The quality and scale of data is related to the classification performance of extracted features, and further, the accuracy of electrical appliances identification. In fact, the voltage and current signals are inevitably interfered by noise in the processes of transmission, and result in radiometric distortion and trajectory anomaly. In addition, when V¨CI trajectory features sample data of certain appliances are very small or new appliances are appear in status-monitor of appliances, insufficient train information will lead to a pool recognition result for the deep leaning network. Therefore, in the NILM method that is based on V-I trajectory and deep leaning network, a V-I trajectory image data preprocessing procedure is often needed to eliminate errors, noises and trajectory distortion first, and the poor recognition accuracy of electrical appliances due to the "$small - sample$" data in status-monitor is still need to improve.

In this paper, based on non-intrusive load monitoring, in the complex measured environment with noise interference, a preprocessing algorithm is proposed to improve the quality of V-I trajectory image data, and the deep learning network is improved to realize the V-I trajectory feature extraction and electrical appliance identification. In order to solve the problem of small sample data and the increase time cost for retraining learning network, a novel NILM approach, which is based on transfer learning and convolution neural network, is proposed. And then, the proposed method is applied to electrical appliance recognition in actual engineering and higher recognition accuracy than tradition method is obtained. The remainder of this paper is organized as follows. In Section 2, the NILM framework is introduced. Section 3 a preprocessing algorithm is proposed. The novel NILM algorithm based on transfer learning and convolution neural network is studied in Section 4. Experiments and results analysis are reported in Section 5. Finally, the conclusions are given in Section 6.

2 Non-intrusive load monitoring system framework

In this paper, the theoretical knowledge and related technologies of machine learning and pattern recognition are used to realize the recognition of household electrical appliances based on non-intrusive load monitoring data. The principle block diagram of non-intrusive load monitoring system is shown in Fig. 1. It includes five parts: data acquisition, data preprocessing, feature extraction, load

identification and feature database [15].

The basic functions of each part are described in detail below.

(1)Data acquisition

The hardware experiment platform built includes household appliances, data acquisition devices and computers. The upper computer software installed on the computer is used for data reception, storage and processing. When the appliance is powered up and activated, the data acquisition device receives the signal data and transmits it to the computer.

(2)Data preprocessing

The function of data preprocessing is to preprocess the collected raw data, such as filtering, denoising, outlier processing, etc. By filtering and cleaning the collected data and abnormal values, the interference of abnormal noise can be reduced, and the accuracy and representativeness of the extracted features can be ensured as far as possible. Therefore, the preprocessing process will directly affect the quality of the feature extracted, and then affect the recognition effect of the identification algorithm of household appliances.

(3)Feature extraction

The feature extraction module is used to extract features in a format supported by machine learning algorithms from datasets consisting of formats such as signals and images. The key is how to extract the signature feature from signals of appliances by proper methods in non-intrusive household appliance load monitoring system.

(4)Load identification

After selecting and extracting signature features of appliances, machine learning method is usually used for model learning and parameter estimation to achieve the purpose of identification of household appliances.

(5)Feature database

The feature database contains electrical features of various household electrical appliances.

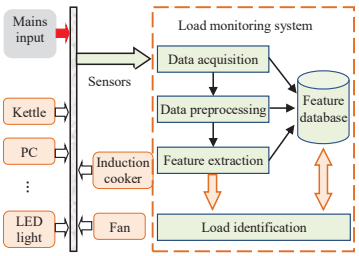


Fig. 1. Principle block diagram of the non-intrusive load monitoring system.

3 V-I trajectory image data preprocessing

3.1 Household electrical appliance classification

The same kind of electrical appliances have similar electrical characteristics. According to the electrical principle and characteristics of household appliances, the appliances can be divided into the following four classes [18].

(1) Resistive load: characteristics of household appliances are similar to resistance. The ideal V-I trajectory of resistive load is straight line, and there is no phase difference between current and voltage. Resistive load are mostly heating appliances, such as electric kettles, rice cookers, water dispensers, electric teapots, etc.

(2) Motor load: refrigerators, fans, vacuum cleaners, washing machines are the motor appliances load. V-I trajectory of the load is an irregular elliptical ring.

(3) Inductive load: induction cooker and fluorescent lamp are typical inductive loads. Their steady-state V-I trajectory are shown as a closed circle or ellipse.

(4) Rectifier load: the rectifier appliance load includes notebook computers, chargers, television sets, etc. The odd harmonic content in the steady current of rectifier load is very high, and the V-I trajectory looks like a horizontal ”8”.

Classification of electrical appliances based on similar electrical characteristics can not only reduce the number of features, but also it can help improve the efficient of identifying appliances. The above analysis can be summarized in the following Table1.

Based on the appliance classification, the current and voltage waveforms of

Table 1. Household electrical appliance classification

Load type	Household appliances
Resistive load	Electric kettle,rice cooker,water heater
Motor load	Refrigerators,fans,washing machines
Inductive load	Induction cooker
Rectifier load	Laptop computer,mobile phone charger

four typical household appliances are shown in Fig.2.

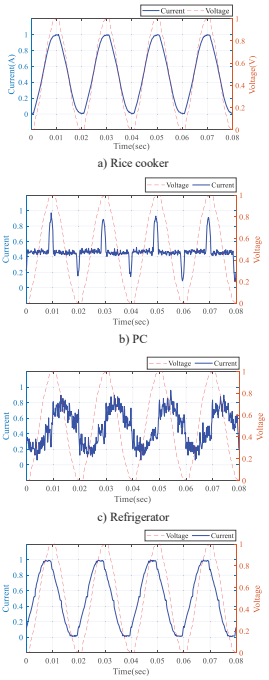


Fig. 2. Current and voltage waveforms (data is normalized)

3.2 V-I Trajectory

Based on High-order harmonic characteristics, the phase angle difference between voltage and current, and the electronic appliance conduction charac-

teristics can be reflected by the V¨CI trajectory features [13, 14]. Based on the voltage and current data in Figure3, the V-I trajectories of the four classes household appliances are plotted in Fig.3. Apparently, from Fig.3, V-I trajectories are different with different class household appliances under ideal conditions.

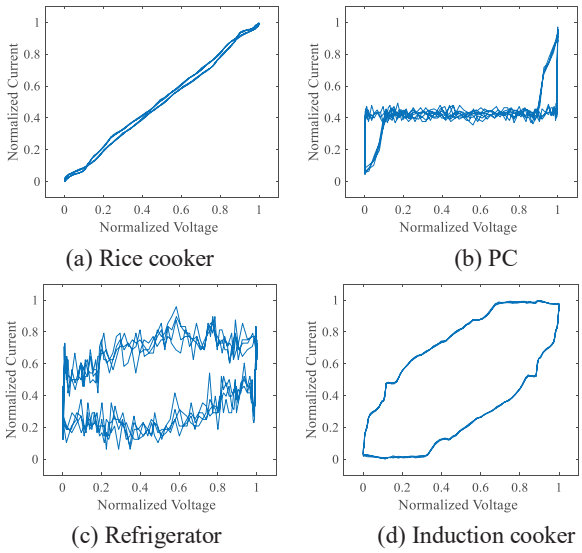


Fig. 3. V-I Trajectory of four types of household appliances.

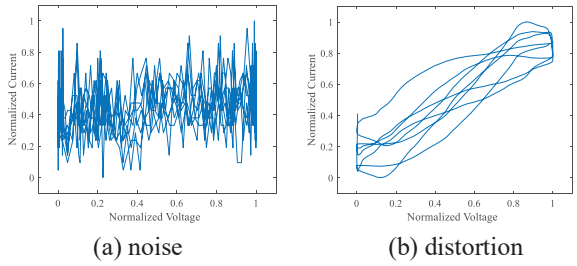


Fig. 4. V-I Trajectory of four types of household appliances.

3.3 Data preprocessing

Noise not only make the characteristics of V-I trajectory category representation are disappeared, but also it leads to trajectory distortion. Fig. 4(a) shows that V-I trajectory of rice cooker is interfered by noise. Compared with the ideal V-I trajectory in Fig.4(a), the V-I trajectory distortion of rice cooker arises from noise interference in Fig.4(b)

3.4 V-I Trajectory image Data Screening Algorithms

The low quality data in Fig.4 can cause a decrease in recognition accuracy of electrical appliances. However, until now, there is still no efficient method that can improve data quality of V-I trajectory. In order to obtain the high quality data, further than, to improve the recognition accuracy of electrical appliances, a data processing algorithm is illuminated in this section.

1)Data evaluation

The quality of the measured electrical signal is different with the variant of environment and time. In order to distinguish bad data, it is need to evaluate the quality of the measured current signal before data processing. Here, according to the correlation between the signal and the normal reference signal, the quality evaluation is preliminary carried out.

In order to research the correlation of data, Pearson correlation coefficient is used to study the linear correlation between two household appliances [19]. It is defined as the quotient of covariance and standard deviation of two samples, and then we have

$$\rho_{X,Y} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} = \frac{\mathrm{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{1}$$

In (1), is the mean of the sample, is the standard deviation of the sample, and n is the number of samples. is the correlation coefficient of the two sample data, which indicates the present correlation of the two data. The value range of is [-1, 1]. The correlation coefficient corresponds to the correlation degree as shown in Table 2.

Table 2. Coefficient of correlation and degree of correlation

$\rho_{X,Y}$ range	Degree of correlation
[0.0 0.3]	No correlation or micro correlation
[0.3 0.5]	Medium correlation
[0.5 0.8]	Strong correlation
[0.8 1.0]	Extremely strong correlation

2)Data Screening Algorithms

In a real environment, the voltage and current signals are inevitably interfered by noise in the processes of transmission. The distortion V-I trajectory, which is plotted by using the voltage and current signal data, will result in a pool train effect of learning network and a low accuracy of electrical appliance recognition, even if the quality of the measured data is improved by the data evaluation and filter. In tradition method, the V-I trajectory data is eliminated as long as them is some difference from the ideal data and this could result in the loss of a lot of useful data. Therefore, in the Section, a data screening algorithm based on restricted domain(R-domain) is proposed to eliminate effectively the exceptional data and improve the utilization of data.

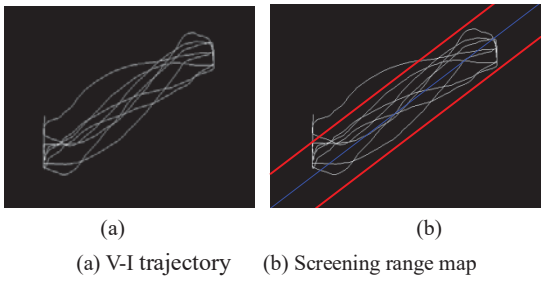


Fig. 5. Screened V-I trajectory

The V-I trajectory to be screened is shown in Fig. 5 (a), and the R-domain is shown in Fig. 5 (b), if the image data is $m \times n$, the central line (the thin blue line in Fig. 9(b)) can be expressed as:

$$y = kx + b_1 \tag{2}$$

In the formula, x is the pixels of rows, y is the pixels of the corresponding columns on the straight line, $k$is the slope and b_1 is the intercept.
The R-domain is defined as the range between the upper and lower thick red lines in the trajectory shown in Fig. 5 (b). Therefore, the R-domain can be defined as

$$y = kx + b_1 \begin{cases} y_1 = kx + b_1 - b \\ y_2 = kx + b_1 + b \end{cases} \tag{3}$$

Where, b is the change value of the line intercept that is the number of pixels in which the straight line moves up and down.y_1 is the upper boundary of the R-domain (the thick red line on the center line in Fig. 5(b)), and y_2 is the lower boundary of the R-domain (the thick red line at the lower part of the center line in Fig. 5(b)).

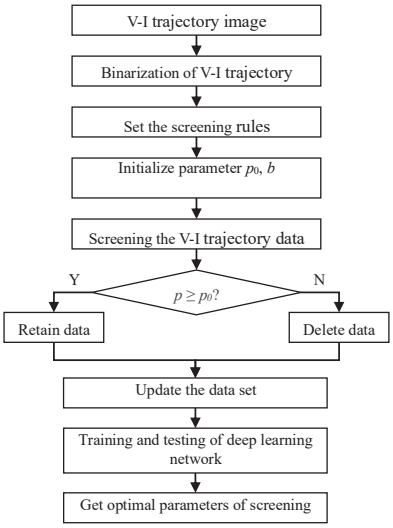


Fig. 6. Flow chart of data screening algorithm based on R- domain

Based on finite restricted domain(R-domain), the algorithm is described as follows:

Step1: Transform the V-I trajectory images into gray images, into black-white images.

Step2: Set screening rules. Let p is defined as the percentage of pixel number in R-domain to the total pixel number of the V-I trajectory image. For the V-I

trajectory of resistive appliances, if p is greater than or equal to a threshold value ($p0$), the trajectory image is selected as a resistive sample of the training data set of deep learning network, otherwise the image data is discarded. For the V-I trajectory of inductive appliances, if $p0$ is smaller than the threshold value, the V-I trajectory is selected as an inductive sample, otherwise the image data is also discarded.

Step3: Initialize various parameter (such as the threshold, y_1 and y_2) values.

Step4: Adjust the size of R-domain based on the initialization threshold. Based on step2, screened data sets are used to train and test the deep learning network, and the optimal parameters b is obtained.

The diagram of the screening algorithm is shown in Fig.6.

The key of the data screening algorithm is to select the appropriate R-domain. From the equation (3), the R-domain is change with the intercept b. So, what is the value of b is important for the data screening algorithm. The Fig.7 shows that the values of p are different with b the variant of b.

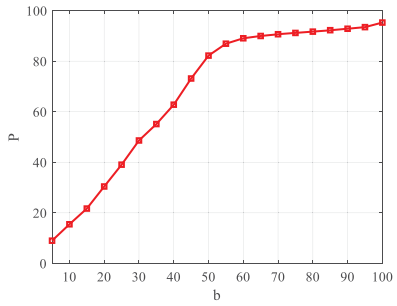


Fig. 7. The values of p are different with b the variant of b

4 V-I trajectory image data preprocessing

4.1 The improved Convolutional Neural Network

The convolutional neural network can automatically extract features directly from the image to complete the classification and recognition task. It can also

avoid the disadvantages of the artificial feature in traditional identification methods [20]. Therefore, the convolutional neural network is has attracted widespread attention in pattern recognition. However, it is necessary to construct a deep learning network and then adjust the network model according to the electrical principle and characteristics of household appliances data. AlexNet is a classical model of convolutional neural network (CNN) in image classification [21]. The network is consists of five convolutional layers, three pooling layers, three fully connected layers, and finally connects the softmax classification layer. ZFNet is fine-tuning model of AlexNet [22]. In this paper, ZFNet is used to conduct initial training on household appliances, and improve the network. In order to make the network suitable for the electrical appliance identification and save the cost of training, we improves the ZFNet into a network with four convolutional layers, three pooling layers and two fully connected layers as shown in Fig. 8 ($referredtoasZFNet_9$).

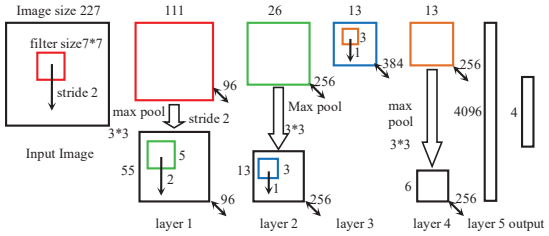


Fig. 8. Structural schematic of the improved network

4.2 Deep transfer learning for household appliances recognition

In order to solve the problem of the small sample data and high cost for the deep leaning network, transfer learning [23] is used to study the method of household appliances recognition. The household appliances identification block diagram of deep transfer learning is shown in Fig. 9.

Source data sets are usually large-scale labeled data sets. The selection of source data sets is very important in transfer learning. Here, ImageNet set, which is the largest database in image recognition, is used as the source data set to train learning model.

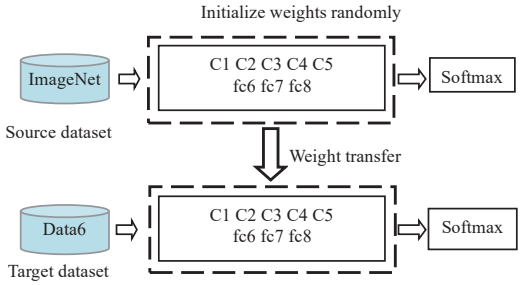


Fig. 9. The block diagram of deep transfer learning

4.3 HOG feature of household appliances

After measuring the voltage and current data of household appliances and constructing the data set of V-I trajectory of appliances, the local features of HOG [24] is extracted and input into the SVM to classify the household appliances in the next step. Fig. 10 shows the visualization of HOG features. The left image is the original image of the feature extracted,and the right image is the visual image of the extracted HOG feature.

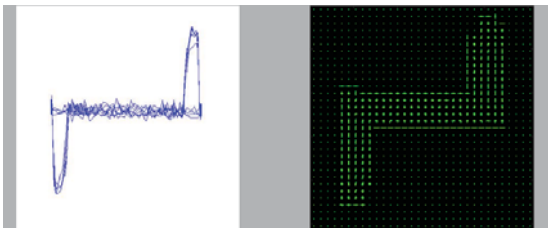


Fig. 10. HOG feature visualization

4.4 Household appliances recognition based on area feature of V-I trajectory

Since 2007, H.Y. Lam used V-I trajectory to identify household appliances. However, they mainly studied the shape feature of steady-state V-I trajectory of household appliances, such as total area, area of left and right partitions, asymmetry, cyclic direction, average curvature, self-intersection points, peak value of

middle segment, slope of middle segment, etc. Each shape feature is related to some characteristics of household appliances [25].

In order to compare with the household appliances recognition method based on deep transfer learning in this paper, the traditional method will be used to extract the relative pixel area feature of V-I trajectory, and the SVM will be used to identify appliances.

The area around the V-I trajectory is the total area of the trajectory, which can be calculated directly according to the coordinates of the trajectory points. However, when the V-I trajectory of appliances is complex, it is not easy to calculate the total area by these methods. For this reason, this paper proposes the relative pixel area feature based on the traditional total area feature of V-I trajectory.

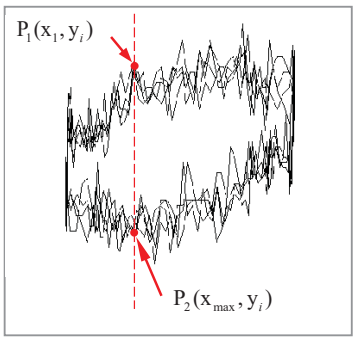


Fig. 11. Schematic diagram for calculating relative pixel area

As shown in Fig. 11, if the V-I matrix of the appliance is Img, the size is $m \times n$, after binarization, the position vector of all the trajectory pixels in each column of the image matrix Img is $P_1(x_1, y_i), P_1(x_2, y_i), \cdots, P_1(x_{\max}, y_i),$, and the minimum row pixel $P_1(x_1, y_i)$ and the maximum $P_2(x_{\max}, y_i)$ are found, take the row difference $(x_{\max} - x_1)$, which is called the column area. The sum of all the column areas is the total pixel area. The relative value of the total pixel area to the total number of pixels is defined as a Relative Pixel Area (RPA).

Relative pixel area can be described as:

$$RPA = \frac{\sum_{i=1}^{n} (x_{\max} - x_{\min})|_{y=y_i}}{m \times n} \tag{4}$$

In the formula, m and n respectively represent the number of rows and columns of the image matrix Img;x_1 represents the minimum number of rows of the $i-th$ trajectory pixel in the V-I trajectory;$x_{\max}$ is the maximum number of rows of the $i-th$ trajectory pixel in the V-I trajectory.
The relative pixel area features are extracted from the V-I trajectory of appliances in the data set. The distribution of the features is shown in Fig. 12. From Fig.12, there is a clear distinction between the different appliances.

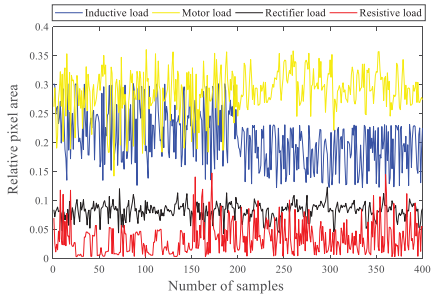


Fig. 12. Relative Pixel Area of Household appliances

5 V-I trajectory image data preprocessing

The low quality data in Fig.4 can cause a decrease in recognition accuracy of electrical appliances. However, until now, there is still no efficient method that can improve data quality of V-I trajectory. In order to obtain the high quality data, further than, to improve the recognition accuracy of electrical appliances, a data processing algorithm is illuminated in this section.

5.1 Data screening experiment

An experimental platform is built to simulate the power environment of a real family, as shown in Fig.13. The household appliances to be tested are placed

in normal working state, and their voltage and current signals are measured while the household appliances work steadily.

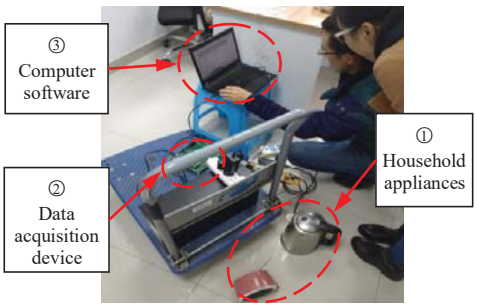


Fig. 13. Data acquisition schematic.

In order to find the optimal R-domain and achieve the best screening effect, a detailed analysis is carried out. First, the threshold $p0$ is initially set to70%. All data of the resistive load and inductive load are screened, and the parameter b is taken as 30, 40, 50, 60, 70, 80, 90, and 100. After screening data, and nine data sets of 400 V-I trajectory of each type of appliances were constructed using the filtered data.

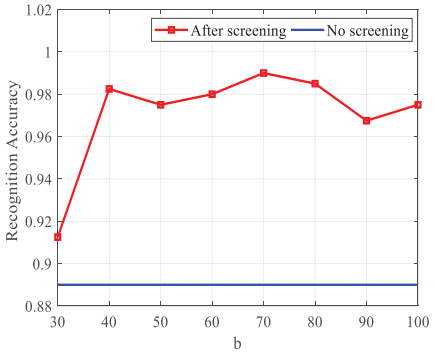


Fig. 14. Recognition accuracy

There are 300 V-I trajectory images for each kind of appliances in the training sample set, and 100 V-I trajectory images for each kind of appliance are used as test samples. The recognition results of the four classes of appliances are shown in Table 3. The recognition results with/without screening algorithm are shown in Fig. 14.

Table 3. Recognition results

b	Data set	Recognition accuracy%				
		Resistive load	Inductive load	Rectifier load	Motor load	Average
No screened	Data1	95	94	90	77	89.00
30	Data2	100	66	99	100	91.25
40	Data3	98	96	99	100	98.25
50	Data4	100	94	96	100	97.50
60	Data5	100	96	96	100	98.00
70	Data6	100	100	96	100	99.00
80	Data7	100	99	95	100	98.50
90	Data8	99	98	90	100	96.75
100	Data9	99	97	94	100	97.50

Table 3 shows that when the data set is Data6, the average recognition rate is 99%, which is the highest after screening.

As can be seen from Fig. 14, the recognition rate of the dataset Data1 without the screening algorithm is only 89%, which is lower than the recognition rate of other screened datasets. It can be seen that the recognition effect by using the data screening algorithm is significantly improved.

At the same time, after deep learning training, the loss and accuracy in the training process are analyzed, and the curves of loss with the iterations under different R-domains are shown in Fig. 15.

As can be seen from Fig. 15, as the iteration increases, the loss gradually decreases. For the same iteration (the 100th iteration), when the data set is Data7, the loss is the smallest. When the data set is data6, the loss is the second. Thus, under the same network model, when the data set is Data6 and

Data7, the network converges faster.

In summary, the data set Data6 is the best in all data set after screening. It shows that b=70 is the optimal parameter in R-domain. The subsequent experiments in this paper will use the data set Data6 as the V-I trajectory image data set for appliance identification.

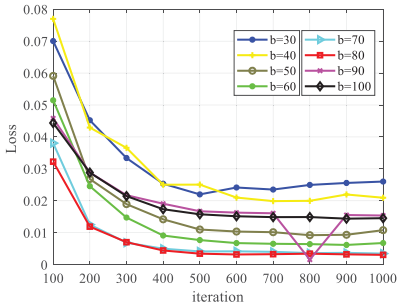


Fig. 15. Loss with iteration

5.2 Recognition results based on improved Convolutional Neural Network

Screened dataset Data6 is used to train the $ZFNet_9$ ($improvedZFNet$) model under the framework of CAFFE (Convolution Architecture for Feature Extraction). The maximum iterations is 5000. The recognition results based on the train data set are shown in Table 4.

Table 4. Recognition results based on the train data set

Network model	Layer number	Training time (h)	Test accuracy
ZFNet	11	6.25	72.5%
$ZFNet_9$	9	5.46	78%

As can be seen from the Table4, the training time of $ZFNet_9$ is about 12% shorter than before, and the recognition rate is also increased by about 7.5%.

It can be seen from the above that, the training speed and recognition rate are improved by using the proposed network. In the training process, the loss and accuracy of the network are shown in Fig. 16.

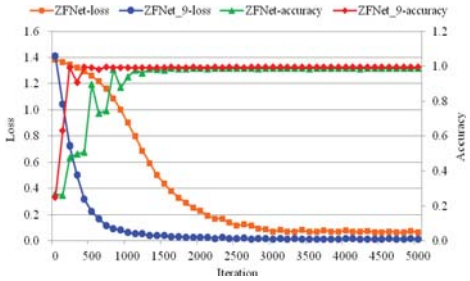


Fig. 16. Comparison of training process between ZFNet and $ZFNet_9$ model

It can be seen from the Fig.16 that the speed of convergence is accelerated and training time is induced by using the network improved.

From the above experiments, it can be seen that although the training time has been reduced to a certain extent and the recognition accuracy has been improved, the improvement is limited. The recognition accuracy of the electrical appliances is still needs to be improved further due to the fewer sample data in V-I trajectory image data set. The deep learning training of small sample data can easily lead to over-fitting. Only when the size of the sample data set is large enough, high quality features can be obtained. However, the sample data are not easy to get in practical application.

In order to further improve the recognition accuracy of electrical appliances, the proposed method, based on deep transfer learning, is applied to identify household appliances.

5.3 Recognition results based on deep transfer learning

ImageNet is the source data set, which is trained by AlexNet model. After carrying out 310000 iterations, the network parameters are got and are transferred to the training network of V-I trajectory image data set. The target data set is Data6.The maximum iterations is set to 1000. The experimental results

with/without transfer learning are shown in Table 5.

Table 5. Experimental results with/without transfer learning

Training method	Recognition accuracy	Training time (h)
No Transfer learning	78%	5.46
Transfer learning	99%	2.73

From the Table5, it can be seen that the accuracy of household appliances recognition has been significantly improved and the training time has been shortened by nearly 50%. It can be seen from the above that the transfer learning not only greatly improves the accuracy of appliances identification, but also obviously reduces the training time.

The loss of the training process of 2000 iterations is shown in Fig.17. Here, $AlexNet_TL$ denotes the training process after using transfer learning and AlexNet represents the training process without transfer learning.

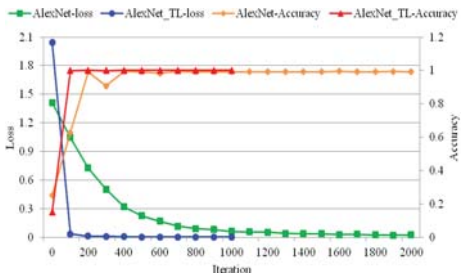


Fig. 17. Comparison of training processes

It can be clearly seen from the Fig. 16 that the accuracy after using transfer learning algorithm can faster reach a stable value than the method without using the transfer learning algorithm, and the loss converges faster than that before the transfer.

To sum up, compared with before transfer learning, the proposed method

not only shortens the training time of network obviously and accelerates the convergence, but also greatly improves the accuracy of household appliances recognition. The recognition ratio achieves up to 99%.

In the traditional method, the local feature and relative pixel area feature of HOG are extracted from V-I trajectory image, and the recognition of household appliances have been successfully implemented. HOG features are extracted in the image and relative pixel area features are extracted in the trajectory image. To test the performance of the proposed method, two traditional methods (HOG+SVM, RPA+SVM) are used to realize the recognition of household appliances. The experimental results of the three methods are shown in Table 6.

Table 6. Comparison of recognition effectiveness

Method	Transfer learning	HOG+SVM	RPA+SVM
Accuracy	99%	95%	88%

As can be seen from the Table 6 above, when the relative pixel area (RPA) of V-I trajectory is used as feature, the accuracy of the recognition is relatively low. When extracting the HOG local features of V-I trajectory, the recognition rate is higher. The recognition rate of household appliances is the highest with the transfer learning algorithm, which shows that the recognition effect of the household appliance recognition method based on deep transfer learning is the best.

6 Conclusion

In this paper, based on machine learning theory, the recognition method of household appliances based on deep learning is studied. The experimental results show that the method has a good recognition effect and has a good prospect of engineering application. This paper has two major contributions: (1) Aiming at the problem that the measured data of household appliances are disturbed by a large amount of noise, which results in the unavailability of a large number of V-I trajectory image data of household appliances, a data screening algorithm based on the V-I trajectory of household appliances in restricted domain is

proposed. And then the data sets of V-I trajectory of household appliances are constructed by using the data of V-I trajectory images screened by different parameters. Based on the deep learning network, the optimal screening parameter b=70 and the best-performing data set Data6 are found. (2) In order to solve the problem that new household appliances can not be recognized and retraining takes a lot of time, and the sample size of V-I trajectory is small, which results in unsatisfactory recognition effect, the transfer learning is introduced to study household appliances recognition. Up to 99% of the household recognition accuracy is achieved. This makes it more in line with the actual situation of appliances identification, reduces the training time, and improves the recognition effect of the model. The experimental results show that the proposed method has more advantages than the traditional method. The proposed method has strong robustness, because it can reduce the influence of noise interference in the measured data to a certain extent.

7 Acknowledgement

This work was supported by the Equipment Development Department Funds grant(61403120304) , the Fundamental Research Funds for the Central Universities grant (2019XJ04), and Science and Technology Major Project of the Science and Technology Department of Sichuan Province (2018GZDZX0043).

References

1. Nur Farahin Esa, Md Pauzi Abdullah, and Mohammad Yusri Hassan. A review disaggregation method in non-intrusive appliance load monitoring. *Renewable and Sustainable Energy Reviews*, 66:163–173, 2016.
2. Sarah Darby et al. The effectiveness of feedback on energy consumption. *A Review for DEFRA of the Literature on Metering, Billing and direct Displays*, 486(2006):26, 2006.
3. Haroon Rashid, Pushpendra Singh, Vladimir Stankovic, and Lina Stankovic. Can non-intrusive load monitoring be used for identifying an appliance¡¯s anomalous behaviour? *Applied energy*, 238:796–805, 2019.
4. B Neenan, J Robinson, and RN Boisvert. Residential electricity use feedback: A research synthesis and economic framework. *Electric Power Research Institute*, 3, 2009.

5. Soumyajit Ghosh, Arunava Chatterjee, and Debashis Chatterjee. Improved non-intrusive identification technique of electrical appliances for a smart residential system. *IET Generation, Transmission & Distribution*, 13(5):695–702, 2018.
6. Michael Zeifman and Kurt Roth. Nonintrusive appliance load monitoring: Review and outlook. *IEEE transactions on Consumer Electronics*, 57(1):76–84, 2011.
7. George William Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992.
8. George W Hart and Anastasios T Bouloutas. Correcting dependent errors in sequences generated by finite-state processes. *iEEE Transactions on information Theory*, 39(4):1249–1260, 1993.
9. José A Hoyo-Montaño, Naim León-Ortega, Guillermo Valencia-Palomo, Rafael A Galaz-Bustamante, Daniel F Espejel-Blanco, and Martín G Vázquez-Palma. Non-intrusive electric load identification using wavelet transform. *Ingeniería e Investigación*, 38(2):42–51, 2018.
10. D Srinivasan, WS Ng, and AC Liew. Neural-network-based signature recognition for harmonic source identification. *IEEE Transactions on Power Delivery*, 21(1):398–405, 2005.
11. Hsueh-Hsien Chang. Non-intrusive demand monitoring and load identification for energy management systems based on transient feature analyses. *Energies*, 5(11):4569–4589, 2012.
12. Dominik Egarter, Venkata Pathuri Bhuvana, and Wilfried Elmenreich. Paldi: Online load disaggregation via particle filtering. *IEEE Transactions on Instrumentation and Measurement*, 64(2):467–477, 2014.
13. Taha Hassan, Fahad Javed, and Naveed Arshad. An empirical investigation of vi trajectory based load signatures for non-intrusive load monitoring. *IEEE Transactions on Smart Grid*, 5(2):870–878, 2013.
14. Hong Yin Lam, GSK Fung, and WK Lee. A novel method to construct taxonomy electrical appliances based on load signaturesof. *IEEE Transactions on Consumer Electronics*, 53(2):653–660, 2007.
15. A Longjun Wang, B Xiaomin Chen, C Gang Wang, and D Hua. Non-intrusive load monitoring algorithm based on features of v–i trajectory. *Electric Power Systems Research*, 157:134–144, 2018.
16. Leen De Baets, Joeri Ruyssinck, Chris Develder, Tom Dhaene, and Dirk Deschrijver. Appliance classification using vi trajectories and convolutional neural networks. *Energy and Buildings*, 158:32–36, 2018.
17. Wen-Sheng Wang, Jin Wang, and KW Wang. Application of som neural network and c means method in load classification. In *Proceedings of the CSU-EPSA*,

volume 23, pages 36–39. Editorial Board of Proceedings of the CSU-EPSA, School of Electrical ¡-, 2011.

18. KH Ting, Mark Lucente, George SK Fung, WK Lee, and SYR Hui. A taxonomy of load signatures for single-phase electric appliances. In *IEEE PESC (Power Electronics Specialist Conference)*, pages 12–18, 2005.
19. Kaustav Basu, Vincent Debusschere, Ahlame Douzal-Chouakria, and Seddik Bacha. Time series distance-based methods for non-intrusive load monitoring in residential buildings. *Energy and Buildings*, 96:109–117, 2015.
20. Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Computer Science*, 2015.
21. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
22. Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
23. Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.
24. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. 2005.
25. Nur Iksan, Jaka Sembiring, Nanang Haryanto, and Suhono Harso Supangkat. Appliances identification method of non-intrusive load monitoring based on load signature of vi trajectory. In *2015 International Conference on Information Technology Systems and Innovation (ICITSI)*, pages 1–6. IEEE, 2015.

Optimizing the Green Open Vehicle Routing Problem by Membrane-Inspired Hybrid Heuristic Algorithm

Yunyun Niu[1], Zehua Yang[1], and Jianhua Xiao[2⋆]

[1] School of Information Engineering,
China University of Geosciences in Beijing,
Beijing 100083, China
[2] The Research Center of Logistics, Nankai University,
Tianjin 300071, China
jhxiao@nankai.edu.cn

Abstract. In this work, a membrane-inspired hybrid heuristic algorithm, MIHA, is proposed to deal with the green open vehicle routing problem with time windows (GOVRPTW). The MIHA has a cell-like and three-level nested membrane structure. Tabu search, genetic algorithm and neighborhood search algorithms were used as sub-algorithms of MIHA. Elementary membranes added extra attractors to tabu search. Genetic algorithm, especially the crossover operator, was designed to retain good gene segments of solutions. Tabu search helped genetic algorithm escape from local optimal solutions. They were combined in the framework of membrane system. Computational experiments were performed on realistic instances based on the real road conditions of Beijing, China. Experimental results showed that our algorithm was more competitive than peer algorithms.

Keywords: membrane computing, P system, open vehicle routing problem, carbon emission, tabu search

1 Introduction

The most widely studied and used model for route planning is the vehicle routing problem (VRP) introduced by Dantzig and Ramser in 1959 [1]. In short, VRP is the determination of the optimal set of routes to be performed by a fleet of vehicles, in order to satisfy the demand of a given set of customers. Several variants of this problem have been formulated and studied. In particular the vehicle routing problem with time windows (VRPTW) is used for its practical applications. In this variant customers must be served in a specific time interval, which are included as time windows constraints [2]. In other problems vehicles need not return to the depot after delivering goods to customers, and instead go to other locations. This variant of VRP, in which vehicles do not return to the

⋆ Corresponding author

depot is called the open vehicle routing problem (OVRP) [3] or the open vehicle routing problem with time windows (OVRPTW) when time windows constraints are specified [4].

In 1983, Bodin et al. [5] proposed the first solution approach for the OVRP. From then on, several researchers have used various heuristics and meta heuristics to solve the OVRP. The most common meta heuristics are based on search, such as tabu search [6–9], neighbourhood-based search [10–13], or threshold accepting algorithm [14, 15]. Other researchers have applied bio-inspired and populations-based meta heuristics, such as particle swarm optimization [16–18], ant colony optimization [19, 20] or genetic and evolutionary computing [21–23].

Outsourcing logistics operations to third party logistics would reduce costs by better resources utilization and operations efficiency in freight transportation. Companies hire vehicles from other companies for delivering their goods. The vehicles will not be supposed to return to the company as usual. The problem can be modeled as a variant of the OVRP. Nowadays green road freight transportation has become a hot topic all over the world. The logistics industry is under the pressure of reducing carbon emission. Green open vehicle routing problems (GOVRPs) were proposed based on fuel consumption models [24]. These problems are NP-hard which makes them intractable with large instance problems. Until now, there have been few solving methods. Lots of work needs to be done in this area.

Membrane computing is a branch of natural computing. It provides a parallel distributed computing model called P system [25, 26]. Some types of P systems have been used as modeling notations for ecosystems and pedestrian behavior [27–30]. Moreover, P systems have provided nondeterministic frameworks and distributed parallel for computing or optimizing that have been applied in various aspects of engineering [31–37]. Readers can find circumstantial evaluations of miscellaneous P systems in the literature [38–41].

In this work, a membrane-inspired hybrid heuristic algorithm, MIHA, was proposed to deal with the green open vehicle routing problem (GOVRPTW). The MIHA has a cell-like and three-level nested membrane structure, see Figure 1. Skin membrane is the first level, where genetic algorithm (GA) is implemented. The adjacent inner membranes labeled by 1, ..., 6 are the second level, where tentative solutions can be found by different tabu algorithms. In each level-2 membrane, there is an elementary membrane called a level-3 membrane, where neighborhood search operations will be done to help adjust the search direction of corresponding level-2 membrane. There are also communication channels between skin membrane and level-2 membranes. On the one hand, the GA operators, especially the crossover operator, were designed to retain good gene segments. On the other hand, tabu search algorithms with different attractors help GA escape from local optimal solutions. Experimental results proved our opinions very well. Comparing with peer algorithms, the MIHA turned out to be more competitive.

