Evolutionary membrane computing: A comprehensive survey and new results

Gexiang Zhang [a], Marian Gheorghe [b], Linqiang Pan [c], Mario J. Pérez-Jiménez [d]

[a] School of Electrical Engineering, Southwest Jiaotong University, Chengdu 610031, PR China
[b] Department of Computer Science, University of Sheffield, Regent Court, Portobello Street, Sheffield S1 4DP, UK
[c] Key Laboratory of Image Processing and Intelligent Control, Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, PR China
[d] Department of Computer Science and Artificial Intelligence, University of Sevilla, Avda. Reina Mercedes s/n, Sevilla 41012, Spain

Keywords:
Membrane computing
Evolutionary computation
Evolutionary membrane computing
Membrane-inspired evolutionary algorithm
Automated design of membrane computing model

A B S T R A C T

Evolutionary membrane computing is an important research direction of membrane computing that aims to explore the complex interactions between membrane computing and evolutionary computation. These disciplines are receiving increasing attention. In this paper, an overview of the evolutionary membrane computing state-of-the-art and new results on two established topics in well defined scopes (membrane-inspired evolutionary algorithms and automated design of membrane computing models) are presented. We survey their theoretical developments and applications, sketch the differences between them, and compare the advantages and limitations.

1. Introduction

Natural computing is a fast growing interdisciplinary field interested in developing concepts, computational paradigms and theories inspired from various natural processes and phenomena. A thorough overview of this field has been recently produced [63]. Membrane computing and evolutionary computations are two nature-inspired theories which are discussed in this paper. Membrane computing (MC) refers to the branch of natural computing that investigates a class of computing models, also called membrane systems or P systems, abstracted from the compartmentalized structure and functioning of biological membranes within a living cell, cell tissues or colonies of cells [55]. The first P system model was introduced by Gheorghe Păun in 1998, and since then the MC research has been continuously and rapidly progressing. There are, basically, three main types of P systems: cell-like P systems, tissue-like P systems and neural-like P systems [55]. In all cases, there are basic components (membranes, cells, neurons, etc.) hierarchically arranged, through a rooted tree, for cell-like P systems, or distributed across a network, like a directed graph, for tissue-like P systems, with a common environment.

☆ This paper is written according to the invited plenary lecture presented at the 2012 Asian Conference on Membrane Computing. GZ acknowledges the support by the National Natural Science Foundation of China (61170016 and 61373047), the Program for New Century Excellent Talents in University (NCET-11-0715) and SWJTU supported project (SWJTU12CX008). MG acknowledges the support of CNCSIS-UE-FISCSU Project Number PNII-IDEI 643/2008. LP acknowledges the support of National Natural Science Foundation of China (61033003, 91130034 and 61320106005).

* Corresponding author. Tel.: +86 028 87601579.
E-mail address: zhgxdylan@126.com (G. Zhang).

Neural-like P systems consider neurons as their cells organized with a network structure as a directed graph. In the past fourteen years, plenty of research results have been achieved at the theoretical level, for example, various variants of P systems with Turing computing power have been developed and polynomial or linear solutions to a variety of computationally hard, NP-complete or PSPACE-complete, problems have been obtained [55]. Applications of MC in various fields, including parallel and distributed algorithms, graphics, linguistics, economy [10] and more recently in systems and synthetic biology [16], have been investigated. To date, the applications with a practical or engineering use are intensively investigated and deserve more attention.

Evolutionary computation (EC) is a branch of natural computing that covers a wide range of problem-solving optimization techniques based on principles of biological evolution, such as natural selection and molecular genetics, and the collective behavior of decentralized and self-organized systems [12]. The main evolutionary paradigms are: genetic algorithm (GA) [24,23], genetic programming (GP) [34–36], evolution strategies (ES) [65,1], evolutionary programming [15], particle swarm optimization [32,53], ant colony optimization [11], differential evolution [60] and quantum-inspired evolutionary algorithm [82]. Each paradigm uses population-based probabilistic search, a powerful tool for broad exploration and local exploitation of the model space [31]. This strategy improves algorithms for local search and other global search algorithms such as simulated annealing [33] and tabu search [22]. Also, their stochasticity improves the heuristic artificial intelligence [46,47] and machine-learning algorithms [44,66]. EC mostly involves meta-heuristic optimization techniques and generally includes two broad areas: evolutionary algorithms (EAs) and swarm intelligence [4]. These techniques are being increasingly widely applied to various problems, ranging from practical applications in industry and commerce to leading-edge scientific research [12]. Amongst various research directions in EC, it is widely acknowledged that the appropriate hybridization of EC and various techniques is very useful to improve the algorithm performance [3,2,82].