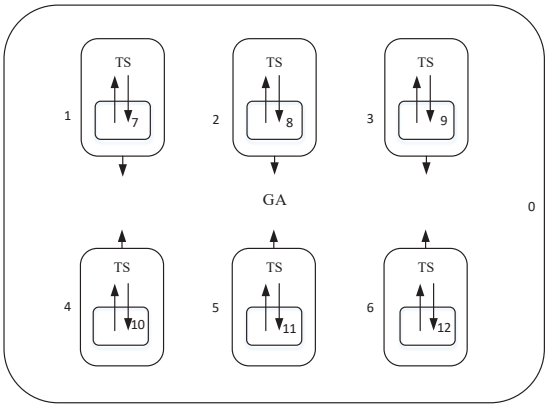


Fig. 1: Three subsystems of MIMOA

2 Fuel Consumption Model of Open Vehicle Routing Problem

The green open vehicle routing problem (GOVRPTW) is defined on a complete directed graph $\mathcal{G} = (N, A)$ where $N = \{0, \ldots, n\}$ is the set of nodes, $A = \{(i, j) : i, j \in N, i \neq j, j \neq 0\}$ is the set of arcs, and node 0 corresponds to the depot. The customer set is $N_0 = N \setminus \{0\}$, and each customer i has a positive demand q_i. The distance from i to j is denoted by d_{ij}. Variable f_{ij} is the total amount of flow on each arc $(i, j) \in A$. Let w be the weight of a vehicle. Therefore, the total load of vehicle on arc (i, j) is $w + f_{ij}$. The binary variable x_{ij} is equal to 1 if and only if a vehicle travels on arc $(i, j) \in A$. The binary variable z_{ij}^{rh} is equal to 1 if and only if a vehicle travels on arc $(i, j) \in A$ at speed $v^r,\ r = 1, \ldots, R$. Furthermore, t_i corresponds to the service time of node $i \in N_0$, which must start within the time window $[a_i, b_i]$. If a vehicle arrives at customer i before a_i, it will wait until a_i before servicing the node. y_j is the service start time at $j \in N_0$. The total time spent on a route in which $j \in N_0$ is the last visited node before returning to the depot is defined by s_j. As vehicles are hired from other companies, they depart from the depot at different times $y_{0i} \in [a_0, b_0]$. Vehicles do not come back to the depot after serving customers. We used the comprehensive emissions model of Barth et al. (2005), Scora and Barth (2006), and Barth and Boriboonsomsin (2008) to estimate fuel consumption and emissions for a given time instant [42–44]. The mathematical model of GOVRPTW is defined as follows.

$$Minimize \sum_{(i,j)\in A} \lambda f_c k N V d_{ij} \sum_{r=1}^{R} z_{ij}^r / v^r \tag{1}$$

$$+ \sum_{(i,j)\in A} \lambda f_c \gamma \alpha_{ij} d_{ij} (w x_{ij} + f_{ij}) \tag{2}$$

$$+ \sum_{(i,j)\in A} \lambda f_c \beta \gamma d_{ij} \sum_{r=1}^{R} (v^r)^2 z_{ij}^r \tag{3}$$

$$+ \sum_{j\in N_0} f_d s_j \tag{4}$$

subject to

$$\sum_{j\in N_0} x_{0j} \leq |N_0| \tag{5}$$

$$\sum_{i\in N} x_{ij} = 1,\ \forall j \in N_0 \tag{6}$$

$$\sum_{j\in N} x_{ij} \leq 1,\ \forall i \in N_0 \tag{7}$$

$$\sum_{i=1}^{n} x_{i0} = 0 \tag{8}$$

$$\sum_{j\in N} f_{ij} - \sum_{j\in N} f_{ji} = q_i,\ \forall i \in N_0 \tag{9}$$

$$q_j x_{ij} \leq f_{ij} \leq (Q - q_i) x_{ij},\ \forall (i,j) \in A \tag{10}$$

$$y_i - y_j + t_i + \sum_{r=1}^{R} d_{ij} z_{ij}^r / v^r \leq M_{ij}(1 - x_{ij}),\ \forall i \in N, j \in N_0, i \neq j \tag{11}$$

$$a_i \leq y_i \leq b_i,\ \forall i \in N_0 \tag{12}$$

$$\sum_{r=1}^{R} z_{ij}^r = x_{ij},\ \forall (i,j) \in A \tag{13}$$

$$x_{ij} \in \{0,1\},\ \forall (i,j) \in A \tag{14}$$

$$z_{ij}^r \in \{0,1\},\ \forall (i,j) \in A,\ r = 1,\ldots,R \tag{15}$$

$$f_{ij} \geq 0,\ \forall (i,j) \in A \tag{16}$$

$$y_i \geq 0,\ \forall i \in N_0 \tag{17}$$

The objective of the GOVRPTW is to minimize the total cost of fuel consumption, CO_2 emissions, and the total driver wage. The first three terms of the objective function represent the cost of fuel consumption and of CO_2 emissions. In particular, term (1) computes the cost induced by the engine module, term (2) reflects the cost induced by the weight module and term (3) measures the cost induced by the speed module, where $\lambda = \xi/\kappa\psi$, $\gamma^h = 1/1000 n_{tf} \eta$ and $\alpha = \tau + gsin\theta + gC_r cos\theta$ are constants, and $\beta = 0.5 C_d \rho A$ is a vehicle-specific constant. Finally, term (4) computes the total driver wage. A list of and values for parameters for light-duty vehicles is given in Table 1.

Table 1: Vehicle parameters.

Notation	Description	Typical values
ξ	Fuel-to-air mass ratio	1
g	Gravitational constant (m/s^2)	9.81
ρ	Air density (kg/m^3)	1.2041
C_r	Coefficient of rolling resistance	0.01
η	Efficiency parameter for diesel engines	0.45
f_c	Fuel and CO2 emissions cost (£/liter)	1.4
f_d	Driver wage $(£/s)$	0.0022
κ	Heating value of a typical diesel fuel (kj/g)	44
ψ	Conversion factor $(g/s$ to $L/s)$	737
n_{tf}	Vehicle drive train efficiency	0.45
v^l	Lower speed limit (m/s)	5.5 (or 20 km/h)
v^u	Upper speed limit (m/s)	27.8 (or 100 km/h)
θ	Road angle	0
τ	Acceleration (m/s^2)	0
w	Curb weight (kg)	3500
Q	Maximum payload (kg)	4000
f	Vehicle fixed cost (£/ day)	42
k	Engine friction factor $(kj/rev/liter)$	0.25
N	Engine speed (rev/s)	38.34
V	Engine displacement (liter)	4.5
C_d	Coefficient of aerodynamics drag	0.6
A	Frontal surface area (m^2)	7.0

The maximum number of vehicles available for each type is imposed by constraints (5). We consider an unlimited number of vehicles. Constraints (6)—(8) ensure that each customer is visited exactly once, and the vehicles do not need to return to the depot. Constraints (9) and (10) define the flows. Constraints (11) and (12) are time window constraints, where $M_{ij} = max\{0, b_i + s_i + d_{ij}/v^r - a_j\}$. Constraints (13) means that there is only one speed level for each arc.

3 Algorithm

In this section, a membrane-inspired hybrid heuristic algorithm, MIHA, is proposed to deal with the GOVRPTW. The MIHA has a cell-like and three-level nested membrane structure, see Figure 1. Skin membrane is the first level; and the adjacent inner membranes labeled by 1, ..., 6 are the second level. In each level-2 membrane, there is a elementary membrane called level-3 membrane. Level-2 membranes can send tentative solutions to the skin membrane through unidirectional channels. However, the communication channels between level-2 membranes to the corresponding level-3 membranes are bi-directional.

The main procedure of the MIHA is described in Algorithm 1. In initialization stage, initial solutions are generated in six level-2 membranes by using different

operators, then initial population is formed in the skin membrane. Searching processes in level-2 membranes are guided by tabu search algorithm, while evolution in the skin membrane is according to genetic operators. After every I_{guide} steps, archive solutions of each level-2 membrane can be sent to the skin membrane and help to update the current population P.

Algorithm 1 Main framework of MIHA

Require: maximum number of iterations I_{max}, Tabu-list size T, Archive size Ar, iteration number before sending archive solutions to the skin membrane I_{guide}, and population size P.
Ensure: x_{best}.
1: Initialization
2: **for** $i = 1$ to I_{max} **do**
3: **for** $j = 1$ to 6 **do**
4: Execute TS in level-2 membrane
5: Execute GA in the skin membrane
6: **if** i mod $I_{guide} == 0$ **then**
7: **for** $i = 1$ to 6 **do**
8: Send solutions in Archive to the skin membrane
9: Update current population
10: Evaluate current population, and select the best solution x_{best}

3.1 Initialization

In level-2 membranes, six different operators are used to generate different initial solutions. These operators are listed as follows.

(1) Random: Randomly choose routes which satisfy constraints (5)–(17).
(2) Nearest neighborhood heuristic (NNH): It creates a set of routes according to the distance from current node. The first route begins with an unrouted customer x_0, which is nearest to the depot. Then choose an unrouted customer x_1, which is nearest to x_0. Continue this process in the same way until no customer can be assigned to the route. Start a new route unless all the customers are routed.
(3) Modified nearest neighborhood heuristic (mNNH): It creates a set of routes according to both distance from current node and demand of the next customer. In a delivery system, a vehicle can reduce q_j payload after servicing customer j. Let's define $\overline{\Delta f_{ij}} = q_j/d_{ij},\ i \in N,\ j \in N_{ucs}$, where N_{ucs} is defined as the unrouted customer set. The mNNH creates a set of routes sequentially. The first route begins with an unrouted customer $r = argmax_{j \in N_{ucs}}\{|\overline{\Delta f_{0j}}|\}$. Then, calculate $\overline{\Delta f_{ij}}$ between the current node r and the other unrouted nodes. Choose the node with the greatest current $\overline{\Delta f_{ij}}$ value as the next node if the node does not violate any of the constraints. Then, update the current node r and search for the next node in

the same way. When no customer can be assigned to the route, a new route is started. If all the customers are already routed, the process stops.

(4) Insert I_1 algorithm: Customer u^* is selected by using Equation 18 as follows based on the insertion heuristics I_1 [45]. The best possible position for node u is also calculated by Equation 20.

$$c_2(i^*, u^*, j^*) = optimum[c_2(i, u, j)] \tag{18}$$

$$c_2(i, u, j) = \lambda d_{0u} - c_1(i, u, j),\ \lambda \geq 0 \tag{19}$$

$$c_1(i^*, u, j^*) = min[c_1(i, u, j)], \tag{20}$$

$$c_1(i, u, j) = \alpha_1(d_{iu} + d_{uj} - \mu d_{ij}) + \alpha_2(y_{ju} - y_j),\ \mu, \alpha_1, \alpha_2 \geq 0,\ \alpha_1 + \alpha_2 = 1 \tag{21}$$

(5) Earliest deadline first: Customers are selected according to earliest deadline. This operator tries to choose the customer with earliest close time in each step.

(6) Shortest waiting time first: This operator tries to select customers with shortest waiting time.

After initialization, neighborhood search algorithm are used in each level-2 membrane to generate 20 neighbors. The total 120 neighbors will be sent to the skin membrane to form the initial population for the genetic algorithm.

3.2 Neighborhood search algorithm

Three neighborhood search algorithms are used in our MIHA method.

(1) Random operator: It selects two nodes randomly from the solution and randomly finds possible positions for them.

(2) High-cost-node improvement operator: The operator tries to reassign the high cost node $u^* = argmax_{u \in N}\{d_{iu} + d_{uj}\}$, where i is the preceding node, and j is the following node.

(3) Long-wait-time improvement operator: The operator tries to reassign the node with long wait time $u^* = argmax_{u \in N}\{a_u - e_u\}$, where e_u is the arriving time at customer u.

3.3 Tabu search in level-2 membrane

After initialization, tabu search is implemented in each level-2 membrane, see Algorithm 2. First, use current solution to create neighbors. If any neighbor solution is better than others in archive, replace the worse one. Randomly choose one solution from the archive and send it to the inner membrane. Use this solution to create neighbors in the corresponding level-3 membrane. Compare each of these neighbor solutions to solutions in archive. If it is better than some solution in archive, it will replace that solution. Then select the best solution which is in the archive but not in tabu list at the same time, and update the current solution. Finally, add current solution into the tabu list.

Algorithm 2 Tabu search in level-2 membrane

1: $neighbors$ =NeighborSearch(x_i)
2: UpdateArchive($Archive, neighbors$)
3: $neighbors = InnerMembrane(neighbors)$
4: UpdateArchive($Archive, neighbors$)
5: x_i =SelectCurrentSolution($Archive, TabuList$)
6: UpdateTabuList($TabuList, x_i$)

3.4 Evolution in skin membrane

After initialization, population in skin membrane evolve according to the GA. The binary tournament is adopted to choose parent chromosomes for genetic operators. A pair of chromosomes are chosen randomly, and the one with lower fitness value is picked out for reproduction. This process will be repeated until sufficient parent chromosomes are obtained. Route-exchange crossover [46] is used to retain better gene segment. Sequences of routes in one chromosome are reproduced and shared with other chromosomes. When a route is inserted to another chromosome as a new route, duplicated customers are deleted from the original route to ensure feasibility of the chromosome. Single point mutation is used as mutation operator in the skin membrane.

3.5 Communication with skin membrane

After every I_{guide} steps, archive solutions of each level-2 membrane can be sent to the skin membrane and help update the current population. They are merged with current individuals in skin membrane. The best P solutions are selected and combined to be a new population. GA operators implemented in skin membrane, especially the crossover operator, is designed to retain good gene segments of solutions found by tabu search in level-2 membrane. Moreover, various solutions obtained by tabu algorithms with different attractors help GA escape from local optimal solutions.

4 Computational Results

In this section, we tested the proposed algorithm on a set of real-world instances. We took the real road of Beijing city as the research background, and used the locations contains the urban zones of Beijing and suburban areas as experimental data [24].

The presented algorithm is coded in Matlab and was tested on a PC with an Intel Core 2.4 GHz processor, 8G RAM, and the Microsoft Windows 7 operating system. Parameters involved in our algorithm are listed in table 2.

Our algorithm is tested on 10 customer sets including 60 nodes to minimize the total cost. For each instance, the results collected over 20 runs are reported. Table 3–Table 6 are used to prove the effects of different parts in our algorithm. The columns display the best solution (BS), mean solution (MS), the worst solution (WS), standard deviation (STD), and elapsed time (ET).

Table 2: Parameters.

Notation	Description	Typical values
I_{max}	The maximum iteration number	500
I_{guide}	Iteration number before communicating with skin membrane	50
Ar	Archive size	100
Ns	Neighborhood size	100
L	Tabu-list size	30
$(\alpha_1, \alpha_2, \mu, \lambda)$	Insertion parameters	$(0.5, 0.5, 1, 1)$
P	Population size	100
p_1	crossover rate	1
p_2	mutation rate	0.1

4.1 Effect of search in level-3 membranes

In this subsection, the effect of search in level-3 membranes was analyzed. Experiments were conducted on the 60-node instances considering the use of light-duty vehicles. Level-3 membranes can provide another attractor to tabu searching process, and make it easier to obtain better solutions. The MIHA without level-3 membranes can be denoted as MIHA$_{-level3}$ for convenience. In Table 3, we compared the experimental results of MIHA and MIHA$_{-level3}$. It is proved that the level-3 membranes have great advantages in finding solutions with smaller total cost.

4.2 Effect of GA in skin membrane

In this subsection, we analyzed the effect of genetic algorithm in skin membrane. The MIHA without genetic algorithm can be denoted by MIHA$_{-GA}$. In MIHA, genetic algorithm in skin membrane obtains solutions from level-2 membranes and forms the initial population. During the next evolutionary process, crossover operator tries to combine good genetic segments from different individuals, while mutation operator helps extend the search scope. We listed the computational results by using MIHA and MIHA$_{-GA}$ in Table 4. It is proved that GA in skin membrane has good impact on the performance of our algorithm.

4.3 Effect of membrane structure

In this subsection, we tried to prove the effect of membrane structure. Membrane computing provides a parallel distributed framework. We denoted our algorithm without membrane structure to be MIHA$_{-MS}$. As shown in Table 5, without membrane framework, the MIHA$_{-MS}$ can not find competitive solutions as the results obtained by using MIHA.

Table 3: Computational results by using MIHA and MIHA$_{-level3}$.

	instance	BS(RMB)	MS(RMB)	WS(RMB)	STD(RMB)	ET(s)
MIHA	BJ60_01	10864.7334	10914.4491	10939.8273	21.2864	179.3093
	BJ60_02	10217.4691	10310.1949	10395.6818	54.3404	167.1625
	BJ60_03	11321.6848	11423.8391	11489.1644	53.3224	168.3861
	BJ60_04	11800.7563	11847.7284	11915.0534	31.7880	179.6456
	BJ60_05	10890.8377	10909.8830	10947.0117	14.5797	190.0507
	BJ60_06	11875.3018	11916.5380	11942.8835	21.5401	157.5884
	BJ60_07	12528.3080	12634.5020	12685.8953	54.0499	138.2126
	BJ60_08	11783.3490	11867.4716	11932.7904	45.1478	162.5045
	BJ60_09	11573.5410	11705.3953	11908.5312	107.6724	173.2149
	BJ60_10	12939.9900	13196.1938	13269.0754	90.2917	161.0214
	Average	11579.5971	11672.6195	11742.5914	49.4019	167.7096
MIHA$_{-level3}$	BJ60_01	10875.6553	10928.4416	10957.5933	22.3405	98.4386
	BJ60_02	10353.4233	10437.4359	10528.0329	46.5592	98.8066
	BJ60_03	11414.4174	11509.0854	11649.5848	76.5323	98.8509
	BJ60_04	11808.4673	11848.1219	11945.9823	35.8897	103.6991
	BJ60_05	10899.0969	10928.9037	10994.4671	28.3997	114.5028
	BJ60_06	11883.8319	11932.5517	11992.1479	36.4949	99.5624
	BJ60_07	12569.4389	12735.7666	12914.4059	114.1379	87.9780
	BJ60_08	11814.2420	11863.5460	11935.7837	40.0692	92.5350
	BJ60_09	11650.6475	11742.9264	11858.7899	72.9718	90.5239
	BJ60_10	13166.0801	13239.2438	13320.9522	41.7384	84.2999
	Average	11643.5301	11716.6023	11809.7740	51.5134	96.9197

Table 4: Computational results by using MIHA and MIHA$_{-GA}$.

	instance	BS(RMB)	MS(RMB)	WS(RMB)	STD(RMB)	ET(s)
MIHA	BJ60_01	10864.7334	10914.4491	10939.8273	21.2864	179.3093
	BJ60_02	10217.4691	10310.1949	10395.6818	54.3404	167.1625
	BJ60_03	11321.6848	11423.8391	11489.1644	53.3224	168.3861
	BJ60_04	11800.7563	11847.7284	11915.0534	31.7880	179.6456
	BJ60_05	10890.8377	10909.8830	10947.0117	14.5797	190.0507
	BJ60_06	11875.3018	11916.5380	11942.8835	21.5401	157.5884
	BJ60_07	12528.3080	12634.5020	12685.8953	54.0499	138.2126
	BJ60_08	11783.3490	11867.4716	11932.7904	45.1478	162.5045
	BJ60_09	11573.5410	11705.3953	11908.5312	107.6724	173.2149
	BJ60_10	12939.9900	13196.1938	13269.0754	90.2917	161.0214
	Average	11579.5971	11672.6195	11742.5914	49.4019	167.7096
MIHA$_{-GA}$	BJ60_01	10887.3386	10918.3082	10945.5878	19.3695	160.4212
	BJ60_02	10265.5463	10332.7347	10414.2800	47.0151	147.2111
	BJ60_03	11347.9277	11471.3450	11593.1845	75.5207	147.7990
	BJ60_04	11802.0435	11841.7084	11935.3495	37.6027	149.1469
	BJ60_05	10896.8324	10915.9710	10940.7710	12.3304	167.5803
	BJ60_06	11880.3074	11937.5616	11971.5064	28.7858	168.7334
	BJ60_07	12532.3226	12656.8304	12847.2887	83.8907	127.9422
	BJ60_08	11804.3203	11874.9623	11932.1919	38.2917	136.4170
	BJ60_09	11629.4012	11685.8443	11849.0145	61.8985	138.6707
	BJ60_10	13119.1625	13260.5937	13353.1819	73.0125	126.3682
	Average	11616.5203	11689.5860	11778.2356	47.7718	147.0290

Table 5: Computational results by using MIHA and MIHA$_{-MS}$.

	instance	BS(RMB)	MS(RMB)	WS(RMB)	STD(RMB)	ET(s)
MIHA	BJ60_01	10864.7334	10914.4491	10939.8273	21.2864	179.3093
	BJ60_02	10217.4691	10310.1949	10395.6818	54.3404	167.1625
	BJ60_03	11321.6848	11423.8391	11489.1644	53.3224	168.3861
	BJ60_04	11800.7563	11847.7284	11915.0534	31.7880	179.6456
	BJ60_05	10890.8377	10909.8830	10947.0117	14.5797	190.0507
	BJ60_06	11875.3018	11916.5380	11942.8835	21.5401	157.5884
	BJ60_07	12528.3080	12634.5020	12685.8953	54.0499	138.2126
	BJ60_08	11783.3490	11867.4716	11932.7904	45.1478	162.5045
	BJ60_09	11573.5410	11705.3953	11908.5312	107.6724	173.2149
	BJ60_10	12939.9900	13196.1938	13269.0754	90.2917	161.0214
	Average	11579.5971	11672.6195	11742.5914	49.4019	167.7096
MIHA$_{-MS}$	BJ60_01	11326.5402	11449.6812	11643.8202	100.8547	14.6030
	BJ60_02	10704.9149	10823.7735	10950.9690	76.9992	14.2598
	BJ60_03	12034.3366	12134.3792	12192.6607	52.7816	13.8523
	BJ60_04	12365.0652	12559.8677	12853.2036	145.7351	15.0256
	BJ60_05	11231.7932	11292.1902	11369.0203	49.6178	15.7620
	BJ60_06	12286.2834	12444.0914	12588.4879	88.0559	14.1395
	BJ60_07	13241.8030	13392.2743	13478.2894	82.8872	12.0423
	BJ60_08	12170.5549	12361.7430	12505.8710	108.1388	13.4651
	BJ60_09	12291.5100	12420.7206	12572.0969	73.2170	12.9955
	BJ60_10	13354.1939	13583.4122	13781.8097	128.4438	12.1163
	Average	12100.6995	12246.2133	12393.6229	90.6731	13.8261

4.4 Effect of tabu search

To analyze the effect of tabu search in level-2 membranes, we compared our algorithm to the MIHA without tabu search, MIHA$_{-TS}$. Tabu search in level-2 membranes is replaced by greedy algorithm. Although membrane framework remains in that case, the experimental results are less competitive than those obtained by using MIHA, see Table 6.

Table 6: Computational results by using MIHA and MIHA$_{-TS}$.

	instance	BS(RMB)	MS(RMB)	WS(RMB)	STD(RMB)	ET(s)
MIHA	BJ60_01	10864.7334	10914.4491	10939.8273	21.2864	179.3093
	BJ60_02	10217.4691	10310.1949	10395.6818	54.3404	167.1625
	BJ60_03	11321.6848	11423.8391	11489.1644	53.3224	168.3861
	BJ60_04	11800.7563	11847.7284	11915.0534	31.7880	179.6456
	BJ60_05	10890.8377	10909.8830	10947.0117	14.5797	190.0507
	BJ60_06	11875.3018	11916.5380	11942.8835	21.5401	157.5884
	BJ60_07	12528.3080	12634.5020	12685.8953	54.0499	138.2126
	BJ60_08	11783.3490	11867.4716	11932.7904	45.1478	162.5045
	BJ60_09	11573.5410	11705.3953	11908.5312	107.6724	173.2149
	BJ60_10	12939.9900	13196.1938	13269.0754	90.2917	161.0214
	Average	11579.5971	11672.6195	11742.5914	49.4019	167.7096
MIHA$_{-TS}$	BJ60_01	10885.7841	10926.8821	10956.9339	21.4261	178.9296
	BJ60_02	10348.5065	10431.9705	10567.5762	64.6180	163.2735
	BJ60_03	11408.0875	11544.4161	11691.5340	81.8656	181.3918
	BJ60_04	11806.7144	11874.6096	12000.3970	59.1001	189.1059
	BJ60_05	10891.2894	10906.0981	10939.9388	13.5889	211.0158
	BJ60_06	11928.3036	11991.9485	12038.1416	27.5477	160.3498
	BJ60_07	12788.3859	12943.2357	13091.7043	93.8455	148.9768
	BJ60_08	11838.6827	11979.3826	12068.2326	62.0261	168.0495
	BJ60_09	11698.5996	12038.9623	12193.1661	139.9844	167.9968
	BJ60_10	13350.1401	13450.0882	13526.5397	52.9312	150.3822
	Average	11694.4494	11808.7594	11907.4164	61.6933	171.9472

5 Conclusion

This work focused on membrane-inspired evolutionary algorithm to deal with a real and green open vehicle routing problem. The problem considered in this paper is to construct open routes for vehicles to visit all customers within their time windows. The overall objective is to minimize the total cost that is composed of cost of emissions, operational costs and cost of drivers. A hybrid heuristic algorithm was designed in the framework of membrane system. It benefited from parallel distributed structure and special communication strategy. The computational results proved its competitiveness.

Open vehicle routing problem (OVRP), a kind of vehicle routing problem (VRP), has unlimited potential value in the vigorous development of Sharing Economy. The practical importance of this problem has not received enough attention from researchers. In the future, more realistic models involving open routes can be addressed, such as close-open vehicle routing problem and OVRP with heterogeneous fleet. Our algorithm can be used to those problems with possible modifications. It is predictable that membrane-inspired algorithm can do more in the fields of logistics and transportation.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61872325, 61772290); the China Scholarship Council (201806405004); the Fundamental Research Funds for the Central Universities (63192616); the Science and Technology Development Strategy Research Program of Tianjin (18ZLZXZF 00320); and the Collaborative Innovation Center for China Economy.

References

1. Dantzig, G.B., Ramser, R.H.: The truck dispatching problem. Manag. Sci. 680–91 (1959).
2. Toth, P., Vigo, D.: The vehicle routing problem. Monogr. Discret. Math. Appl. 9 (2002)
3. Schrage, L.: Formulation and structure of more complex/realistic routing and scheduling problems. Networks. 11, 229–232 (1981)
4. Repoussis, P.P., Tarantilis, C.D., Ioannou, G.: The open vehicle routing problem with time windows. J. Oper. Res. 58, 355–367 (2007)
5. Bodin, L., Golden, B., Assad, A., Ball, M.: Routing and scheduling of vehicles and crews: The state of the art. Comput Oper Res. 10(2), 63–211 (1983)
6. Brandao, J.: A tabu search heuristic algorithm for open vehicle routing problem. Eur. J. Oper. Res. 157, 552–564 (2004)
7. Derigs, U., Reuter, K.: A simple and efficient tabu search heuristic for solving the open vehicle routing problem. J. Oper. Res. Soc. 60, 1658–1669 (2009)
8. Fu, Z., Eglese, R., Li, L.: Corrigendum to the paper: A new tabu search heuristic for the open vehicle routing problem. J. Oper. Res. Soc. 57, 1017–1018 (2006)
9. Russell, R., Chiang, W., Zepeda, D.: Integrating multi-product production and distribution in newspaper logistics. Comput. Oper. Res. 35, 1576–1588 (2008)
10. Fleszar, K., Osman, I.H., Hindi, K.S.: A variable neighbourhood search algorithm for the open vehicle routing problem. Eur. J. Oper. Res. 195(3), 803–809 (2009)
11. Pisinger, D., Ropke, S.: A general heuristic for vehicle routing problems. Comput.Oper. Res. 34(8), 2403–2435 (2007)
12. Salari, M., Toth, P., Tramontani, A.: An ILP improvement procedure for the open vehicle routing problem. Comput. Oper. Res. 37, 2106–2120 (2010)
13. Zachariadis, E., Kiranoudis, T.: An open vehicle routing problem metaheuris-tic for examining wide solution neighborhoods. Comput. Oper. Res. 37, 712–723 (2010)
14. Tarantilis, C.D., Ioannou, G., Kiranoudis, C.T., Prastacos, G.P.: A threshold accepting approach to the open vehicle routing problem. RAIRO Oper. Res. 38, 345–360 (2004)

15. Tarantilis, C.D., Ioannou, G., Kiranoudis, C.T., Prastacos, G.P.: Solving the open vehicle routing problem via a single parameter meta-heuristic algorithm. J.Oper. Res. 56, 588–596 (2005)
16. MirHassani, S., Abolghasemi, N.: A particle swarm optimization algo-rithm for open vehicle routing problem. Expert Syst. Appl. 38, 11547–11551 (2011)
17. Wang, W., Wu, B., Zhao, Y., Feng, D.: Particle swarm optimization for open vehicle routing problem. in: D.S. Huang, K. Li, G.W. Irwin (Eds.), Proceedings of the 2006 International Conference on Intelligent Computing: Part II (ICIC06),Springer-Verlag, Berlin, Heidelberg, pp. 999–1007. (2006)
18. Zhen, T., Zhu, Y., Zhang, Q.: A particle swarm optimization algorithm for the open vehicle routing problem. in: Proceeding 2009 International Conference on Environmental Science and Information Application Technology, IEEE, pp. 560–563 (2009)
19. Li, X., Tian, P.: An ant colony system for the open vehicle routing problem. in: M.Dorigo (Ed.), Lecture Notes in Computer Science (ANTS 2006), Springer, Berlin, pp. 356–363, 4150. (2006)
20. Li, X., Tian, P., Leung, S.: An ant colony optimization metaheuristic hybridized with tabu search for the open vehicle routing problem. J. Oper. Res. Soc. 60, 1012–1025 (2009)
21. Pan, L., Fu, Z.: A clonal selection algorithm for open vehicle routing problem. in:Proceeding 2009 Third International Conference on Genetic and Evolutionary Computing, pp. 786–790 (2009)
22. Repoussis, P.P., Tarantilis, C.D., Braysy, O., Ioannou, G.: A hybrid evolution strategy for the open vehicle routing problem. Comput. Oper. Res. 37, 443–455 (2010)
23. Yu, S., Ding, C., Zhu, K.: A hybrid GA-TS algorithm for open routing optimization of coal mines material. Expert Syst. Appl. 38, 10568–10573 (2011)
24. Niu, Y., Yang, Z., Chen, P., Xiao, J.: Optimizing the green open vehicle routing problem with time windows by minimizing comprehensive routing cost. J. Clean Prod. 171, 962–971 (2018)
25. Păun, Gh.: Computing with membranes. Journal of Computer and System Sciences. 61(1), 108–143 (2000)
26. Pan, L., Păun, Gh., Pérez-Jiménez, M.J.: Spiking neural P systems with neuron division and budding. Science China Information Science. 54(8), 1596–1607 (2011)
27. Barbuti, R., Bove, P., Milazzo, P., Pardini, G.: Minimal probabilistic P systems for modelling ecological systems. Theoretical Computer Science. 608, 36–56 (2015)
28. Sakellariou, I., Stamatopoulou I., Kefalas, P.: Using membranes to model a multi-agent system towards underground metro station crowd behaviour simulation. E-CAI 2012 workshop. Montpellier, France, August 28, 5–10 (2012)
29. Niu, Y., Zhang, Y., Zhang, J.: Running cells with decision-making mechanism: intelligence decision P System for evacuation simulation. Int J Comput Commun. 13, 865–880 (2018)
30. Lucie, C., Erzsébet, C., Luděk, C., Petr, S.: P colonies vol. 1, no.3, Pages: 178–197 (2019)
31. Nishida, T. Y.: Membrane algorithms: Approximate algorithms for NP-complete optimization problems, Applications of Membrane Computing, 303–314 (2006)
32. Zhang, G., Rong, H., Neri, F., Pérez-Jiménez, M.J.: An optimization spiking neural P system for approximately solving combinatorial optimization problems. Int. J. Neural. Syst. 24(5), 1–16 (2014)

33. Zhang, G., Rong, H., Cheng, J., Qin, Y.: A population-membrane-system-inspired evolutionary algorithm for distribution network reconfiguration. J. Electron. 23(3), 437–441 (2014)
34. Zhang, X., Li, J., Zhang, L.: A multi-objective membrane algorithm guided by the skin membrane. Nat. Comput. 15(4), 597–610 (2016)
35. Zhang, X., Tian, Y., Jin, Y.: Approximate non-dominated sorting for evolutionary many-objective optimization. Inform. Sciences. 369, 14–33 (2016)
36. Petr Sosík.: P systems attacking hard problems beyond NP: a survey vol. 1, no.3, Pages: 198–208 (2019)
37. Ciobanu, G., Pérez-Jiménez, M.J., Păun, Gh. eds.: Applications of membrane computing. Springer Berlin Heidelberg. 287(1), 73–100 (2006)
38. Zhang, X., Wang, S., Niu, Y., Pan, L.: Tissue P systems with cell separation: attacking the partition problem. Science China Information Sciences. 54(2), 293–304 (2011)
39. Pan, L., Păun, G.: Spiking neural P systems: an improved normal form. Theoretical Computer Science. 411, 906–918 (2010)
40. Zhang, G., Liu, C., Rong, H.: Analyzing radar emitter signals with membrane algorithms. Mathematical and Computer Modelling. 52(11–12), 1997–2010 (2010)
41. Pan, L., Daniel, D.P., Perez-Jimenez, M.J.: Computation of ramsey numbers by P system with sctive membranes. International Journal of Foundations of Computer Science. 22, 29–38 (2011)
42. Barth, M., Younglove, T., Scora, G.: Development of a Heavy-duty Diesel Modal Emissions and Fuel Consumption Model. Technical Report. UCB-ITSPRR-2005-1, California PATH Program, Institute of transportation Studies, University of California at Berkeley (2005)
43. Scora, M., Barth, G.: Comprehensive Modal Emission Model (CMEM), Version 3.01, User Guide. Technical Report. URL: < *http* : *//www.cert.ucr.edu/cmem/docs/CMEM_User_Guide_v*3.01*d.pdf* > (accessed 17.02.2014) (2006)
44. Barth, M., Boriboonsomsin, K.: Energy and emissions impacts of a freeway-based dynamic eco-driving system. Transportation Research Part D. 14, 400–410 (2009)
45. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. 35(2), 254–265 (1987)
46. Tan, K.C., Cheong, C.K., Goh, C.K.: Solving multi-objective vehicle routing problem with stochastic demand via evolutionary computation. Eur. J. Oper. Res. 177(2), 813–839 (2007)

Improved spectral clustering algorithm based on Tissue-like P system

Zhe Zhang[1] Xiyu Liu[2*]

Business School, Shandong Normal University, Jinan, China
zaq1230123@163.com xyliu@sdnu.edu.cn

Abstract. Traditional spectral clustering algorithms usually use Gaussian kernel functions to form the similarity matrix. This method is sensitive to the selection of scale parameters. Moreover, the traditional spectral clustering algorithm randomly initializes the center in the clustering stage may affect the clustering results. This paper presents an improved spectral clustering algorithm based on tissue-like P system (SCAP-TP), which uses a new construction method of similarity matrix to consider the distance between data points and the intrinsic structure of the data set. In the clustering stage, a tissue-like P system is designed as the computational framework of the AP clustering algorithm, which greatly improves the efficiency of the clustering algorithm, and also explores a new direction for the application of membrane computing. Three UCI data sets and three artificial data sets are used for making comparison between SCAP-TP algorithm and some existing methods. Experiments results prove the superiority of the proposed algorithm.

Keywords: Tissue-Like P Systems · Spectral Clustering · AP Algorithm · Laplace Matrix.

1 Introduction

Spectral clustering is an algorithm evolved from graph theory, which transforms the clustering problem of data into the partitioning problem of graphs. Spectral clustering treats each data point as a vertex on the graph. The similarity of data points is regarded as the weight of the edge. By dividing the graph, the sum of the edge weights in the subgraph is as high as possible, the sum of the edge weights between the graphs is as low as possible, thus completing the clustering problem of the data. In 2000, according to the theory of spectral partitioning, Shi [1] proposed a standard cut set criterion based on 2-way partitioning (Normalized Cut). Hagen and Kahng [2] proposed the proportional secant objective function (Ratio Cut). Although the spectral clustering algorithm has achieved good results in recent years, the algorithm is still in the early stage of development, and there are still some problems to be further studied. For example, the algorithm is sensitive to the selection of scale parameters when constructing similar matrices using Gaussian kernel functions, and Zelnik-Manor and Perona [3] have shown that different scale parameter values have different effects on

the results. In order to solve this problem, Zhang [4] proposed a construction method of similar matrix based on local density. Li and Guo [5] utilized neighbor propagation principle to get the similarity matrix. Yessica [6] presented Powered Gaussian kernel similarity function to solve this problem.

Membrane computing (P system) is a computational model abstracted by the Gh. Paun [7] based on the cellular structure and function of the organism. According to different structures, P systems can be divided into three types: cell-like P systems, tissue-like P systems and neural-like P systems. These computational models have been proven to have powerful computing power, and because of the uncertainties and maximum parallelism of P systems, these models also have high computational efficiency. Since the advent of the P systems, it has received extensive attention from scholars in related fields. More and more variants of the P system have been proposed for different research fields. Recently, membrane computing has begun to combine with clustering issues. Jiang et al. [8] proposed a variant of tissue-like P systems with active membranes to realize the clustering process. Zhao [9] constructs a cell-like P systems with promoters and inhibitors to improve DBSCAN algorithm. Liu et al. [10] proposed a consensus clustering algorithm based on K-medoids. Peng [11] proposed a tissue-like membrane system to adapt a multi-objective clustering framework for fuzzy clustering. Peng [12] proposed an extended membrane system to processing the automatic fuzzy clustering problem. Gong [13] combines the MST clustering algorithm with the membrane calculation to effectively improve the quality of the cluster.

In this paper, for the problem that spectral clustering is sensitive to scale parameters and sensitive to random selection initial values in the clustering stage, we first introduce the method of constructing similar matrix by Li and Guo [5], which can simultaneously consider the distance between data points and the intrinsic structure of data set. The feature vector is obtained by spectral clustering. Then use the Affinity Propagation (AP) algorithm for clustering. We constructed a tissue-like P systems as the computational framework of the AP clustering algorithm, which greatly improved the efficiency of the algorithm in clustering stage. The remaining work of this paper is arranged as follows: In section 2, we briefly introduce the related knowledge of tissue-like P systems, spectral clustering and AP algorithm. In section 3 we propose an improved spectral clustering method based on tissue-like P system. Section 4 compares the effects of proposed algorithm with other algorithms on different data sets. The conclusion is arranged in section 5.

2 Related works

2.1 Spectral clustering

The idea of spectral clustering comes from the theory of spectral partitioning, which transforms the clustering problem into the partitioning problem of undirected graphs. Each data point is treated as a vertex on the undirected graph $G=(V,E)$, the edge E represents the connection between the data points, the weight w_{ij} on the edge represents the similarity between the two points, and W

is a similar matrix. By dividing the undirected graphs, different subgraphs are obtained, so that the sum of weights in the subgraphs are as high as possible, and the sum of weights between the different subgraphs are as low as possible, this completes the clustering of the data set.

Spectral clustering can solve the partitioning problem of graphs by eigenvectors of Laplacian matrix. In general, the similarity matrix relies on the use of Gaussian kernel functions in the fully connected graph to define the weights between the vertices $W_{ij} = exp(-dist(x_i, x_j)/2\sigma^2)$, where σ is a scale parameter. Then the degree matrix D, $d_{ii} = \sum w_{ij}$, $D = diag(d_i)$ is obtained through the similar matrix. A Laplacian matrix L is constructed from the similarity matrix W and the degree matrix D. There are usually three ways to construct the Laplacian matrix L: (1) unnormalized Laplacian matrix L=D-W (2) Normalized symmetric Laplacian matrix $L = I - D^{-1/2}WD^{-1/2}$ (3) Normalized asymmetric Laplacian matrix $L = D^{-1}WD^{-1}$. Then, the eigenvectors corresponding to the first k eigenvalues are calculated by Laplacian matrix to form the feature matrix U, and U is normalized to obtain a new feature matrix Y. Each row of the feature matrix Y is regarded as a data point, the feature matrix Y is clustered, and then the clustering result is mapped to the original data set, and the entire spectral clustering algorithm is complete.

2.2 Tissue-like P systems

The tissue-like p system deals with a mechanism in which multiple cells interact with each other in the same environment to participate in the calculation. A tissue-like P system of degree m is defined as [13]:

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, i_0) \tag{1}$$

Where:

(1) O is a finite non-empty alphabet that contains objects in the cell.
(2) syn is the connection between cells. $syn \subseteq (1, 2, \ldots, m) \times (1, 2, \ldots, m)$.
(3) i_0 is the label of the output cell.
(4) σ_i is i-th cell in the following form:

$$\sigma_i = (Q_i, s_{i,0}, w_{i,0}, P_i), 1 \leq i \leq m \tag{2}$$

Where Q_i is a finite set of states; $s_{i,0} \in Q_i$ is the initial state; $w_{i,0} \in Q^*$ is the initial set of objects; P_i is a finite set of rules, and the form of the rule is $sw \to s'xy_{go}z_{go}$ .The execution of the system from any state s to the next state s' is as follows: in state s , in each cell, the multiple set w will be replaced by x, y, z, and x remains in the current cell, y is sent to the cells that connected to the current cell through the communication channel, z is sent out to the environment, so the system enters the next state s'.

2.3 AP clustering algorithm

The AP algorithm is an algorithm proposed by Frey and Dueek [15] in 2007. It does not need to specify the number of clusters in advance. The idea of the AP

algorithm is to treat all the data as nodes of the network, and then calculate the cluster center through the information transfer of each side of the network. In the entire information transfer mechanism, there are two kinds of information transmitted between nodes, responsibility and availability. The AP algorithm continuously updates the responsibility and availability of each point through the iteration until k high-quality exemplars are generated, and then the remaining points are allocated to the corresponding clusters.

First, the AP algorithm takes the similarity between the data points as an input, and the similarity between the two points is defined as $s_{ij} = -\|x_i - x_j\|^2$ .The greater the value of the similarity, the closer the distance between the two points is, which is convenient for subsequent calculations. The similarity matrix is defined as S .

Then preference P indicates the tendency of the point to be selected as the cluster center. If there is no prior knowledge, all data points are considered as potential exemplar representatives, and generally P is set to the median of the elements in the similar matrix S.

Then we can calculate the responsibility and availability. The core of the AP algorithm is the process of repeated delivery of these two pieces of information. The degree of responsibility $r(i,k)$ represents the possibility that the point k is suitable as a representative point of the point i.

$$r(i,k) = \begin{cases} S(i,k) - max_{j \neq k}(a(i,k) + r(i,k)), i \neq k \\ S(i,k) - max_{j \neq k}(S(i,k)), i = k \end{cases} \tag{3}$$

The degree of availability $a(i,k)$ indicates that point i selects point k as the degree of attribution of its representative point.

$$a(i,k) = \begin{cases} min(0, r(k,k) + \sum_{j \neq i,k} max(r(j,k),0)), i \neq k \\ \sum_{j \neq k} max(r(j,k),0), i = k \end{cases} \tag{4}$$

According to the above two formulas, the algorithm is iterated. In order to prevent the oscillation during the iteration, Meng et al. [16] introduced the damping coefficient, whose main function is to adjust the stability of the algorithm iteration. The adjusted formula is:

$$r_{t+1}(i,k) = \lambda * r_t(i,k) + (1-\lambda) * r_{t+1}(i,k) \tag{5}$$

$$a_{t+1}(i,k) = \lambda * a_t(i,k) + (1-\lambda) * a_{t+1}(i,k) \tag{6}$$

Finally, $k = argmaxa(i,k) + r(i,k)$ is used to determine the cluster center. If $i = k$, then i is the clustering center. If $i \neq k$, then k is the clustering center of i. The algorithm stops when the algorithm reaches the maximum number of iterations or if the cluster center no longer changes in several iterations.

3 Improved Spectral Clustering Algorithm

3.1 Construction of Similarity Matrix

The construction of similar matrix has a great influence on the spectral clustering algorithm. The traditional spectral clustering algorithm uses the Gaussian kernel

function to construct the similar matrix, $w_{ij} = exp(-dist(x_i, x_j)/2\sigma^2)$, where σ is a scale parameter. Different scale parameters may produce different results. To improve this problem, Li and Guo [5] proposed a powered Gaussian kernel spectral clustering. In this algorithm, the powered Gaussian kernel similarity function is used to construct the similarity matrix, $w_{ij}^{\gamma} = exp(-dist(x_i, x_j)/\beta)^{\gamma}$, where γ is a power parameter, which is generally set to 5. $\beta = max_i(min\|x_i - x_j\|)$, β can take into account the distance between data points and the structure of the data set. In this paper, we borrow this method to form a similarity matrix.

3.2 The Proposed Tissue-like P system

After constructing a similarity matrix using the formula $w_{ij}^{\gamma} = exp(-dist(x_i, x_j)/\beta)^{\gamma}$, according to the flow of the spectral clustering method, the degree matrix, the Laplacian matrix and the eigenvectors corresponding to the first k eigenvalues are calculated respectively. After the normalization process, the AP clustering algorithm is used to cluster the new feature matrix. The AP algorithm does not need to specify the initial cluster center, which overcome the influence of the traditional spectral clustering on the clustering result when the K-means algorithm is used in the clustering stage. Considering the parallelism of membrane computing, we construct a tissue P system as the computational framework of AP clustering algorithm. Figure.1 is the proposed tissue-like P system with the following structure:

$$\Pi = (O, \sigma_1, \ldots, \sigma_n, \psi_1, \ldots, \psi_n, syn, i_0) \tag{7}$$

Where:

(1) O is a finite non-empty alphabet that contains objects in the cell

(2) $syn \subseteq ((1,1), \ldots, (1,n), (2,1) \ldots, (2,n), \ldots, (n,1), \ldots, (n,n))$ is the connection between cells. .

(3) i_0 is the label of the output cell, $i_0 = 0$

(4) σ_i is the i-th cell of the lower n cells, and its form is as follows:

$$\begin{aligned}\sigma_i =&(s, s_i, w_{i,1}, \ldots, w_{i,n}, a_{i,1}, \ldots, a_{i,n}, s_i w_{i,1} a_{i,1} \rightarrow z_1(r_{i,1}, go),\\ &s_i w_{i,2} a_{i,2} \rightarrow z_2(r_{i,12}, go), \ldots, s_i w_{i,n} a_{i,n} \rightarrow z_n(r_{i,n}, go)), 1 \leq i \leq n\end{aligned} \tag{8}$$

In the cell σ_i, s represents a finite set of states. In this paper, in order to satisfy the information transfer mechanism of the AP algorithm, we define S as the designated number of each cell, so s_i refers to the cell number is s_i in the tissue-like P system. $w_{i,1}, w_{i,2}, \ldots, w_{i,n}$ refers to the object in cell s_i, representing the similarity between the *i-th* data point and the remaining data points. $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ is also an object in the cell s_i, representing the availability between the *i-th* data point and other data points. The rule $s_i w_{i,n} a_{i,n} \rightarrow z_n(r_{i,n}, go)$ refers to the calculation of the responsibility $r_{i,j}$ between the *i-th* data point and the *j-th* candidate center using the formula 5, and sends it to the cell labeled z_j. When the new object $r_{i,j}$ enters the cell, the original $r_{i,j}$ dissolves.

ψ_i is the *i-th* cell in the upper n cells, and its form is as follows:

$$\begin{aligned}\psi_i =&(z, z_j, r_{1,j}, \ldots, r_{n,j}, z_i r_{1,j} \rightarrow s_1(a_{1,j}, go),\\ &z_i r_{2,j} \rightarrow s_2(a_{2,j}, go), \ldots, z_i r_{n,j} \rightarrow s_n(a_{n,j}, go)), 1 \leq i \leq n\end{aligned} \tag{9}$$

In ψ_i, z is also a finite set of cell states, and z_j refers to the cell number z_j in the tissue-like P system. $r_{1,j}, r_{2,j}, \ldots, r_{n,j}$ is an object in the cell, representing the responsibility between the *j-th* data point and other data points. The rule $z_i r_{i,j} \rightarrow s_i(a_{i,j}, go)$ indicates that the availability $a_{i,j}$ between the *j-th* data point and the *i-th* data point is calculated according to the formula 6, and $a_{i,j}$ is sent to the cell labeled s_i. When the new object $a_{i,j}$ enters the cell, the original $a_{i,j}$ dissolves.

The whole process is repeated in the Tissue-like P system until the maximum number of iterations is reached, and the system stops. Then according to the formula $k = argmaxa(i,k) + r(i,k)$. We can get a set of cluster centers. The detailed steps of SCAP-TP algorithm is shown in Algorithm 1.

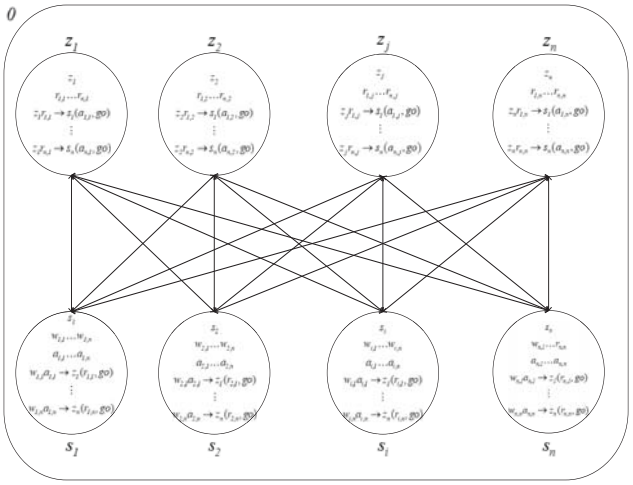


Fig. 1. The Proposed Tissue-like P System

4 Experimental analysis

In this part, we compare the algorithm proposed in this paper with the k-means, NJW ($\sigma = 0.1, \sigma = 0.5$), MPSC [17] algorithm on three artificial data sets and three UCI data sets. The specific information of the data sets are as shown in Table.2:

Table 1. SCAP-TP algorithm

Algorithm 1. SCAP-TP algorithm
Step1.Use the Powered Gaussian kernel function to construct the similar matrix $W.w_{ij} = exp(-dist(x_i, x_j)/2\sigma^2).\beta = max_i(min\|x_i - x_j\|)$
Step2. Degree matrix D ($d_{ii} = \sum w_{ij}$).
Step3. Construct a normalized symmetric Laplacian matrix L, $L = I - D^{-1/2}WD^{-1/2}$.
Step4. Calculate the feature vector v corresponding to the first k eigenvalues of L, and construct the feature matrix U.
Step5. Normalize the feature matrix U to obtain a normalized matrix Y, which contains n points in space reduced to k dimensions.
Step6. Treat each row of Y as a point and Clustering them by AP clustering algorithm based on the Tissue-like P system that we formed.

Table 2. Information of Data Sets

Data Sets	Objects	Attributes	Classes	Source
Iris	150	4	3	UCI
Wine	178	13	3	UCI
Seeds	210	7	3	UCI
Spiral	944	2	2	Artificial
Twomoons	2000	2	2	Artificial
Threecircles	3603	2	3	Artificial

The three artificial data sets are showed in Figure. 2. In the experiment, the power parameter γ is set to 5, and in the AP algorithm clustering phase, the maximum number of iterations is 500, and the iteration convergence coefficient is 50. For each data set, the number of cells in tissue-like P system is equal to the number of data points in the data set. All the experiments are conducted on the computer with Intel core i5-3230M CPU, 4GB RAM. The experiments environment is Matlab 2016b.

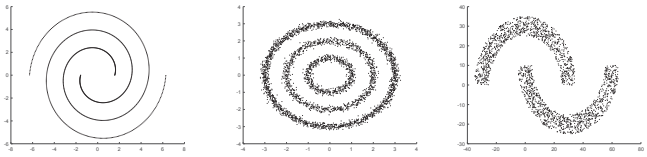
The experimental results on the artificial dataset are shown in Figure.3. As can be seen from Figure.3, our algorithm can handle data sets of different structures very well.

For the clustering problem of UCI data sets, we use the correct rate and running time as evaluation indicators. The accuracy is calculated as:

$$\rho_{ACC} = \frac{1}{N}\sum max_j|v_i \bigcap v_j| \tag{10}$$

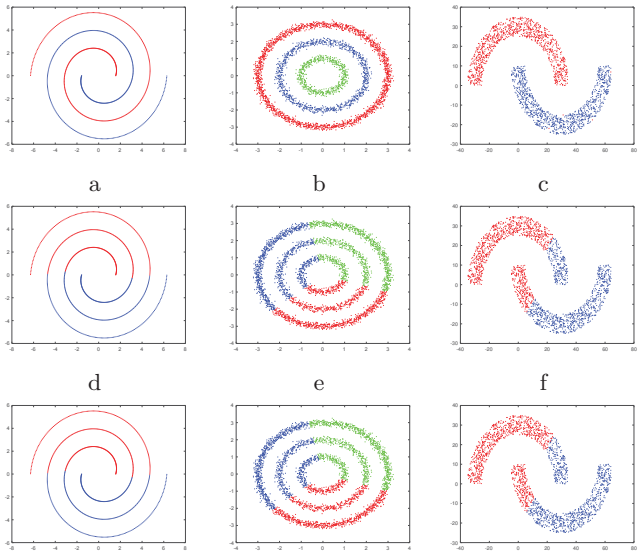
The experimental results of the UCI data set are as follows:

When testing the UCI dataset, our algorithm performs better than other algorithms, and because of the tissue-like P system as the computational framework, runtime is also optimized.



a b c

Fig. 2. Artificial Data Sets



a b c

d e f

g h i

Fig. 3. (a)(b)(c) clustering results of SCAP-TP algorithm in artificial data sets respectively; (d)(e)(f) clustering results of NJW algorithm with $\sigma = 0.1$ in artificial data sets respectively; (g)(h)(i) clustering results of NJW algorithm with $\sigma = 0.5$ in artificial data sets respectively

Table 3. Clustering Results of Different Algorithms on UCI Data Sets

Data Sets	Evaluation Index	K-means	NJW	MPSC	SCAP-TP
Iris	Accuracy	0.7273	0.8534	0.9067	0.9067
Iris	Time(s)	0.3521	0.5160	0.5889	0.4154
Seeds	Accuracy	0.7008	0.7905	0.7194	0.8857
Seeds	Time(s)	0.5732	0.5890	0.4641	0.4675
Wine	Accuracy	0.4730	0.4267	0.5505	0.6742
Wine	Time(s)	0.5394	0.4840	0.4022	0.3686

5 Conclusion

This paper proposes an improved spectral clustering algorithm based on tissue-like P system (SCAP-TP) algorithm to solve the clustering problem. Its basic

idea is to use powered Gaussian function to construct a similar matrix to overcome the influence of scale parameters on spectral clustering, Then, the normalized feature matrix is obtained according to the process of spectral clustering. and the feature matrix is clustered by AP algorithm. In order to improve the efficiency of the AP algorithm, we construct a tissue-like P system as the computing framework of the AP algorithm. Compared with other algorithms, the superiority of the proposed algorithm in clustering data sets with different structures is proved. In the future, we will pay more attention to the combination of clustering algorithm and membrane computing to improve the performance of the algorithm, and also explore the direction of membrane computing.

References

1. Shi J, Malik J, F.: Normalized cuts and image segmentation. IEEE Transactions on pattern Analysis and machine intelligence **22**(8), 888–905 (2000)
2. Hagen L, Kahng AB, F.: New spectral methods for ratio cut partitioning and clustering . IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **11**(9), 1074–1085 (1992)
3. Zelnik Manor L, Perona P, F.: Self-tuning spectral clustering. Advances in neural information processing systems **22**, (2004)
4. Zhang X, Li J, Yu H, F.: Local density adaptive similarity measurement for spectral clustering. Pattern Recogn Lett **32**(2), 352–358 (2011)
5. Li XY, Guo LJ, F.: Constructing affinity matrix in spectral clustering based on neighbor propagation. Neuro computing **97**, 125–130 (2012)
6. Nataliani Y, Yang M S, F.: Powered Gaussian kernel spectral clustering. Neural Computing and Applications **31**, 557–572 (2019)
7. Paun, Gheorghe, F.:Computing with Membranes. Journal of Computer and System Sciences **61**(6), 108–143 (2000)
8. Zhenni Jiang, Xiyu Liu, Minghe Sun, F.: A Density Peak Clustering Algorithm Based on the K-Nearest Shannon Entropy and Tissue-Like P System. Mathematical Problems in Engineering **2019**, (2019)
9. Zhao Y, Liu X, Li X, F.: An improved DBSCAN algorithm based on cell-like P systems with promoters and inhibitors. PLoS ONE **13**(12),(2018)
10. Liu X, Zhao Y, Sun W, F.: K-Medoids-Based Consensus Clustering Based on Cell-Like P Systems with Promoters and Inhibitors. Communications in Computer and Information Science **681**, 95–108 (2016)
11. Peng H, Shi P, Wang J, F.: Multiobjective fuzzy clustering approach based on tissue-like membrane systems. Knowledge-Based Systems **125**, 74–82 (2017)
12. Peng H, Wang J, Shi P, F.: An Extended Membrane System with Active Membranes to Solve Automatic Fuzzy Clustering Problems. International Journal of Neural Systems **26**(03), (2016)
13. Gong P, Liu X, S.: An Improved MST Clustering Algorithm Based on Membrane Computing. In: Springer, Cham. International Conference on Human Centered Computing 2017, 10745:1-12.
14. Paun, Gheorghe and Rozenberg, Grzegorz and Salomaa, Arto, T.: The Oxford Handbook of Membrane Computing. Oxford University Press, Inc.,New York, NY, USA (2010)
15. B. J. Frey, D. Dueck, F.: Clustering by passing messages between data points. Science **315**, 972–976 (2007)

16. J. Meng, H. Hao, Y. Luan, F.: Classifier ensemble selection based on affinity propagation clustering. Journal of Biomedical Informatics. **60**, 234–242 (2016)
17. Wang Lijuan, Ding Shifei, Jia Hongjie, F.: An improvement of Spectral Clustering via Message Passing and Density Sensitive Similarity. IEEE Access **7**, 101054–101062 (2019)

A review of membrane computing models for ecosystems and a case study on giant pandas

Yingying Duan[1], Gexiang Zhang[1⋆], Dunwu Qi[2], Luis Valencia-Cabrera[3], Haina Rong[1], and Mario J. Perez-Jimenez[3].

[1] School of Electrical and Engineering, Southwest Jiaotong University, Chengdu 610031, China
[2] Chengdu Research Base of Giant Panda Breeding, Chengdu, Sichuan, China
[3] Department of Computer Science and Artificial Intelligence, Universidadde Sevilla, Sevilla, Spain

Abstract. Ecosystem modeling based on membrane computing is emerging as a powerful way to study the dynamic of (real) ecological populations. These models, providing distributed parallel devices, have shown a great potential to imitate the rich features observed in the behavior of species and their interactions, key elements to understand and model ecosystems. Compared with differential equations, membrane computing models, *a.k.a.* P systems, can model more complex biological phenomena, due to their modularity, their ability to enclose the evolution of different environments and simulate in parallel different interrelated processes.

In this paper, a comprehensive survey of membrane computing models for ecosystems is given, taking a giant panda ecosystem as an example to assess the models performance. This work aims at modeling a number of species using P systems with different membrane structure types to predict the number of individuals depending on parameters such as reproductive rate, mortality rate, and rescue or release. Firstly, the computing models are introduced conceptually, explaining the use of the rules. Next, various modeled species (including endangered animals, plants, and bacteria) are summarized, and some computer tools are presented. Then, a discussion follows on the use of P systems for ecosystem modeling. Finally, a case study on giant pandas in Chengdu Base is analyzed, concluding that the study in this field by using single environment systems can provide a valuable tool to deepen into the knowledge about the evolution of the overall ecosystem. This could ultimately help in the decision making processes of the managers of the ecosystem to increase the species diversity and modify the adaptability. Also, we should consider the impacts of natural disasters on population dynamics of species. To this purpose, the analysis performed has provided a considerably more feasible prediction data than those so far been harvested.

Keywords: Membrane Computing; Ecosystem Modeling; Endangered Species

⋆ Gexiang Zhang is corresponding author (phone:13882184673; e-mail: zhgxdylan@126.com)

1 Introduction

Membrane Computing is a fast-growing branch of natural computing [83]. The computational devices within this paradigm are called membrane systems or P systems, and they have attracted major interest since their appearance. Nowadays membrane computing community is actively combining deep theoretical researches with practical applications.

On the one hand, theoretical studies focus on how to design membrane computing models according to the compartmentalized structure and the functioning of biological membranes within living cells, and how to evaluate their computational power and computational complexity, addressing efficiency aspects. Different types of membrane structures abstracted from biological cells can be distinguished in membrane systems. The most widely studied are cell-like P systems [79, 81, 112] (inspired in the compartmentalized hierarchical structure inside a cell), tissue-like P systems [7, 67] (with the focus in the interconnection among cells, not entering into details of the internal compartments inside each cell), and spiking neural P systems [78, 80, 101] (inspired from the transmission of electrical pulses, *a.k.a.* spikes, among neurons). Along the last twenty years, plenty of research results have proved that a number of variants of these computational models are equivalent in power to Turing computing power (computationally complete), and many of them obtained efficient solutions (polynomial solutions, even linear in some cases) to a variety of computationally hard, NP-complete [89, 99] or PSPACE problems [104, 1].

On the other hand, application researches of membrane computing models aims to effectively apply the introduced models to handle several practical problems, from basic ones to the modeling of complex systems. For *automatic design of membrane computing models* (ADMCM), regarded as an automatic computation device, the models can adaptively achieve basic arithmetic operations [132]. Thus, in [49] Huang *et al* applied P systems with the Q-bit representation to compute power n^2, something also achieved in [77] by Ou *et al*, who applied P systems calculate power n^2 with a different approach using an elitist genetic algorithm (GA). within this same research line, in [17, 131] five types of automatic P systems were used to compute adding, subtraction, multiplication, division and power. In some studies, spiking-like P systems are used as a computing device to resolve several arithmetic problems; for instance, the addition of n natural numbers and the multiplication of two arbitrary natural numbers with a given length of binary bits [88, 134]. Of course, apart from providing automatic design and arithmetic operations, P systems are applied to hard problems such as vertex cover [62, 102], quadratic assignment [73], 3-coloring [31, 74] and non-semilinear sets [2, 103]. Besides, they have been applied to solve image processing problems [19, 86, 100, 121, 125, 128], complex optimization problems [4, 132, 110, 130, 35], and intelligent control problems for robots [10, 84, 122]. These applications show that membrane computing models are useful tools to solve many practical problems of very different nature. It seems clear that each type of computationally hard problem addressed by membrane computing models somehow implies

an extension of application fields, providing theoretical and practical foundations opening paths that can be worth exploring, and widening the application space.

Regarding the contributions of membrane computing to model complex systems, relevant achievements have been made along the last decade, with a special attention to the study of real ecosystems (focusing on endangered or invasive species, among others) and population dynamics. Thus, endangered species on ecosystems present reproduction rates usually low, along with mortality rates abnormally high, due to reasons derived from the biology of the species themselves, the increased threats by others, the effects of human activities or natural disasters. It is often the case that these endangered species are not considered to be free of danger even when the threat is vanishing, because of their scarcity in the undisturbed fragments, so that isolated population sometimes cannot survive after destruction and become extinct. Hence, the qualitative and quantitative understanding of the inherent laws or processes underlying the disturbances in the population size and distribution (*i.e.*, the population dynamics) has become critical for the successful management and conservation of endangered species [72]. Apart from their natural mortality, most species have suffered the effects of several nature disasters. In certain studies, some types of natural disasters have caused or may potentially cause the risk of species biodiversity collapse, or even some species extinction in certain regions [54, 105, 109, 114]. For example, references in [15, 37, 93, 98, 108] studied the impacts of climate change on the parameters related with the population dynamics of congbird [98], divergence of species responses [37], the impact for agricultural welfare [108], parasite biodiversity [15], and an information fusion of numerous natural or environmental factors by using multi-mathematic models [93]. In order to accurately assess the impacts of these factors, several mathematical models are used to analyze the impacts of these elements on population dynamics; *e.g.*, differential equations [38], generalized linear models (GLMs) [71], generalized additive models (GAMs) [126], ecological niche factor analysis (ENFA) [48], or machine-learning methods such as maximum entropy method (Maxent) [91], Bayesian approaches [40] and neural networks [117]. According to the predicted results of these models, the parameters related can affect the population change in varying degrees. The researches mentioned that it would be worth providing projections for endangered species population dynamics under the influence of potential natural disasters, in order to protect them. Besides, it would be advisable to address the research on the population dynamics of the species following different approaches.

This work focuses on membrane computing (MC) models of species in certain ecosystems, so it involves two main fields: membrane computing and population dynamics. Concerning the first one, MC has among its strengths the availability of rigorous and complete, strongly-founded theoretical-practical developments; in addition, it provides parallel distributed devices in a framework with flexible evolution rules. With respect to the latter one, the population dynamics of the species has obvious biological processes involved, such as feeding, reproduction, mortality, rescue, release, biochemical reactions of bacteria, and this dynamics could be also affected by the potential impacts of natural disasters on popu-

lations. Based on the characteristics of membrane systems and the features of endangered species, these evolutionary behavior of such species can be expressed by the rules of membrane systems. Hence, along this paper, we recapitulate ecosystem models using different types of P systems. Firstly, we consider how to map between the elements related with the species and their interactions in the ecosystem and those related with the definition of P systems. So far, there are several references to study the applications of P systems on different real ecosystems. The common characteristics are summarized as: (a) each individual is represented by an object in the P system; (b) the different behaviors of individuals of the species are abstracted as the rules in the P systems; (c) a certain living environment in the ecosystem is abstracted as a membrane structure. At every moment, all individuals will evolve synchronously. For these species inhabiting different geographical regions but subject to the same processes, this situation can also be represented by using multienvironment P systems, where communication is possible between different environments. For other species such as plants and bacteria, the evolutionary processes are also modeled according to their characteristics, and possibly distinguishing environments with different parameters. Through the analysis above, we have depicted some of the most relevant facts taken into account when modeling ecosystems based on P systems, in order to accurately predict data about the biological evolution of the species, aiming to capture in these models (i.e., to mimic) the relevant elements of the real biological phenomena under study.

The rest of this paper is arranged as follows. Section 2 introduces membrane computing models for ecosystems. After that, Section 3 summarizes the applications of such membrane computing models to the population dynamics of certain ecosystems. Then, Section 4 lists several simulation software tools to perform virtual experiments for P systems models of ecosystems. Later on, in Section 5, a case study on the population dynamics of giant pandas is analyzed. Finally, some conclusions and possible further developments are discussed in Section 6.

2 Membrane computing models for ecosystems

As outlined in Section 1, different types of mathematical models have been applied to ecosystems. These models are representations imitating the real systems under study, using a certain formalism. In particular, some of these approximations are computational models, what means they follow the rules of some computing paradigm that regulates their behavior, and can be *computed* directly in their computational devices, or be simulated following the same exact rules; on the contrary, non-computational models (*e.g.* differential equations) require the use of approximated methods in order to be computed by some computing device.

When membrane computing is used to create a representation of an ecosystem, incorporating their main parameters, individuals, processes, etc. involved in their dynamics, this is considered a computational model, because is *a model of*

the ecosystem that is based on a computational paradigm (in this case, membrane computing), whose computation follows the exact rules of the formal model, not requiring any approximate method to be computed. These models of ecosystems are based on membrane computing, so they are usually called membrane-computing based models, using for this representation some type of membrane systems (commonly referred to as P systems).

As computational devices, membrane systems or P systems are abstracted from the structure and the functioning of livng cells. There are three main classes of P systems: (1) cell-like P systems, inspired from living cells; (2) tissue-like P systems, inspired from the interactions of cells in tissues; and (3) SN P systems (Spiking Neural P systems), inspired from neural systems.

Concerning the population dynamics of ecosystems, with regards to the membrane structure mostly cell-like and tissue-like P systems have been used to build these models. Thus, cell-like P systems have been generally applied to model ecosystems where a single environment is involved, whereas the latter ones haven been applied to systems involving more than one environment implying communications. However, a third type of system, initially referred to as multi-environment system, has been widely adopted to combine the internal structure on cell-like P system (in tissue-like systems the cells only present one level, without internal organization) plus the existence of different regions with cells inside and allowing communication among environments (what is not possible in cell-like systems).

With respect to the dynamics of the systems involved in these computational models, mostly two main paths have been followed when dealing with multi-environment systems: a stochastic approach (there are zero or several cell-like P systems inside each environment) and a probability approach (there is one and only one cell-like P system in each environment); for further details refer to [132]. Nowadays, the stochastic approaches are generally associated with computational models at a *micro* level (*e.g.*, involving molecular interactions), not being widely used to model ecosystems at a *macro* level. Consequently, in what follows along the paper we only consider the computational models of P systems following the probabilistic approach, also termed *population dynamic* P systems (PDP systems).

PDP systems are variants of P systems introducing probability mechanisms into P systems. According to the number of environments, PDP systems can be divided into single environment PDP systems (with a cell-like P system inside a single environment, see Fig.1 (a)) and multi-environment PDP systems (with several environments, each one containing a single P system inside, see Fig.1 (b)). These two types of PDP systems are introduced as follows, including syntactic and semantic aspects.

Definition 2.1 ([11]). A *single-environment* P system of degree n with $n \geq 1$ is a tuple

$$\Pi = (\Gamma, \mu, M_1, ..., M_n, R, \{f_r\}_{r \in R}) \tag{1}$$

where

- Γ is a finite alphabet constructed by all the objects in the PDP system.
- μ is a membrane structure (MS), consisting of n membranes, labeled as 1, 2,...,n. The skin membrane is marked as 1. We associate electrical charges with membranes from the set {-,0,+}, negative, neutral and positive.
- $M_i, 1 \leq i \leq n$, are finite multisets over Γ, representing the multisets of objects initially placed in the n regions delimited by the membranes of the hierarchical structure μ.
- $R_i (R_i \in R)$, $1 \leq i \leq n$, are a finite sets of rules of the following forms:
  - Rules of the first type: $r_1 \equiv u[v]_i^{\alpha} \xrightarrow{p_r} u'[v']_i^{\beta}$
  - Rules of the second type: $r_2 \equiv u[v]_i^{\alpha} \xrightarrow{1-p_r} u'[\lambda]_i^{\beta}$

where, u, v are a multiset over Γ and p_r is a real number between 0 and 1 associated with the rule, α and β are electric charges where α, $\beta \in \{-, 0, +\}$. In each computation step, the same left-hand side of the rule can produce different evolutionary states (*e.g.*, surviving or not), and the sum of these rules sharing their left-hand side (including the electrical charge) must always be equals to 1.

Rule analysis. For modeling ecosystem, the rules are abstracted from the behavior of species, food distributions, natural disasters, bacterium reactions, *etc*. From the point of view of the certainty of the application of the rules, two kinds of operational rules can be distinguished. The first type would be rule without explicit probability written; this is equivalent to a probability of 1; that is, these rules will be executed whenever selected, what will happen whenever they are applicable and no rules are competing for the same objects involved. The other type of rules, in this sense, would be the rules with probability lower than 1; that is, rules that, once selected, will be executed depending on their probability.

The pattern transformation of the above system is as follows: taking Fig.1(a) as an example, object v from region delimited by region 1 is transferred into region in membrane 2, and then begin to evolve by using some of the two rules in 2. Thus, if rule r_1 is selected according to its probability, then object v is rewritten into object $v_{'}$ in cell 2 (of course, this could be any multiset); if r_2 is selected instead, then object v is removed from the skin membrane. The system halts when reaching a given condition, typically a number of iterations or cycles of the evolution of the system, because usually when modeling complex systems there is no beginning or end (differently from P systems generating numbers, computing functions or solving computationally hard problems); instead, in this case the result of the computation is indeed the observation of the system itself, including whatever elements (individuals and other possible variables involved) subject to study.

Definition 2.2 ([21]). A *multi-environment* P system of degree (m, n) with $m \geq 1$, $n \geq 1$, taking T time units, $T \geq 1$, is a tuple

$$\begin{aligned} \Pi = (G, \Gamma, \Sigma, T, R_E, \mu, \Pi, \{f_{r,j} | r \in R_\Pi \wedge 1 \leqslant j \leqslant m\}, \\ \{\mathcal{M}_{i,j} | 1 \leq i \leq n, 1 \leq j \leq m\}, \{E_j | 1 \leq j \leq m\}) \end{aligned} \tag{2}$$

where

- $G = (V, S)$ is a directed graph such that $(x, x) \in S$, for each $x \in V$. Let $V = \{e_1, e_2, ..., e_m\}$ whose elements are called environments.
- Γ is the working alphabet and $\sum \subsetneqq \Gamma$ is an alphabet describing the objects that can be presented in the different environments.
- R_E is a finite sets of communication rules between two environments, of the form

$$r_{e_j,e_{j_l}} \equiv (x)_{e_j} \xrightarrow{p_{(x,j_1,j_2,\ldots,j_h)}} (x'_1)_{e_{j_1}} ... (x'_h)_{e_{j_h}} \tag{3}$$

where $x, x'_1, \ldots, x'_h \in \sum$, $(e_j, e_{j_l}) \in S (l = 1, ..., h)$ and $p_{(x,j_1,j_2,\ldots,j_h)}(t) \in$ [0,1]. For the same left-hand size $(x)_{e_j}$, the sum of functions associated with the rules from R_E equal to 1.

- $\Pi = (\Gamma, \mu, R_\Pi)$ is a P system skeleton representing the m P systems respectively placed inside the m environments (with the same alphabet, membrane structure and rules). Each environment e_j contains exactly one P system with this skeleton Π. The only difference among them will be derived from the different parameters and initial multisets that can be initially placed inside the P system of each environment.
- $\mathcal{M}_{i,j}$, $1 \leq i \leq n$, $1 \leq j \leq m$, are the multisets of objects initially present inside each of the n membranes of the m environments.
- E_j, $1 \leq j \leq m$, are the multisets of objects initially present in the m environments.

Taking Fig.2 (b) as an example: in the above system, there is a PDP system like Fig.1. (a) inside each environment. Object x from environment e_j, $1 \leq j \leq 4$, can move to another environment e_k (maybe to more than one at the same time) using rule $r_{e_j,e_{j_l}}$; during its transmission, object x from environment e_j can be rewritten into $x'_i, 1 \leq i \leq h$, in environments e_{j_1} to e_{j_h}.

When studying real world ecosystems, the first type (single-environment) can only study the population dynamics of species in a region, while the second type (multi-environment) is used to model various species distributed in more than one environment. For the latter, each environment e_j contains one P system with the P system skeleton Π (if we want to identify the different multisets placed inside the P system of each environment, these systems might be referred to as Π_j). Besides, it is worth emphasizing the important role played by the information communication among environments (by sending one object to one or more neighboring environments, possibly transforming this object into a different one inside each target environment). At the same time, the m P systems placed in the different regions are executed synchronously (let us also note that, inside each one of these m P systems, a second level of parallelism is present through

the parallel execution of rules in the n internal membranes of each system). The above analysis shows that a single-environment PDP system is a special case of a multi-environment PDP system.

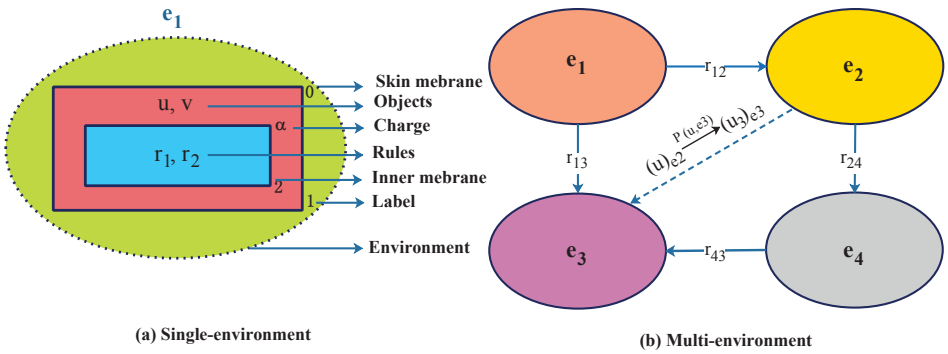


Fig. 1. A portion of classified PDP systems used for modeling ecosystems. (a) A single-environment PDP system with two membranes. (b) A multi-environment PDP system with four environments with the same P system skeleton placed inside each environment (their internal structure being omitted).The system shows population activity of the four environments e_1 to e_4.

In what following the main uses of PDP systems to model ecosystems are analyzed.Thus, a synthesis of several papers published since 2013 illustrates that different types of P systems have been used for predicting population dynamic of ecosystems. Well-known membrane systems can be used for modeling ecosystems in order to assess the projected number of individuals of certain species and their distribution (in terms of ages and locations). So far, a number of endangered species (listed in literature in section 1) have been studied. They focus on different species, study different processes and phenomena, and there are some differences in the definition of rules such as counts or types and subtly different membrane structures (*e.g.*, single vs multi-environment, or different number of membranes in the P system skeleton). However, the general structure of the systems and their dynamics can be extracted for a general protocol, as explained in [26]. A simplified version of such protocol is outlined in the following steps described below.

Step 1. Obtain biological data of the species studied. In the present study pedigree data is available. The information includes the number of female (male) individuals, age, and birthday or death date. Depending on the biology of the species (usually animals), we need to further know other information not recorded in datasets, typically related with processes of interest; for example, their living habits or feeding needs.

Step 2. Define a conceptual model. According to the evolutionary behavior of the species, *i.e.*, feeding, reproduction, mortality, and so on, a preliminary

general model (conceptual model) is abstracted from these basic processes, and then each module of this model is given a certain priority.

Step 3. Define the computational model: starting from the conceptual model, the computational model is built based on a mathematical framework Π, in our case PDP systems. Natural evolutionary behaviors from conceptual model are symbolized, representing the underlying processes with the elements of the computational model. The necessary mapping for this model involves: (a) designing the membrane structure of the skeleton P systems (cell-like structure) to place inside the environments, including the initial multisets representing individual objects (an animal $\leftrightarrow$ an object) and symbolic food; (b) designing the evolutionary rule sets capturing the main processes affecting the biology of the species under study, according to their living behaviors. Eventually, a complete multi-environment PDP system (or some other equivalent complete model for the ecosystem) is established. This model should be ready to analyze in terms of its functioning under different scenarios.

Step 4. Choose simulation software: the previous model designed might be analyzed with manual traces, to validate against real data and later use to formulate hypothesis and check the behavior of the system under potential scenarios of interest. However, the manual analysis of big complex systems is not only tedious or error prone, but also impractical and directly intractable in certain case studies. Thus, we need simulation tools where we can debug the models, experimentally validate them and finally using them for intensive virtual experiments under different scenarios of major interest for the ecosystems under study. In the case of PDP systems and similar types of membrane systems, the most widely used software has been the framework provided by `P-Lingua` and `MeCoSim`. This software is used to run experiments to estimate the population size of the species under study in the coming years under different conditions, performing the simulation from the P system defined in Step 3 and the initial data provided by the user. To sum up the process, MeCoSim contains three files: (a) a config file, where we can define the custom app including the setting of simulations, input tables and output fields/charts, along with other input/output information and parameter-related data (please read user manual [137]); (b) data file, whose purpose is to record experimental data sets, *i.e.*, input data, output data and all the parameters corresponding to a specific experiment; (c) model file, where the specification of the P system itself is given, including membrane structure, initial multisets received from the scenario and grammatical rules capturing the rules of the P systems (both skeleton and environment rules).

Step 5: Output predicted data sets. Taken the statistic data sets of a certain year as input, along with all the parameters related with the biology of the species and the conditions of the ecosystem, the system predicts a set of experimental results by using MeCoSim environment, and then running the model loaded for certain number of cycles (usually years) to get the output.

The protocol depicted above provides an organized sequence of steps to design a model based on PDP system and use it in a practical way to get new insights from the study of the phenomena under study. This provides a theoret-

ical but also practical framework for the use of size-based indicators to monitor the ecosystem changes of species. From a conservation and management viewpoint, a key advantage followed with these models based on PDP systems and the tools available (where many potential scenarios can be analyzed by simply changing the input data) is that predictions can be obtained according to the evolutionary behavior of species, rather than relying solely on historical baselines that may not be relevant under current or future environmental conditions. The applications of models in the context studied can also include analysis of how several parameters -including growth ratio, reproduction ratio, and mortality ratio, among others- affect predicted changes in the number of species.

3 Application of P systems to ecosystems

In this section, we revise various species that were modeled using different types of P systems. In [4], it is concluded that P systems is a suitable modeling technique to predict population dynamics of various species because of the following features:

• Formulating ecosystem modeling as population dynamic P systems overcomes some of the limitations of purely size- or species-based approaches.

• Natural behaviors of each individual are identifiable: identifiability of individual behaviors such as survival or mortality is known (conforming to the natural evolution law of real species).

• The evolution direction of each individual is uncertain: this uncertainty arises from the probabilistic handling of the underlying natural behavior.

• Increasing super-large samples may change distribution functions of DEs, but in PDP system this issue is not present.

• There is no need for function constraints or condition assumptions such as sampling independence, functional differentiability, existence of expectation or variance when predicting population dynamics. Besides, the modeling methods proposed can further contribute to interpretability and robustness.

Based on the advantages outlined above, the population dynamics of the species present in several ecosystems have been studied in the past with different applications of P systems, as summarized in the following subsections.

3.1 Bearded vulture

The ecosystem modeled in that work is located on the southern slope of the central Pyrenees (Aragon region, Spain), a mountainous area belong to the Eurosiberian biogeographic region, which encompasses the three geomophological regions of the Pyrenees: the 'Axial Pyrenees', 'Internal Sierras', and 'External Sierras'. Bearded vultures are distributed in different regions within those areas.

The bearded vulture is a cliff-nesting and territorial large scavenger. This species is the only vertebrate that feeds almost exclusively on bone remains of herbivores living in the three habitats mentioned, *i.e.*, animals like red deer, fallow deer, roe deer and sheep. The remains of these animals was predicted to be

the major limiting factor for the survival of avian scavengers during winter and summer [66]. Bearded vulture has a mean lifespan in wild birds of 21.4 years[136]. In general, the mean age of the successful reproduction is 11.4 years [4]. With every spawning, usually only one chick survives due to the aggression, although the species can produce (as frequently does) two eggs. Recently, field technicians from the Conservation of the Bearded Vulture have carried out annual breeding surveys, indicating that the fertility ratio of the species in the Pyrenees is around 30%, which makes this species become one of the rarest raptors.

Taking into consideration all the evolutionary characteristics and the core parameters affecting the changes in the population size of the species, different types of P systems were used to model ecosystems related to bearded vulture. In the initial phase, a bearded vulture model was presented by a single-environment P system, whereas also different rule selection methods started to be explored. Thus, in [11], based on the principle of bio-chemistry reactions, they selected rules by using stochastic constants, so that a rule will be executed if the condition of intrinsic reactivity given a certain threshold is met. However, this technique cannot exploit in general the full range of rules of the system, involving a number of different processes subject to different natural laws. Thus, in order to solve the drawbacks of limiting the use of rules mentioned, a probability-based population dynamic P system is defined, where rules are chosen in a probability-based roulette way. Experimental results show that, in comparison with previous P system, this system can better simulate the trends of population dynamics of bearded vultures, in the sense of showing a higher accuracy with respect of the validation dataset.