MC has the rigor and sound theoretical development for all variants of membrane systems and provides a parallel-distributed framework and flexible evolution rules. While EC has outstanding characteristics, such as easy-understanding, robust performance, flexibility, convenient use for real-world problems and a very large scope of applications. These features suggest the exploration of the interactions between MC and EC, called evolutionary membrane computing (EMC) in this paper. Until now, the possible interplay of MC and EC has produced two research topics: membrane-inspired evolutionary algorithms (MIEAs) and automated design of membrane computing models (ADMCM). On the one hand, MIEAs, initially called membrane algorithms (MA) [50,94], creates a bridge between MC and various real-world applications. This hybrid method is considered as the application with a practical use of MC in computer science [55]. It is worth pointing out that it is more appropriate to use the concept MIEA, instead of MA, to describe this kind of approximate optimization algorithms in this paper due to three reasons: (1) MA is normally associated with the name of the implementation algorithms of MC models, instead of the hybrid optimization algorithms; (2) only a small portion of publications on MIEAs used the term MA; and (3) In *Chapter 19* of the collective paper [21], this class of hybrid optimization algorithms has been generically called MIEA. The automated synthesis of MC models or of a high level specification of them is envisaged to be obtained by applying various EC algorithms. On the other hand, ADMCM aims to circumvent the programmability issue of membrane based models for membrane systems [14,30]. The difference between MIEA and ADMCM can be illustrated by using Fig. 1, where they have different inputs and outputs.

Since the introduction of MC, attention has been paid to EMC. In recent years, research into EMC, a novel and promising interdisciplinary research direction, has become a rapidly expanding field, as shown in Fig. 2. The preparation of this survey paper is motivated by the following points:

1. No survey on EMC has yet appeared in the specialized literature. It is necessary to provide an overview of the state-of-the-art of EMC so as to allow newcomers to the area to obtain a clear understanding of key research problems and developments in this field, including those that are currently under way.
2. An introduction of EMC to the broader scientific community for research purposes is aimed.
3. Some confusing aspects related to the EMC research that appeared in the current literature has to be clarified.

This paper presents an overview of the state-of-the-art of EMC and new results. The main contributions can be summarized as follows:

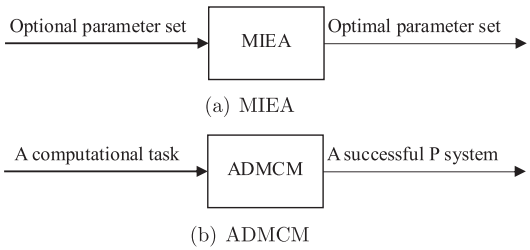

Fig. 1. Difference between MIEA and ADMCM.

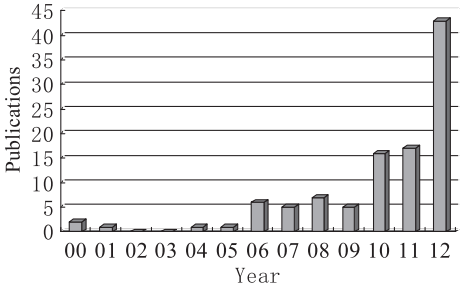

Fig. 2. Growth of publications on EMC.

1. An overview of EMC state-of-the-art on two established topics in well defined scopes, MIEAs and ADMCM, are presented. This is a systematic review of recent efforts to develop a theory of EMC.
2. New research results on MIEAs and ADMCM are provided to show the progress of EMC development.
3. A unified framework is introduced to easily and clearly summarize the EMC investigations on theoretical and application aspects.
4. In this study, we introduce and discuss basic concepts related to EMC, survey theoretical developments and applications, sketch the differences between various EMC variants, and compare the advantages and limitations of the manifold flavors.
5. This work is supplemented with a comprehensive list of pointers to literature on EMC.

The rest of this paper is arranged as follows. Section 2 introduces EMC and provides an overview of the EMC work done so far. Section 3 presents some recently experimental results on EMC. Finally, some conclusions and some possible further developments are discussed in Section 4. It worth noting that the acronyms involved in this paper are listed in Appendix A.

2. Evolutionary membrane computing (EMC)

EMC aims to complement the existing set of MC applications [10,16] with others regarding optimizations, learning problems, based on various meta-heuristics. In this respect, two research topics, MIEA and ADMCM, have been investigated. An MIEA concentrates on developing new variants of meta-heuristic algorithms for solving complex optimization problems by using the hierarchical or network membrane structures, evolution rules and computational procedures utilized by P systems and the methods and well-established techniques employed by EC [94,83]. In various variants of MIEAs, membrane structures have significant effects on their design. Thus we consider the membrane structure as the criterion to classify a variety of MIEA approaches into two groups: hierarchical structure based MIEAs and network structure (NS) based MIEAs. The former is subclassified into four categories: nested membrane structure (NMS), one-level membrane structure (OLMS), hybrid membrane structure (HMS) and dynamic membrane structure (DMS), and the latter is divided into two subcategories: statically network structure (SNS) and dynamically network structure (DNS).

The ADMCM investigations cover three key topics: abstract rewriting systems on multisets (ARSM), parameter optimization of P system models (POPSM) and automatic design of P systems (ADPS). The topics investigated within the EMC framework are shown in Fig. 3. Subsequently, the work reported in the literature on the usage of EMC for different problems is overviewed.

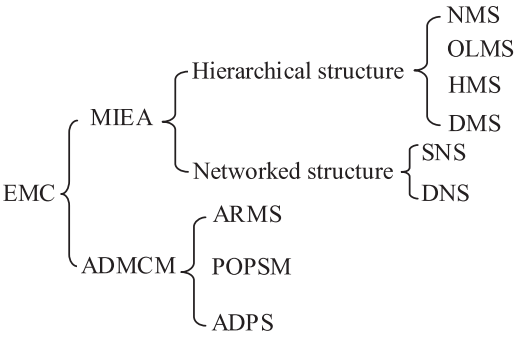

Fig. 3. The unified framework of EMC.

2.1. Membrane-inspired evolutionary algorithms (MIEAs)

MIEAs integrate the membrane structures, evolution rules and computational mechanisms of P systems with the search principles of various meta-heuristics. Until now two classes of membrane structures, the hierarchical structure of a cell-like P system (formally, a rooted tree) and the network structure of a tissue-like P system (formally, a directed graph), have been used to design a variety of MIEAs. In the sequel we will summarize the MIEA with respect to the above structures.

2.1.1. Hierarchical structures

As shown in Fig. 3, the hierarchical structures considered in MIEAs can be grouped into four types: nested membrane structure (NMS), one-level membrane structure (OLMS), hybrid membrane structure (HMS) and dynamic membrane structure (DMS). This subsection will first focus on the first two types and then move to briefly introduce the other ones. Finally a summary on the hierarchical membrane structure based MIEAs is provided.

(1) Nested membrane structure (NMS)

The first version of MIEAs was introduced in [48] and further expounded in [49–51]. For convenience, we call the algorithm membrane algorithms (MA) in this paper. An MA is composed of three components:

(i) An NMS of order $q \geqslant 1$ containing q membranes, which is shown in Fig. 4, where the outermost one is the skin membrane and the innermost one is an elementary membrane, that is, the structure is formally a rooted tree with only one branch;
(ii) Inside the ith region delimited by the ith membrane, $i = 1, 2, \ldots, q-1$, a subalgorithm is running and two tentative solutions of an optimization problem are obtained; inside the innermost region, one can find a subalgorithm and only one tentative solution;
(iii) Solution transporting mechanisms between adjoining regions.

After the initial setting, MA works as follows:

(i) Inside the ith region, the tentative solutions are updated by the subalgorithm associated with the ith region, $i = 1, 2, \ldots, q$.
(ii) The ith membrane ($2 \leqslant i \leqslant q-1$) sends the copies of the better and worse solutions into its adjoining inner and outer regions, respectively. The innermost membrane, membrane q, only collects the copy of the better solution coming from the $(q-1)$th region.
(iii) The next population in each region is constructed by selecting the two best individuals from the solutions left in the region and received from the adjacent regions.
(iv) Steps (i)–(iii) are repeated until the termination condition is satisfied.
(v) The computing result is collected in the innermost membrane.

In MIEAs with NMS, several meta-heuristic approaches were used as subalgorithms. For the convenience of description, this paper introduces the concept of *algorithm in membrane (AIM)* to unify various subalgorithms, algorithm components and an independent algorithm in a membrane into one single concept. In [48–50], a tabu search was considered as AIM to solve traveling salesman problems (TSP). In [52,51], Brownian and genetic algorithms (GA) were applied to design AIM for the solution of TSP and job-shop scheduling problems (JSSP). In [77], a DNA sequence design problem was successfully solved by using GA as AIM. The authors in [38] also used GA and a local search as AIM to design two versions of MIEAs, membrane algorithm for min storage (MA4MS) and MA4MS with a local search (MA4MS LS), to solve min storage problems. It is worth noting that these problems, TSP, JSSP and min storage problems, are well-known NP-hard optimization ones. In [72,79], an MA was constructed by applying NMS and GA to solve function optimization problems and the proton exchange membrane fuel cell model parameter estimation problems. In [97], an improved bio-inspired algorithm based on membrane computing (IBIAMC) was combined with a sequential quadratic programming (SQP) algorithm to design an MA with NMS to solve complex constrained problems and the gasoline blending scheduling problem (GBSP). Additionally the applications of MIEAs with NMS and results regarding the performances of algorithms are listed in Table 5.

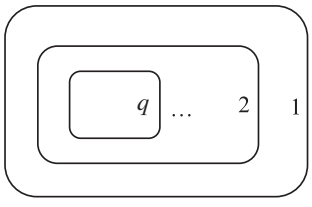

Fig. 4. A nested membrane structure.

Remarks: NMS-based MIEAs were designed by integrating NMS and transport rules of a cell-like P system with several meta-heuristic approaches. AIMs are separated by membranes and communications are performed only between adjacent regions. So this class of algorithms can be easily implemented in parallel, distributed, or grid computing systems. Empirical studies have shown that NMS-based MIEAs have better performance than several traditional approximation techniques, such as simulated annealing (SA) and GA, with respect to a given problem [50,51].

The existing work on NMS mainly considered GA and local search techniques, such as tabu, Brownian and SQP, as AIM. As mentioned in [49], AIM can be designed by using other meta-heuristic approaches such as SA, evolution strategy, particle swarm optimization (PSO); however, some meta-heuristics do not fit the framework, for example, most of differential evolution (DE) algorithms are not suitable for NMS as more than two individuals are needed in the process of evolution. Until now AIM is the same for all the membranes, but one can consider distinct algorithms as well. Experimental investigations in the literature also indicate that the choice of AIM is of great importance to the algorithm performance and the way AIM is chosen, to a large extent, depends on the problem.