According to the analysis conducted in this work, the population belonging to a single environment usually contains a small number of individuals, with their genetic variability probably lost; this loss may happen not only during the founding event, but also during subsequent generations, when the population remains small and the exchange of individuals with other populations is minimal. In order to capture other scenarios the initial scope is widened. Thus, it is frequently the case of ecosystems with certain separated regions with some degree of communication among them; there, in order to modify the breeding rate by increasing genetic diversity and enhance the survival rate of individuals, they can move among areas; this is captured in the case studied by a multi-environment P system with two nested membranes. In this system, each environment contained 17 different types of animals corresponding to 13 species. Besides, there was communication between two environments, that is, individuals from the same species coming from different regions could mate, thus increasing genetic diversity and modifying the survival rate of bearded vulture. In [66], Margalida *et al* also studied multi-environment population dynamic P systems. Their experimental results showed that these types of P systems can truly reflect the trend of beard vulture, improving the results compared with those of single-environment P systems.

3.2 Zebra mussel

The zebra mussel (*Dreissena polymorpha*) is a freshwater mussel living in several of the major river basins, including Ribarroja reservoir in the north of Spain. This species is an invasive species. Its appearance in Spain and several European countries resulted in adverse impacts on industry, economy and ecology [138, 139]. Zebra mussel is a dioecious species with an r-selected reproductive strategy, consisting in external fertilization and planktonic larval stages. Its success colonizing new environments may be attributed to high fecundity, efficient larval dispersal, few natural controls and its ability to adhere to hard substrates [140].

Zebra mussel has become a dangerous threat by feeding competition and alternation of river sediments, to native mussels. As these native mussels are threatened or endangered, current control strategies in Spain water bodies are therefore limited to avoid spreading of zebra mussel by regulating boating and fishing activities. For these reasons, different biochemical and histological biomakes have undertaken to study the impacts in the population dynamics of zebra mussel, thus aiming to control the dispersal of zebra mussel, and over other species. In general, traditional approaches applied logistic regression [142], classification and regression tree model [141], rule-based genetic algorithms [143], and maximum entropy method (Maxent) [144] to analyze the alteration of population dynamics of zebra mussel, obtaining a series of good results. However, the use of such equations in the case of the zebra mussel ecosystem imposes some restrictions on its ecological analysis. Hence, some scholars used P systems to predict the change of population dynamics of zebra mussel. Using PDP systems, the maximal advantages are that it is possible to mimic the evolutionary features of animals or to explore the birth or mortality trend of populations, giving the traceability of each adult or larvae individual during the evolution of the system. In [13], according to the categories of species and distribution of their regions, a multi-environment population dynamics P system with 5 cells and 17 areas is used to model the zebra mussel of Ribarroja reservoir. Since zebra mussel must breed in a strict temperature, this parameter is also considered in this P system. Comparing with statistical data, the deviation rate is controlled within 10%. Subsequently, Colomer *et al* [23] used a PDP system with 40 cells and 17 areas to model zebra mussel. This model divided 40 cells into different functions providing the first or second cycle evolution of this species. Simulation results showed that PDP models provide very useful tools to model complex, partially desynchronized, processes that work in a parallel way. According to the analysis above, P system-based model can predict better results when handling characteristics such as the ones present in this species, thus increasing the confidence in the effectiveness of the approach followed.

3.3 Pyrenean chamois

Pyrenean chamois (genus *Rupicapra*) is a mountain ungulate distributed over most of the medium- to high-altitude mountain ranges of Andorra, France

and Spain [145]. There are about 53000 individuals of pyrenean chamois living in these places. The status of the species has not always been so favorable; for example, in the late 60's the population decreased down to the edge of extinction due to indiscriminate hunting.

In order to realistically simulate the evolutionary behavior of this species and estimate the effects of introducing a pestivirus affecting the species, Colomer *et al* [21] modeled Pyrenean chamois using a multi-environment PDP system with thresholds including maxim density. According to the requirements of modeling pyrenean chamois, the processes mainly considered where that of feeding, reproduction and mortality, being these evolutionary behaviors properly abstracted as some rules in the PDP system designed.

As the population dynamics of pyrenean chamois in a significant part of certain regions was highly influenced by an infection of a border disease virus (BDV)[136], this model considered the impacts of diseases caused by BDV on the population dynamics of the species under study.

The changing trend of population scales of species may be affected by many factors. Weather change problems (*i.e.*, increased frequency of extreme weather) can cause the risk of biodiversity loss or collapse, as it may happen to pyrenean chamois. For some species, it may even cause their extinctions in some regions, hence this model also considered the impacts of weather on population dynamics. Besides, other natural factors are also considered in this model. In a P system, natural disasters can be abstracted in different ways; for instance, increasing the mortality of pyrenean chamois. In this model, there is a membrane structure with 10 membranes, where the individuals of each of these regions are subject to the effects of multi-natural conditions. In addition, there are four environments (each one including the same skeleton with the 10 regions just mentioned) corresponding to four study areas where ecological data was available (national reservoir of hunting in Pallars-Aran, RNC Cerdanya-Alt Urgell, RNC Cadi, and RNC Freser-Setcases), all of them belonging to the Catalan Pyrenees.

According to the simulation results obtained with the model designed, belonging to ecological data of 22 years, some differences were observed with respect to the real data available to validate the model. It was concluded after a deeper study that these differences between the values obtained with the model and the statistical data could be further reduced by introducing more nature conditions into P systems. The module, flexible and extensible nature of membrane systems would make it possible to introduce the new elements without major changes in the existing model, and reusing the existing structures and rules.

Due to the fact that monitoring data including weather change varies continuously, and due to the existing relations among different natural conditions (see the selected examples), it is likely that for assessing population dynamics we should focus on multi-index fusion, not just single-index study. A few potential examples can enlighten this thought:

- Hot weather can increase the spread of diseases;
- Bad weather can make species including chamois move to other areas;
- Invasions of alien species may also bring new diseases.

3.4 Giant pandas

Giant pandas are listed as endangered species by IUCN Red Data Book (recently updated to class VU - vulnerable - in the last assessment, 2016). According to the fourth giant panda survey by National Forestry Administration (NFA), only both 1864 wildlife giant panda and 375 captive giant panda including 166 male and 209 female survive on earth by end of 2013. It is a sharply debated question whether panda populations are just beginning to regain lost ground or are already healthier than they have been for many years [44]. Recent studies have clarified that giant panda in more than 200 countries and regions around the world are almost extinct or extinct because present survival rates become exceptionally low.

Facing the survival status of giant pandas, approaches on assessing population size are built. Because P systems can incorporate and quantify several behaviors of species, *i.e.*, reproduction, feeding, mortality and the direct or indirect species interactions over communications, P systems are used to model giant pandas. In [50], based on the ecological data of giant pandas in Chengdu Research Base of Giant Panda Breeding, a probability membrane system is designed with a membrane structure consisting of two nested membranes, a series of objects and evolution rules to represent the giant panda ecosystem. Experimental results showed that the model can approximately simulate the trends of population dynamics of giant pandas. In order to properly mimic the natural behaviors and lifestyles of giant panda in real natural environment, based on the ecological data of giant pandas in China Giant Panda Conservation Research Center, in [113], a release module is added to the previous population dynamics P system with a two-layer nested membrane. The simulation results indicated that the maximum deviation rate between the prediction results and the actual data was basically controlled within $\pm 4.13\%$, which improved the accurate of solutions using P system compared with [50]. Although these systems have already been dedicated to further improve the quality of simulation results mentioned, they are still limited to single-environment conditions, with no clear-cut studying for multi-environment P systems so far. Besides, natural disasters are also not considered in the studied systems to this day.

3.5 Other biological systems

In this section, we survey ecosystem modeling for other biological systems including the quorum sensing regulatory networks of the bacterium vibrio fischeri and that of arabidopsis thaliana. We will study the impacts of P systems on the behavior of populations of other ecosystems. For such ecosystems, P systems are used to model the biological phenomena rather than the behavior of individuals. The model designs and experiment assays were applied to the following ecosystems:

Vibrio fischeri. Quorum sensing is a cell density-dependent gene regulation system that can manage expression of specific sets of genes. Certain pathogenic bacteria use quorum sensing to regulate genes encoding extracellular virulence

factors[3]. The cell density control of luminescence in the symbiotic marine bacterium *Vibrio fischeri* is the best-studied quorum sensing system. Hence, several references used P systems to model this bacteria, concluding that these systems can really simulate the overall effect of the colony. In [7], Bernardini *et al* used a single-environment P system with membrane structure (9,1) to model the quorum sensing regulatory networks of the *Vibrio Fisheri*. In this model, there are 9 compartments, each one representing a bacterium. Inside each of the compartents, rules are used to regulate the reactions of the luminescence genes. Besides, objects can move between a compartment and the environment. Single-environment techniques limited the communication between different types of bacteria. In [96], Romero-campero *et al* presented a multi-environment P system with stochastic constants in order to allow the individuals of bacterial cells to communicate each other. Through experimental results, this multi-environment computational models of quorum sensing in *Vibrio fischeri* can efficiently predict the change of bacterial density.

Arabidopsis thaliana. Gene regulatory networks are useful models based on versatile frameworks for biologists to understand the interactions among genes in living organisms. In order to better reproduce the behaviour and the dynamics of gene networks, an accurate tool -a PDP system- is provided to simulate the behaviour of different types of gene networks of species. Thus, in [118], Luis *et al* first use membrane computing models to reproduce the behaviour of a gene network constructed by the improved LAPP method. In this model, the state of each gene in the network at every moment needed to be coded by a series of binary numbers in the existing environments. Then, through interactions (regulated by the rules defined), the next state of the genes will be produced. The contribution of each interaction is calculated from the previously generated objects in order to assess the global influence. Once global clock equals to 0, the system can stop, obtaining the optimal gene networks. Subsequently, Luis *et al* [119] applied the defined LN DP systems to reconstruct gene logical networks and gene dynamics of *Arabidopsis thaliana* in order to regulate the flowering processes associated to *Arabidopsis thaliana* on a long day scenario. By simulating using the software *MeCoSim*, the designed model proved to match the output data obtained by the latter algorithm.

4 Simulation software

P system simulators have become important computational tools in the processes of model debugging, validation and later virtual experimentation, among other purposes. In this section, we introduce some simulation software products to design models for ecosystems by means of P systems.

The primitive simulators are P-Lingua dedicated languages, which are a class of software frameworks applied to specify and simulate P systems [3]. P-Lingua has been successfully used to fix ecosystem modeling problems [3], formal model checking and several computationally hard problems[136]. Each model displays characteristic semantic constraints that determine the way in which the rules

are selected. Hence, it is necessary for simulation software to take into account different scenarios when the computational tools of P systems come to the fore. Nowadays several P system models, *i.e.*, P systems with active membranes [3], tissue P system models [1] and spiking neural P systems (SN P) systems [1], among others, can be performed in the simulators.

With the development of simulators, in recent years, different software applications have been applied to the simulation and validation of biological systems. Here, we will introduce several developed software tools in the following sections:

MetaPlab (Italy; 2008). In [16], Castellini *et al* applied a software called $MetaPlab$ (MP), a computational framework for metabolic P systems, to model biological phenomena related to metabolism. Metaplab framework consists of the following four layers: (i) MP graph: it takes MP systems as inputs and visualizes them. (ii) MP store data structure. This layer is applied to store all the elements of MP in a suitable Java object form. (iii) Data processing. It is a plugin-based module dealing with biological data. (iv) MP vistas. It copes mainly with the graphical representation of MP structures and dynamics. For more details refer to the web site [146]. Through experiment verification, $MetaPlab$ can deal with dynamic computation problems, flux discovery and regulation discovery problems, compared with traditional simulators.

BioSimWare (Italy; 2010). In [3], Besozzi *et al* presented $BioSimWare$. $BioSimWare$ was a novel software providing a user-friendly framework for complex biological systems, ranging from cellular processes to biological phenomena. $BioSimWare$ implemented several stochastic algorithms to simulate the dynamics of single- or multi-environment models, as well as automatic tools to analyze the effect of variation of the system parameters. $BioSimWare$ supports SBML format, and can automatically convert stochastic models into the corresponding deterministic formulation. Prediction data showed that this software can offer a better comprehension for complex biological.

MeCoSim(Spain; 2010). In [90], the simulation environment $MeCoSim$ was first presented. This software provided a multifunction application for the study, analysis, modeling, visual simulation, model checking and optimization of all the possible variants of P systems covered by P-Lingua framework, plus some linked external simulators. In this software, some plugins have been developed to provide some analysis and model monitoring capabilities. Based on the powerful programming function of the software, it has been successfully applied as an assistant tool for the iterative design of ecosystem models such as literatures mentioned.

Due to the different scope of the first two software products, $MeCoSim$ can be a kind of feasible simulation tool for P systems to model ecosystems following an approach similar to the one described in previous section of the paper. Nevertheless, for $MeCoSim$, there are still relevant challenges to be considered. The core point is that the simulators cannot dynamically tune the ecological parameters (according to real biological parameters, parameters such as reproduction rate must change every year) in the process of performing P systems; that is, these parameters will remain unchanged along the whole simulation, resulting in

the bigger deviation rate between prediction data and real biological data. This is a critical problem to be solved, possibly opening a new line of future work for the software developers.

5 A case study on giant pandas

This section analyzes a particular case study where the methodology and tools explained in previous subsections are applied. Specifically, the species of interest will be the giant panda, and the following subsections will describe in further detail the purpose, process and conclusions derived from our study.

5.1 Data Availability

The present work aims to capture the main facts and processes related with biological data and evolution of certain populations of giant pandas in captivity. More specifically, the geographical environment researched is Chengdu Research Base of Giant Panda Breeding (GPBB, for short). Taking a closer look, let us remark that the captive giant panda population of GPBB consists of individuals in GPBB, those in Chengdu Zoological Garden, and those who were born in GPBB but are living outside of GPBB; The reference basis for our research is *giant panda pedigree data* complied by Chinese association of zoological gardens, including data belonging to 12 years (from 2005 to 2016). The adequate processing of these data sets could lead to the extraction of relevant statistical data referred to the number of female and male giant pandas per year, including the age of each individual, being these data the basis and driving force, along with the deep study of the processes involved in the biological evolution of the species, to model the ecosystem.

5.2 Model Design

Population dynamic P systems (PDPs) based models involve two stages: a conceptual model, followed by a computational model; the former one is used to build the different behavior modules of the species in the ecosystem, such as Giant Panda, and give several key parameters required by the model; subsequently, the latter phase is applied from the schema of the previous model to design the specific PDP system detailing the structure, objects involved and setting the computation rules governing the processes analyzed in the biology of the species in the ecosystem and its evolution, according to the conceptual model introduced.

(a) Conceptual Model of PDPs

The main goal of this stage, in our case study, is the design of a novel population dynamics P system based model for captive Giant Pandas in the regions described at the beginning of this section. The processes of interest in the evolution of the ecosystem include the biological aspects related with its life cycle and other possible phenomena happening in the environment; these processes should

Table 1. Summary of studies that have used a new frontier approach, termed PDP systems with different constraints, to assess the number of endangered species under conditions of different types.

A case study (Endangered)	Region/Condition	Comments
M. Cardona *et al.* 2008 [11] **Bearded Vulture**	Region: the cliff-nesting and territorial mountains in Catalan Pyrenees (Northeastern Spain). Condition: Single-environment	(2,1) with two electrical charges (0 or +) where the skin region is used to fix reproduction and mortality and the inner one to fix feeding; Five wild and domestic ungulates are included as carrion (prey) species.
M. Cardona *et al.* 2008 [12] **Bearded Vulture**	Region: Catalan Pyrenees(NE) Condition: Single-environment	The structure of this system is the same as that of [11]. The only difference is: this system is a dynamic P system with the probabilistic approach, while the former used stochastic constants (a rule can be used when the reaction condition reaches a given constant).
M. Cardona *et al.* 2010[14] **Scavenger Birds**	Region: Catalan Pyrenees(NE) Condition: Single-environment	(2,1) with two charges. This system considers not-nomadic species (also called invasion alien species - see part (b) in section 3 -) and density regulation in order to coexist. Subsequently, this model contains 13 species including two new scavenger birds.
M.A. Colomer *et al.* 2010[21] **Pyrenean Chamois**	Region: Catalan Pyrenees(NE) Condition: Multi-environment	(11,4,1) with three electrical charges (-,0,+). The model mainly considers four influencing factors: introduced disease such as pestivirus infection, climate change (refer to part (a) in section 3), hunting, and migrations between areas.
M.A. Colomer *et al.* 2010[24] **Bearded Vulture**	Region: the cliff-nesting and territorial mountains in Catalan Pyrenees (Northeast, Spain). Condition: Multi-environment	The computational model of the probabilistic P system is the same as that of [21] (please refer to third case in this table for the detailed introduction about the model of a P system).
M. Cardona *et al.* 2011[13] **Scavengers/Zebra mussel**	Region: Catalan Pyrenees (NE Spain) /a fluvial reservoir (Ribaroja-Ebro river, Northeast Spain) Condition: Multi-environment	For the scavengers, the structure is the same as [11], hence many details have been skipped. For mussels, the structure is (5,17,1) with tree electrical charges. This model mainly focuses on factors such as water temperature (see part (a) in section 3 for impacts of the factor), fixation of the mussel to the substrate, movement of larvae and density regulations.
M.A. Colomer *et al.* 2011[22] **Scavenger Birds**	Region: (Spain) Catalan Pyrenees/Pyrenean and Pre-pyrenean mountains. Condition: Multi-environment	(2,2) with environment change module where any of species will move to another area when the capacity reaches a threshold. The model studied: (a) 13 species, including three avian scavengers (three types of vultures) as predator species, plus six wild ungulates and four domestic ungulates as prey species; (b) the interactions between species; (c) the communication between two areas; (d) load capacity regulation.
M.A. Colomer *et al.* 2011 [20] **Plant Communities**	Region: (sub)Alpine(NE Spain) Condition: Multi-environment	(5,5) with climatic variability (part (a) in section 3) and orographic factors (part (c)). More importantly, the model first emphasizes on the impact of the plant community module on population dynamics. The remaining modules are similar to those in previous models.
M.A. Colomer *et al.* 2012 [25] **A carnivore that predates on ungulates and five ungulates**	Region: Catalan Pyrenees(NE) Condition: Single-environment	(11,2) with three electrical charges. This model mainly considers the impacts of environment factors such as weather, orography and soil conditions on carnivore size.

Table 2. (Contined) summary of studies that have used a new frontier approach, termed PDP systems with different constraints, to assess the number of endangered species under conditions of different types.

Case study	Region/Condition	Comments
A. Margalida *et al*. 2011[65]. **Scavenger Birds**	Region: Catalan Pyrenees(NE) Condition: Multi-environment	The model only considers wild ungulates due to the limitation of domestic carcasses. Undoubtedly, this causes an impact on the biomass. The model of (2,2) structure verified that when only considering wild ungulates the ecosystem cannot offer enough food for predators.
A. Margalida *et al*. 2012[66]. **European vultures** as **the Bearded vulture, Egyptian vulture, and Cinereous vulture**	Regions: 10 municipalities in Catalonia, Northern Spain. Food source: the four scenarios of food availability considered. Condition: Multi-environment	Taking 10 areas and 4 avian scavengers as the research object, the model considers the impact of climate variations, such as seasons (summer and winter) (part (a) in section 3), food shortage, density regulation, and changes in species habitats (insufficient resources) on population dynamics.
M.A. Colomer *et al*. 2014[23]. **Zebra mussel**	Region: Reservoir of Ribarroja Condition: Multi-environment	(40,17), where the first 20 membranes are used for 20 weeks reproductive cycle, 16 for the weeks of second reproductive cycle, and the last two membranes are used to preform mortality;
Z. Huang, G. Zhang, *et al*. 2017 [50]. **Domestic Giant Panda**	Two regions: Chengdu Research Base of Giant Panda Breeding (GPBB)/China Conservation and Research Center for Giant Panda(CCRCGP) (Wolong). Condition: Single environment	(2,1) where two membranes are used to evolve and store object information; The evolution process of the species: RMF+Rescue module, where RMF is also modified as RFM, FMR or other forms.
H. Tian, G. Zhang, *et al*. 2018 [113]. **Domestic Giant Panda**	Two regions: GPBB/CCRCGP Condition: Single-environment	The membrane structure is the same as in [50], and the only difference is that release module is added to the previous module, that is, RMF+Rescue module+Release module.
F. Bernardini, *et al*. 2005 [7]. **The quorum sensing regulatory networks of the bacterium Vibrio Fischeri.**	Region: Marine; Condition: Single-environment; Evolutionary Rule choices: in the stochastic way	(9,1) where multisets of objects are used to model bags or soups of chemicals whereas rules are used to model generic biochemical processes.
F. Romero-Campero *et al*. 2008. [96]. **Quorum Sensing in Vibrio Fischeri**	Region: Marine; Condition: a parametric multi-environment P system; Rule choices: stochastic approach.	(N;25), multi-compartmental P system where N bacteria are randomly placed inside a multi-environment with 25 different regions, that is, there is an uncertain number of bacteria in each region.
L. Valencia-Cabrera, *et al*.[118] 2013. **Gene regulatory networks**	Condition: single-environment	The first membrane computing model is applied to reconstruct the behavior of logic networks of species.
L. Valencia-Cabrera, *et al*. [119] 2013. **Gene regulatory networks**	Case study: Arabidopsis thaliana Condition: single-environment	Based on [118], P systems are used to reproduce a logic gene network of (real) arabidopsis thaliana in order to regulate the flowing processes.

be simulated by computers according to the model designed and the input data about giant panda populations and related parameters. PDPs prediction focuses on the changes in the female or male population size and distribution of ages. In the following subsection, we design the conceptual model that will strat defining the modules that will be finally applied into the computational model.

Before introducing the conceptual model, we first describe the life cycle of giant panda, the classification of age groups and types of food required. In this simplified case study, the whole evolution behavior of Giant Panda considered consists of four processes: reproduction, mortality, feeding and rescue. Thus, the ecosystem model to design should also contain modules for these four processes involved.

In this model, according to the specialists' understanding of Giant Panda life habits, age groups are classified into six ranges; a detailed introduction is in Table 1, where the classification standards of female and male may have a little difference. For food required, we mainly consider three kinds of food as the necessary feeding sources: bamboos, bamboo shoots and other food (including milk, fruits and so on).

Table 3. The classification of age groups

	Infancy	Sub_adult	Adult	Middle-age	Quinquagenarian	Senile
Female_GP	[0, 1]	[2, 4]	[5, 8]	[9, 17]	[18, 27]	[28, 34]
Male_GP	[0, 1]	[2, 4]	[5, 6]	[7, 17]	[18, 27]	[28, 36]

The whole setup (an entire year) of this conceptual model can mainly consist of the four models depicted in Fig. 2, where the first three models are executed sequentially, while the rescue module runs in parallel with respect to them. It involves a series of rules when executing each model that are abstracted from different evolution behaviors. At every instant, each individual will evolve according to a series of rules in a parallel way with other individuals. The model graph of this conceptual model is illustrated in Fig. 2.

As shown in Fig. 2, the conceptual model considered in PDPs can be grouped into four modules: reproduction module, mortality module, feeding module and rescue module. This subsection will first focus on these four modules:

I. Reproduction module. Many new individuals are born every year. Depending on the born rate, the number of new individuals will emerge depending on the number of female individuals in the breeding age, with certain variations due to the probabilistic nature of the rules, capturing the inherent randomness present in nature up to a certain degree. In further extensions of this model, we could determine the number of new individuals also depending on additional or alternative direct or indirect factors.

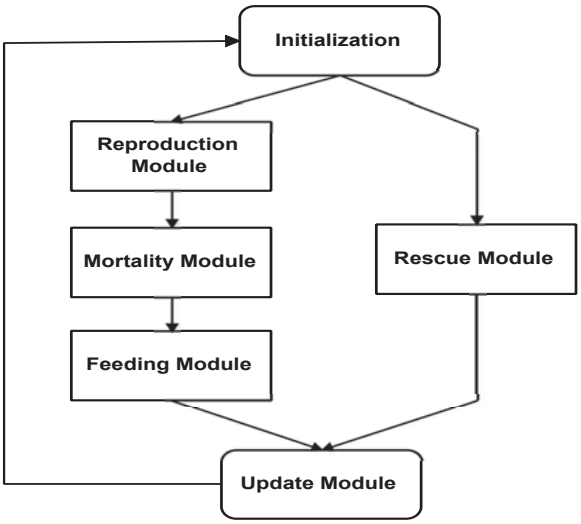


Fig. 2. A conceptual model graph

II. Feeding module. During each cycle (in our case, a year), plenty of food is provided to captive Giant Pandas, where the main types of food are bamboo, bamboo shoots and others (fruits or milk). In captive environment, all required food can be satisfied. However, it is worth incorporating this process in our models in order to also track the food consumption and, more importantly, we can also consider analyzing scenarios with certain damage producing scarcity of food sources due to some natural disasters like earthquakes or climatic anomalies.

III. Mortality module. Some individuals of the population can die with a certain mortality rate in this phase, also subject to probabilistic rules. In comparison with wild giant pandas, the fundamental results showed that the mortality of captive giant pandas is significantly lower, and the longevity of captive individuals is clearly higher, given the improved life conditions and medical technologies of captive population, also subject to improvements along the years. In this model, both parameters about mortality and longevity can be tuned on the basis of statistical data.

IV. Rescue module. In nature, population size is affected not only by the birth and death of individuals, but also by the addition of new rescue individuals to the ecosystem. The rescue module mainly describes the change of rescued information including the number of giant pandas rescued from the wild, the sex and the age. Historically, the number of rescued individuals ranges from at least one panda without rescue every year to at most two rescued individuals. In addition, each individual has a certain probability of being rescued. The maximum life span of (virtual) rescued giant pandas is the same as that of (natural) wild giant pandas. At the same time, it is difficult to find and rescue young and older giant pandas because of their weak mobility, so the rescue

situation of giant pandas at this age is not considered. In simulation, the number of rescued giant pandas of each sex and age is pseudo-random. In this way, they will be added to current population at the end of a cycle. Because the rescue module only simulates the phenomenon that rescued individuals may occur every year, the steps of performing this module are not affected by other modules.

Before designing this model we have just presented, it was necessary to obtain data and qualitative information about the different processes and behavioral facts related with Giant Panda life cycle. This module had to consider natural factors such as reproduction habits, mortality rates, specific evolutionary behavior and conduct patterns, determined according to the actual situation observed. Concerning rescue module, the data about rescued individuals per age and gender were collected from past experience, and the causes for these rescues analyzed with the experts in charge of managing the ecosystem. Some decisions were made concerning the estimation of rescued individuals per year, and some increase in the comparative age of these individuals when incorporated into captive life was applied.

(b) Computational (Mathematical) Model of PDPs

The main goals of the research conducted was to assess the evolution in the population size along the years under certain given conditions and initial populations, by using a model base on P systems. This step involved a number of processes and parameters related with the biology of the species and the ecosystem, that should be translated into the concepts belonging to the formal model used: P systems (*i.e.*, structures - environments, membranes -, objects, evolution rules - skeleton and environment rules -). Besides, the proper semantic constraints and considerations inherent to P systems had to be taken into account, along with the accuracy of these conditions (like the application of certain probabilities associated with the rules) in mimicking the natural processes involved. For PDPs, once the simple P system skeleton to be placed inside the environment was defined, we mainly focus on designing the evolutionary rules to capture the processes taking place in our subject ecosystem.

Due to the fact that only one target species was the subject of our study, and no movement among regions was considered as part of the initial design, no complex environment interactions was required. Therefore, a single environment was enough for this case study. Inside such environment, the P system skeleton was designed with two membranes: the external membrane, denoted as a skin membrane (directly contained inside the single environment), labeled as 1; and the inner membrane (used to perform most of the evolution operations), labeled as 2. Then, inside those two membranes all processes occur, sequencing the proper operations of three of the modules and simultaneously performing the actions related with the other one: the rescue. In every module, all the individuals subject to the rules of the module evolve in parallel. A trace of a simulation of the model along a cycle (a year in this case) is illustrated in Fig. 3. In what follows the model will be described in a semi-formal way through its main constituent parts.

The PDP system Π modeling our case study can be defined as the tuple:

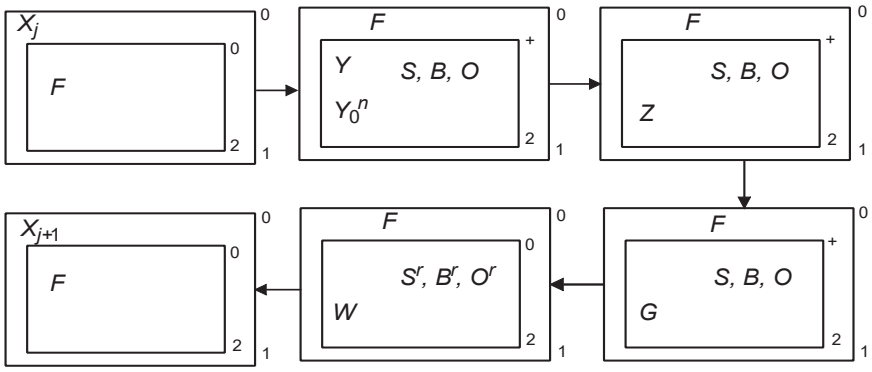


Fig. 3. Evolution patterns of a simulation cycle

$$(G, \Gamma, \Sigma, T, R_E, \mu, R, \{f_{r,1}|r \in R\}, M_{1,1}, M_{2,1}, E_1) \tag{4}$$

where

- G is an alphabet of objects;
- Γ is an evolutionary object set, where

$$\Gamma = \{X_{i,j}, Y_{i,j}, Z_{i,j}, W_{i,j} : 1 \leq i \leq 2, 1 \leq j \leq k_{i,5}\} \tag{5}$$

where

Symbol i represents female (i=1) or male (i=2) Giant Panda, j represents the age of Giant Panda.

In the initial phase, each individual (giant panda) is abstracted as an object, therefore the situations of objects should change as the life span of individuals change, for exampe, $X_{i,j} \rightarrow Y_{i,j} \rightarrow Z_{i,j} \rightarrow W_{i,j}$. Object $X_{i,j}$ is abstracted as a i-year-old male or female (j) panda individual before reproduction modules, object $Y_{i,j}$ is abstracted as a j-year-old male or female (i) panda individual in mortality modules, object $Z_{i,j}$ is abstracted as a j-year-old male or female (i) survival panda individual, object $X_{i,j}$ is abstracted as a j-year-old male or female (i) panda individual after feeding, symbol $C_{i,j}$ is abstracted as a j-year-old male or female (i) rescue panda individual.

In formula (2), symbols S,B and O denote different foor such as bamboos, bamboo shoots and others (fruits or milk). Symbols F and A are auxility alphabets, where F is to promote food production, A is to trigger rescue behaviors.

- Σ is an environment alphabet set that is an empty set in an initial state;
- T is a simulation circle;
- μ is a hierarchical membrane structure with two membranes labeled as 1 and 2, where the skin membrane (environment) is labeled as 2, $\mu = [[]_1]_2$;
- Both $M_{1,1}$ and $M_{2,1}$ are object sets over V associated with regions 1 and 2, where

- $M_{2,1} = \{X_{i,j}^{q_{i,j}} : 1 \leq i \leq 2, 1 \leq j \leq k_{i,5}\}$, where $q_{i,j}$ is the number of all the j-year-old male or female (i) pandas;
- $M_{1,1} = \{F, A\}$.

In this system, rewriting rules of all modules consist of initialization rules, reproduction rules, rescue rules, mortality rules, feeding rules and uodate rules. rule set R of computational model are designed as follows:

(a) Initialization rules: food supply

$$r_1 \equiv [F]_2^0 \rightarrow F[S^{g_3} B^{g_4} O^{g_5}]_2^-$$

(b) Reproduction rules

- Panda individuals unreaching reproduction age

$$r_2 \equiv X_{i,j}[]_2^0 \rightarrow [Y_{i,j}]_2^-, \, i \in [1,2], j \in [0, k_{i,12}]$$

- Female reproduction individuals during the reproductive period

$$r_3 \equiv X_{2,j} \xrightarrow{g_y} [Y^y Y_{2,j}], \, 1 \leq y \leq 2, k_{2,12} \leq j \leq k_{2,13}$$

- Female unreproduction individuals during the reproductive period

$$r_4 \equiv X_{2,j}[]_2^0 \xrightarrow{1-g_1-g_2} [Y_{2,j}]_2^-, k_{2,12} \leq j \leq k_{2,13}$$

- Male panda individuals

$$r_5 \equiv X_{1,j}[]_2^0 \rightarrow [Y_{1,j}]_2^-, \, k_{1,12} \leq j \leq k_{1,13}$$

- Aged panda individuals

$$r_6 \equiv X_{i,j}[]_2^0 \rightarrow [Y_{i,j}]_2^-, \, 1 \leq i \leq 2, k_{i,13} \leq j \leq k_{i,5}$$

- Neonatal female or male panda individuals

$$r_7 \equiv [Y]_2^- \rightarrow [Y_{i,0}]_2^+, 1 \leq i \leq 2$$

(c) Rescue Rules

-Number of giant pandas rescued from the wild field

$$r_8 \equiv [A]_2^- \xrightarrow{pc_c} AC_c[]_2^+, cmin \leq c \leq cmax$$

- Sex for rescued giant pandas

$$r_9 \equiv [C \xrightarrow{pg_i} C_i]_1^0, 1 \leq i \leq 2$$

- Age for rescued giant pandas

$$r_{10} \equiv [C_i \xrightarrow{pg_i} C_{i,j+1+round(j/3)}]_1^0, 1 \leq i \leq 2, 0 \leq j \leq cmaxage$$

(d) Mortality Rules

- Survival individuals of infancy giant pandas

$$r_{11} \equiv [Y_{i,j} \xrightarrow{1-k_{i,6}} Z_{i,j}]_2^+, 1 \leq i \leq 2, 0 \leq j < k_{i,1}$$

- Mortality individuals of infancy giant pandas

$$r_{12} \equiv [Y_{i,j} \xrightarrow{k_{i,6}} \lambda]_2^+, 1 \leq i \leq 2, 0 \leq j < k_{i,1}$$

- Survival individuals of young giant pandas

$$r_{13} \equiv [Y_{i,j} \xrightarrow{1-k_{i,7}} Z_{i,j}]_2^+, 1 \leq i \leq 2, k_{i,1} \leq j < k_{i,2}$$

- Mortality individuals of young giant pandas

$$r_{14} \equiv [Y_{i,j} \xrightarrow{k_{i,7}} \lambda]_2^+, 1 \leq i \leq 2, k_{i,1} \leq j < k_{i,2}$$

- Survival individuals of adult giant pandas

$$r_{15} \equiv [Y_{i,j} \xrightarrow{1-k_{i,8}} Z_{i,j}]_2^+, 1 \leq i \leq 2, k_{i,2} \leq j < k_{i,3}$$

- Mortality individuals of adult giant pandas

$$r_{16} \equiv [Y_{i,j} \xrightarrow{k_{i,8}} \lambda]_2^+, 1 \leq i \leq 2, k_{i,2} \leq j < k_{i,3}$$

- Survival individuals of middle-age giant pandas

$$r_{17} \equiv [Y_{i,j} \xrightarrow{1-k_{i,9}} Z_{i,j}]_2^+, 1 \leq i \leq 2, k_{i,3} \leq j < k_{i,4,1}$$

- Mortality individuals of middle-age giant pandas

$$r_{18} \equiv [Y_{i,j} \xrightarrow{k_{i,9}} \lambda]_2^+, 1 \leq i \leq 2, k_{i,3} \leq j < k_{i,4,1}$$

- Survival individuals of middle aged and old giant pandas

$$r_{19} \equiv [Y_{i,j} \xrightarrow{1-k_{i,10}} Z_{i,j}]_2^+, 1 \leq i \leq 2, k_{i,4,1} \leq j < k_{i,4,2}$$

- Mortality individuals of middle aged and old giant pandas

$$r_{20} \equiv [Y_{i,j} \xrightarrow{k_{i,10}} \lambda]_2^+, 1 \leq i \leq 2, k_{i,4,1} \leq j < k_{i,4,2}$$

- Survival individuals of old giant pandas

$$r_{21} \equiv [Y_{i,j} \xrightarrow{1-k_{i,11}} Z_{i,j}]_2^+, 1 \leq i \leq 2, k_{i,4,2} \leq j < k_{i,5}$$

- Mortality individuals of old giant pandas

$$r_{22} \equiv [Y_{i,j} \xrightarrow{k_{i,11}} \lambda]_2^+, 1 \leq i \leq 2, k_{i,4,2} \leq j < k_{i,5}$$

- Longevity giant pandas

$$r_{23} \equiv [Y_{i,k_{i,5}} \to \lambda]_2^+, 1 \leq i \leq 2$$

(e) Feeding Rules. Giant pandas in different periods needs to acquire different quantities of food, therefore we divide feeding rules into three periods like infancy, young and other periods.

- Feeding rules for infancy giant pandas

$$r_{24} \equiv [Z_{i,j} S^{f_{i,1}} B^{f_{i,2}} O^{f_{i,3}} \to W_{i,j}]_2^0, 1 \leq i \leq 2, 0 \leq j < k_{i,1}$$

- Feeding rules for young giant pandas

$$r_{25} \equiv [Z_{i,j} S^{f_{i,4}} B^{f_{i,5}} O^{f_{i,6}} \to W_{i,j}]_2^0, 1 \leq i \leq 2, k_{i,1} \leq j < k_{i,2}$$

- Feeding rules for giant pandas during other periods

$$r_{26} \equiv [Z_{i,j} S^{f_{i,7}} B^{f_{i,8}} O^{f_{i,9}} \to W_{i,j}]_2^0, 1 \leq i \leq 2, k_{i,2} \leq j < k_{i,2}$$

(f) Update rules

- Empty food rules

$$r_{27} \equiv [S \to \lambda]_2^0$$

$$r_{28} \equiv [B \to \lambda]_2^0$$

$$r_{29} \equiv [O \to \lambda]_2^0$$

- Update circle rules

$$r_{30} \equiv [W_{i,j}]_2^0 \to X_{i,j+1}[]_2^0, 1 \leq i <\leq 2, 0 \leq j \leq k_{i,5}$$

$$r_{31} \equiv F[]_2^0 \to [F]_2^0$$

$$r_{32} \equiv A[]_2^0 \to [A]_2^0$$

where

symbol $k_{i,1}$ indicates that captive giant pandas are in sub-adulthood; symbol $k_{i,2}$ indicates that these pandas are in adulthood; symbol $k_{i,3}$ represents that some pandas are in middle age stage; symbol $k_{i,4,1}$ represents that some pandas are in middle-aged old stage; symbol $k_{i,4,2}$ represents that some pandas are in old age stage; symbol $k_{i,5}$ denotes the longevity age of giant pandas. Symbols $k_{i,6}$-$k_{i,11}$ denotes the mortality of giant pandas in different stages, where $k_{i,6}$ is in infancy stage, $k_{i,7}$ is in sub-adulthood, $k_{i,8}$ is in adulthood, $k_{i,9}$ is in middle age stage, $k_{i,10}$ is in middle-aged old stage, $k_{i,11}$ is in old age stage. Symbols $k_{i,11}$ and $k_{i,12}$ show the age of breeding giant pandas at the begining and at the end stages, respectively.

Symbol g_1 denotes the probability of giant panda fertility in reproductive period, g_2 denotes the probability that giant pandas in reproductive period breed twins. Symbol g_3 is the number of supplied bamboos within a year, g_4 is the

number of supplied bamboo shoots within a year, symbol g_5 is the number of other supplied food like fruits or milk within a year.

Symbol $f_{i,1}$ is the amount of bamboos consumed by all reserached giant panda individuals in a year, $f_{i,2}$ is the amount of bamboo shoots consumed by all individuals in a year, $f_{i,3}$ is the amount of others consumed by all individuals in a year. Symbol $f_{i,4}$ is the amount of bamboos consumed by a sub-adult panda within a year, $f_{i,5}$ is the amount of bamboo shoots consumed by a sub-adult panda within a year, $f_{i,6}$ is the amount of other food consumed by a sub-adult panda within a year. Symbol $f_{i,7}$ is the amount of bamboos consumed by a adult panda within a year, $f_{i,8}$ is the amount of bamboo shoots consumed by a adult panda within a year, $f_{i,9}$ is the amount of other food consumed by a adult panda within a year.

Symbol *cmin* defines the minimum number of rescued wild giant pandas, *cmax* defines the maximum quantity of rescued wild individuals, *cmaxage* defines maximum age for rescuing wild individuals, pc_c is the probability of rescuing c wild individuals in a year, pg_i is the probability of rescuing wild individuals that the sex of each panda is i, pa_j is the probability of rescuing wild individuals that the age of each panda is j.

The values of these constants have been obtained experimentally. For each probabilistic parameter, if it is large, there is a larger population fluctuation, whereas if it is small, there is not an obvious change in size, both of which become impossible to obtian reliable estimates. Parameters reflect the severity of natural stochasticity (k decreases with increasing environmental disasters). It goes without saying that these parameters have considerable significance in population and conservation biology [68].