(2) One-level membrane structure (OLMS)

The class of MIEAs with OLMS was first presented in [88] when a quantum-inspired evolutionary algorithm (QIEA) was considered as an AIM. OLMS is a special hierarchical membrane structure coming from a cell-like P system. In [88], the P system-like framework consists of a membrane structure, as shown in Fig. 5. The membrane structure has m elementary membranes, labeled as $1, 2, \ldots, m$, placed inside the skin membrane, denoted by 0. The rules are composed of two types: evolution rules in each of the compartments 1 to m which are transformation-like rules for updating an individual according to the evolutionary mechanism of a meta-heuristic approach and communication rules which send the best fit individual from each of the m regions delimited by m elementary membranes into the skin membrane and then the overall best fit individual from the skin back to each region.

The steps for implementing MIEAs with OLMS can be described as follows.

(i) Construct OLMS, as shown in Fig. 5.
(ii) Randomly allocate n individuals of a population into m elementary membranes ($m \leqslant n$) so that there is at least one individual in each elementary membrane. Thus, the number of individuals in an elementary membrane varies from 1 to $n - m + 1$.
(iii) Determine the number of iterations for the subpopulation inside each elementary membrane to independently perform a meta-heuristic approach. To be specific, the number g_i ($i = 1, 2, \ldots, m$) of iterations for the ith elementary membrane is generated randomly between 1 and a certain integer number.
(iv) Inside each elementary membrane, the evolutionary process of the meta-heuristic approach is independently performed.
(v) The communication rules are used to exchange specific information between regions. For instance, the fittest individual from each elementary region is sent into the region delimited by the skin membrane and the overall fittest one within the skin membrane is communicated to each region.

In MIEAs with OLMS, the initial population of individuals is scattered across the membrane structure. The number $n_i, 1 \leqslant i \leqslant m$, of objects for each region is randomly chosen. In any step the current generation is assessed compartment by compartment to select the best fit individual. Every $g_i (1 \leqslant i \leqslant m)$ generations for each compartment, the communication rule is performed once. The process will stop when the best fit solution will remain unchanged for several generations (we must fix in advance that number).

OLMS has been combined with several meta-heuristics, such as QIEA [88], PSO [99,75,76,58], GA [96,78], differential evolution (DE) [8], ant colony optimization (ACO) [87], quantum shuffled frog leaping algorithm [17] and quantum particle swarm optimization [18], to design various variants of MIEAs for solving various problems. In [88,87,91,94,83], three classes of well-known NP-hard problems, single-objective knapsack problems, satisfiability problems and traveling salesman problems, were used as application examples to verify the algorithm performance. In [89], OLMS was applied to construct a multi-objective optimization algorithm to solve multi-objective knapsack problems. In [42,99,8,13,6,78,75,76], benchmark functions were used to test the introduced methods. Moreover, MIEAs with OLMS were also used to solve some real-world applications: radar emitter signal processing [94,8,99,43], digital filter design [42], broadcasting problems of P systems (BPoPS) [93], controller design [74], GBSP [96], image segmentation [73,58], image decomposition [87] and spectrum allocation problems of cognitive radio systems [17,18]. It is worth noting that the membrane algorithm with quantum-inspired

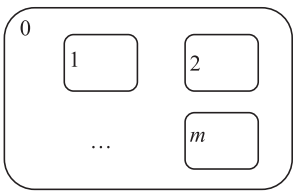

Fig. 5. A one level membrane structure.

subalgorithms (MAQIS) in [87] used multiple distinct components of QIEA, instead of identical subalgorithms, as AIM in different compartments. The applications of MIEAs with OLMS and their results obtained are listed in Table 5.

Remarks: OLMS is different from the star topology defined in [81]: which has a center node and therefore communications are performed only between each sub-node and the center node; the communication in a star topology is a local process. OLMS has no center node and the communication is usually a global process and it can be executed between any two or more elementary membranes. OLMS can be extended to a dynamic structure with active membranes, such as membrane division and dissolution [84]. Thus a star topology might be regarded as a special case of OLMS. The investigations on OLMS, especially the comparative analysis of dynamic behaviors of quantum-inspired evolutionary algorithms based on P systems (QEPS) with its counterpart algorithm, QIEA, in [90,86], show that OLMS-based MIEAs have better optimization performance than their counterpart approaches because of their improved capacity of balancing exploration and exploitation, which is derived from their better balance between convergence and diversity. As compared with NMS, OLMS usually has much smaller number of membranes and cross-membrane communications and therefore is more suitable for a parallel distributed implementation, and this is something to be considered by further research. NMS is a linear topology structure, in which the communication or information exchange between adjacent regions is a local process. According to the comparative experiments conducted on knapsack problems in [88], NMS is inferior to OLMS when a QIEA is used as AIM. The OLMS approach is used in some other contexts [6,73], but the authors seem not to be aware about its original definition provided in [88].

The use of OLMS approach in MIEA is in its initial stage and further investigation should clarify better its role. Some further research avenues can be:

(i) Apart from QIEA, DE, PSO and ACO, some other meta-heuristic approaches could be used as AIM to design approximate optimization algorithms.
(ii) Further investigations should be conducted on OLMS-based MIEAs and their counterpart methods to better reveal the role of a P system in the hybrid optimization algorithms.
(iii) More real-world problems, such as multi-objective or constrained complex problems, should expand the application areas of OLMS-based MIEAs.
(iv) Implementations of these algorithms onto parallel distributed hardware platforms are expected to reveal the improved performance of them.
(v) So far each region is associated with the same AIM, but different algorithms can be used across the regions of the P system structure. The first study is reported in [87].

(3) Hybrid membrane structure (HMS)

HMS, as the name suggests, could be achieved by combining various features. In [95,26,67,27], HMS was constructed by putting two NMS inside the skin membrane, where AIM was designed by using GA or PSO and rules consisting of transformation, communication, rewriting, splicing and/or uniport. In [57], HMS was designed by using m NMS, each of which has two membranes, inside the skin membrane. In this work, differential evolution (DE) was used as AIM and transformation and communication rules are adopted. The two types of HMS could be thought as the hybridization of NMS and OLMS. In [98], two layers of OLMS were applied to design the membrane structure. In [81], NMS and a star topology were combined with DE to establish two variants of MIEAs, respectively.

HMS-based MIEAs were applied to solve four sorts of problems. In [95,26,57,98], benchmark functions were discussed. In [26,27], optimal controllers for a marine diesel engine and a time-varying unstable plant were designed by applying a single-objective and a dynamic multi-objective MIEA, respectively. In addition, an image segmentation problem was solved in [57]. These applications are shown in Table 5.

Remarks: HMS is a very flexible membrane structure because there are many possibilities of building a model for such an approach, but at the same time it is a complex and hard to construct model due to difficulties in getting a good intuition behind selecting its features and formulating reasonable explanations for the choices made. It is obvious that HMS is more complex than NMS and OLMS. Further work on HMS could be focused on convincing comparisons of HMS-based MIEAs with their counterpart algorithms, theoretical analysis and comparisons with NMS and OLMS.

(4) Dynamic membrane structure (DMS)

DMS is a membrane structure that changes with the number of iterations in the process of evolution. In [25,41,39,40], the membrane structure was constructed at the beginning by placing a certain number of membranes inside the skin membrane; as the number of iterations increases, all the membranes inside the skin membrane merge into one membrane by using the merge rule of a P system and then the membrane inside the skin membrane is divided into a certain number of membranes by applying the division rule of a P system; the two processes of merging and division are repeated until the computation halts. This use of the dynamic structure is different as show below. In [25], the initial membrane structure and the one obtained through recurrent division operations consist of a certain number of NMS contained in the skin membrane. While in [41,39,40], DMS is a changing OLMS, i.e., the initial membrane structure and further divided regions are all elementary membranes inside the skin membrane, whereas the result of merging the regions is an elementary membrane inside the skin membrane. In [80], the initial membrane structure used a star topology and when a membrane is dissolved another one is divided and overall the number of regions is kept constant. In this structure, communication, export, import, dissolution and division rules were used. In [84], the DMS of a P system with active membranes was used to design an

Table 1
Comparisons of MIEAs with hierarchical structures.

Types	Structures	Similarities	Differences	Features	Suggestions
NMS	Fig. 4		Linear topology	Simple, easy realization in parallel	Systematic analysis, parallel implementation, more applications
OLMS	Fig. 5	Derived from a cell-like	Connected membranes and star topologies	Simple, easy realization in parallel	Extension, parallel implementation, more applications
HMS		P system	No predefined topology	Complex, diverse	Systematic analysis, more applications
DMS			Variable topology	Complex, Variable	Systematic analysis, more applications

approximate optimization algorithm, where the number of elementary membranes inside the skin membrane varies with the number of iterations in a non-deterministic way by using membrane separation and merging rules. At each iteration, the number of elementary membranes inside the skin membrane is variable. In these circumstances, three meta-heuristic approaches were considered. In [25,39,80], GA was used as AIM. In [40], cellular automata and chaotic search are combined to design AIM. While in [41,84], AIM was designed with QIEA. In the skin membrane of DMS in [84], a local search, tabu search, was applied to finely tune the current best solution.

Applications of DMS-based MIEAs mainly focused on S-boxes in cryptographic algorithms [80], radar emitter signal analysis [41], benchmark functions [39,25,40] and satisfiability problems [84]. Table 5 summarizes these applications.

Remarks: The dynamic structure of the underlying P system is the key difference between DMS and previously presented methods, namely NMS, OLMS and HMS. This feature might have a direct influence on the population diversity of the algorithms involved. The diversity has, on the other hand, an impact on the algorithms performance as the dynamic structure will help getting a balance between exploration and exploitation. Especially for a multi-objective problem, the better the population diversity is, the more Pareto solutions can be obtained. It seems that DMS might be beneficially used in solving a multi-objective optimization problem, however, this needs to be further analyzed and more solidly proved. The potential of DMS over NMS, OLMS and DMS has to be further investigated in order to be confirmed.

(5) Summary of MIEAs with hierarchical structures

This subsection summarizes the four types, which are reported in the literature, of hierarchical membrane structures derived from a cell-like P system. The comparisons between NMS, OLMS, HMS and DMS based MIEAs are drawn in Table 1, where similarities, differences, features and suggestions for further work are summed up. The hierarchical membrane structure of a cell-like P system can be used in many other ways and variants of MIEAs for solving various problems can be devised. One might also think at using other types of evolution rules that appear normally in cell-like P systems.