Step 1: Obtain data set. Data sets about giant pandas come mainly from two regions: Chengdu Research Base of Giant Panda Breeding (GPBB for short) and China Conservation and Research Center for Giant Panda (CCRCGP for short).

Step 2: Initialization. Designing a successful ecomembrane system can require a plenty of parameters such as mortality, reproduction rate, rescue rate and so on. In the initialized configuration, these parameters should be initialized first.

Step 3: Design a conceptual model. Some behavior modelers like mortality module are abstracted from daily behavior of species. According to evolutionary circle, they are given different priorities so as to perform successfully. The model described by the whole process is called a conceptual model.

Step 4: Design a computational model. It is a mathematical modeling of the conceptual model in Step 3. (Please see subsection (b))

Step 5: Output. Simulate and obtain a predicted number of giant pandas.

In summary, for our given expamples-a giant panda population prediction method based membrane systems, the detailed introduction of this method is as follows: we first need to count the basic information of giant pandas in the researched region, i.e., counts, age, sex, and so on; then, we design a concep-

tual with execution sequence according to the fragmented habits (reproduction, feeding, death and rescue) of all researched pandas; next, we can also design a computational model that it contains a standard mathematical formula and a set of rules abstracted from the envolutionary habits of species accroding to a given conceptual; finally, we will obtain a series of the computational resutls by doing a plently of simulation experiments, and output an optimal data. Theroretical analysis indicates that in absence of real data as a reference this method can effectively analyze variation trends of population quantities and evaluate the number of panda individuals in the future.

Table 4. Values of the ecological parameters used in this model for Giant Panda (GP for short) (F=Female, M=Male, A=$\pm 1.35 \times 10^7$), other explanations refer to part 5.2.

Species	i	$k_{i,1}$	$k_{i,2}$	$k_{i,3}$	$k_{i,4,1}$	$k_{i,4,2}$	$k_{i,5}$	$k_{i,6}$	$k_{i,7}$	$k_{i,8}$	$k_{i,9}$	$k_{i,10}$	$k_{i,11}$	$k_{i,12}$	$k_{i,13}$
GP (F)	1	1	4	8	17	27	34	0.09	0.001	0.007	0.008	0.1	0.15	6	20
GP (M)	2	1	4	6	17	27	36	0.05	0.001	0.005	0.0058	0.034	0.091	5	20
Species	i	g_1	g_2	g_3	g_4	g_5	$f_{i,1}$	$f_{i,2}$	$f_{i,3}$	$f_{i,4}$	$f_{i,5}$	$f_{i,6}$	$f_{i,7}$	$f_{i,8}$	$f_{i,9}$
GP (F)	1	0.191	0.098	A	A	A	0	0	182	2920	2920	292	11680	10950	1276
GP (M)	2	0.191	0.098	A	A	A	0	0	182	2920	2920	292	11680	10950	1276

5.3 Experiments

In this subsection, we used the software tool-P-lingua (MeCoSim) [1] to test our experiments. This software is new programming language able to define P systems of different types (frameworks) including the probabilistic framework mentioned in this paper. Subsequently, we first introduce experimental design and simulation (a), and then we analyze the experimental results. Besides, for PDPs, we give idential parameter settings obtained experimentaly.

(a) Experimental design and simulation

PDPs scale individual-level processes up to ecosystem structure and dynamic. Here, we presents in three steps how pedigree data of giant panda can be used to inform population dynamic P systems (PDPs), where population size of giant panda in each year is shown as Fig.6.

First, the individual pedigree data of captivity giant panda that can be used to parameterize PDPs include: food consumption; number of recuse individuals; individual sex and individual age; and the division of individual life span about offspring, adult and old. Once parameterized, PDP can be used to predict the number of individuals at the first time step. This process needs to first calculate number of reproductive and mortility individuals. These are used to calculate the fixed size-specific survival, reproductive, and mortality rates (see Fig. 7). Reproductive and moratility rates determine the change that increases and decreases of population size in and out of the studied sepcies. Because each individual enters

into next status in a probability way, at each time step, number of these individuals varies. The predicted changes at population size through time are uncertain numerically but are controlled within a given confidence interval. The numerical density of species is summed across all individuals at differnt age groups. These are outputted at each time step along with predicted changes. Changes in population size can be described by fitting, at each time step, a straight line (see Fig.8 where the parameters used in PDP systems are derived from the data of Table 5). Predicted changes in population can then be confronted with empirical data for comparison or repeating the above process in conjuntion with a statistical procedure to formally estimate parameters and their uncertainty (see Fig.9).

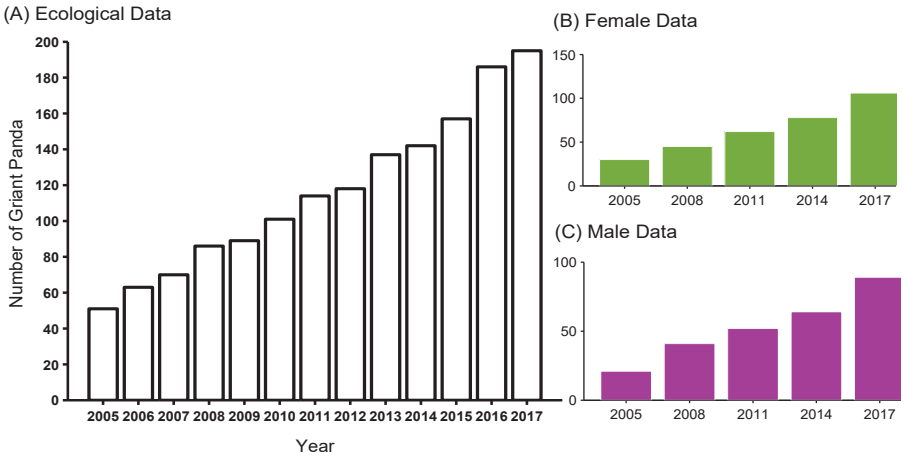


Fig. 4. Population Size of Endangered Giant Panda Published in the Last 13 Years where all the used data are showed though three histograms. (A) Ecological Data: the total number of real giant panda in each year. (B) Female data: the number of female individuals in a selected year. (C) Male data: the number of male individuals in a selected year. (That is, $N_A = N_B + N_C$, N presents the number of giant panda). Such data sets are called statistical data sets which are generally used as input for prediction.

(b) The analysis for experimental results

In Fig.8, by comparing varying trajectories of the number of prediction species to trends in the number of statistical species, we identified the changing regular of population size across the GPBB. In five subgraphs above, we provide strong evidence that the average rate of changes in giant panda populations are 1.86% (data from 2005 as input) per year shown in (a), almost consistent with statistical data, and that similar rates of changes occur across other four group experiments we provided, *i.e.*, 10.26% (2008 as input), 12.84% (2011 as input), 3.41% (2014 as input) (see four other figures in Fig.8). This rate of each group means that on average, the number of griant panda will be really predicted with

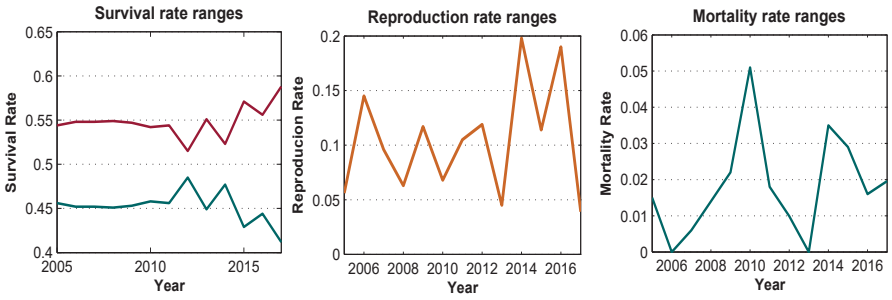


Fig. 5. The statistical survival rate, reproduction rate and mortality rate of endangered giant panda from GPBB for 13 years from 2005 to 2017, given three group trajectories reflecting the dominant caused of changes in population size: natural factors; human behavior; or enigmatic factors. The purpose of three graphs is to offer the basis reference for setting parameters in the process of running PDP systems such that the data in Table 5 are tuned on the basis of Fig. 7.

a small error rate within serveral years. According to these results, we find that the deviation of the number of species per year are controlled within 10% except for special point (Fig.8.) although different input can also led to different prediction results in terms of predicting the number of species in the same year (Fig.9). These devaition results from models show that there is no single, fully integrated model that can simulate all possible species states, and there causes number variation in the combination of parameters including growth rate, birth rate and mortality rate rather than a single cause because of the difficulty of parameterizing interactions.

5.4 Discussion

Data are needed to parameterize and serve for population dynamic P systems (PDPs) to apply them to predict number of individuals of real ecosystems in several years and assess how accurate their predictions of ecosystems structure and function are. Because PDPs are computationally inexpensive feasible for them to moel ecosystem, there could be scope for the development of PDPs as real-time prediction models. Prediction from PDPs could be combined with data collected from GPBB including individual growth rates, indiviudal mortality rates, individual reproductive rates, number of individuals, and number of rescue individuals. More research on the level of complexity needed to accurately predict ecosystem dynamic with uncertain is needed.

Models have been confronted by data in different ways: qualitative comparisons of prediction data from PDP models with standard data from statistics and then more qualitative assessment of PDP models by calculating deviation rate and variance. Choosing the model with earlier history data as a reference, Comparing between prediction data and earlier history data is an approach that is being used for evaluating PDP models.

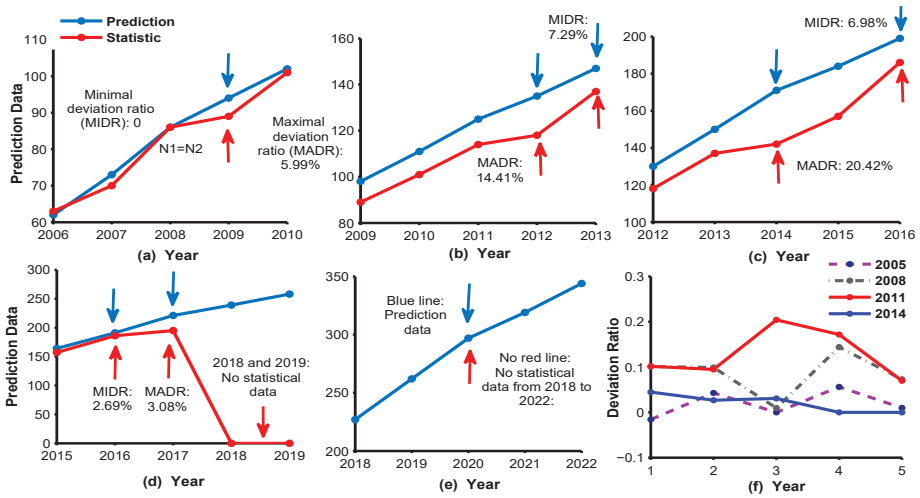


Fig. 6. Prediction data changes and Statistical data changes as different years's input data. There is a significant and roughly difference in the average per-year prediction data and statistical data between the size-rise and the size-decline phases across the 5 years analyzed. Each pair of values corresponds to the predicted or statistic result of the same year. For (**a**), taking 2005 as an input, a PDP system is used to predict five-year population size from 2006 to 2010, respectively; for (**b**), taking 2008 as an input, prediction years from 2009 to 2013; for (**c**), taking 2011 as an input, prediction years from 2012 to 2016; for (**d**), taking 2014 as an input, prediction years from 2015 to 2019; for (**e**), taking 2017 as an input, prediction years from 2018 to 2022; for (f), this graph describes the comparison of the deviation ratio of five groups of data set, the special introduction is given in Fig.9. In five figures above, we list the minial deviation ratio and maximal ratio of each input year (shown by the arrows), where $ratio = (N_1 - N_2)/N_2)$ in which N_1 represents prediction data, N_2 represents statistical data in reality. In (d) and (e), because the pedigree data only counted the data before 2017, there are no data for these years from 2018 to 2022 (lack of red line).

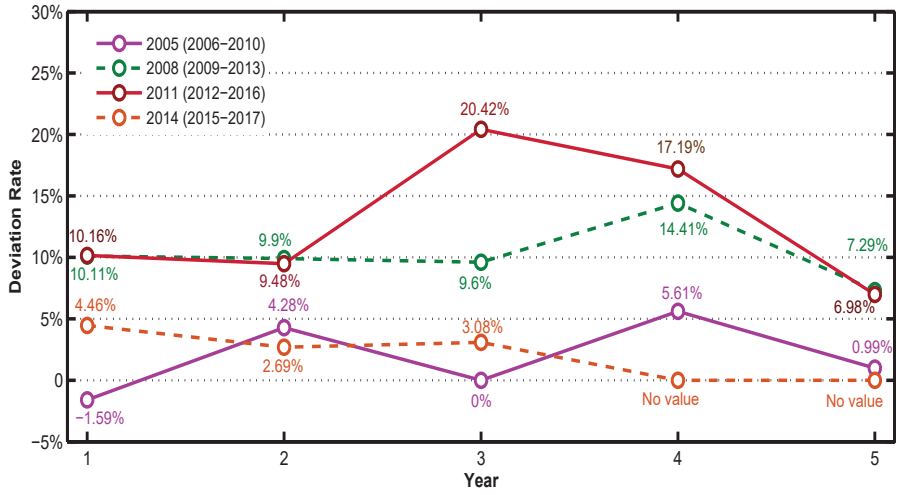


Fig. 7. How the difference occurs in the process of prediction by PDP models. (**dotted** line to **Solid** line) The changes of five-year time intervals (2006-2010) deviation rates (2005 as input); The changes of five-year (2009-2013) deviation rates (2008 as input); The changes of five-year (2012-2016) deviation rates (2011 as input); The changes of five-year (2015-2017) deviation rates (2014 as input); The deviation trajectory generated by the prediction errors. Overlapping prediction parts, *i.e.*, 2009-2010 {input 2005(5.61% and 0.99%), 2008(10.11% and 9.9%)}, 2012-2013 {input 2008(14.41% and 6.98%), 2011(10.11% and 9.48%)}, and 2015-2016 {input 2011(17.19% and 6.98%), 2014(4.46% and 2.69%)}, indicate the input data of different years can predict different results for the same year. In this graph, 'No value' means no deviation rate due to the lack of statistical data in reality.

Uncertainty also comes from our imperfect knowledge about what drivers the change of eocsystems. This is especially critical in modeling ecosystems, where different parameters make different prediction results. Tools to formly integrate data and assess parameter uncertainty are begining to be used in conjunction with PDPs and Data. One key advantage of the PDP framework is that it can account for the effect of parameters and obtain prediction data calculated in a probability way under parameter effect on model output. Because this model can evaluate how uncertainty changes number of individuals, this model is particularly useful for predicting data.

6 Conclusions and Future works

Predicting the change trend of population size is increasingly urgent as changing environmental and climate conditions continue to modify the population dynamic of endangered species. Record species changes have hastened efforts to identify exectinction risks and ameliorate the ultimate causes of decline. In this work, we have investigate the computational models for ecosystems, especially for endangered species. Since early modeling approaches ignored species traits and functional groups, PDP systems applied to threatened and declining populations can now overcome many drawbacks associated with differential equations. Subsequently, we give a global overview of modeling ecosystems using PDP systems. Most of the researches are focused on model definitions with different constraints such as structures, climate conditions, invasion ways and habitat destructions or losses, and their applications to different species. A large number of experiment results verified that PDP systems can predict better results. The key strength of PDP systems is that each sub-rule imitates one of spcies behaviors, making use of all speceis data, leading to the best possible prediction on population size. As a case study, we define a computational model of a single-environment PDP system and apply it to modeling Giant Panda. Compared with statistical data, better prediction data are obtained. These results verified that PDP systems are improving recovery of threateded and declining species, which means that modeling ecosystems using PDP systems have the potential to fix some natural problems in reality.

Building models of giant panda that integrate modeling processes from biogeochemical to the basic evolutionary cycles of this species is only a preliminary work. In the future, some of main conditions needs to be considered: 1) modeling multi-region captive giant panda requires more complex models than single-region species, i.e., communications between different species. 2) Considering the impacts of climate changes, invasion alien species and habit destructions such as earthquake on species requires more real physical models to design the model of a PDP system, and it may be challenging to get more efficient data collection. Our team is trying to solving these problems in order to more accurately estimate ecosystems.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (61972324, 61672437, 61702428), the Sichuan Science and Technology Program (2018GZ0185, 2018GZ0086), New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0043) and Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06).

References

1. Alhazov, A., Martín-Vide, C., Pan, L: Solving graph problems by P systems with restricted elementary active membranes. In Aspects of Molecular Computing, 1-22(2003).
2. Alhazov, A., Cojocaru, S.: Small asynchronous P systems with inhibitors defining non-semilinear sets. Theoretical Computer Science, 12-19(2017).
3. Andersen, M.C., Adams, H., Hope, B., et al: Risk assessment for invasive species. Risk Analysis: An International Journal, 24(4), 787-793(2004).
4. Bahuguna, D., Abbas, S., Dabas, J.: Partial functional differential equation with an integral condition and applications to population dynamics. Nonlinear Analysis: Theory, Methods & Applications, 69(8), 2623-2635(2008).
5. Bax, N., Williamson, A., Aguero, M., et al: Marine invasive alien species: a threat to global biodiversity. Marine policy, 27(4), 313-323(2003).
6. Bayley, P.B., Peterson, J.T.: An approach to estimate probability of presence and richness of fish species. Transactions of the American Fisheries Society, 130(4), 620-633(2001).
7. Bernardini, F., Gheorghe, M.: Population P Systems. Journal of Universal Computerence, 10(5):509-539(2004).
8. Bie D, Gutiérrez-Naranjo M A, Jie Z, et al: A membrane computing framework for self-reconfigurable robots[J]. Natural Computing, 1-12(2018).
9. Bortolussi, L., Lanciani, R., Nenzi, L.: Model checking Markov population models by stochastic approximations. Information and Computation, 262, 189-220(2018).
10. Buiu, C., Vasile, C., Arsene, O.: Development of membrane controllers for mobile robots. Information Sciences, 187, 33-51(2012).
11. Cardona, M., Colomer, M. A., Pérez-Jiménez, M. J., et al: Modeling ecosystems using p systems: The bearded vulture, a case study. In International Workshop on Membrane Computing, 137-156(2008).
12. Cardona, M., Colomer, M., Pérez Jiménez, M.D.J., et al: A P System modeling an ecosystem related to the bearded vulture. In Proceedings of the Sixth Brainstorming Week on Membrane Computing, 51-66(2008).
13. Cardona, M., Colomer, M. A., Margalida, A., et al: A computational modeling for real ecosystems based on P systems. Natural Computing, 10(1), 39-53(2011).
14. Cardona, M., Colomer, M.A., Margalida, A., et al: AP system based model of an ecosystem of some scavenger birds. In International Workshop on Membrane Computing, 182-195(2009).
15. Carlson, C.J., Burgio, K.R., Dougherty, E R , et al: Parasite biodiversity faces extinction and redistribution in a changing climate. Sci Adv, 3(9):e1602422(2017).
16. Castellini, A., Manca, V.: MetaPlab: A Computational Framework for Metabolic P Systems. In Membrane Computing: 9th International Workshop, 5391, 157-168 (2009).

17. Chen, Y., Zhang, G., Wang, T., et al: Automatic design of a P system for basic arithmetic operations. Chinese Journal of Electronics, 23(2): 302-304(2014).
18. Cheng,J., Zhang, G., Zeng, X.: A novel membrane algorithm based on differential evolution for numerical optimization. International Journal of Unconventional Computing, 7(3): 159-183(2011).
19. Christinal, H.A., et al: Thresholding 2d images with cell-like p systems. Romanian Journal of Information Science and Technology, 13 (2), 131-140(2010).
20. Colomer, M., Fondevilla, C., Valencia Cabrera, L.: A new P system to model the subalpine and alpine plant communities. Proceedings of the Ninth Brainstorming Week on Membrane Computing, 91-112(2011).
21. Colomer, M.A., Lavín, S., Marco, I., et al: Modeling population growth of Pyrenean chamois (Rupicapra p. pyrenaica) by using P-systems. In International Conference on Membrane Computing, 144-159(2010).
22. Colomer, M.A., Margalida, A., Sanuy, D., Pérez-Jiménez, M.J.: A bio-inspired computing model as a new tool for modeling ecosystems: the avian scavengers as a case study. Ecological modelling, 222(1), 33-47(2011).
23. Colomer, M., Margalida, A., Valencia, L., et al: Application of a computational model for complex fluvial ecosystems: The population dynamics of zebra mussel Dreissena polymorpha as a case study. Ecological Complexity, 20, 116-126(2014).
24. Colomer, M.A., Martínez-del-Amor, M.A., Pérez-Hurtado, I., et al: A uniform framework for modeling based on P systems. In 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), 616-621(2010).
25. Colomer, M.A., Pérez-Hurtado, I., Pérez-Jiménez, M.J., et al: Comparing simulation algorithms for multienvironment probabilistic P systems over a standard virtual ecosystem. Natural Computing, 11(3), 369-379(2012).
26. Colomer, M.À., Margalida, A., Pérez-Jiménez, M.J.: Population dynamics P system (PDP) models: a standardized protocol for describing and applying novel bio-inspired computing tools. PloS one, 8(4), (2013).
27. Conway, G.R.: Mathematical models in applied ecology. Nature, 269(5626), 291(1977).
28. Costantino, R.F., Desharnais, R.A., Cushing, J.M., et al: Chaotic dynamics in an insect population. Science, 275(5298), 389-391(1997).
29. Cushing, J.M., Costantino, R.F., Dennis, B.: Nonlinear population dynamics: models, experiments and data. Journal OF Theoretical Biology, 194: 1-9(1998).
30. DeYoung, B., Heath, M., Werner, F., et al: Challenges of modeling ocean basin ecosystems. Science, 304(5676), 1463-1466(2004).
31. Díaz-Pernil, D., Gutiérrez-Naranjo, Miguel A., Pérez-Jiménez, M.J., et al: A Linear-time Tissue P System Based Solution for the 3-coloring Problem. Electronic Notes in Theoretical Computer Science, 171(2): 81-93(2007)
32. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, et al: A linear time solution to the partition problem in a cellular tissue-like model. Journal of Computational & Theoretical Nanoscience, 7(5), 884-889(6)(2010).
33. Díaz-Pernil, D., Miguel, A. Gutiérrez-Naranjo, Peng, H.: Membrane computing and image processing: a short survey. Journal of Membrane Computing, 58-73(2019).
34. Doherty, T.S., Glen, A.S., Nimmo, D.G., et al: Invasive predators and global biodiversity loss. Proceedings of the National Academy of Sciences, 113(40), 11261-11265(2016).

35. Dong, W., Zhou, K., Qi, H., et al: A tissue P system based evolutionary algorithm for multi-objective VRPTW. Swarm and Evolutionary Computation, 39: 310-322(2018).
36. Early, R., Bradley, B.A., Dukes, J.S., et al: Global threats from invasive alien species in the twenty-first century and national response capacities. Nature Communications, 7, 12485(2016).
37. Fei S., Desprez J.M., Potter K.M., et al: Divergence of species responses to climate change. Science Advances, 3(5): e1603055(2017).
38. Fisher, R.A.: The wave of advance of advantageous genes. Annals of eugenics, 7(4), 355-369(1937).
39. Gallardo, B., Clavero, M., Sánchez, M.I., et al: Global ecological impacts of invasive species in aquatic ecosystems. Global change biology, 22(1), 151-163(2016).
40. Gamboa, F., Gassiat, E.: Bayesian methods and maximum entropy for ill-posed inverse problems. The Annals of Statistics, 25(1), 328-350(1997).
41. Grziwotz, F., Strauß, J.F., Hsieh, C.H., et al: Empirical Dynamic Modelling Identifies different Responses of Aedes Polynesiensis Subpopulations to Natural Environmental Variables. Scientific reports, 8(1), 16768(2018).
42. Guo, H., Yuan, X.: Leslie Matrix and Its Application-Prediction on the Development of the Population of the Giant Panda in Fuping. Journal of Southwest Nationalities College: Natural Science Edition, 22(2), 175-178(1996).
43. Guo, H., Yuan, X., Li, G.: Prediction on the development of the population of the Giant Panda in chengdu research base of giant panda breeding. Exploration of Nature, (2), 74-77(1997).
44. Guo, J.: Giant Panda Numbers Are Surging–or Are They?. Science, 316(5827), 974-975(2007).
45. Haddad, N. M., Brudvig, L. A., Clobert, J.: Habitat fragmentation and its lasting impact on Earth's ecosystems. Science advances, 1(2), (2015).
46. Hanski, I.: Habitat fragmentation and species richness. Journal of Biogeography, 42(5), 989-993(2015).
47. He, J., Xiao, J., Shao, Z.: An adaptive membrane algorithm for solving combinatorial optimization problems. Acta Mathematica Scientia, 34(5):1377-1394(2014).
48. Hirzel, A.H., Hausser, J., Chessel, D., et al: Ecological-niche factor analysis: how to compute habitat-suitability maps without absence data?. Ecology, 83(7), 2027-2036(2002).
49. Huang, X., Zhang, G., Rong, H., Ipate, F.: Evolutionary design of a simple membrane system. In International Conference on Membrane Computing, 203-214(2011).
50. Huang, Z., Zhang, G., Qi, D.: Application of Probabilistic Membrane Systems to Model Giant Panda Population Data. Computer Systems & Applications, 26(8): 252-256(2017).
51. Hulme, P.E.: Trade, transport and trouble: managing invasive species pathways in an era of globalization. Journal of applied ecology, 46(1), 10-18(2009).
52. Jiménez-Valverde, A., Peterson, A.T., Soberón, J., et al: Use of niche models in invasive species risk assessments. Biological invasions, 13(12), 2785-2797(2011).
53. Klausmeier, C.A.: Habitat destruction and extinction in competitive and mutualistic metacommunities. Ecology Letters, 4(1), 57-63(2001).
54. Lambers, J.: Extinction risks from climate change. Science, 348(6234): 501-502(2015).
55. Langkilde, T., Thawley, C.J., Robbins, T.R.: Behavioral adaptations to invasive species: benefits, costs, and mechanisms of change. Advances in the Study of Behavior, 49, 199-235(2017).

56. Ledesma, L., Manrique, D., Rodríguez-Patón, A.: A tissue P system and a DNA microfluidic device for solving the shortest common superstring problem. Soft Computing, 9(9), 679-685(2005).
57. Lee, C.E.: Evolutionary genetics of invasive species. Trends in ecology & evolution, 17(8), 386-391(2002).
58. Li, Z., Zhang, L., Su, Y., et al: A skin membrane-driven membrane algorithm for many-objective optimization. Neural Computing and Applications, 30(1), 141-152(2018).
59. Li, H., Liang, Y., Cao, D., Xu, Q.: Model-population analysis and its applications in chemical and biological modeling. TrAC Trends in Analytical Chemistry, 38, 154-162(2012).
60. Lindgren, E., Andersson, Y., Suk, J.E., et al: Monitoring EU Emerging Infectious Disease Risk Due to Climate Change. Science, 336(6080):418-419(2012).
61. Liu, C., Du, Y.: A membrane algorithm based on chemical reaction optimization for many-objective optimization problems. Knowledge-Based Systems, 165: 306-320(2019).
62. Lu, C., Zhang, X.: Solving vertex cover problem by means of tissue P systems with cell separation. International Journal of Computers Communications & Control, 5(4), 540-550(2010).
63. Manalastas, P.: Membrane Computing with Genetic Algorithm for the Travelling Salesman Problem. In Theory and Practice of Computation, 116-123(2013).
64. Manca, V., Marchetti, L.: Solving dynamical inverse problems by means of Metabolic P systems. BioSystems, 109(1), 78-86(2012).
65. Margalida, A., Colomer, M. A., Sanuy, D.: Can wild ungulate carcasses provide enough biomass to maintain avian scavenger populations? An empirical assessment using a bio-inspired computational model. PloS one, 6(5), (2011).
66. Margalida, A., Colomer, M. A.: Modelling the effects of sanitary policies on European vulture conservation. Scientific reports, 2, 753(2012).
67. Martin-Vide, C., Păun, Gh., Pazos, J., et al: Tissue P systems. Theoretical Computer Science, 296(2), 295-326(2003).
68. Moilanen, A., Hanski, I.: Habitat destruction and coexistence of competitors in a spatially realistic metapopulation model. Journal of Animal Ecology, 64(1), 141-144(1995).
69. Molnar, J.L., Gamboa, R.L., Revenga, C., et al: Assessing the global threat of invasive species to marine biodiversity. Frontiers in Ecology and the Environment, 6(9), 485-492(2008).
70. Nakagiri, N., Tainaka, K. I., Tao, T.: Indirect relation between species extinction and habitat destruction. Ecological Modelling, 137(2-3), 109-118(2001).
71. Nelder, J.A., Wedderburn, R.W.: Generalized linear models. Journal of the Royal Statistical Society: Series A (General), 135(3), 370-384(1972).
72. Nicole, J.: Threat of invasive species to bats: a review. Mammal review. (2017).
73. Niu, Y., Subramanian, K.G., Venkat, I., et al.: A tissue P system based solution to quadratic assignment problem. International Journal of Foundations of Computer Science, 23(07), 1511-1522(2012).
74. Niu, Y., Xiao, J., Jiang, Y.: Time-free solution to 3-coloring problem using tissue P systems. Chinese Journal of Electronics, 25(3), 407-412(2016).
75. Orellana-Martin, D., Valencia-Cabrera, L., Nez, A.R.N., et al: The unique satisfiability problem from a membrane computing perspective. Romanian journal of information science and technology, 21.3: 288-297(2018).

76. Orozco-Rosas, U., Montiel, O., Sepúlveda, R.: Mobile robot path planning using membrane evolutionary artificial potential field. Applied Soft Computing, 77: 236-251(2019).
77. Ou, Z., Zhang, G., Huang, X.: Automatic design of cell-like P systems through tuning membrane structures, initial objects and evolution rules. International Journal of Unconventional Computing, 9(5-6): 425-443(2013).
78. Pan, L., Păun, G.: Spiking neural P systems with anti-spikes. International Journal of Computers Communications & Control, 4(3), 273-282(2009).
79. Păun, A., Păun, Gh.: The power of communication: P systems with symport/antiport. New Generation Computing, 20(3), 295-305(2002).
80. Păun, A., Păun, Gh.: Small universal spiking neural P systems. BioSystems, 90(1), 48-60(2007).
81. Păun, Gh.: P systems with active membranes: attacking NP-complete problems. Journal of Automata, Languages and Combinatorics, 6(1), 75-90(2001).
82. Păun, Gh: Membrane computing: an introduction. Springer Science & Business Media, (2012).
83. Păun, Gh., Rozenberg, Gh., Salomaa, A.: Membrane computing with external output. Fundamenta Informaticae, 41(3), 313-340(1998).
84. Pavel, A.B., Buiu, C.: Using enzymatic numerical P systems for modeling mobile robot controllers. Natural Computing, 11(3), 387-393(2012).
85. Peng, H., Wang, J., Ming, J., et al.: Fault diagnosis of power systems using intuitionistic fuzzy spiking neural P systems. IEEE transactions on smart grid, 9(5), 4777-4784(2018).
86. Peng, H., Wang, J., Shi, P: A novel image thresholding method based on membrane computing and fuzzy entropy. Journal of Intelligent and Fuzzy Systems Applications in Engineering and Technology, 24(2), 229-237(2013).
87. Peng, H., Wang, J.: A hybrid approach based on tissue P systems and artificial bee colony for IIR system identification. Neural Computing and Applications, 28(9), 2675-2685 (2017).
88. Peng, X.W., Fan, X.P., Liu, J.X., et al: Spiking Neural P Systems for Performing Signed Integer Arithmetic Operations. Journal of Chinese Computer Systems, 34(2), 360-364(2013).
89. Pérez-Jiménez, M.J.: The P versus NP problem from the membrane computing view. European Review, 22(1), 18-33(2014).
90. Pérez-Hurtado, I., Valencia, L., Pérez-Jiménez, M.J., Colomer, M.A., Riscos-Núñez, A.: MeCoSim: A general purpose software tool for simulating biological phenomena by means of P Systems. In Proceedings 2010 IEEE Fifth International Conference on Bio-inpired Computing: Theories and Applications, pp. 637-643(2010).
91. Phillips, S.J., Anderson, R.P., Schapire, R.E.: Maximum entropy modeling of species geographic distributions. Ecological modelling, 190(3-4), 231-259(2006).
92. Pimm, S.L., Jenkins, C.N., Abell, R., et al: The biodiversity of species and their rates of extinction, distribution, and protection. Science, 344(6187):1246752-1246752(2014).
93. Proistosescu, C., and P.J. Huybers: Slow climate mode reconciles historical and model-based estimates of climate sensitivity. Science Advances 3-7(2017).
94. Prugh, L.R., Hodges, K.E., Sinclair, A.R.: Effect of habitat area and isolation on fragmented animal populations. Proceedings of the National Academy of Sciences, 105(52), 20770-20775(2008).

95. Reina-Molina, R., Díaz-Pernil, D., Real, P., Berciano, A.: Membrane parallelism for discrete Morse theory applied to digital images. Applicable Algebra in Engineering, Communication and Computing, 26(1-2), 49-71(2015).
96. Romero-Campero, F.J., Pérez-Jiménez, M.J.: A model of the quorum sensing system in Vibrio fischeri using P systems. Artificial life, 14(1), 95-109(2008).
97. Rybicki, J., Hanski, I.: Species–area relationships and extinctions caused by habitat loss and fragmentation. Ecology letters, 16, 27-38(2013).
98. Saether, B.E., Tufto, J., Engen, S., Jerstad, K., et al: Population Dynamical Consequences of Climate Change for a Small Temperate Songbird. Science, 287(5454):854-856(2000).
99. Sharaf H, Badr A, Farag I.: Using P system with Innate Immunity to solve NP Complete Problems. Computing & Information Systems, 14(2), 2010.
100. Song, T., Pang, S., Hao, S., et al: A parallel image skeletonizing method using spiking neural p systems with weights. Neural Processing Letters, 1-18(2018).
101. Song, T., Pan, L., Păun, G.: Asynchronous spiking neural P systems with local synchronization. Information Sciences, 219, 197-207(2013).
102. Song, T., Zheng, H., He, J.: Solving vertex cover problem by tissue P systems with cell division. Applied Mathematics & Information Sciences, 8(1), 333(2014).
103. Sosik, P.: A catalytic P system with two catalysts generating a non-semilinear set. Romanian J. Inf. Sci. Technology, 16(1): 3-9(2013).
104. Sosik, P., Rodríguez-Patón, A.: Membrane computing and complexity theory: A characterization of PSPACE. Journal of Computer and System Sciences, 73(1), 137-152(2007).
105. Soultan, A., Wikelski, M., Safi, K.: Risk of biodiversity collapse under climate change in the Afro-Arabian region. Scientific reports, 9(1), 955(2019).
106. Spatz, D.R., Zilliacus, K.M., Holmes, N.D., et al: Globally threatened vertebrates on islands with invasive species. Science advances, 3(10), (2017).
107. Strayer, D.L., Eviner, V.T., Jeschke, J.M., et al: Understanding the long-term effects of species invasions. Trends in ecology & evolution, 21(11), 645-651(2006).
108. Stevanovic, M., Popp, A., Lotze-Campen, H., et al: The impact of high-end climate change on agricultural welfare. Science Advances, 2(8): e1501452-e1501452(2016).
109. Stohlgren, T.J., Schnase, J.L.: Risk analysis for biological hazards: what we need to know about invasive species. Risk Analysis: An International Journal, 26(1), 163-173(2006).
110. Strohm, S., Tyson, R.C.: The effect of habitat fragmentation on cyclic population dynamics: a reduction to ordinary differential equations. Theoretical ecology, 5(4), 495-516(2012).
111. Swihart, R.K., Feng, Z., Slade, N.A., Mason, D.M., et al: Effects of habitat destruction and resource supplementation in a predator–prey metapopulation model. Journal of Theoretical Biology, 210(3), 287-303(2001).
112. Tanentzap, A.J., Walker, S., Theo Stephens, R.T., et al: A framework for predicting species extinction by linking population dynamics with habitat loss. Conservation Letters, 5(2), 149-156(2012).
113. Tian, H., Zhang, G., Rong, H., et al: Population model of giant panda ecosystem based on population dynamics P system. Journal of Computer Applications, 38(5): 1488-493(2018).
114. Tilman, D., May, R. M., Lehman, C. L., et al: Habitat destruction and the extinction debt. Nature, 371(6492), 65(1994).
115. Urban, Mark C.: Accelerating extinction risk from climate change. Science, 348.6234: 571-573(2015).

116. Vanbergen, A.J., Espíndola, A., Aizen, M. A.: Risks to pollinators and pollination from invasive alien species. Nature ecology & evolution, 2(1), 16(2018).
117. Vander Zanden, M.J., Olden, J.D., Thorne, J.H., Mandrak, N.E.: Predicting occurrences and impacts of smallmouth bass introductions in north temperate lakes. Ecological Applications, 14(1), 132-148(2004).
118. Valencia-Cabrera, L., Garcia-Quismondo, M., Pérez-Jiménez, et al: Modeling Logic Gene Networks by Means of Probabilistic Dynamic P Systems. IJUC, 9(5-6), 445-464(2013).
119. Valencia Cabrera, L., García Quismondo, M., Pérez-Jiménez, M., et al. Analysing gene networks with pdp systems. Arabidopsis thailiana, a case study. In Proceedings of the Eleventh Brainstorming Week on Membrane Computing, 257-272(2013).
120. Wahlberg, N., Moilanen, A., Hanski, I.: Predicting the occurrence of endangered species in fragmented landscapes. Science, 273(5281), 1536-1538(1996).
121. Wang, B., Chen, L., Cheng, J.: New result on maximum entropy threshold image segmentation based on P system. Optik, 163: 81-85(2018).
122. Wang, X., Zhang, G., Neri, F., et al: Design and implementation of membrane controllers for trajectory tracking of nonholonomic wheeled mobile robots. Integrated Computer-Aided Engineering, 23(1): 15-30(2016).
123. Wang, X., Zhang, G., Zhao, J., et al: A modified membrane-inspired algorithm based on particle swarm optimization for mobile robot path planning, International Journal of Computers, Communications & Control, 10(5): 732-745(2015).
124. Willis K.J., Bhagwat S.A.: Biodiversity and climate change. Science, 326(5954): 806-807(2009).
125. Yahya, R., Hasan, S., George, L., Alsalibi, B.: Membrane computing for 2D image segmentation. Int. J. Advance Soft Compu. Appl, 7(1), 35-50(2015).
126. Yee, T.W., & Mitchell, N.D.: Generalized additive models in plant ecology. Journal of vegetation science, 2(5), 587-602(1991).
127. Zhang, G., Cheng, J., Wang, T., Zhu, J.: Membrane Computing: Theory and Applications. Beijing: China Science Publishing. 2015.
128. Zhang, G., Gheorghe, M., Li, Y.: A membrane algorithm with quantum-inspired subalgorithms and its application to image processing, Natural Computing—An International Journal, 11(4): 701-717(2012).
129. Zhang, G., Pan, L.: A survey of membrane computing as a new branch of natural computing. Chinese journal of computers, 33(2): 208-214(2010).
130. Zhang, G., Rong, H., Neri, F., Pérez-Jiménez, M. J.: An optimization spiking neural P system for approximately solving combinatorial optimization problems. International Journal of Neural Systems, 24(05), 1440006(2014).
131. Zhang, G., Rong, H., Ou, Z., Pérez-Jiménez, M.J., et al: Automatic Design of Deterministic and Non-Halting Membrane Systems by Tuning Syntactical Ingredients, IEEE Transactions on NanoBioscience, 13(3): 363-371(2014).
132. Zhang, G., Zhou, F., Huang, X., et al: A novel membrane algorithm based on particle swarm optimization for solving broadcasting problems. Journal of Universal Computer Science, 18(13): 1821-1841(2012).
133. Zhang, X., Li, J., Zhang, L.: A multi-objective membrane algorithm guided by the skin membrane. Natural Computing, 15(4):597-610(2016).
134. Zhang, X., Zeng, X., Pan, L.: A spiking neural p system for performing multiplication of two arbitrary natural numbers. Chinese journal of computers. 32(12): 2362-2372(2009).