2.1.2. Network structures

Except for the hierarchical membrane structure based MIEAs, the other types of MIEAs are inspired from network structure (NS) of a tissue-like P system, as shown in Fig. 3. NS is divided into two categories: SNS and DNS, which are reviewed below. Finally the network membrane structure based MIEAs are summarized.

(1) Static network structure (SNS)

Two types of SNS were reported in the literature. In [28,29], SNS was constructed by applying seven cells[1] to solve multi-objective optimization problems. In this structure, three cells are used for different single objective optimizations and three cells are applied for optimizing all objectives simultaneously; the seventh cell collects the results; the channels between the cells were prescribed and the communication rule of a tissue-like P system was applied; AIM was designed by using GA. The other variant of SNS was designed by placing five cells in a common environment. In this method, fully connected channels between the five cells were established to fulfill the communications; each cell contains a representative and widely used variant of a differential evolution (DE) algorithm; each DE algorithm independently evolves inside one cell according to its own evolutionary mechanism and at the same time communicates with other cells through channels; transformation and communication rules of a tissue P system are considered.

Four types of problems were solved by applying SNS based MIEAs. In [28,29], multi-objective benchmark functions were used to test the algorithm performance. In addition, simulated moving bed and controller design were considered as application problems in [28] and in [29], respectively. In [85], twenty-one manufacturing parameter optimization problems (MPOP) with various types of constraints were discussed to show the advantages of the presented algorithm over twenty-two optimization algorithms recently reported in the literature. The applications are listed in Table 5.

Remarks: A tissue-like P system provides numerous variants of SNS, but only two types were considered in the design of MIEAs in the literature. More possible SNS could be used to construct more MIEAs. Except for GA and DE, many other meta-heuristics might be considered to be AIM. Moreover, the applications of SNS-based MIEAs need to be expanded. More deeply, a systematic analysis on SNS-based MIEAs is of great importance to explore how to more appropriately combine a tissue-like P system with various meta-heuristics. For instance, what is the reason of introducing SNS like in [28,29]? It is worth

[1] The authors used the concept *membrane*, instead of *cell*.

pointing out that SNS could be suitable for a small-scale network as its computational complexity increases rapidly when the problem size grows.

(2) Dynamic network structure (DNS)

As a special tissue-like P system, a population P system has a dynamic membrane structure. In [92], DNS was designed with three cells for organizing three types of QIEAs, where communications between cells are performed at the level of genes, instead of the level of individuals reported in the existing MIEAs in the literature. If the communication between any pair of cells will be performed, a channel can be dynamically built. Extensive experiments conducted on knapsack problems and a real-world application, a distribution system reconfiguration problem in a power system, show the effectiveness and superiority of the introduced approach. The applications are shown in Table 5.

Remarks: This research direction is very new and promising because the communication channel will be dynamically established if necessary in this structure and therefore could be extended to a complex structure with a number of cells to solve high dimensional, multi-modal optimization problems. As compared with SNS, DNS is more suitable for integrating many distinct meta-heuristic approaches. Additionally, systematic analysis, more variants of MIEAs and more applications need to be investigated.

(3) Summary of MIEAs with NS

The comparisons between SNS and DNS based MIEAs are drawn in Table 2, where similarities, differences, features and suggestions for further work are summarized. Further work could be focused on parallel implementation, deep analysis, more algorithms and more applications.

2.1.3. Summary of MIEAs

As a hybrid approximate optimization algorithm integrating P systems with meta-heuristic approaches, MIEA provides a fertile research direction with a well-defined scope, a set of open questions, and therefore further studies are necessary to prove the usefulness of P systems-based approaches for real-world applications. This direction is the only one with a practical use in computer science and aims to explore the great potential of membrane computing as a distributed processing mechanism [61,55]. The summarization of the MIEA work is shown in Table 5, where the first column lists the abbreviations for various algorithms studied in the literature; the third and fourth columns highlight the main membrane structure (MS) flavors and the meta-heuristic approaches applied inside the membranes or cells, respectively; the last two columns sum up the problems MIEAs have been applied to and the comparative results provided in the corresponding literature. Furthermore, the classification of the MIEA applications in terms of the problem characteristics is summarized as shown in Table 6. Future research questions may address issues regarding various membrane structures, systematic analysis of the roles of a P system in a MIEA, novel combinations of the MC and EC, and the exploration of problem-oriented or meta-heuristic-oriented membrane structures.

2.2. Automated design of membrane computing models (ADMCM)

To circumvent the programmability issue of membrane-based models for complex systems, ADMCM is envisaged to obtain the automated synthesis of membrane computing models or of a high level specification of them by applying various meta-heuristic search methods. In this respect, the work done in the literature could be classified into three categories: ARSM, POPSM and ADPS. They will be reviewed in the sequel.

2.2.1. Work on ADMCM

(1) Abstract rewriting system on multisets (ARSM)

The preliminary ADMCM work can be dated back to the early 2000. Suzuki and Tanaka made the first attempts [69,70,68] to introduce a genetic programming approach into artificial cell systems via a cell-like P system model called ARSM, a rewriting membrane computing model consisting of a multiset of symbols, a set of rewriting rules (reaction rules) and membranes arranged in a hierarchical structure (a rooted tree). ARMS are particular types of P systems and exhibit complex behaviors such as non-linear oscillations. In ARSM, the rewriting rule promoting a reaction was regarded as an enzyme and the evolutionary mechanism of genetic programming was applied to evolve the system. The fitness of an enzyme was defined as the number of steps to reach the solution. Thus the best enzyme that can solve a problem within the smallest number of steps can be obtained. ARSM were used to model and analyze an ecological system in [68]. This investigation looks very natural and it is expected that some further continuations will appear.