135. Zhao, J., Butters, T.D., Zhang, H., et al: Image-Based Model of Atrial Anatomy and Electrical Activation: A Computational Platform for Investigating Atrial Arrhythmia. IEEE Transactions on Medical Imaging, 32(1): 18-27(2013).
136. Zou, A., Hou, Z., Tan, M., et al: A behavior controller based on spiking neural networks for mobile robots. Neurocomputing, 71(4/6):655-666(2018).
137. http://www.p-lingua.org/mecosim/
138. Mackie, G., Schloesser, D. Comparative biology of zebra mussels in Europe and North America: an overview. American zoologist, 36(3), 244-258(1996).
139. Ardura, A., Zaiko, A., Borrell, Y. Novel tools for early detection of a global aquatic invasive, the zebra mussel Dreissena polymorpha. Aquatic Conservation: Marine and Freshwater Ecosystems, 27(1), 165-176(2017)
140. Ackerman, J.D., Sim, B., Nichols, S.J. A review of the early life history of zebra mussels (Dreissena polymorpha): comparisons with marine bivalves. Canadian Journal of Zoology, 72(7), 1169-1179 (1994).
141. Mingyang, L., Yunwei, J., Kumar, S. Modeling potential habitats for alien species Dreissena polymorpha in Continental USA. Acta Ecologica Sinica, 28(9), 4253-4258(2008).
142. Hallstan, S., Grandin, U., Goedkoop, W. Current and modeled potential distribution of the zebra mussel (Dreissena polymorpha) in Sweden. Biological Invasions, 12(1), 285-296(2010).
143. Chen, Q., Mynett, A. Applications of Soft Computing to Environmental Hydroinformatics with Emphasis on Ecohydraulics Modelling. In Practical Hydroinformatics, (405-420)(2009).
144. Drake, J. M., Bossenbroek, J. M. Profiling ecosystem vulnerability to invasion by zebra mussels with support vector machines. Theoretical Ecology, 2(4), 189-198(2009).
145. Masini, F., Lovari, S. Systematics, phylogenetic relationships, and dispersal of the chamois (Rupicapra spp.). Quaternary Research, 30(3), 339-349(1988).
146. MetaPlab website, http://mplab.sci.univr.it

A Grid-Density Based Algorithm by Weighted Spiking Neural P Systems with Antispikes and Astrocytes in Spatial Cluster Analysis

Deting Kong, Yuan Wang, Di Wang, Xiyu Liu, and Jie Xue*

Business School, Shandong Normal University
East road of Wenhua, No.88, Jinan, Shandong, China
Jiexue@sdnu.edu.cn

Abstract. In this paper, we propose a novel clustering approach based on P systems and grid- density strategy. We present grid-density based approach for clustering high dimensional data, which first projects the data patterns on a two-dimensional space to overcome the curse of dimensionality problem. Then, through meshing the plane with grid lines and deleting sparse grids, clusters are found out. In particular, we present weighted spiking neural P systems with antispikes and astrocyte (WSNPA2 in short) to implement grid- density based approach in parallel. Each neuron in weighted SN P system contains a potential, which can be expressed by a computable real number. Spikes and anti-spikes are inspired by neurons communicating through excitatory and inhibitory impulses. Astrocytes have excitatory and inhibitory influence on synapses. Experimental results on multiple real-world datasets demonstrate the effectiveness and efficiency of our approach.

Keywords: Spiking Neural P Systems, Grid-density based Clustering Approach, Multidimensional Datasets

1 Introduction

Spiking neural P systems (SN P in short) are a kind of parallel and distributed neural-like computation model in the field of membrane computing [1,2]. SN P systems can generate and accept the sets of Turing computable natural numbers [9], generate the recursively enumerable languages [12] and compute the sets of Turing computable functions [3].

Inspired by different biological phenomena and mathematical motivations, several families of SN P systems have been constructed, such as SN P systems with anti-spikes [17], SN P systems with weight [14], SN P systems with astrocyte [9], homogenous SN P systems [23], SN P systems with threshold [24], fuzzy SN P systems [15], sequential SN P systems [18], SN P systems with rules on synapses [20], SN P systems with structural plasticity [8]. For applications, SN P systems are used to design logic gates, logic circuits [14] and operating systems

[4], perform basic arithmetic operations [25], solve combinatorial optimization problems [11], diagnose fault of electric power systems [21]. Păun who initiated the P systems pointed out that solving real problem by membrane computing need to be addressed. The comparative analysis of dynamic behaviors of a hybrid algorithm indicates that the combination of evolutionary computation with membrane systems can produce a better algorithm for balancing exploration and exploitation [22,10,16]. However, the hybrid algorithm does not use objects and rules defined by P systems. On account of P system is still in the phase of solving addition, subtraction, multiplication, and division [13]. How can these algorithms deal with complex functions? Different from researches above, the whole process of clustering algorithm proposed in this paper is implemented through changes of objects by rules in membranes. In which, objects encode data. Membrane rules working on objects achieve the clustering goal.

Grid-based clustering is usually used for the more complex and high-dimension data. Data space is partitioned into certain number of cells. Cells are basic units for clustering operations [19]. OPTIGRID [26] is designed to obtain an optimal grid partitioning. CLIQUE is probably the most intuitive and comprehensive clustering technique [27]. The shifting grid approach (SHIFT) has been reported to be somehow similar to the sliding window technique. AGRID combines density and grid-based approaches to cluster large high-dimensional data [56]. Smart Grids integrate several disciplines, which migrate from the traditional centralized power delivery infrastructures to the distributed nature [28,7].

Based on the above considerations, this paper develops a hybrid optimization method, grid-density based algorithm by weighted SN P systems with anti-spike and astrocyte. Characteristic of each dimension is calculated and compared by rules independently in different membranes synchronously. Communications among membranes is utilized to explore clusters. Experimental results on multiple realworld datasets demonstrate the effectiveness and efficiency of our approach.

2 Weighted Spiking Neural P Systems with Antispikes and Astrocytes

Weighted spiking neural P systems with antispikes and astrocytes (called WSNPA2) of degree $m \geq 1$ is a construct of the form

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, ast_1, \ldots, ast_k, In, Out)$$

where, O is the set of spikes, $O = \{a, \bar{a}\}$, a is spike, $\bar{a}$ is antispike. The empty string is denoted by λ; $\sigma_1, \sigma_2, \ldots, \sigma_m$ are neurons, m is the degree of neurons, of the form $\sigma_i = (n_i, R_i)$, $1 \leq i \leq m$, Where, n_i is the initial number of spikes contained in σ_i, R_i is a finite set of rules with: $(1)E/s^c \rightarrow s$, s is spikes or antispikes, c is the number of spikes in the rule, $c \geq 1$, E is a regular expression over a or a; $(2)s^e \rightarrow \lambda$, e is the number of spikes, $e \geq 1$. $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\} \times \omega$ are synapses between neurons, ω is the weight on synapse (i, j), $\omega \in Z$. For each (i, j), there is at most one synapse (i, j, ω). A rule $E/s^c \rightarrow s$

is applied as follows. If neuron σ_i contains r spikes/antispikes, $r \geq c$, then the rule can fire, c numbers of spikes/antispikes are consumed, $r - c$ numbers of spikes/antispikes remain in σ_i and one spikes/antispikes is released. The number of spikes/antispikes is multiplied by ω and pass immediately to all neurons with $(i, j, \omega) \in syn$. $s^e \rightarrow \lambda$ is forgetting rules. e numbers of spikes/antispikes are omitted from the neuron immediately.

For spikes a^q and antispikes a^p ($p, q \in Z$ are numbers of spikes and antispikes), an annihilation rule $a\bar{a} \rightarrow \lambda$ is applied in a maximal manner. a^{q-p} or $a(a)^{p-q}$ remain for the next step, provided that $q \geq p$ or $p \geq q$, respectively. $ast_1, \ldots, ast_k$ are astrocytes, of the form $ast_i = (syn_{asti}, t_i)$, where $syn_{asti} \subseteq syn$ is the subset of synapses controlled by the astrocyte, t_i is the threshold of the astrocyte. Suppose that there are k spikes passing along the neighboring synapses syn_{asti}. If $k \geq t_i$, then ast_i has an inhibitory influence on syn_{asti}, and the k spikes are transformed into one spike by $a^k \rightarrow a$. a will be sent into the neuron connected to ast_i. Otherwise, $k < t_i$, then ast_i has an excitatory influence on syn_{asti} , all spikes survive and reach their destination neurons.

$In, Out \in \{1, 2, \ldots, m\}$ indicate the input and output neurons, respectively.

3 Grid-Density Based Clustering Algorithm for Multi-di mensional Dataset

3.1 Identify the two well-informed features

Generally, in grid-based methods, the computations will grow exponentially with high dimensions, because of the evaluations should be done over all grid points. For example, a cluster analysis with N dimensions and L grid partitions in each dimension, would result in L^N grids. To avoid this curse of dimensionality problem, we try to project data in actual feature space into a 2D space, aim to discover the initial locations of potential clusters in a plane. The plane comprised by the two well-informed features $n_i, n_j \in N$ will be covered by a $L \times L$ lattice of grids with M data objects $X_p (p = 1, \ldots, M)$.

At first, each dimension of objects is partitioned into $K = [\sqrt{M}]$ bins, $B = \{b_1, b_2, \ldots, b_k\}$ is as

$$c_i(b_k) = \mid \{X_p, p = 1, \ldots, M, \mid X_{pi} - b_k \mid < \mid X_{pi} - b_{k'} \mid ; b_k, b_{k'} \in B, b_{k'} \neq b_k\} \mid \quad (1)$$

x_{pi} is the value of feature n_i in data pattern X_p and $\mid . \mid$ is the cardinality operator representing the number of elements in a set. The number of peaks in K bins are considered as the centroid of data collections and are accounted as the effectiveness measure, ε, of feature n_i. Figure 1 depicts this histogram for the 13 features of the known Wine data set where their ε's are 3,2,1,2,1,2,0,4,3,3,3,4,5, respectively. According to these ε's, the features n_8, n_{12} and n_{13} are selected. Two of the three should be selected as the two well-informed features for Wine data set. These features will then be used to do the cluster analysis.

$$\varepsilon(n_i) = \mid \{c_i(b_k) : c_i(b_k) > c_i(b_{k-1}) \; and \; c_i(b_k) > c_i(b_{k+1})\} \mid \quad (2)$$

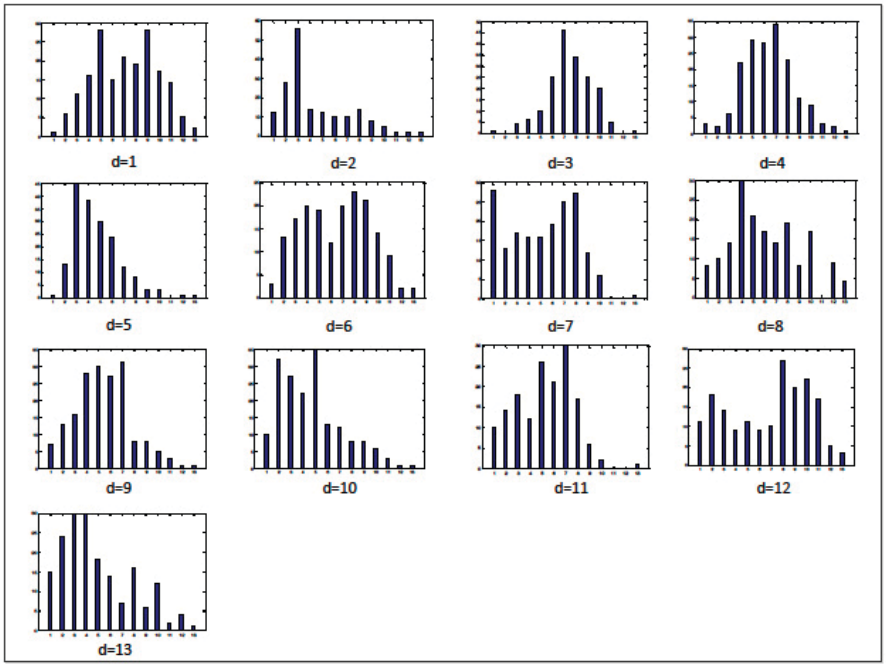


Fig. 1. histogram for the 13 features of Wine data set

3.2 Clustering by Grid-Density Based Algorithm

The plane comprised by the two well-informed features will be covered by a $H = L \times L$ lattice of grids. Grids are denoted by $G = \{g_1, g_2, \ldots, g_H\}$. $C(g_h), h \in \{1, 2, \ldots, H\}$ is the number of data X_p partitioned in grid g_h according to (3).

$$C(g_h) =\mid \{X_p : p = 1, \ldots, M, X_p \in \ g_h, g_h \in G\} \mid \tag{3}$$

Next, non-dense grids are deleted. A grid is dense if $C(g_h) > \theta, \theta \in N^+$ is a threshold defined before computation. After getting the initial members of grid graph G, G is refined by finding out dense grid. Those sparse girds are discarded. The refined grid graph is defined as:

$$G_r = \{g_h \mid C(g_h) > \theta\} \subseteq G \tag{4}$$

Each grid $g_h \in G_r$ has 4 neighbors connected with it as shown in Figure 2. A cluster is a set of neighbors of dense grids. The process of clustering algorithm is shown in Table below.

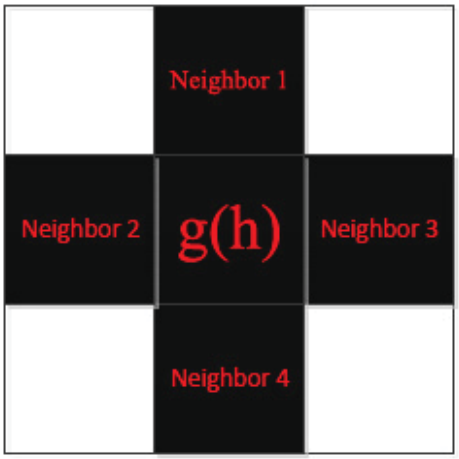


Fig. 2. Neighbors of grid $g_h \in G_r$

Algorithm: Grid-Density based clustering algorithm
Inputs: $\Omega = \{X_{pi}, 1 \le p \le M, 1 \le i \le N\}, H = L \times L, \theta : density threshold$
Outputs: $CS = \{CS_1, CS_2, \ldots, CS_t\}$
Begin
for all features $n_i, i = 1, 2, \ldots, N$
use $K = [\sqrt{M}]$ bins to partition the feature n_i
obtain the number of data in each bin $B = \{b_1, \ldots, b_k\}$ by (1)
compute the effectiveness measure $\varepsilon(n_i)$ for n_i by (2)
rank $\varepsilon(n_i)$
get the two top-ranked features
project data patterns into $H = L \times L$ grids
obtain the capability $C(g_H)$ of each grid by (3)
select dense grid by (4)
form cluster set by combing neighbor dense grids
return the t clusters, $CS = \{CS_1, CS_2, \ldots, CS_t\}$
End

---

4 Multi-WSNPA2 Design for Grid-Density Based Clustering

4.1 Grid-Density Based Clustering by Multi-WSNPA2

In this section, the weighted spiking neural P system with antispkies and astrocytes is designed for grid-density based clustering. Objects in each neuron

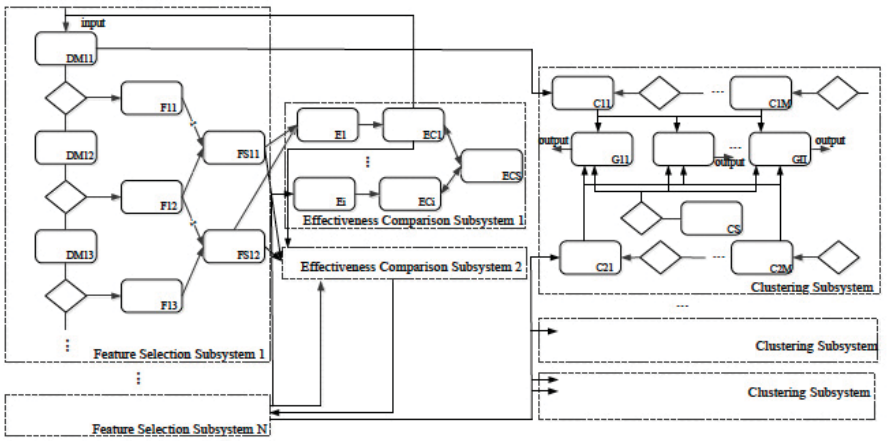


Fig. 3. Structure of WSNPA2 for grid-density based clustering algorithm

are organized as spikes and antispikes with real-valued numbers corresponding to $\Omega = \{X_{pi}, 1 \leq p \leq M, 1 \leq i \leq N\}$. Feature selection and cluster analysis are implemented by rules of WSNPA2. WSNPA2 is divided into three subsystems: feature selection, effectiveness comparison and clustering. The structure of WSNPA2 is shown in Fig.3, where ovals represent neurons, rhombic stand for astrocytes and arrows indicate channels. WSNPA2 for grid-density based clustering algorithm is described as the following construct

$$\Pi = (O, \sigma_{S1}, \sigma_{S2}, \sigma_{S3}, syn, R, ast_{S1}, ast_{S3}, \sigma_{in1}, \ldots, \sigma_{inN}, \sigma_{out1}, \ldots, \sigma_{out_t})$$

where, $O = \{a, \bar{a}\}$. At beginning, the input neuron contains x_{pi} numbers of spike a; σ_{S1} stands for neurons in feature selection subsystems,$\sigma_{S1} = \{DMiz, Fiz', FSiz''\}, 1 \leq i \leq N, 1 \leq z = z' \leq [\sqrt{M}], 1 \leq z'' \leq 2[\sqrt{M}]/3$; σ_{S2} represents neurons in effectiveness comparison subsystems, $\sigma_{s_2} = \bigcup_{1 \leq i \leq N}(E_i \cup EC_i) \cup \{ECS\}$; σ_{S3} describes neurons in clustering subsystems, $\sigma_{S3} = \{\{C_{i'j}, i^{'} \in \{1, 2\}, 1 \leq j \leq N\}, \{G_{gg'}, 1 \leq g, g^{'} \leq L\}, CS\}$; The number of astrocytes ast_{S1} in the feature selection system is $N * N$ between each two DM_{iz}. The number of astrocytes ast_{S3} in the clustering system is $L \times L \times 2 + 1$; Input neurons $\sigma_{in1}, \ldots, \sigma_{inN}$ are in the feature selection system. Output neurons $\sigma_{out1}, \ldots, \sigma_{out_t}$ are in the clustering system.

There are several different clustering subsystem work in parallel for different grid number $H = L * L$ and density threshold θ, which means the whole system can output variant clustering results simultaneously.

syn represents synapse among neurons:

$syn(DM_{iz}, ast_{S1}), 1 \leq i \leq N, 1 \leq z \leq [\sqrt{M}]$

$syn(F_{iz'}, ast_{S1}), 1 \leq i \leq N, 1 \leq z^{'} \leq [\sqrt{M}]$

$syn(F_{iz'}, F\ S_{iz''}), 1 \le i \le D, 1 \le z^{'} \le 1 \le z'' \le 2[\sqrt{M}]/3$

$syn(F\ S_{iz''}, E_i), 1 \le i \le N, 1 \le z'' \le 2[\sqrt{M}]/3$

$syn(E_i, EC_i), 1 \le i \le N$

$syn(EC_i, ECS), 1 \le i \le N$

$syn(DM_{i1}, C_{i1}), 1 \le i \le N$

$syn(C_{i'j}, ast_{S3}), i^{'} \in \{1, 2\}, 1 \le j \le N$

$syn(C_{i'j}, G_{gg'}), i^{'} \in \{1, 2\}, 1 \le j \le N, 1 \le g, g^{'} \le L$

$syn(G_{gg'}, ast_{S3}), i^{'} \in \{1, 2\}, 1 \le g, g^{'} \le L$

$syn(ast_{S3}, CS)$

R is the following set of firing and forgetting rules:([ ]x means the rule works in neuron x, otherwise, the rule executes through all neurons)

$[a^{x_{pi}} \rightarrow a^{x_{pi}}, x_{pi} < t_{ih}]_{DM_{iz}}, a^{x_{pi}} \rightarrow a, x_{pi} > t_{ih}$

$[a^{f} \rightarrow a^{f}]_{F_{iz'}}, [a^{f_2 - f_1} \rightarrow a]_{FS_{iz''}}, f_2 - f_1 > 0$

$[a^{2m} \rightarrow a^{2m+2}]_{E_i}, [a^{2m+2}/a^{m} \rightarrow a^{m}]_{E_i}, [a^{m} \rightarrow a^{m}]_{EC_i}$

$[a \rightarrow a]_{E_i}, [a \rightarrow a]_{EC_i}, [a^{m} \rightarrow \bar{a}^2]_{ECS}, [\bar{a}^2\ a^{m} \rightarrow \lambda]_{E_i^{'}}$

$[a^{m} \rightarrow \bar{a}^2]_{ECS}, [\bar{a}^2\ a^{m} \rightarrow \lambda]_{E_i^{'}}, [\bar{a}^2\ a^{x_{ij}} \rightarrow a^{x_{ij}}]_{DM_{iz}}$

$[a^{x_{i'j}} \rightarrow a^{x_{i'j}}]_{CS_{i'j}}, a^{x_{i'j}} \rightarrow a, x_{i'j} > \theta_{i'}$

$[a \rightarrow a]_{G_{gg'}}, [a^4/a^3 \rightarrow \bar{a}^2]_{G_{gg'}}, [a^{n} \rightarrow \lambda\ ;\ n < \theta]_{G_{gg'}}$

$[a^{n} \rightarrow a\ ;\ n \ge \theta]_{G_{gg'}}, [a \rightarrow a]_{G_{gg'}}$

4.2 Overview of Computations

Data set of M observations are codified by spikes $a^{x_{pi}}$, $1 \le i \le N$, $1 \le p \le M$. The computation of the P system is split in three subsystems. When $a^{x_{pi}}$ arrive in neuron DM_{i1}, the computation begins in parallel.

In feature selection subsystem, threshold t_{ih} in each astrocytes ast_{S1ih} is $t_{ih} = [h * (X_{i\ max} - X_{i\ min})/[\sqrt{M}]]$, $1 \le i \le N$, $1 \le h \le [\sqrt{M}]$. If $x_{pi} > t_{ih}$, it is said that x_{pi} belongs to the current neuron DM_{iz}. Rule 2 add a spike in DM_{iz}. Otherwise, $a^{x_{pi}}$ pass through DM_{iz} to find its neuron (bin) by rule 1. After all $a^{x_{pi}}$ execute with rule 1 and rule 2, the peak of each dimension is chosen by rule 3 and rule 4.

All peaks of dimension i gain by spike a in neuron E_i. Then, effectiveness comparison subsystem starts. The maximum number of peaks of each dimension is selected by rule 5-9. Rule 5 and 6 copy peaks a^m into a^{2m+2} and sends a^m into neuron EC_i for preparation. Then, different number of a^m is descended one by one by rule 8. Rule 9 helps ECS collect all dimensions without the one with

maximum number of peaks. The serial number of the neuron who sends out $\bar{a}^2$ by rule 10 is chosen as the first dimension for clustering. Other effectiveness comparison subsystem will work in the same way except that the chosen dimension is deleted by rule 11.

Rule 12 activates the input neurons of the two selected features. The clustering subsystem begins. Rule 13-14 put observations into suitable bins in their own dimensions. $(\theta_{i'} = [i^1 X_{i\ max} - X_{i\ min})/L])$. Then, rule 15 select the grid who has two spikes. It is chosen as initial grid for cluster . Rule 16 activates the input neurons of the two selected features again. Rule 17-19 finds dense grids. Rule 12-16 will continue to work until there are no spike input. The clustering result is obtained by the serial number of neurons with a output by rule 19.

5 Experiments and Analysis

The experiments set out to investigate the performance of the proposed approach compared to classical clustering algorithms. We conduct experiments using ten real-world datasets. Table below summarizes these data sets, ordered in their number of attributes.

Table 1. Ten real-world datasets of UCI.

Data set	Number of attributes	Number of classes	Number of objects
Haberman	3	2	306
Iris	4	3	150
Thyroid	5	4	215
Ecoli	7	8	336
Diabetes	8	3	768
Breast	9	3	684
Glass	9	6	214
Wine	13	3	178
Vehicle	18	4	846
Ionosphere	33	2	351

The amount of necessary resources to define multi-WSNPA2 of grid-density based clustering for the ten datasets are shown in Table.2.

To compare the algorithm with k-means and AHC (agglomerative hierarchical clustering) in more precise notion, their clustering performance in terms of accuracy is depicted in Table. 3. This AHC uses the ward linkage[27] which is appropriate for Euclidean distance. The accuracy of clusters evaluates the right objects of clusters in each class.

Table 2. The amount of necessary resources to define multi-WSNPA2 of the ten datasets.

Data set	parallel steps	Initial cells	Initial objects	Number of rules
Haberman	314	52	36517	929
Iris	155	49	2.0782e+03	617
Thyroid	223	73	2.7634e+04	1097
Ecoli	344	128	1.1750e+03	2389
Diabetes	778	222	2.7639e+05	6170
Breast	694	235	19331	6183
Glass	222	132	2.1698e+04	1958
Wine	190	173	1.5998e+05	2340
Vehicle	866	524	1581507	15268
Ionosphere	359	618	2.6438e+03	11628

Table 3. The accuracy of clusters evaluates the right objects of clusters in each class.

Data set	the algorithm	K-means	AHC
Haberman	47.82	48.64	50.06
Iris	93.33	89.79	91.54
Breast	93.99	96.06	95.83
Wine	97.75	95.20	97.73
Ionosphere	72.93	70.20	70.54
Average	**81.16**	79.97	81.14

Clearly, the performance is comparable to k-means and AHC and even better as its averages (in bold-face) show.

Table 4. Comparison of time consuming among the three algorithms.

Data set	the algorithm	K-means	AHC
Haberman	0.07	0.08	0.07
Iris	0.03	0.05	0.08
Thyroid	0.05	0.04	0.06
Ecoli	0.07	0.04	0.09
Breast	0.15	0.05	2.48
Glass	0.05	0.07	0.07
Wine	0.04	0.07	0.05
Vehicle	0.19	0.14	0.46
Ionosphere	0.08	0.06	0.14
Average	**0.08**	0.07	0.38

The intrinsic maximal parallelism of P systems can be exploited to produce a speed-up for solutions. In order to achieve this, the model needs several ingredients, among them the ability to generate an exponential workspace in polynomial time. The computational cost is more than k-means as the last stage of its algorithm is repetitive. Table .4 compares the time consuming against k-means and AHC where the fastest (in average) is shown in boldface.

6 Conclusion

This paper discusses the use of weighted spiking neural P system with antispike and astrocyte to appropriately develop a novel hybrid method with grid-density based algorithm for solving clustering problems which first projects the data patterns on a two-dimensional space to overcome the curse of dimensionality problem. To choose these two well-informed features, a simple and fast feature selection algorithm is proposed. Then, through meshing the plane with grid lines and deleting sparse grids, clusters are found out. In particular, we present weighted spiking neural P systems with antispikes and astrocyte (WSNPA2 in short) to implement grid-density based approach in parallel. Each neuron in weighted SN P system contains a potential, which can be expressed by a computable real number. Spikes and anti-spikes are inspired by neurons communicating through excitatory and inhibitory impulses. Astrocytes have excitatory and inhibitory influence on synapses. Characteristic of each dimension is calculated and compared by rules independently in different membranes synchronously. Communications among membranes is utilized to explore clusters. Experimental results on multiple real-world datasets demonstrate the effectiveness and efficiency of our approach to classical k-means and AHC algorithms.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (No. 61802234, 61876101), Natural Science Foundation of Shandong Province (No. ZR2019QF007), and China Postdoctoral Project (No. 2017M612339)

References

1. Leporati A., Zandron C., Ferretti C., Mauri G.: Solving numerical NP-complete problems with spiking neural P systems. Lecture Notes in Computer Science 4860, 336–352 (2007)
2. Leporati A., Mauri G., Zandron C., G. Paun, M. Prez-Jimnez: Uniform solutions to SAT and subset sum by spiking neural P systems. Natural Computing 8(4), 681–702 (2009)
3. Paun A.,Paun G.: Small universal spiking neural P systems. Biosystems 90, 48–60 (2007)
4. Adl A., Badr A., Farag I.: Towards a Spiking Neural P Systems OS. CoRR 1012.0326 (2010)

5. Hinneburg A., Keim D.: Optimal grid-clustering: towards breading the curse of dimensionality in high-dimensional clustering. In: Proc. of the 25th Inter-national Conference on Very Large Data Bases (1999)
6. Monti A., Ponci F.: Power grids of the future: Why smart means complex. IEEE Computer Society 7–11 (2010)
7. Vicente D., Vellido A.: A review of hierarchical models for data clustering and visualization. In: Giraldez R, Riquelme JC, Aguilar-Ruiz JS (eds) Tendencias de la Minera de Datos en Espana (2004)
8. Cabarle F., Adorna H.,Perez-Jimenez M., Song T.: Spiking neural P systems with structural plasticity. Neural Computing Applications 26, 1905–1917 (2015)
9. Paun G.: Spiking neural P systems with astrocyte-like control. Journal of Universal Computer 13, 1707–1721(2007)
10. Zhang G., Marian G.,Wu C.: A quantum-inspired evolutionary algorithm based on P systems for Knapsack problem. Fundamenta Informaticae 87, 93–116 (2008)
11. Zhang G., Rong H., Neri F., Perez-Jimenez M.: An optimization spiking neural P system for approximately solving combinatorial optimization problems. Internarional Journal of Neural Systems 24(5), 1440006 (2014)
12. Chen H., Freund R., Ionescu M., Paun G., Prez-Jimnez M.: On string languages generated by spiking neural P systems. Fundamenta Infirmaticae 75, 141–162 (2007)
13. Cheng J., Zhang G., Zeng X.: A novel membrane algorithm based on differential evolution for numerical optimization. International Journal of Unconventional Computing 7, 159–183(2011)
14. Wang J., Hoogeboom H., Pan L., Paun G., Perez-Jimenez M.: Spiking neural P sys- tems with weights. Neural Computation 22, 2615–2646 (2010)
15. Wang J., Shi P., Peng H., Perez-Jimenez M., Wang T.: Weighted fuzzy spiking neural P systems. IEEE Transactions on Fuzzy Systems 21, 209–220 (2013)
16. Huang L., Suh I.H., Abraham A.: Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants. Information Sciences 181, 2370–2391 (2011)
17. Pan L., Paun G.: Spiking neural P systems with anti-spikes. International Journal of Computers Communication and Control (2011)
18. Ibarra O., Paun A., Rodriguez-Paton A.: Sequential SN P systems based on min/max spike number. Theory Computer Science 410, 2982–2991 (2009)
19. Agrawal R., Gehrke J., Gunopulos D., Raghavan P.: Automatic subspace clustering of high dimensional data for data mining applications. In: Proceedings of the 1998 ACM SIGMOD international conference on Management of data, pp. 94–105 (1998)
20. Song T., Zou Q., Li X., Zeng X.: Asynchronous spiking neural P systems with rules on synapses. Neurocomputing 151, 1439-1445 (2015)
21. Wang T., et al.: Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems. IEEE Transactions on Power System 30, 1182-1194 (2015)
22. Nishida T.: Membrane algorithm with Brownian sub algorithm and genetic sub algorithm. International Journal of Foundations of Computer Science 18, 1353-1360 (2007)
23. Zeng X., Zhang X., Pan L.: Homogeneous spiking neural P systems. Fundamenta Infirmaticae 97, 275-294 (2009)
24. Zeng X., Zhang X., Song T., Pan L.: Spiking neural P systems with thresholds. Neural Computation 26, 1340-1361 (2014)

25. Liu X., Li Z., Liu J., Liu L., Zeng X.: Implementation of arithmetic operations with time- free spiking neural P systems. IEEE Transactions on Nanobioscience 14, 617-624 (2015)
26. Zeng X., Song T., Zhang X., Pan L.: Performing Four Basic Arithmetic Operations With Spiking Neural P Systems. IEEE Transactions on Nanoscience 11(4), 366-374 (2012)
27. Zhao Y., Song J.: AGRID: an efficient algorithm for clustering large high dimensional data sets. In: Proc. of The 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD03), pp. 271-282 (2003)
28. Yan Y., Qian Y., Sharif H., Tipper D.: A survey on smart grid communication infrastruc- tures: Motivations, requirements and challenges. IEEE Communications Surveys and Tutorials 15(1), 28-42 (2013)

Generating Hilbert Words in Array Representation with P Systems

Rodica Ceterchi[1], Luping Zhang[2], Linqiang Pan[2,5], K. G. Subramanian[2,3], and Gexiang Zhang[4]

[1] University of Bucharest
Faculty of Mathematics and Computer Science
14 Academiei St, 010014 Bucharest, Romania
[2] School of Artificial Intelligence and Automation,
Huazhong University of Science and Technology,
Wuhan, 430074, China
[3] Faculty of Science, Liverpool Hope University,
Hope Park, Liverpool L16 9JD, UK
Honorary Visiting Professor
[4] Nature-Inspired Computation and Smart Grid Lab
School of Electrical Engineering, Southwest Jiaotong University
Chengdu, 610031, China

In Memoriam Professor Rani Siromoney

Abstract. Construction of finite grammars to generate languages of digitized picture patterns, considered as arrays of symbols, has been a problem of interest with several studies having been done in the area of two-dimensional formal languages. On the other hand, with the introduction of the novel computing model of P system in the area of membrane computing, P systems were developed for handling the problem of picture array generation, with the rewriting involved being sequential as in Chomsky grammars or parallel as in Lindenmayer systems. We introduce the array representation for the finite approximations of the Hilbert space-filling curve, and we generate them with P systems with parallel array rewriting.

1 Introduction

In two-dimensional language theory [13], a variety of array grammars based on different aspects such as the type of rules, the rewriting mode and others, have been proposed and their generative powers have been investigated. In the recent past, the novel computing model of P system in the area of membrane computing [19] was considered as a convenient framework (see for example, [2, 24, 25, 27]) to handle the problem of picture array generation in view of the ability of the P system model to regulate the array generation process. Motivated by these

[5] Corresponding Author: lqpan@mail.hust.edu.cn, lqpanhust@gmail.com

studies and utilizing the P system model, the problem of generation of approximation patterns of different kinds of space-filling curves has been investigated [4, 6, 7, 9].

The Hilbert curve [14], proposed by Hilbert in 1891 as a variant of the Peano curve [18], is a continuous space-filling curve (SFC) passing through every point of a square. The Hilbert words are the finite approximations of the Hilbert curve. In [6] the authors have introduced array representations for the Peano curve, as well as for the related Wunderlich curve of type 1. Array-rewriting rules were introduced, which, acting in parallel, are able to produce the words of the respective SFCs, in array representation, and P systems with two membranes were used as a control mechanism for the parallel array-rewriting.

In the present paper we apply this approach to the Hilbert SFC. The problem of generating Hilbert words in string representation (as chain code words) with P systems was addressed in [5]. Here we introduce the array representation (following [6]) and array rewriting rules, using two types of P systems as control mechanism. We prove correctness using again a linearization procedure, which is here more refined than the one introduced in [6], and is the most important main contribution of the paper. While there are many methods of controlling the rewriting in formal language theory [20], the advantage of the P system is that the number of membranes used is small (here only one or two) and the specification of the target in the rules controls the rewriting.

While the curves considered in [6] were approximated by words represented as $3^n \times 3^n$ arrays, the curves considered in this paper are approximated by words represented as $2^n \times 2^n$ arrays. There are other differences as well, which make the problem of applying the same technique non-trivial. Peano words and Wunderlich type 1 words have their starting and ending points in opposite corners of the square. Hilbert words have their starting and ending points on the same side of the square. There are implications of these facts on the linearization method.

The paper is organized as follows: after Introduction and Preliminaries, we recall in Section 3 the generation of Hilbert words in string representation. Section 4 introduces the array rewriting rules, and states the main result, the generation of Hilbert arrays with P systems. Section 5 presents the linearization method for Hilbert arrays, and proves the main result.

2 Preliminaries

2.1 Space filling curves and chain-code words

Among different syntactic approaches for two-dimensional picture generation, chain code grammars introduced in [16] are string grammars with terminal alphabet $\{l, r, u, d\}$. The symbols, called *chain-code symbols*, are interpreted as moving and drawing a line one unit in the two-dimensional plane, to the left, right, up or down from the current position. The pictures generated by such chain code grammars are called *chain code pictures*.

Based on the chain code grammar features, *chain code P system* models have been introduced in [5] (also [4]) to generate the patterns of approximations of space-filling curves.

2.2 The P Systems used to control parallel rewriting

The P systems used in previous work ([5], [4], [6], [7]) to control the parallel application of the rewriting rules (either string or array rewriting) have the following simple structure:

$$\Pi = (N, T, [_1 [_2]_2]_1, \{w\}, \emptyset, R_1, R_2, 2).$$

They have two membranes, out of which the inner one, with label 2, has the sole role of collecting the results (no rules, no initial object). In membrane 1, rules R_1 are rewriting rules acting in parallel, a group with target indication *here*, denoted by morphism γ (respectively Γ), and another group with target indication *in*, denoted by morphism f (respectively F).

The derivation mode is *target agreement*: all rules with the same target are applied simultaneously.

According to [11], a second P system can be considered, to accomplish the same task, namely

$$\Pi = (N, T, [_1 \ ]_1, \{w\}, R_1)$$

with only one membrane. All rules are in R_1, with two types of *labels*: label *gen* (from *generate*) appended to the γ rules, and label fin (from *final*) appended to the f rules.

The derivation mode is *label agreement*: all rules with the same label are applied simultaneously.

The correctness proofs are extremely precise, and are the same for both systems. We have to show that for any natural $n \geq 1$, $f(\gamma^n(w))$ = the n-th approximation of the desired SFC in string representation, (respectively, $F(\Gamma^n(w))$ = the n-th approximation of the desired SFC in array representation).

3 The Hilbert Curve in String Representation and its Generation with P Systems

The Hilbert curve, introduced in [14], was codified using chain code symbols for the first time in the paper [26]. Several generative mechanisms were used to produce the finite approximations, called Hilbert words. We recall in this section their generation with P systems from [5].

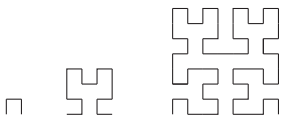


Figure 1: First three approximations of the Hilbert curve, H_1, H_2, H_3.

The first Hilbert word is $H_1 = urd$, and for $n > 1$ the Hilbert words are given by the inductive formula

$$H_n = \rho'(H_{n-1})uH_{n-1}rH_{n-1}d\lambda'(H_{n-1})$$

where ρ', λ' and δ are homomorphisms on $\Sigma = \{u, r, l, d\}$ given below:

$\rho'(u) = r, \rho'(d) = l, \rho'(l) = d, \rho'(r) = u$ – symmetry w.r.t. Oy axis and rotation right 90° degrees;

$\lambda'(u) = l, \lambda'(d) = r, \lambda'(l) = u, \lambda'(r) = d$ – symmetry w.r.t. Oy axis and rotation left 90° degrees. (We have kept the notations from [7].)

$\delta(u) = d, \delta(d) = u, \delta(r) = l, \delta(l) = r$ – symmetry w.r.t. origin.

Let γ denote the string morphism

$$U \to RuUrUdL, R \to UrRuRlD, L \to DlLdLrU, D \to LdDlDuR$$

and let f denote the string morphism

$$U \to \lambda, R \to \lambda, L \to \lambda, D \to \lambda.$$

Theorem 1. *([5]) With the above notations we have:*

$$f(\gamma^n(U)) = H_n.$$

Using this, we have in [5] the following result:

Theorem 2. *The context-free parallel chain code P system*

$$\Pi_H = (\{U, R, L, D\}, \{u, l, r, d\}, [_1[_2]_2]_1, \{U\}, \emptyset, R_1, R_2, 2)$$

where $R_2 = \emptyset$ and R_1 contains the γ rules with target indication "here", and f rules with target indication "in", functioning in target agreement mode, generates the Hilbert words $H_n (n \geq 1)$.

We can now also state the following:

Theorem 3. *The context-free parallel chain code P system*

$$\Pi'_H = (\{U, R, L, D\}, \{u, l, r, d\}, [_1]_1, \{U\}, R_1)$$

with only one membrane, with rules in R_1 the γ rules with label gen, and the f rules with label fin, functioning in label agreement mode, generates the Hilbert words $H_n (n \geq 1)$.

4 2-D Hilbert arrays

We introduce here the 2D representations of the finite approximations of the Hilbert curve, and the array rewriting rules for their generation. We follow closely the model of previous work [6] on the Peano and Wunderlich type 1 curves.

The **Basic patterns** of the Hilbert words are the following four 2×2 arrays:

$$(U) \quad \begin{matrix} 2\ 3 \\ 1\ 4 \end{matrix} \quad \begin{matrix} r\ d \\ u\ * \end{matrix} \quad with * = d, r \tag{1}$$

$$(R) \quad \begin{matrix} 4\ 3 \\ 1\ 2 \end{matrix} \quad \begin{matrix} *\ l \\ r\ u \end{matrix} \quad with * = u, l \tag{2}$$

$$(L) \quad \begin{matrix} 2\ 1 \\ 3\ 4 \end{matrix} \quad \begin{matrix} d\ l \\ r\ * \end{matrix} \quad with * = d, r \tag{3}$$

$$(D) \quad \begin{matrix} 4\ 1 \\ 3\ 2 \end{matrix} \quad \begin{matrix} *\ d \\ u\ l \end{matrix} \quad with * = u, l \tag{4}$$

Consider the alphabet of non-terminals

$$\bar{N} = \{Ud, Ur, Ru, Rl, Ld, Lr, Du, Dl\},$$

where each element is a 1×1 array.

Denote by Γ the array morphism of the eight rewriting rules bellow:

$$U* \to \begin{matrix} Ur\ Ud \\ Ru\ L* \end{matrix} \quad with * = d, r \tag{5}$$

$$R* \to \begin{matrix} D*\ Rl \\ Ur\ Ru \end{matrix} \quad with * = u, l \tag{6}$$

$$L* \to \begin{matrix} Ld\ Dl \\ Lr\ U* \end{matrix} \quad with * = d, r \tag{7}$$

$$D* \to \begin{matrix} R*\ Ld \\ Du\ Dl \end{matrix} \quad with * = u, l \tag{8}$$

Denote by F the array morphism of the four rewriting rules below

$$Ud \to d, Ur \to r, Ld \to d, Lr \to r, Ru \to u, Rl \to l, Du \to u, Dl \to l \tag{9}$$

We intend to prove the following (equivalent) results:

Theorem 4. *The P system*

$$\Pi_H = (N, T, [_1 [_2]_2]_1, \{Ud\}, \emptyset, R_1, R_2, 2)$$

with $R_2 = \emptyset$ and R_1 containing the rules Γ with target indication here, and the rules F with target indication in, functioning in the derivation mode target agreement, will produce in membrane 2 the Hilbert words codified as 2D arrays.

Theorem 5. *The P system*

$$\Pi'_H = (N, T, [_1\ ]_1, \{Ud\}, R_1)$$

with R_1 containing the rules Γ with label gen and the rules F with label fin will produce the Hilbert words codified as 2D arrays.

The proof is the object of the next section.

5 The linearization method for Hilbert arrays

We introduce in this section the linearization of Hilbert arrays. Linearization of 2D arrays for the Peano and Wunderlich curves was first introduced in [6]. The case for the Hilbert curve is different: the Peano related curves have starting and ending points in opposite corners of the square, and the starting point can be determined from the argument; but the Hilbert words have starting and ending points on the same side of the square, and the starting corner has to be specified.

The purpose of linearization is to show that Hilbert 2-D arrays can be transformed into corresponding 1-D strings of chain-code symbols which are precisely the chain-code Hilbert words, and that the array rewriting can thus be reduced to the string rewriting case.

Let $\alpha \in \{se, sw, ne, nw\}$ denote the four corners of the square, (se, sw, ne, nw short for southeast, southwest, northeast and northwest)..

Recall that the nonterminal array alphabet is

$$\bar{N} = \{Ud, Ur, Ru, Rl, Ld, Lr, Du, Dl\}$$

where the elements are 1×1 arrays. Note that the entries of the arrays in $\bar{N}$ have a special form.

Recall that $N = \{U, D, R, L\}$ are the string nonterminals, and that $T = \{u, d, r, l\}$ are the string terminals, the chain code symbols.

Consider the injection $\phi : \bar{N} \to N \times T$ which takes a 1×1 matrix to its entry, consisting of the catenation of a symbol in N with a symbol in T. (Note that not all combinations are allowed!) We have

$$\phi(Ud) = Ud, \quad \phi(Ur) = Ur, \quad \phi(Du) = Du, \quad \phi(Dl) = Dl,$$

$$\phi(Ru) = Ru, \quad \phi(Rl) = Rl, \quad \phi(Ld) = Ld, \quad \phi(Lr) = Lr.$$

Since ϕ acts like the identity, we will omit it in the sequel, but we will keep in mind the different semantics. While Ud for instance can be the argument of Γ, $\phi(Ud)$ can be the argument of γ, but we will write the result as $\gamma(Ud)$.

We plan to define inductively a family $lin_\alpha^{(n)}$ of morphisms on $\bigcup_{n\geq 0} M_{2^n \times 2^n}[\bar{N}]$, which takes arrays into strings over $\bar{N}$ (which, via ϕ, will be identified with strings over $N \cup T$). For every $n \geq 0$,

$$lin_\alpha^{(n)} : M_{2^n \times 2^n}[N] \to N^{2^{2n}},$$

will be a partial function, with domain Γ^n, and such that for all $n \geq 1$ we have

$$lin_\alpha^{(n)} \circ \Gamma^n = \gamma^n \circ \phi.$$

In the following we will omit the ϕ.

Case $n = 0$**.**

We make the convention that $lin^{(0)}$ applied to 1×1 arrays is the identity (i.e. produces the entry of that array). Actually, $lin^{(0)} = lin^{(0)} \circ \phi$. Note there is no index denoting corner. The superscipt (0) will be omited in the sequel.

Case $n = 1$**.**
The partial function

$$lin_{\alpha}^{(1)} : M_{2\times 2}[N] \to N^{2^2},$$

with domain Γ, is defined by the following eight relations:

$$lin_{sw}^{(1)}(\Gamma(U*)) = lin_{sw}^{(1)} \begin{pmatrix} Ur\ Ud \\ Ru\ L* \end{pmatrix} = RuUrUdL*, \qquad with\ * = \{d, r\} \tag{10}$$

$$lin_{ne}^{(1)}(\Gamma(L*)) = lin_{ne}^{(1)} \begin{pmatrix} Ld\ Dl \\ Lr\ U* \end{pmatrix} = DlLdLrU*, \qquad with\ * = \{d, r\} \tag{11}$$

$$lin_{sw}^{(1)}(\Gamma(R*)) = lin_{sw}^{(1)} \begin{pmatrix} D*\ Rl \\ Ur\ Ru \end{pmatrix} = UrRuRlD*, \qquad with\ * = \{u, l\} \tag{12}$$

$$lin_{ne}^{(1)}(\Gamma(D*)) = lin_{ne}^{(1)} \begin{pmatrix} R*\ Ld \\ Du\ Dl \end{pmatrix} = LdDlDuR*, \qquad with\ * = \{u, l\}. \tag{13}$$

Note that $lin_{\alpha}^{(1)}$ applied to an array starts in the α corner of the array and will access each entry of the array. At each entry, which is of the form Xy, with $X \in \{U, R, L, D\}$ and $y \in \{u, r, l, d\}$,

- it writes the entry Xy into the output string
- goes to the neighbouring entry indicated by y.

This will be true for all the other members of the family.

We recall from section 3 the definition of the string morphism γ:

$$\gamma(U) = RuUrUdL, \quad \gamma(R) = UrRuRlD,$$

$$\gamma(L) = DlLdLrU, \quad \gamma(D) = LdDlDuR.$$

From the definition of $lin^{(1)}$ we note that:

$$lin_{sw}^{(1)}(\Gamma(U*)) = RuUrUdL* = \gamma(U)* = \gamma \circ lin(U*),\ with\ * = \{d, r\} \tag{14}$$

$$lin_{ne}^{(1)}(\Gamma(L*)) = DlLdLrU* = \gamma(L)* = \gamma \circ lin(L*),\ with\ * = \{d, r\} \tag{15}$$

$$lin^{(1)}_{sw}(\Gamma(R*)) = UrRuRlD* = \gamma(R)* = \gamma \circ lin(R*),\ with\ * = \{u, l\} \tag{16}$$

$$lin^{(1)}_{ne}(\Gamma(D*)) = LdDlDuR* = \gamma(D)* = \gamma \circ lin(D*),\ with\ * = \{u, l\} \tag{17}$$

We have thus established

Lemma 1. *For $X = U, L, R, D$, and the appropriate values for $*$, we have (written in compact form):*

$$lin^{(1)}_{\alpha} \circ \Gamma(X*) = \gamma(X)* = \gamma \circ lin(X*).$$

Case $n = 2$.

We want $lin^{(2)}_{\alpha} : M_{4\times4}[N] \to N^{16}$, to act on $\Gamma^2(N)$, which are 2×2 arrays resulting from the application of array rewriting rules Γ twice.

For instance, in view of Lemma 1, we have:

$$\begin{aligned} &lin^{(2)}_{sw} \circ \Gamma^2(U*) = (lin^{(2)}_{sw}(\Gamma(\Gamma(U*))) = lin^{(2)}_{sw}\Gamma\left(\begin{pmatrix} Ur\ Ud \\ Ru\ L* \end{pmatrix}\right) \\ &= lin^{(2)}_{sw}\begin{pmatrix} \Gamma(Ur)\ \Gamma(Ud) \\ \Gamma(Ru)\ \Gamma(L*) \end{pmatrix} \end{aligned} \tag{18}$$

The 4×4 array which is the argument of $lin^{(2)}$ above is composed of four 2×2 arrays, which are defined, and on which $lin^{(1)}$ is defined, (case $n = 1$) which are 'glued' according to the basic pattern

$$\begin{matrix} r\ d \\ u\ * \end{matrix}$$

defined in relation (1). Thus, the sequence of equalities above can be continued by

$$\begin{aligned} &= lin^{(1)}_{sw} \circ \Gamma(Ru) \\ &\quad lin^{(1)}_{sw} \circ \Gamma(Ur) \\ &\quad lin^{(1)}_{sw} \circ \Gamma(Ud) \\ &\quad lin^{(1)}_{ne} \circ \Gamma(L*) \\ &= \gamma(R)u\gamma(U)r\gamma(U)d\gamma(L)* \quad = \gamma^2(U*). \end{aligned} \tag{19}$$

On the other arguments, the pattern can be any other one of three basic patterns described by formulas (2), (3) and (4), the definition of $lin^{(2)}$ follows from the pattern, and the computations are similar.

Thus we also have

Lemma 2. *For $X = U, L, R, D$, and the appropriate values for $*$, we have (written in compact form):*

$$lin_\alpha \circ \Gamma^2(X*) = \gamma^2(X)* = \gamma^2 \circ lin(X*).$$

Case n (Induction hypothesis).
Assume we have defined

$$lin_\alpha^{(n)} : M_{2^n \times 2^n}[N] \to N^{2^{2n}},$$

a partial function, with domain Γ^n, such that the following are true for for $X = U, L, R, D$ respectively, and the corresponding values of $*$:

$$lin_\alpha^{(n)} \circ \Gamma^n(X*) = \gamma^n(X)* = \gamma^n \circ lin(X*).$$

Case $n+1$:
We define now the partial function

$$lin_\alpha^{(n+1)} : M_{2^{n+1} \times 2^{n+1}}[N] \to N^{2^{2n+1}},$$

with domain $\Gamma^{n+1}(\bar{N})$. The arrays of $\Gamma^{n+1}(\bar{N})$ are each composed of four $2^n \times 2^n$ subarrays, arranged according to one of the four basic patterns. We have:

$$\Gamma^{n+1}(U*) = \begin{pmatrix} \Gamma^n(Ur)\ \Gamma^n(Ud) \\ \Gamma^n(Ru)\ \Gamma^n(L*) \end{pmatrix}, \qquad with\ * = d, r \tag{20}$$

$$\Gamma^{n+1}(L*) = \begin{pmatrix} \Gamma^n(Ld)\ \Gamma^n(Dl) \\ \Gamma^n(Lr)\ \Gamma^n(U*) \end{pmatrix}, \qquad with\ * = d, r \tag{21}$$

$$\Gamma^{n+1}(R*) = \begin{pmatrix} \Gamma^n(D*)\ \Gamma^n(Rl) \\ \Gamma^n(Ur)\ \Gamma^n(Ru) \end{pmatrix}, \qquad with\ * = u, l \tag{22}$$

$$\Gamma^{n+1}(D*) = \begin{pmatrix} \Gamma^n(R*)\ \Gamma^n(Ld) \\ \Gamma^n(Du)\ \Gamma^n(Dl) \end{pmatrix}, \qquad with\ * = u, l \tag{23}$$

We define:

$$\begin{aligned} &lin_{sw}^{(n+1)}(\Gamma^{n+1}(U*)) := \\ &= lin_{sw}^{(n)}(\Gamma^n(Ru))\, lin_{sw}^{(n)}(\Gamma^n(Ur))\, lin_{sw}^{(n)}(\Gamma^n(Ud))\, lin_{ne}^{(n)}(\Gamma^n(L*)) \end{aligned} \tag{24}$$

$$\begin{aligned} &lin_{ne}^{(n+1)}(\Gamma^{n+1}(L*)) := \\ &= lin_{ne}^{(n)}(\Gamma^n(Dl))\, lin_{ne}^{(n)}(\Gamma^n(Ld))\, lin_{ne}^{(n)}(\Gamma^n(Lr))\, lin_{sw}^{(n)}(\Gamma^n(U*)) \end{aligned} \tag{25}$$

$$\begin{aligned} &lin_{sw}^{(n+1)}(\Gamma^{n+1}(R*)) := \\ &= lin_{sw}^{(n)}(\Gamma^n(Ur))\, lin_{sw}^{(n)}(\Gamma^n(Ru))\, lin_{sw}^{(n)}(\Gamma^n(Rl))\, lin_{ne}^{(n)}(\Gamma^n(D*)) \end{aligned} \tag{26}$$

$$lin_{ne}^{(n+1)}(\Gamma^{n+1}(D*)) := \\ = lin_{ne}^{(n)}(\Gamma^n(Ld))\, lin_{ne}^{(n)}(\Gamma^n(Dl))\, lin_{ne}^{(n)}(\Gamma^n(Du))\, lin_{sw}^{(n)}(\Gamma^n(R*)) \tag{27}$$

Lemma 3. *For $X = U, L, R, D$ respectively, and appropriate $*$, we have:*

$$lin_{\alpha}^{(n+1)} \circ \Gamma^{n+1}(X*) = \gamma^{n+1}(X)* = \gamma^{n+1} \circ lin(X*).$$

The proof is immediate by straightforward computations and the induction hypothesis. □

Lemma 4.

$$lin_{\alpha}^{(n)} \circ (F \circ \Gamma^n) = (f \circ \gamma^n) \circ lin_{\alpha}$$

Proof. For $X = U, D, R, L$ and the appropriate values for $*$, we have:

$$lin_{\alpha}^{(n)} \circ F \circ \Gamma^n(X*) = f \circ lin_{\alpha}^{(n)} \circ \Gamma^n(X*) = (f \circ \gamma^n)(lin_{\alpha}(X*)). \tag{28}$$

□

The proof of Theorem 2:

Starting with the nonterminal 1×1 array Ud in membrane 1, by applying n times (in parallel) the rules with target *here* we obtain the array $\Gamma^n(Ud)$, still in membrane 1. One application of the rules with target *in* leads to the array $F(\Gamma^n(Ud))$ in membrane 2. By the third relation of the previous lemma, the linearization of $F(\Gamma^n(Ud))$ is the string $f(\gamma^n(U))d$. According to Theorem 1, we have $f(\gamma^n(U))d = H_n d$, where H_n is the nth Hilbert word. □

6 Conclusions and future work

Based on previous work on Peano type SFC words [6], we have introduced here 2D array representations of Hilbert words. We have shown how they can be generated with P systems with array rewriting rules acting in parallel. The linearization procedure, which allows the transition from arrays to strings and from array-rewriting to string rewriting, is here explored more in depth, and the formalism is much more refined.

Other variants of Hilbert words in array representation can be readily obtained using this approach.

Methods and concepts exposed here can be extended to 3D cases.

A more vast area of further research is to use P systems to implement or model some of the many applications of SFCs (see [1]).

Acknowledgements

This work was started during the two weeks visit of RC at the Key Laboratory of Image Information Processing and Intelligent Control, of Education Ministry of China, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, China, June 24 – July 7, 2019.

The work of LZ and LP and was supported by the National Natural Science Foundation of China (61772214).

The work of GZ was supported by the National Natural Science Foundation of China (61972324, 61672437, 61702428), the New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0043), the Sichuan Science and Technology Program (2018GZ0185, 2018GZ0086) and Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06).

The authors are grateful to Rudolf Freund for discussions and suggestions during the CMC20 conference in Curtea de Argeş, Romania, August, 2019.

References

1. Bader, M.: Space-filling Curves - An Introduction with applications in Scientific Computing. Texts in Computational Science and Engineering. Springer-Verlag (2013).
2. Ceterchi, R., Mutyam, M., Păun, Gh., Subramanian, K.G.: Array - rewriting P systems. Natural Comput. 2 (2003) 229–249.
3. Ceterchi, R., Nagar, A.K., Subramanian, K.G.: Approximating polygons for space-filling curves generated with P systems, C. Graciani et al. (Eds.): Pérez-Jiménez Festschrift, LNCS 11270, (2018) 57–65. https://doi.org/10.1007/978-3-030-00265-7_5
4. Ceterchi, R., Nagar, A.K., Subramanian, K.G.: Chain Code P System Generating a Variant of the Peano Space-filling Curve, T. Hinze et al. (Eds.): CMC 2018, LNCS 11399, Springer Nature (2019). https://doi.org/10.1007/978-3-030-12797-8_6
5. Ceterchi, R., Subramanian, K.G., Venkat, I.: P Systems with parallel rewriting for chain code picture languages. Proc. 11th Conference on Computability in Europe (CiE), 145–155 (2015).
6. R. Ceterchi, A.K. Nagar, L.Pan, K.G. Subramanian: P Systems Generating Array Representations of Peano Type Space-Filling Curves. Proceedings of the 20th International Conference on Membrane Computing, CMC20, August 5–8, 2019, Curtea de Argeş, Romania (Gh. Păun editor) Bibliostar, Râmnicu Vâlcea (2019) 309–324.
7. R. Ceterchi, K.G. Subramanian: P Systems for Generating Pictures in String Representations: The Case of Space-Filling Curves, Proceedings of the 20th International Conference on Membrane Computing, CMC20, August 5–8, 2019, Curtea de Argeş, Romania (Gh. Păun editor) Bibliostar, Râmnicu Vâlcea (2019) 63–80.
8. Dassow, J., Habel, A., Taubenberger, S. : Chain-code pictures and collages generated by hyperedge replacement, Lecture Notes in Comp. Sci., 1073 (1996) 412–427.
9. Dharani A., Stella Maragatham R., Nagar A. K., Subramanian K. G.: Chain Code P System for Generation of Approximation Patterns of Sierpiski Curve. IWCIA2018, LNCS 11255 (2018) 43–52

10. Drewes, F. : Some Remarks on the Generative power of collage grammars and chain-code Grammars, Lecture Notes in Comp. Sci., 1764 (2000) 1–14.
11. Freund R.: Playing with Derivation Modes, Proceedings of the 20th International Conference on Membrane Computing, CMC20, August 5–8, 2019, Curtea de Argeş, Romania (Gh. Păun editor) Bibliostar, Râmnicu Vâlcea (2019) 109–122.
12. Gheorghe, M., Păun, Gh., Pérez Jiménez, M. J., Rozenberg, G. : Research frontiers of membrane computing: Open problems and research topics. Int. J. Found. Comput. Sci. 24(5) (2013) 547–624.
13. Giammarresi D., Restivo A., Two-dimensional languages In: Rozenberg G., Salomaa, A. (eds.) *Handbook of Formal Languages*, vol. **3**, pp. 215–267. Springer, Heidelberg (1997).
14. Hilbert, D. : Über die stetige Abbildung einer Linie auf ein Flächenstück. Math. Annln. 38 459–460 (1891).
15. Kitaev, S. : Mansour, T., Seebold, P. : The Peano curve and counting occurrences of some patterns, J. Autom., Lang. Combin. 9(4) (2004) 439–455.
16. Maurer, H.A., Rozenberg, G., Welzl, E. : Using string languages to describe picture languages, Inform. Control 54 (1982) 155–185.
17. Moore, E.H. : On certain crinkly curves. Trans. Amer. Math. Soc. 1 (1900), 72–90.
18. Peano, G. : Sur une courbe qui remplit toute une aire plane. Math. Annln. 36 157–160 (1890).
19. Păun, Gh. : Computing with membranes. J. Comp. System Sci. 61 (2000), 108-143.
20. Salomaa, A. : Formal languages. Academic Press. London. 1973.
21. Sagan, H. : Space-filling curves. Springer-Verlag. New York. 1994.
22. Seebold, P. : Tag system for the Hilbert curve. Discrete Math. and Theor. Comp. Sci. 9 (2007) 213–226.
23. Sierpiński, W. : Sur une nouvelle courbe continnue qui remplit toute une aire plane. Bull. Acad. Sci. de (Sci. math et nat.,) Série A) 462–478 (1912).
24. Siromoney, R., Subramanian, K.G. : Space-filling curves and Infinite graphs. Lecture notes in Comp. Sci. 153 (1983) 380–391.
25. Subramanian, K.G. : P systems and picture languages, Lecture Notes in Comp. Sci. 4664 (2007) 99–109.
26. Subramanian, K.G., Siromoney, R.: On Array Grammars and Languages. Cybernetics and Systems. 18 (1987) 77-98.
27. Subramanian, K.G., Venkat, I., Pan, L. : P Systems generating chain code picture languages, Proc. Asian Conf. Membrane Computing, (2012) pp.115–123.
28. Wunderlich, W. : Über Peano-Kurven. Elem. Math. 28 (1973) 1–10.

Automatic Design of Spiking Neural P Systems Based on Genetic Algorithms

Jianping Dong[1], Michael Stachowicz[2], Gexiang Zhang[1] *, Matteo Cavaliere[2], Haina Rong[1], and Prithwineel Paul[1]

[1] School of Electrical Engeneering, Southwest Jiaotong University, Chengdu, 610031, China
zhgxdylan@126.com

[2] Faculty of Science and Engineering, Manchester Metropolitan University, Manchester, Britain

Abstract. At present, all known spiking neural P systems (SN P systems) are established by manual design rather than automatic design. The method of manual design poses two problems: consuming a lot of computing time and making unnecessary mistakes. In this paper, we propose an automatic design approach for SN P systems by genetic algorithms. More specifically, the regular expressions are changed to achieve the automatic design of SN P systems. In this method, the number of neurons in system, the synapse connections between neurons, the number of rules within each neuron and the number of spikes within each neuron are known. A population of SN P systems is created by generating random accepted regular expressions. A genetic algorithm is applied to evolve a population of SN P systems toward a successful SN P systems with high accuracy and sensitivity for carrying out specific tasks. An effective fitness function is designed to evaluate each candidate SN P system. In addition, the elitism, crossover and mutation are also designed. Finally, experimental results show that the approach can successfully accomplish the automatic design of SN P systems for generating natural numbers and even natural numbers by using the .NET framework.

Keywords: Spiking neural system; genetic algorithm; fitness function; mutation probability; membrane computing.

* Corresponding author.

1 Introduction

As a bridge between computer science and natural science, natural computing is a wide research area with several branches, which includes cellular automations [1], neural computations [2], elvoving algorithms [3–6], swarm intelligence [7], artificial immune systems [8], membrane computing [9, 10], etc. Membrane computing is a new and hot research direction in recent years. Membrane computing models, called P systems or membrane systems, are abstracted from the structure and functioning of the living cell, as well as from the cooperation of cells in tissues, organs and other populations of cells [11, 12]. Currently, membrane systems can be divided, according to different membrane structure, into cell-like P systems, tissue-like P systems and SN P systems [13]. Until now, many variants of membrane computing models have been investigated and also some models have been used in real life application. [14–18].

A good membrane computing model is the basis of its applied investigations and software and hardware implementations. In recent years, with the development of membrane computing models, experts and scholars have started the automatic design method of membrane computing models. Currently, the automatic design methods of membrane systems can be classified into two groups [19, 21]: the heuristic algotithms and the reasoning techniques. Genetic algorithms and quantum-inspired evolutionary algorithms are used to evolve a population of P systems [20]. A genetic approach was used to design an artificial cell system, which can be regarded as a cell-like P system with a single membrane [22]. An evolving design solution of membrane systems was proposed to implement the design of square of 4 in the membrane system based on simulation software P-Lingua [23]. A fitness function with a penalty factor was presented to evaluate a P system [24]. Cell-like P systems and tissue-like P systems have been automatically designed to fullfill some specific tasks, like computing square of 4 and of n ($n \geq 2$ is a natural number) [25–27]. A deterministic and non-halting membrane system by tuning membrane structures, initial objects and evolution rules was proposed in [28]. The reasoning techniques were used to design P systems in [29]. However, this idea has never been extended to the third generation neural networks, which closely the activity of biological neurons, and more specifically SN P systems [15].

This paper makes the attempt to propose an automatic design approach of SN P systems by genetic algorithms. First of all, we establish a population of

SN P systems with changing regular expression on the predefined number of neurons in each SN P system, the synapse connections between neurons, the number of rules within each neuron and the number of spikes within each neuron. Secondly, a genetic algorithm is applied to evolve a population of SN P systems toward a successful SN P system with high accuracy and sensitivity for carrying out specific task. An effective fitness function is designed to evaluate each candidate SN P system. Finally, experimental results show that the approach can successfully accomplish the automatic design of SN P systems for generating natural numbers and even natural numbers by using the .NET framework.

The paper is structured as follows. Section 2 describes the problems of the automatic design of SN P systems. Section 3 presents the automatic design approach for SN P systems based on genetic algorithms in detail. Experimental results are shown and analyzed in Section 4. Finally, some conclusions are drawn in Section 5.

2 Problem Description

In this section, we briefly review SN P systems, and then the problems of the automatic design of SN P systems are described.

2.1 Spiking Neural P System

A SN P system consists of five main elements: the number of neurons in each SN P system, the synapse connections between neurons, the number of rules within each neurons, the regular expressions which define each rule and the number of spikes within each neuron.

A SN P system [15] of degree $m \geq 1$ is a tuple $\Pi = (O, \sigma_1, \cdots, \sigma_m, syn, i_o)$, where:

(1) $O = \{a\}$ is the singleton alphabet (a is called spike);
(2) $\sigma_1, \cdots, \sigma_m$ are neurons, identified by pairs

$$\sigma_i = (n_i, R_i), 1 \leq i \leq m \tag{1}$$

where:
(a) $n_i \geq 0$ is the initial number of spikes contained in σ_i.

(b) R_i is a finite set of rules of the following two forms:
   (i) $E/a^c \rightarrow a; d$ where E is a regular expression over O, and $c \geq 1, d \geq 0$;
   (ii) $a^s \rightarrow \lambda$, for some $s \geq 1$, with the restriction that for each rule $E/a^c \rightarrow a; d$ of type (i) from R_i, we have $a^s \notin L(E)$;

(3) $syn \subseteq \{1, \ldots m\} \times \{1, \ldots m\}$ with $(i, i) \notin syn$ for $i \in \{1, \ldots m\}$ (synapses between neurons);

(4) i_o indicates the output neuron (i.e. σ_{io} is the output neuron).

The firing and forgetting rules of a SN P system are described and discussed in [15, 30]. The distinguishing feature of SN P system is that the sequence of configurations can associate a spike train. If the output neuron spikes, then we have 1 and otherwise we have 0. Hence, the spike train can be represented by the sequence of ones and zeros.

2.2 Problem Statement

As briefly introduced above, there are many variants of SN P systems, which are designed by manual design ranther than automatic design. In order to automatically generate a SN P system, we should consider each aspect in a SN P system. In this paper, the number of neurons in system, the synapse connections between neurons, the number of rules within each neurons and the number of spikes within each neuron, according to specific task, are previously determined, but the regular expressions which define each rule and the delays on each rule are randomly generated in a SN P system. Then we can generate a population of SN P systems by same method. In our study, the aim is to use genetic algorithms to get an optimal SN P system by approriately evolving a SN P system. The problem is summarized as follows:

First of all, we define a population of SN P systems $\Pi = \{\Pi_i\}_{i \in H}$, where H is a subset of natural numbers and each SN P system Π_i of degree $m \geq 1$ is described as follows:

$$\Pi_i = (O, \sigma_1, \cdots, \sigma_m, syn, i_o)$$

where

(1) $O = \{a\}$ is a predefined singleton alphabet;

(2) $\sigma_1, \cdots, \sigma_m$ is the neurons from 1 to m.

$$\sigma_i = (n_i, R_i), 1 \leq i \leq m \tag{2}$$

where:

(a) $n_i \geq 0$ is the initial number of spikes contained in σ_i.

(b) R_i is a finite set of rules of the following two forms:

(i) **Spike transfer rules:** $E/a^c \rightarrow a; d$. When fullfilling spike transfer rules and $d = 0$, a spike in the neuron will travel along the synapses connected to other neurons.

(ii) **Spike forgetting rules:** $i.e$, s spikes are consumed.

(3) i_o indicates the output neuron.

In addition, the implementations of all the rules are considered to be nodeterministic through the population of SN P systems. In order to simulate nodeterminism of SN P systems, pseudorandoms are used to determine which rule is to be selected.

3 Automatic Design Method

In this section, the detailed procedure of automatically designing of a SN P system is described. An overview of automatic design method is outlined and each step is explained one by one, which include overview of automatical design method, building a population of SN P systems, Design of fitness function and elitism, crossover and mutations.

3.1 Overview of automatical design method

An overview of automatic design method is decribed using two aspects: generating a population of SN P systems and evolving a population of SN P systems. Initial configuration is predefined, which includes the number of neurons in each P system, the synapse connections between each neuron, the number of rules and the number of spikes within each neuron, a population of SN P systems is created by randomly generating rules at the start. We evolve the population based on genetic algorithms after generating a population. The pseudocode of automatic design method is shown in Fig. 1

The genenral steps of the design method can be summarized as follows:

Step 1: Input some basic parameters, which include m, n_i, syn, i_o, H, $MaxSteps$, $StepRepetition$, $MutationRate$, $MinFitness$, $MaxGeneration$, $BestFitness$ and $ExpectedSet$,

Input: Initial membrane construction and objects and genetic algorithm
1: i=1
2: **while** ($i \leq H$) **do**
3: Generating random $SNPS_i$
4: Caculating fitness value F($SNPS_i$)
5: **if** ($F(SNPS_i) \leq MinFitness || F(SNPS_i) == null$) **then**
6: Generating new $SNPS_i$ and replacing old $SNPS_i$
7: **end if**
8: $i = i + 1$
9: **end while**
10: **while** ($generation \leq MaxGeneration$) **do**
11: Caculating fitness value each SNPS
12: Sorting population accordding to set F($SNPS$)
13: i=1
14: **while** ($i \leq H$) **do**
15: **if** ($i \leq Elitism$ && $i \leq H$) **then**
16: $Newpopulation[i] = Population[i]$
17: **if** (F($SNPS_i > BestFitness$)) **then**
18: $BestFitness = F(SNPS_i)$
19: **end if**
20: **else**
21: Crossover and mutation are operated
22: **end if**
23: **if** ($F(SNPS_i) == 0 || F(SNPS_i) == null$) **then**
24: $F(SNPS_i) = 0$
25: **else**
26: $F(SNPS_i) = F(SNPS_i)$
27: **end if**
28: $i = i + 1$
29: **end while**
30: $genration = gneration + 1$
31: **end while**
Output: Spiking neural P system

Fig. 1. The algorithm of automatic design of SN P systems

where:

(a) m, n_i, syn and i_o represent the number of neurons in each SN P system, the number of spikes in each neuron, the synapse connections between each neuron and the output neurons, respectively.

(b) i_o indicates the output neuron.

(c) H is population size.

(d) $MaxSteps$ represents the maximum steps that each network will take.
(e) $StepRepetition$ is the amount of repetitions each network will generate an output list.
(f) $MutationRate$ is the percentage chance for mutation.
(g) $MinFitness$ represents minimal fitness.
(h) $MaxGeneration$ is the max amount of generations.
(i) $BestFitness$ represents the best fitness through generations.
(j) $ExpectedSet$ is the expected set.

Step 2: Accordding to initial parameters and rules that randomly create, generating the population of SN P systems and caculating the fitness value. $F(SNPS_i)$ and $F(SNPS)$ represent the fitness value of the ith SN P system and the fitness set of all SN P systems in the population, respectively. Investigate whether SN P systems are correct according to the fitness function value of each SN P system in the population.

Step 3: The genetic algorithm is used to automatically design each SN P system in the population. $Elitism$ represents the number of reserving optimum number of SN P systems in the population. The rest of SN P systems are operated by using crossover and mutation.

Step 4: Output of new SN P system with high sensitivity and precision after completion of automatic design.

We can infer that from Fig. 1, the most important three steps include building a population of SN P systems, designing a fitness function and setting elitism, crossover and mutation. We will introduce them one by one in the following subsections.

3.2 Building a population of SN P systems

A SN P system includes the number of neurons, the synapse connections between neurons, the number of rules within each neuron, the regular expressions which define each rule and the number of spikes within each neuron. A SN P system represents an individual (DNA, $SNPS_i$) in the popualtion. Here, an individual is also thought of as a set, which contains above five aspects. As a result, the building of a population of SN P systems can be divided into the following steps.

Step 1: Generate a random individual, where rules are randomly generated and other elements are predefined.

Step 2: Repeat the first step until all the individuals($SNPS_i$) in the population are produced;

Step 3: Check whether each individual is correct;

Step 4: Delete and replace individuals with incorrect and fitness value lower than 0.1;

Step 5: Save the initial population.

With the initial population, it is necessary to have an appropriate evaluation function to guide the population to evolve to the optimal solution. Hence, it is worth to notice that the fitness function plays an important role throughout the automatical design process. We describe the details of the fitness function as follows.

3.3 Design of fitness function

In this subsection, we discuss how to design the fitness function, which is used to calculate the sensitivity and the precision of SN P systems. There are two data sets after the establishment of the SN P systems. One is a real output set $OutputSet$. Another is given expected set $ExpectedSet$. $OutputSet$ represents generating number set of repeating execution SN P systems for a specifical task. $ExpectedSet$ is expected number set for a special task. So a fitness function is established by comparing elements in the real output set and the expected set. The pseudocode of the fitness funtion is shown in Fig. 2

The category of an element in the above two sets is as follows:

(1) The output set is compared with the expected set and for every number that is in both of the sets, the true positive count tp increases;

(2) The output set is compared with the expected set and for every number that is in the output set but not in the expected set, the false positive count fp increases;

(3) The output set is compared with the expected set and for every number that is not in the output set but is in the expected set, the false negative count fn increases;

(4) The true negative values, which are not in the output set and not in the expected set, are not counted, since they are not needed for this design.

In the process of the automatic design, besides the design of fitness function, the elitism, crossover and mutation of genetic algorithms are also very important problems. We will discuss how to evolve accroding to the fitness function in the next subsection.

Input: $OutputSet$, $ExpectedSet$, $tp = 0$, $fp = 0$, $fn = 0$
1: Initialization settings
2: Merging elements from $OutputSet$ and $ExpectedSet$ into $OutExSet$. The length of $OutExSet$ is n
3: $i = 1$
4: **while** ($i \leq H$) **do**
5: $i = i + 1$
6: **if** $OutExSet(i) \in OutputSet$ **then**
7: **if** $OutExSet(i) \in ExpectedSet$ **then**
8: $tp = tp + 1$
9: Turn to **Step 21**
10: **else**
11: $fp = fp + 1$
12: Turn to **Step 21**
13: **end if**
14: **else**
15: **if** $OutExSet(i) \in ExpectedSet$ **then**
16: $fn = fn + 1$
17: Turn to **Step 21**
18: **else**
19: Turn to **Step 21**
20: **end if**
21: **end if**
22: **if** $i \geqslant n$ **then**
23: Turn to **Step 26**
24: **else**
25: Turn to **Step 4**
26: **end if**
27: $Fitness = (\frac{2 \times tp}{2 \times tp + fp + fn}) \times sf$
28: **end while**
Output: Return $Fitness$

Fig. 2. The design of the fitness function

3.4 Elitism, crossover and mutation

DNA consists of genes, which in the case of this paper are represented by a SN P system. Each instance of DNA also contains the fitness for the genes contained within it. The crossover function allows the exchange of genes between two parents, creating a new child DNA with the characteristics of the parents that were used. After the crossover, there is also a chance for the new child DNA to mutate, changing one of the rules in the generated network at random. To ensure variety in each population, population total new random members are added to the population pool with each generation.

Along with crossover and chance for mutation, this algorithm also allows for the use of elitism. This feature allows a selected number of best networks to be introduced with a new generation. This allows the algorithm to ensure that fitness will continue to increase even if poor mutations occur too frequently. The pseudocode of the elitism, crossover and mutation is shown in Fig. 3

Input: A population of SN P systems in the current genenration.
1: Caculating fitness value each SN P system and sorting SN P system according the fitness value.
2: $i = 1$
3: **while** $(i \leq H)$ **do**
4: $i = i + 1$
5: **if** $(i \leq Elitism \;\&\&\; i \leq H)$ **then**
6: $Newpopulation[i] = Population[i]$
7: **else**
8: Parent1=ChooseParent( )
9: Parent2=ChooseParent( )
10: **if** $random1 \leq CrossoverRate$ **then**
11: $Newpopulation[i] = maxParent1, Parent2$
12: **else**
13: $CrossoverChild$=Crossover(Parent1,Parent2)
14: **if** $random2 \leq CrossoverRate$ **then**
15: $Newpopulation[i] = CrossoverChild$
16: **else**
17: $MutateChild$=Mutate($CrossoverChild$)
18: $Newpopulation[i] = MutateChild$
19: **end if**
20: **end if**
21: **end if**
22: **end while**
Output: Return $Newpopulation$

Fig. 3. The design of the elitism, crossover and mutation

The detailed procedure of elitism, crossover and mutation are described as follows:

Elitism: Elitism, the best optimal individual in the current population, is set to 1 in the method of the automatic design, $i.e.$, a SNP system with the high sensitivity and precision can be saved to new popualtion each generation.

Crossover: The crossover is mainly composed of two steps, one is to choose the parent individuals (Parents with a higher fitness will have a higher chance of reproducing, however all parents should have a chance), and the other one is to exchange the corresponding rules in the two parent individuals.

Mutations: After getting new sub-individuals from the crossover of two parent individuals, new sub-individuals are mutated and added to new population, where $MutationRate$ is adjusted according to the detailed problem dynamically. The pseudocode of dynamic adjustment is described in Fig. 4.

Input: $GlobleBestFitness = 0$, $CurrentBestFitness$, $RateChange = 0$, $MutationRate = 0$
1: $i = 1$
2: **while** $(i \leq H)$ **do**
3: $i = i + 1$
4: **if** $GlobleBestFitness \leq CurrentBestFitness$ **then**
5: $GlobleBestFitness = CurrentBestFitness$
6: $RateChange++$
7: **else**
8: $RateChange = 0$
9: **end if**
10: **if** $RateChange \geq 10$ **then**
11: $MutationRate = random(0, 10)$
12: **else**
13: $MutationRate = random(10, 20)$
14: **end if**
15: **end while**
Output: Return $MutationRate$

Fig. 4. The dynamic adjustment of mutation probability

4 Experimentations and Analysis of Results

In this section, the method of the automatic design is first thoroughly test to ensure the systems are generated as expected. First of all, a SN P system

generating all even natural numbers and a SN P system generating all natural numbers are used to simulate the evolution of SN P systems towards a expected configuration to ensure the sentivity and precision of the data being outputed. Secondly, we analyze the experimental results of two known SN P systems and obtain experimental conclusions.

In order to illustrate the performance of the method of the automatic design when simulating a SN P system generating all natural numbers, we do a dynamic behavior analysis from the fitness function value of the experimental testing process. F_{max} is used to testify the convergency of the algorithms.

F_{max}: the maximum fitness value in the current generation. The maximum fitness value F_{max} represents the best SN P system in the current genneration. F_{max} is defined in Eqs.3.

F_{av}: the average fitness value in the current generation. A larger value of F_{av} represents a smaller difference between the expected set and the output set. F_{av} is defined in Eqs.4.

$$F_{max} = max\sum_{i=1}^{H} F(SNPS_i) \tag{3}$$

$$F_{av} = \frac{1}{H}\sum_{i=1}^{H} F(SNPS_i) \tag{4}$$

where, $F(SNPS_i)$ represents the fitness value of ith SN P systems, H is the number of the SN P systems in the population.

4.1 A SN P system generating all natural numbers

The specific sketch of a SN P system generating all natural numbers is shown in Fig. 5.

A SN P system generating all natural numbers mainly contains five elements: four neurons, ten synapse connections between neurons, eight rules and two starting spikes each neuron. Four neurons consist of three general neurons and one output neuron.

In the process of simulated evolution, the basic design parameters are set as follows. Expected output set for the natural numbers system: 1, 2, 3, 4, 5, 6, 7, 8, 9; Population count: 4 members; Maximum number of steps per system: 50;

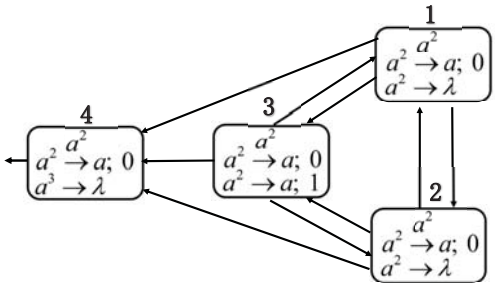


Fig. 5. A SN P system generating all natural numbers

Maximum number of repetitions per system: 50; Maximum number of generations: 200.