Table 2
Comparisons of MIEAs with NS.

Types	Similarities	Differences	Features	Suggestions
SNS	Derived from a tissue-like P system	Static structure	Suitable for a small-scale network	Systematic analysis, parallel implementation, more applications
DNS		Dynamic structure	Suitable for a large-scale network	Extension, parallel implementation, more applications

(2) Parameter optimization of P system models (POPSM)

More recently, new research on applying evolutionary algorithms to estimate the structure and parameters of a system in cell systems biology modeling based on P systems has been developed. In [62,5], a nested evolutionary algorithm (NEA) was introduced to optimize the automated design of biological models comprising the optimization of the model structure and its stochastic kinetic constants. The models were built by using stochastic P systems. A distributed evolutionary algorithm (DEA) is composed of two layers, both of which are realized as GA. The outer layer evolves the model structure by combining different modules taken from a predefined module library, while the inner layer fine-tunes the parameters of the model, i.e., the associated stochastic kinetic constants. In [62], three case studies including molecular complexation, enzymatic reactions and autoregulation in transcriptional networks were discussed. In [5], four case studies of increasing complexity including negative and positive autoregulation as well as two gene networks implementing a pulse generator and a bandwidth detector were considered to test the algorithm performance. Moreover, the way the evaluation functions are constructed was also discussed. In [19,64], the task was to find the optimal parameters for a P system model. In [19], it is discussed a continuous backward problem and the comparisons of five real-valued parameter optimization algorithms. In [64], it is applied PSO to optimize the parameters of a MC model of a synthetic autoinducer-2 (AI-2) signalling system in genetically engineered *Escherichia coli* bacteria. The model is a non-deterministic in silico model of Autoinducer-2 Quorum Sensing formalized by using P systems. These investigations prove the suitability of the use of EAs to design P system models in the field of executable biology by tuning their structure and parameters.

(3) Automatic design of P systems (ADPS)

This research direction is to obtain a successful P system for fulfilling a specifically computational task or solving a problem by applying an automated method. This is a quite complex problem as it involves a great number of parameters (structures, rules and objects) and numerous semantic constraints associated with membrane systems. To date, ADPS is focused on the use of EAs to automatically design a cell-like P system. In general, the design problem can be described as follows:

Formulating a computational task, automatically produce a computational model, in the case a P systems – this can successfully fulfill the task, which is intuitively illustrated in Fig. 1(b). More specifically, a cell-like P system Π for a given computational task can be described as

$$\Pi = (V, \mu, W, R, i_o)$$

where

- V is an alphabet of objects;
- μ is a hierarchical membrane structure with m ($m \geqslant 1$) membranes labeled by the elements of a given set $H, H = \{0, 1, \ldots, m-1\}$, and the skin membrane is labeled as 0;
- W is the vector of initial multisets $w_0, \ldots, w_{m-1}$ over V associated with the regions $0, 1, \ldots, m-1$ delimited by μ, i.e., $W = \{w_0, \ldots, w_{m-1}\}$;
- R is the set of evolution rule sets $R_0, \ldots, R_{m-1}$ associated with the regions $0, 1, \ldots, m-1$ of μ, i.e., $R = \{R_0, \ldots, R_{m-1}\}$;
- i_o is the output membrane of Π.

In this system, V is predefined; μ, W and R need to be designed; i_o = 0, i.e., the output result is inside the skin membrane. More precisely, the problem we aim to solve is the automatic design of a P system, by using an EA approach, i.e.:

Considering a family Π of P systems, $\Pi = \{\pi_i\}_{i \in N}$, where N is the set of natural numbers and each P system π_i has parameters μ, W and R, where μ, W and R are tuned by an EA. W and R, coming from the alphabet V, are generated in the process of design. Finally, the best P system for successfully solving the given computational task will be obtained. It is worth pointing out that these P systems can work in deterministic or non-deterministic manner. The design problem is solved by applying the following steps:

Step 1 Designing the membrane structure μ: a hierarchical membrane structure with m membranes is considered in the cell-like P systems.

Step 2 Definition of an alphabet V: As usual we choose a certain number of letters from English alphabet so that it covers the needs of the initial objects w_i and evolution rules $R_i, i = 0, 1, \ldots, m-1$.

Step 3 Designing the evolution rule set R: The evolution rule set R is obtained by using an EA, where the maximal number of evolution rules in R_i, i.e. the length of R_i, and the types of evolution rules need to be considered.

Step 4 Designing the initial object set W: The initial objects inside each membrane w_i ($i = 0, 1, \ldots, m-1$) are obtained by using an EA, where the maximal number of initial objects inside each membrane w_i ($i = 0, 1, \ldots, m-1$) need to be prescribed.

Step 5 Designing an EA: This step has to consider three points: (1) an encoding technique for membrane structure μ, evolution rule set R, and initial object set W; (2) a fitness function for evaluating a candidate P system; (3) the choices of an EA and its parameter setting.