We obtain the change curves of the average fitness values and the maximum fitness values in Fig. 6.

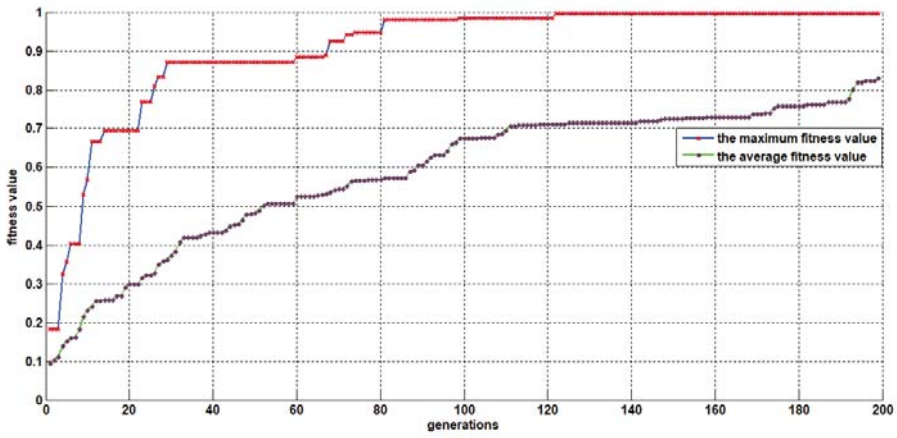


Fig. 6. The change curves of the average fitness values and the maximum fitness values

As in Fig. 6, If the number of iterations increases, the maximum fitness value rapidly increases in initial period and also the average fitness value slowly increases than the maximum fitness value at the same time. After 80 iterations,

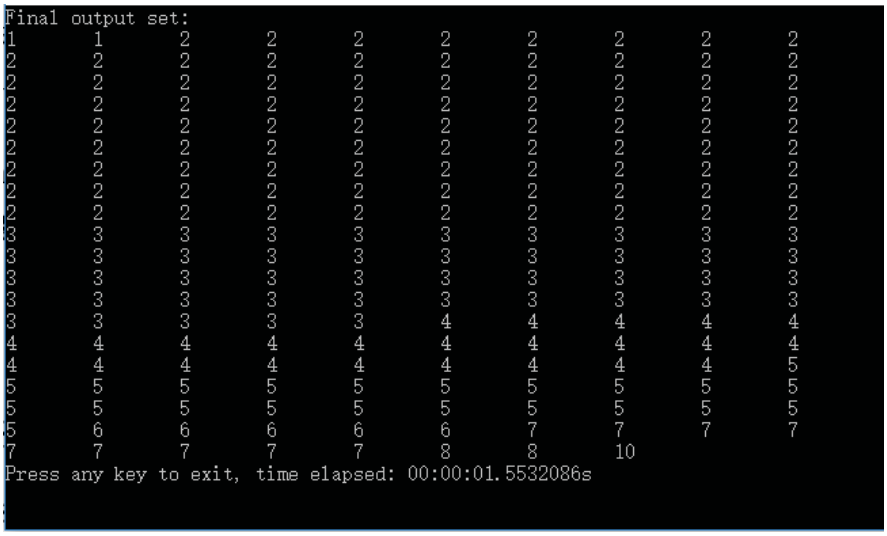


Fig. 7. The output set of SN P systems generating all natural numbers

the maximum fitness value is constant, which indicate that the evolutionary is convergent.

From Fig. 7, the results of the correct natural data output are produced by a natural SNP system and is the same as the expected set.

4.2 A SN P system generating all even natural numbers

The specific sketch of a SN P system generating all even natural numbers is shown in Fig. 8.

A SN P system generating all even natural numbers mainly contails five elements: seven neurons, twelve synapse connections between neurons, twelve rules, two starting spikes in six neurons and no starting spikes in other neurons. Seven neurons consist of six general neurons and one output neuron.

The basic parameters are set as follows.The constraints for this set of tests, unless otherwise stated, are as follows: Expected output set for the even numbers system: 2, 4, 6, 8, 10, 12, 14, 16; Population count: 4 members, increasing by 1 every generation; Maximum number of steps per system: 50; Maximum number of repetitions per system: 50; Maximum number of generations: 200.

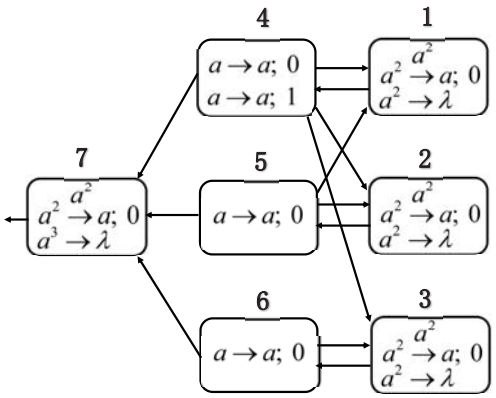


Fig. 8. A SN P system generating all even natural numbers

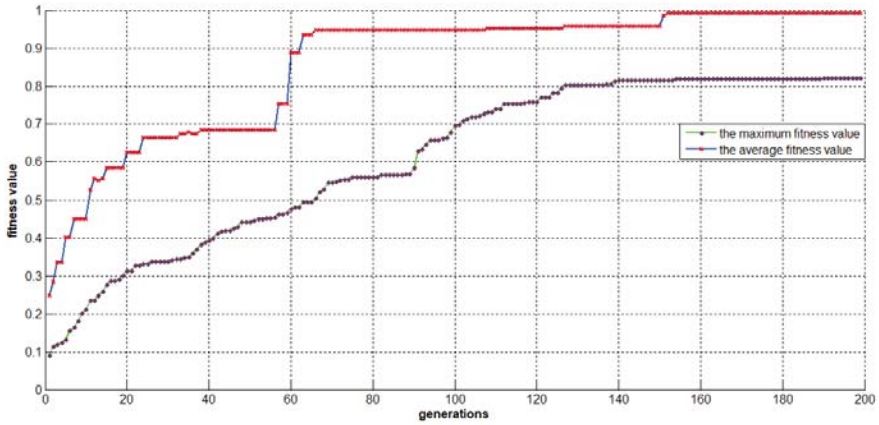


Fig. 9. The change curves of the average fitness values and the maximum fitness values

The results in Fig. 9 and Fig. 10 are the same as those in Fig. 6 and Fig. 7. The greater the number of iterations, the greater the fitness value and the algorithm is convergent. The output results are very close to the expected results. As a result, the expecrimental results of SN P systems generating all natural numbers and SN P systems generating all even natural numbers show that our method is feasible and effective.

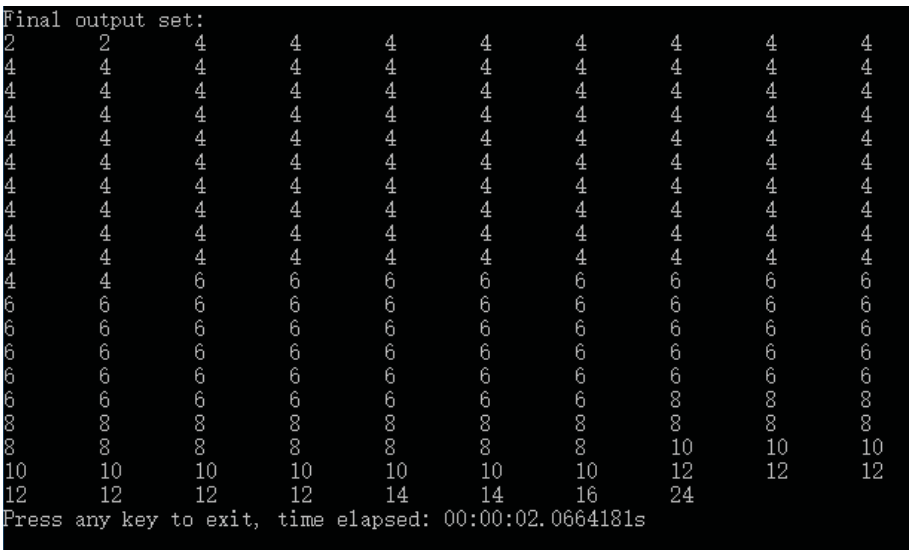


Fig. 10. The output set of SN P systems generating all even natural numbers

5 Conclusions

This paper proposes a clear possibility on how to use genetic algorithms to design an expected SN P system, including the overview approach of the automatic design, the fitness function and the design of elitism, crossover and mutation. After randomly generating a population of SN P systems, the genetic algorithm is used to evolve the initial population to obtain the optimal SN P system. The introduction of genetic algorithm makes it easier to generate new SN P systems than manual design. Moreover, a fitness function, $i.e.$, the comparison between the output set and the expected set, is established to reflect

the precision and sensitivity of SN P systems. The experimental results show that the fitness function designed by this study can effectively guide the search for the optimal SN P system. Finally, the experimental results from two examples show that the approach is effective to automatically design a SN P system based on genetic algorithms. In future works, we will generalize this method to generate more SN P systems, such as generating asynchronous and time-free systems (which are usually quite hard to design manually, but relevant from an application point of view). Moreover, we have only changed the rules in the present approach, while other elements such as the number of neurons and the synapse connections between neurons have not been changed in the process of evolution. Therefore, much attention will be devoted to them in furture.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61972324, 61672437, 61702428), the Sichuan Science and Technology Program (2018GZ0185, 2018GZ0086), New Generation Artificial Intelligence Science and Technology Major Project of Sichuan Province (2018GZDZX0043) and Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06).

References

1. Kari, L., Rozenberg, G.: The Many Facets of Natural Computing. Communications of the ACM, 51(10), 72-83(2008).
2. Wolfram, S.: Statistical mechanics of cellular automata. review of modern physics, 55(3), 601-644(1983).
3. Kazarlis, S., Bakirtzis, A., Petridis, V.: A genetic algorithm solution to the unit commitment problem. IEEE Transactions on Power Systems, 11(1), 83-92(1996).
4. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. IEEE Swarm Intelligence Symposium, 1(1), 33-57(2007).
5. Zhang, G.: Quantum-inspired evolutionary algorithms: a survey and empirical study. Journal of Heuristics, 17(3), 303-351(2011).
6. Zhang, G., Li, N., Jin, W.: Novel Quantum Genetic Algorithm and Its Applications. Frontiers of Electrical and Electronic Engineering in China, 1(1), 31-36(2006).
7. Al-Rifaie, M., John, M., Suzanne, C.: Creativity and Autonomy in Swarm Intelligence Systems. Cognitive Computation, 4(3), 320-331(2012).

8. Farmer, J., Norman, H., Alan, S.: The immune system, adaptation, and machine learning. Physica D, 22(1-3), 187-204(1986).
9. Păun, G.: Membrane Computing. International Symposium on Fundamentals of Computation Theory, 2751(1-3), 284-295(2003).
10. Păun, G.: Introduction to Membrane Computing. Applications of Membrane Computing, 2751(1-3), 1-42(2006).
11. Păun, G., Rozenberg, G.: A guide to membrane computing. Theoretical Computer Science, 287(1), 73-100(2002).
12. Zhang, G.: A membrane algorithm with quantum-inspired subalgorithms and its application to image processing. Natural Computing, 11(4), 701-717(2012).
13. Zhang, G., Pan, L.: A Survey of Membrane Computing As a New Branch of Natural Computing. Chinese Journal of Computers, 2(30), 208-214(2010).
14. Bernardini, F., Gheorghe, M.: Population P systems. Journal of Universal Computer Science, 10(5), 509-539(2004).
15. Ionescu, M., Păun, Gh.: Spiking neural P systems. Fundamenta Information, 71(2), 279-308(2006).
16. Cavaliere, M., Genova, D.: P systems with symport/antiport of rules. Journal of Universal Computer Science, 10(5), 540-558(2004).
17. Martín-Vide, C, Păun, G., Pazos, T.: Tissue P systems. Theoretical Computer Science, 296(2), 295-326(2003).
18. Păun, A., Popa, B.: P systems with proteins on membranes and membrane division. Developments in Language Theory, Lecture Notes in Computer Science, 4036, 292-303(2006).
19. Zhu, M., Zhang, G., Yang, Q., Rong, H., Yuan W., Pérez-Jiménez, M.: P systems-based computing polynomials with integer coefficients: design and formal verification. IEEE transanctions on nanobioscience, 17(3), 272-280(2018).
20. Zhang, G., Gheorghe, M., Pan, L., Pérez-Jiménez, M.: Evolutionary membrane computing: A comprehensive survey and new results. Developments in Language Theory, Inf. Sci., 279, 528-551(2018).
21. Zhang, G., Cheng, J., Wang, T., Wang, X., Zhu J.: Membrane Computing: Theory and Applications. Beijing, China: Science Press, 2015.
22. Suzuki, Y., Tanaka, H.: chemicial evolution among arificial proto-cells. International Conference on Artificial Life 2000, USA, 54-63(2000).
23. Escuela, G., Gutiérrez-Naranjo, M.: An application of genetic algorithms to membrane computing. Proceedings of the 10th Brainstorming Week on Membrane Computing, 101-108(2010).
24. Tudose, C., Lefticaru, R., Ipate, F.: Using genetic algorithms and model checking for P systems automatic design. Studies in Computational Intelligence, 387, 285-302(2012).

25. Huang, X., Zhang, G., Rong, H., Ipate, F.: Automatic Design of Cell-like P Systems through Tuning Membrane Structures, Initial Objects and Evolution Rules. International Journal of Unconventional Computing, 9(5), 425-443(2012).
26. Ou, Z., Zhang, G.: Automatic Design of Cell-like P Systems through Tuning Membrane Structures, Initial Objects and Evolution Rules. International Journal of Unconventional Computing, 9(5), 425-443(2013).
27. Chen, Y., Zhang, G., Wang, T., Huang, X.: Automatic design of celllike P systems through tuning membrane structures, initial objects and evolution rules. Chin. J. Electron., 23(2), 302-304(2014).
28. Zhang, G., Rong, H., Ou, Z., Pérez-Jiménez, M., Gheorghe, M.: Automatic design of deterministic and non-halting membrane systems by tuning syntactical ingredients. IEEE Trans. Nanobiosci., 13(3), 363-371(2014).
29. Yuan, W., Zhang, G., Pérez-Jiménez, M., Wang, T., Huang, X.: P systems based computing polynomials: Design and formal verification. Natural Comput., 15(4), 591-596(2016).
30. Zhang, G., Rong, H., Neri, F., Pérez-Jiménez, M.: An optimization spiking neural P system for approximately solving combinatorial optimization problems. International Journal of Neural Systems, 24(5), 01-16(2014).

Bi-level multi-objective optimization of loss and waste in the wheat processing

Wanying Liang, Hua Yang *, and Kang Zhou

Department of Math and Computer,Development Strategy Institute of reserve of food and material,Wuhan Polytechnic University, Wuhan 430023, Hubei, China{lwy000208@163.com,Huay20@163.com}

Abstract. Wheat, as one of the main processed grain in food industry, with the development of science and technology, the development of wheat processing industry has entered a period of large-scale. However, in order to meet the visual and taste requirements of wheat processing technology, but also to improve the market share of grain products, grain processing enterprises often over-process grain, resulting in unnecessary waste. Therefore, considering that there is still a lot of room for improvement in wheat processing technology, this paper uses the relevant knowledge of information technology and other fields to establish a mathematical model by analyzing the loss rate, yield rate and wheat nutrient loss rate of wheat processing links, and to measure and evaluate the loss and waste of grain processing links. On this basis, the data sets of 8 processes, such as feeding, screening, stone removal, wheat purification, grinding, flour making, powder blending, packaging, are optimized by double layer multi-objective optimization, hierarchical analysis, and a reasonable optimization scheme is put forward.

Keywords: Wheat processing technic ,Metro-logical evaluation,Double-layer Multi-objective Optimization.

1 Foreword

With the improvement of living standards, consumers put forward higher requirements for the fineness of grain. In order to meet the market demand, grain processing enterprises often regard flour and rice with good raw products and good taste as the "first priority", and the processing standards even far exceed the national standards, resulting in a great waste of grain.

Wang [1] studied the food quality and nutrition quality of wheat and wheat processing technic. The appropriate processing technic, flour extraction rate and the extraction scheme were the necessary conditions for the reasonable processing of wheat. Zhao [2] studied excessive flour processing, which not only increased the production cost of enterprises, but also reduced the quality of some noodle products: the one-sided pursuit of refined flour production and consumption, not only caused great loss of beneficial nutrients in wheat, but also lost the real taste

* Corresponding author.

of people's staple flour. Wang [3] in order to improve the overall utilization rate of wheat flour production equipment and wheat flour processing efficiency, based on the research on the production process of a single production line of a wheat flour enterprise, the production related data was collected, combined with the application of system modeling and simulation technology, the processing flow model of wheat flour single production line was established by using Flexsim simulation software, and the simulation experiment was carried out. Through the analysis of simulation data and many simulations, the improvement scheme is put forward to optimize the model, and the overall utilization rate of the equipment is obviously improved. The research conclusions can provide a reference for this kind of wheat flour processing enterprises to a certain extent. In 2015, Zhao [4] , on the basis of defining the evaluation object of grain postpartum loss and waste, according to the characteristics of China's grain postpartum system, adopted objective weighting method to set up index weight, selected six grain varieties, such as rice, wheat, corn, soybean, peanut, rapeseed and so on, based on harvesting, drying, farmers storing grain, storage, transportation, processing, selling and consumption and so on. The evaluation index system of grain postpartum loss and waste was constructed in eight links, and the calculation method of each layer index was given. In the same year, Li [5] studied the processing technic and operation management in the production of special wheat flour. In the process of wheat flour making, skillfully controlling the operation skills of each system, and whether the operation of each grinding system is scientific and reasonable, is directly related to the quality of special wheat flour and the ratio of high quality flour, and is an important measure and guarantee to achieve the best milling effect. Zhao [6] is of great significance to ensure food security in China. Under the background of the rigid growth of grain demand and the increasing difficulty of increasing grain production, the occurrence mechanism of grain postpartum loss is analyzed. On the basis of this analysis, the control measures are put forward from the aspects of constructing modern grain circulation system, improving storage and logistics facilities, guiding enterprises to process moderately, promoting comprehensive utilization of by-products and straightening out grain price to guide residents' rational consumption, so as to reduce grain loss and waste and promote the effective utilization of resources. In 2018, based on the present situation of grain loss and waste at home and abroad, Chen [7] analyzed the causes of grain processing loss and waste in China, and put forward some measures and suggestions. At present, the development of wheat flour processing industry in China has not kept up with the economic development speed, which restricts the development and growth of wheat flour processing industry in China. Wang [8] analyzed and studied the development course and technological process of wheat flour making, and discussed the development prospect of wheat flour processing technic. Jiang [9] described wheat general processing and high-tech processing respectively, and briefly summarized the differences between wheat general processing and high-tech processing. For the future development trend of wheat processing, full automation, green and deep processing are the three main directions.

For the above years of study, there is indeed the problem of loss and waste caused by excessive processing [10] in the process of wheat processing technic. In this paper, based on the loss and waste of 8 processes of wheat processing technic, the investigation data simulation model of output rate, loss rate and nutrient loss rate of grain processing link is established, and the principle of econometric evaluation system is established. According to the loss rate of wheat processing link, the index set of the measurement and evaluation system of the production rate, the measurement and evaluation of the loss investigation of grain processing link is carried out. Finally, the double-layer multi-objective optimization model is constructed, and the optimization scheme is obtained by analyzing the double-layer multi-objective optimization data set of wheat processing link.

2 Simulation Model for investigation of loss and waste in Grain processing

2.1 Model analysis

For a long time, the consumer pays little attention to the nutrition problem of the refined grain, and more focuses on the appearance and the taste of the refined grain. For this reason, in order to meet the needs of people's visual and taste, and to improve the market share of the refined grain, the grain processing enterprises often over-process the grain, resulting in unnecessary waste. In fact, over-processing of food will not only result in a low yield, but also a significant loss of nutrition in the food. The relevant government agencies should supervise the production of the enterprises and support the proper processing of the grain by the enterprises. How do you know that the processing of a certain enterprise is over-processed? We need to set up a simulation model to understand the overall processing technic and processing level of this grain processing enterprise. The postnatal loss waste simulation model of the grain processing link is composed of three parts, which are: an investigation data simulation model of the yield rate of the grain processing links, an investigation data simulation model of the loss rate of the grain processing links, and an investigation data simulation model for the loss of nutrients in grain processing links. In the following, the general simulation model of the loss of the grain processing links is introduced, and the three parts of the simulation model are introduced.

2.2 Model Established

General Simulation Model of the Investigation on the Loss of the Grain Processing links We assume that the sample set of grain processing link loss and waste survey data from the same variety, scale, region and processing technic equipment comes from a whole. Therefore, we need to establish a simulation model of postpartum loss and waste of grain processing link for the same variety, scale, region and processing technic equipment sample set. The steps of establishing the simulation model of postpartum loss and waste in grain processing links are as follows:

(1) The data set of grain processing loss and waste is classified according to different varieties, different scales, different regions and different processing technic and equipment.

(2) For each kind of data set, first of all, the data is preprocessed: after careful screening, the data points that leave the group are removed. Then, the index set that needs to be simulated is calculated or extracted from the preprocessed data set. Finally, the modeling is carried out. Set the index set that needs to be simulated is $P_j(j = 1, 2, 3, \cdots, s)$. P_j denotes the index of the process of the $j(j = 1, 2, 3, \cdots, s)$ step, $P_j^{(i)}$ denote the $i(i = 1, 2, \cdots, t)$ sample of the step processing data.

According to the principle of minimum total difference (sum of distances) between the data of each sample point and the index points that need to be simulated, the model is established as follows:

$$min \sum_{i=1}^{t} \sqrt{\sum_{j=1}^{s} (P_j - P_j^{(i)})^2} \tag{1}$$

Output variables: indicator set is $P_j(j = 1, 2, 3, \cdots, s)$

Input variables: preprocessed sample data is $P_j^{(i)}(j = 1, 2, \cdots, s)(i = 1, 2, \cdots, t)$

Simulation Model of Survey data for production rate of Grain processing links Set a_k represents the $k(k = 1, 2, \cdots, t)$ sample which collected, in the following form

$$a_k = f(a_{k1}, a_{k2}, a_{k3}, a_{k4}, \overrightarrow{a_{k5}}) \tag{2}$$

Where, a_{k1} represents variety of $k - th$ samples, a_{k2} represents regions of $k - th$ samples,a_{k3} represents production enterprise scale of $k - th$ samples,a_{k4} represents type of production equipment of $k - th$ sample,$\overrightarrow{a_{k5}}$represents the output rate corresponding to each production process of $k - th$ sample, it's a vector. For the collected samples, after classification, in order to describe them simply, for a class of samples $\overrightarrow{a_{k5}}$ recorded as C_k,in the form of.

$$C_k = (c_{k1}, c_{k2}, \cdots, c_{ks}) \tag{3}$$

In which,c_{kj}represents the yield of the $k-th$ step of the $k-th$ sample,establishing model.

$$min \sum_{k=1}^{t} \sqrt{\sum_{j=1}^{s} (c_j - c_{kj})^2} \tag{4}$$

Output variables: index set of output rate for each process is $c_j(j = 1, 2, 3, \cdots, s)$.

Input variables: preprocessed sample data is $C_k = (c_{k1}, c_{k2}, \cdots, c_{ks})$.

In the following, from the data of the pre-processed grain sample, the data simulation model of the yield rate of the grain processing link is used, and the corresponding database and simulation system are established, through systematic calculation, the output rate of grain processing links in different provinces and countries under different equipment and scale is obtained.

Simulation Model of investigation data for loss rate of Grain processing links Set a_k represents the $k(k = 1, 2, \cdots, t) - th$ sample of the collection, in the following form.

$$a_k = f(a_{k1}, a_{k2}, a_{k3}, a_{k4}, \overrightarrow{a_{k5}}) \tag{5}$$

Of which, a_{k1} represents variety of $k - th$ samples, a_{k2} represents regions of $k - th$ samples, a_{k3} represents production enterprise scale of $k - th$ samples, a_{k4} represents type of production equipment of $k - th$ sample, $\overrightarrow{a_{k5}}$ represents the loss rate corresponding to each production process of $k - th$ sample, it's a vector. For the collected samples, after classification, in order to describe them simply, for a class of samples $\overrightarrow{a_{k5}}$ recorded as L_k, in the form of.

$$L_k = (l_{k1}, l_{k2}, \cdots, l_{ks}) \tag{6}$$

Of which, l_{kj} represents the yield of the $j-th$ step of the $k-th$ sample,establishing model.

$$min \sum_{k=1}^{t} \sqrt{\sum_{j=1}^{s} (l_j - l_{kj})^2} \tag{7}$$

Output variables: index set of loss rate for each process is $l_j(j = 1, 2, 3, \cdots, s)$.

Input variables: preprocessed sample data is $L_k = (l_{k1}, l_{k2}, \cdots, l_{ks})$.

In the following, from the data of the pre-processed grain sample, the data simulation model of the loss rate of the grain processing link is used, and the corresponding database and simulation system are established, through systematic calculation, the investigation and simulation data of the output rate and loss rate of grain processing links in different provinces and countries under different equipment and scale is obtained.

Investigation data Simulation Model of nutrient loss rate in Grain processing links Set a_k represents the $k(k = 1, 2, \cdots, t) - th$ sample of the collection, in the following form.

$$a_k = f(a_{k1}, a_{k2}, a_{k3}, a_{k4}, a_{k5}, \overrightarrow{a_{k6}}) \tag{8}$$

Of which, a_{k1} represents variety of $k - th$ samples, a_{k2} represents regions of $k - th$ samples, a_{k3} represents production enterprise scale of $k - th$ samples, a_{k4} represents type of production equipment of $k - th$ sample,a_{k5} represents the type of nutrients of $k - th$ sample,$\overrightarrow{a_{k6}}$ represents the corresponding loss rate of a certain nutrient in each production process of the $k - th$ sample, it's a vector. For the collected samples, after classification, in order to describe them simply, for a class of samples $\overrightarrow{a_{k6}}$ recorded as Y_k, in the form of.

$$Y_k = (y_{k1}, y_{k2}, \cdots, y_{ks}) \tag{9}$$

Of which, y_{kj} represents the yield of the$j-th$ step of the $k-th$ sample,establishing model.

$$min \sum_{k=1}^{t} \sqrt{\sum_{j=1}^{s} (y_j - y_{kj})^2} \tag{10}$$

Output variables:index set of certain nutrient loss rate in each process is$y_j(j = 1, 2, 3, \cdots, s)$. Input variables: preprocessed sample data is $Y_k = (y_{k1}, y_{k2}, \cdots, y_{ks})$.

3 Measurement and Evaluation of loss and waste investigation in Grain processing links

Based on the obtained simulation data, under the conditions of different varieties, different scales, different regions and different processing technology and equipment, an evaluation system is established to evaluate the loss and waste of different processing links and different technological processes, to compare the sample data with the simulation data, and to establish the evaluation table of the production condition of the enterprise.

3.1 Principles for establishing a measurement and evaluation index system for loss and waste in grain processing links

In every link of grain processing, in order to pursue fineness and palatability, processing enterprises over-process grain, resulting in serious loss and waste of grain processing link. In order to reduce the loss and waste in the process of grain processing, we need to investigate and study the loss and waste, and establish a measurement and evaluation system in order to accurately grasp the loss and waste of each link of grain processing.

In order to achieve the goals of the highest output rate, the smallest loss rate, the smallest nutrient loss and the smallest energy consumption, it is necessary to classify the waste data set of grain processing links according to different varieties, different scales, different regions and different processing technology and equipment. Then for each kind of data set, the data is preprocessed: after careful screening, the data points that leave the group are removed. Then the index set that needs to be simulated is calculated or extracted from the preprocessed data set. Based on the principles of scientific, maneuverability and dynamics of evaluation index selection, and according to the principle of minimum total difference (sum of distance) between the data of each sample point and the index point that needs to be simulated, an index system composed of index set of output rate, loss rate and nutrient loss rate of each link of grain processing is established by sample set to evaluate different processing links and different equipment. Different process flow and other loss and waste changes, determine the advantages and disadvantages.

3.2 Measuring and evaluating index system of loss and waste in grain processing links

According to the idea of establishing the index system of measurement and evaluation, the index system of grain processing links includes:

(1) Evaluation index set of loss rate of each process is $l_j(j = 1, 2, 3, \cdots, s)$.

(2) Evaluation index set of certain nutrient loss rate in each process is $y_j(j = 1, 2, 3, \cdots, s)$.

3.3 Data sets

In the paper, we have data sets of loss rate and production rate in each link of wheat processing from different regions in china. In the paper we will take the samples under a certain variety, a certain region, a certain scale and equipment as an example, the measurement and evaluation system is used. The difference between the output rate of the corresponding index set and the standard value of the corresponding index set is calculated, and its advantages and disadvantages are evaluated.

4 Optimization model of grain processing link loss and waste investigation

4.1 Double-layer multi-objective optimization model for grain processing links

In order to find out which processing links in the grain processing enterprise is over-processed, and to find the law of the maximum and the minimum loss rate for different production equipment, the two-layer multi-objective optimization model is established.

The first layer, the optimization model of each link of grain processing. Aiming at each link of grain processing, the model is established based on the maximum and the variance of the loss rate, the loss rate of each link is calculated, and the phenomenon of over-processing in which link is judged, in order to optimize the grain processing process. Second, the optimization model of grain processing scale and processing equipment. Starting from different dimensions, the model is established by using the principle of maximum loss rate and maximum variance, the loss rate of each link under different processing scale and different production equipment is calculated, and the law of loss rate in processing process is found out. Set a_k represents the $k(k = 1, 2, \cdots, t) - th$ sample of the collection, in the following form.

$$a_k = f(a_{k1}, a_{k2}, a_{k3}, a_{k4}, l_k^{(j)}) \tag{11}$$

Of which, a_{k1} represents variety of $k - th$ samples, a_{k2} represents regions of $k - th$ samples, a_{k3} represents production enterprise scale of $k - th$ samples, a_{k4} represents type of production equipment of $k - th$ sample, $l_k^{(j)}$represents the loss rate of the$j - th$ procedure of the $k - th$sample, $l^{(j)}$ represents the average loss rate of the $j - th$ procedure for all samples, $p^{(j)}$ represents variance of loss of the $j - th$ procedure for all samples, $\lambda_i (i = 1, 2)$ represents weight, and $\sum_{i=1}^{2} \lambda_i = 1$.

Established model:$max\{\lambda_1 \frac{l^{(j)}}{\sum_{j=1}^{s} l^{(j)}} + \lambda_2 \frac{p^{(j)}}{\sum_{j=1}^{s} p^{(j)}}\}$.

Of which:$l^j = \frac{1}{t} \sum_{k=1}^{t} l_k^{(j)}$,$p^j = \sum_{k=1}^{t} (l_k^{(j)} - l^{(j)})^2$

Output variable: the process that results in the greatest loss due to over-processing of grain

Input variables: preprocessed sample data is $a_k = (a_{k1}, a_{k2}, a_{k3}, a_{k4}, l_k^{(j)})$

4.2 Double-layer multi-objective optimization data set and optimization scheme for wheat processing links

According to Table 1, the data set of the first layer of wheat, the proportion of the influencing factors of the wheat loss, the proportion of the proportion of the eight processing processes of the wheat, the maximum of the screening and the packaging loss, 43.7% and 40.7%, respectively, the net wheat accounts for 6.9%,the powder blending ratio is 6.0%, the powder-making ratio is 1.1%, the stone-removing ratio is 1.2%, the grinding ratio is 0.5%, and the feeding ratio is 0%.

Table 1. Optimal data set for the first layer of Wheat

	Feeding	Screening	Stone removal	Wheat purification	Grinding	Flour making	Flour blending	Packaging
Average value	0	0.127766	0.006538	0.02796	0.0024779	0.005528	0.028004	0.123602
Variance	0	0.028119	0.000247	0.002957	0.000079	0.0002532	0.001936	0.025282
Average value dimensionless	0	0.396942	0.020312	0.086866	0.0076982	0.0171744	0.087003	0.384005
Variance dimension -less	0	0.477604	0.004203	0.050229	0.0013433	0.0043013	0.032891	0.429428
Average variance value after weighting	0	0.437 273	0.012258	0.068547	0.0045208	0.0107378	0.059947	0.406717

Table 2. Optimal data set for the second layer of wheat for large scale

	Feeding	Screening	Stone removal	Wheat purification	Grinding	Flour making	Flour blending	Packaging
Equipment	Large scale							
Average value	0	0.1068	0.009605925	0.032215444	0.007285714	0.012785714	0.022684215	0.0925
Variance	0	0.021277117	0.000334409	0.000944903	0.000354905	0.000771201	0.000651283	0.010199016
Average value dimensionless	0	0.376219261	0.033838334	0.113483808	0.025665038	0.045039625	0.0799086	0.325845334
Variance dimension -less	0	0.616141636	0.009683807	0.027362444	0.010277314	0.022332411	0.018859809	0.29534258
Average variance value after weighting	0	0.496180449	0.021761071	0.070423126	0.017971176	0.033686018	0.049384204	0.310593957

Table 3. Optimal data set for the second layer of wheat for medium scale

	Feeding	Screening	Stone removal	Wheat purification	Grinding	Flour making	Flour blending	Packaging
Equipment	Medium scale							
Average value	0	0.116733333	0.006915783	0.02450654	0.000964971	0.000699588	0.043635118	0.158079802
Variance	0	0.02793521	0.000320147	0.001594491	0.00001199	7.34136056	0.003220371	0.052098583
Average value dimensionless	0	0.332067328	0.019673092	0.069712917	0.002745021	0.001990095	0.124127331	0.449684216
Variance dimension -less	0	0.327923731	0.003758116	0.018717286	0.000140719	0.00008617	0.03780305	0.611570919
Average variance value after weighting	0	0.32999553	0.011715604	0.044215101	0.00144287	0.001038137	0.080965191	0.530627568

Table 4. Optimal data set for the second layer of wheat for small scale

	Feeding	Screening	Stone removal	Wheat purification	Grinding	Flour making	Flour blending	Packaging
Equipment	Small scale							
Average value	0	0.14728125	0.004841243	0.02933594	0.00179279	0.006879422	0.015677437	0.104885974
Variance	0	0.034042764	0.000170252	0.005405845	0.00002835	0.000259476	0.001090714	0.007644494
Average value dimensionless	0	0.474039482	0.015582027	0.094420667	0.005770276	0.022142111	0.050459405	0.337586033
Variance dimension -less	0	0.699865077	0.00350012	0.111135567	0.000582845	0.005334413	0.02242335	0.157158629
Average variance value after weighting	0	0.58695228	0.009541073	0.102778117	0.00317656	0.013738262	0.036441377	0.247372331

Table 5. Optimal data set for the second layer of wheat for medium scale for domestic equipment

	Feeding	Screening	Stone removal	Wheat purification	Grinding	Flour making	Flour blending	Packaging
Equipment	Domestic							
Average value	0	0.157757692	0.006382157	0.032242759	0.003583047	0.005938304	0.021147657	0.108307766
Variance	0	0.034269671	0.000218892	0.003896603	0.000112987	0.000336663	0.001019855	0.007744435
Average value dimensionless	0	0.470413834	0.019030798	0.0961439	0.0106842	0.017707285	0.063059684	0.322960298
Variance dimension -less	0	0.719964597	0.004598651	0.081862937	0.002373731	0.007072878	0.021425936	0.162701271
Average variance value after weighting	0	0.595189215	0.011814725	0.089003418	0.006528965	0.012390081	0.04224281	0.242830785

Table 6. Optimal data set for the second layer of wheat for medium scale for unknow equipment

	Feeding	Screening	Stone removal	Wheat purification	Grinding	Flour making	Flour blending	Packaging
Equipment	Un-know							
Average value	0	0.062783333	0.00687517	0.018680793	0.000083333	0.004639056	0.042859683	0.156739226
Variance	0	0.009962878	0.000334736	0.0009537	0.000000083	0.000085395	0.003843715	0.06568899
Average value dimensionless	0	0.214526091	0.023491957	0.063830913	0.000284744	0.015851318	0.146448424	0.535566553
Variance dimension -less	0	0.12319698	0.004139217	0.011793078	0.00000103	0.001055962	0.047529848	0.812283884
Average variance value after weighting	0	0.168861535	0.013815587	0.037811996	0.000142887	0.00845364	0.096989136	0.673925219

From Tables 1-6, we can know the following results.

(1)From the first layer data set of double-layer multi-objective optimization of wheat processing link, it can be seen that: 0.4372>0.4067>0.0685> . . . >0.0045>0, it can be seen that the loss rate of wheat in the process of screening and packaging is particularly large, followed by wheat and flour blending, and the loss rate of other processes is slightly small and basically the same.

(2)From the second layer data set of the two-layer multi-objective optimization of the wheat processing link, we can see that: Sieve process loss rate: small scale > large scale> medium scale.(0.5869>0.3299>0.4961), Domestic equipment > mixed equipment.(0.5951>0.1688) Packing process loss rate: medium scale> large scale > small scale (0.5306> 0.3105 > 0.2473), Mixed equipment > Domestic equipment (0.6739>0.2428) No matter the size or different equipment, the wheat loses seriously in the screening process and the packaging process, and the small and medium scale loses the most in the screening process and the packaging process respectively, so it is more advantageous to choose the large-scale processing. The loss rate of domestic equipment and mixing equipment in screening process and packaging process is the largest, and the loss rate of mixed equipment in packaging process is 0.67%. The loss rate of domestic equipment and mixing equipment is basically the same in the process of wheat purification, but the mixing equipment also reaches 0.09% in the process of powder blending, which is relatively high. It is shown that the waste of screening and packaging process is serious in small and medium-sized processing enterprises when selecting mixed equipment.

(3)It is suggested to develop large-scale wheat processing enterprises and select domestic equipment. We should give priority to the development of large-scale enterprises, improve the production technology and management level of small and medium-sized enterprises, introduce new technologies, improve the equipment, test the output rate in real time, and reduce the loss of finished products in the processing process.At the same time, for the waste of wheat packaging, enterprises should reasonably adjust the proportion of bags in bulk, it is best to carry out large packing at one time before leaving the factory, reduce the times of sub-loading, at the same time, do seamless links between processing plants, warehouses, means of transportation and transit settings, strengthen logistics construction, and reduce the loss of wheat in the process of packaging and transportation.

5 Conclusion

Based on the study of eight processes of wheat processing technic, three simulation models of grain processing yield, loss rate and nutrient loss rate were established in this paper. According to the obtained simulation data, an evaluation system was established under the conditions of different varieties, different scales, different regions and different processing technic and equipment to evaluate the loss and waste of different processing links and different technological processes. Compare the sample data with the simulation data to evaluate the advantages and disadvantages, and establish the enterprise production status evaluation table. Based on the obtained data results, the bi-level multi-objective optimization is carried out. The first data set of the two-layer multi-objective optimization of the wheat processing link infers that the loss rate of the wheat during the processing, the screening and the packaging process is particularly

large, the net wheat and the powder distribution are the second, and the loss rate of the other processes is slightly small and basically comparable. From the second layer data set of double-layer multi-objective optimization of wheat processing link, it is shown that the screening and packaging process is seriously wasted when mixed equipment is selected in small and medium-sized processing enterprises. It is suggested that large-scale wheat processing enterprises should be developed and domestic equipment should be selected. With the introduction of new technology, the improvement of equipment and the waste of wheat packaging, enterprises should reasonably adjust the proportion of bags in bulk, strengthen logistics construction and reduce the loss of wheat in the process of packaging and transportation.

References

1. Wang,X.: Wheat processing technology and wheat flour quality. Grain and Feed Industry,2006(10): 9-12.
2. Zhao,T.: Grain overprocessing = waste+loss of nutrition. Agricultural Machinery,2013(20):17-20.
3. Zhang,B.: Research on data acquisition and optimization algorithm of wheat processing process. Jiangnan University Master thesis,2015(6).
4. Zhao, X., Cao,B.,Zhao, L.: Study on evaluation index system of grain postpartum loss and waste. Grain Science, Technology and economy,2015(6):6-9.
5. Li,L., Li,S.,Wang ,X.:Processing technology and operation management in the production of special wheat flour. Modern Flour Industry,2015(3).
6. Zhao,H.: Analysis on the mechanism and treatment measures of grain loss. Chinese Journal of Agricultural Resources and Regional Planning,2016,37(11):92-98.
7. Chen,Z., Deng,Y., Zhang,S., Hu,L., Wang,X.: Research on the current situation and countermeasure in the problem of grain processing wastes and losses in China grain. Science and Technology and Economy,2018,43(5):96-99.
8. Wang,L.: Study on processing technology of wheat flour. Henan Agriculture,2018(26):45-46.

9. Jiang,D.: The difference between common and high-tech processing of wheat. China Food,2018(24):114-115.
10. Wan,Z.: Excessive processing has resulted in a huge loss of food resources and nutrients.Heilongjiang grain, 2016(1):55-55.