In this direction, the work started from a simplified version, i.e., considering a family Π of P systems, $\Pi = \{\pi_i\}_{i \in N}$, where each P system π_i has a pre-defined μ and W, but R is obtained by using an optimization approach from a prescribed set of redundant rules $\Re$. In [14], a GA was used for finding a simple P system for calculating 4^2, where the membrane structure μ and initial object set W are settled and the genetic evolution only corresponds to the selection of the subset of rules. A population of P systems was considered and two genetic operators, crossover and mutation, performed the evolution of

the population. This study was extended to a more general P system which computes n^2 in [30] by introducing a QIEA, where the set of rules were encoded by a binary string and evolutionary operations (quantum-inspired gate (Q-gate) update) were performed on genotypic individuals (quantum-inspired bits (Q-bits)), instead of phenotypic individuals (binary bits) or evolution rules of P systems. This approach avoids the difficulty of designing evolutionary operators in the phenotypic space, such as crossover and mutation ones. In [71], several P systems, such as the one generating a language $a^{2^n}b^{3^n}$, were designed by applying different sets of rules and several evaluation methods for a candidate P system were discussed. In [45], the P system computing 4^2 was designed by applying a clonal selection algorithm. In [7], it is presented an automatic design method of a cell-like P system for performing five basic arithmetic operations including addition, substraction, multiplication, division and power. These investigations concentrated on how to design a redundant rule set $\Re$ and how to select a proper subset R from it. As usual the set $\Re$ was built by referring to some related papers and by using experts' experiences.

The further work on ADPS has been extended in [54] by considering three types of design problems for calculating 4^2 as follows:

(i) ***Type 1***: Considering a family Π of P systems, $\Pi = \{\pi_i\}_{i\in N}$, where each P system π_i has a pre-defined μ, and variable W and R which are obtained by using a GA from the alphabet V and from a prescribed set of redundant rules $\Re$, respectively.

(ii) ***Type 2***: Considering a family Π of P systems, $\Pi = \{\pi_i\}_{i\in N}$, where each P system π_i has a pre-defined μ, and variable W and R, where both W and R are obtained by using a GA from the alphabet V. It is worth pointing out that R is generated in the process of design.

(iii) ***Type 3***: Considering a family Π of P systems, $\Pi = \{\pi_i\}_{i\in N}$, where each P system π_i has variable μ, W and R which are attained by using a GA. W and R coming from the alphabet V are generated in the process of design.

It can be seen from the studies above that automatic design of P systems (ADPS) has produced the first step and many problems remain to be addressed. For instance, how to extend the third case in [54] from 4^2 to a more complex computation or the solution of an NP-hard problem, how to encode a P system and how to evaluate a P system are other specific issues to be addressed.

2.2.2. Summary of automated design of membrane computing models (ADMCM)

The three categories, ARSM, POPSM and ADPS, are compared and shown in Table 3, where similarities, differences, features and suggestions of further work are listed. Subsequently the main work on ADMCM is generalized as shown in Table 7.

3. New results on MIEAs and ADMCM

This section will present some experimental results to show some recent progresses on MIEAs and ADMCM.

Table 3
Comparisons of three ADMCM categories.

Types	Similarities	Differences	Features	Suggestions
ARSM	Design of cell-like P systems	Evolving artificial cell systems	Related to executable biology	Extension, more applications
POPSM		Optimization of structures and parameters	Related to executable biology	Reduction of computing time, design of more complex systems
ADPS		Design of three elements: membrane structure, objects and evolution rules	Flexible, possible to design various P systems	Extension, semantics, designing real-world systems

Table 4
Tasks, evaluation functions and successful P systems for eight cases.

No.	Task	Evaluation function $f(t) = f(t-1)+$	Successful P systems
1	$\{2(n-1)\|n \geqslant 1\}$	$\|n_c - 2(t-1)\|$	$R_{11}, R_{12}, R_{13}, R_{14}, R_{15}, R_{16}$
2	$\{2n-1\|n \geqslant 1\}$	$\|n_c - 2t + 1\|$	$R_{21}, R_{22}, R_{23}, R_{24}, R_{25}, R_{26}$
3	$\{n^2\|n \geqslant 1\}$	$\|n_c - n_b^2\| + \|n_b - t\|$	$R_{31}, R_{32}, R_{33}, R_{34}, R_{35}, R_{36}, R_{37}, R_{38}, R_{39}$
4	$\left\{\frac{1}{2}\left[(n-1)^2 + (n-1)\right] \middle\| n \geqslant 2\right\}$	$\left\|n_c - \left\{\frac{1}{2}[(t-1)^2 + (t-1)]\right\}\right\|$	$R_{41}, R_{42}, R_{43}, R_{44}$
5	$\left\{\left[(n-1)^2 + (n-1)\right] \middle\| n \geqslant 2\right\}$	$\|n_c - (t-1)^2 - (t-1)\|$	R_{51}, R_{52}, R_{53}
6	$\left\{\left[(n-1)^2 + 2^n + 2\right] \middle\| n \geqslant 1\right\}$	$\|(t-1)^2 + 2^t - 2 - n_c\|$	R_{61}, R_{62}, R_{63}
7	$\left\{a^{2^n}b^{3^n} \middle\| n > 1\right\}$	$\|n_a - 2^{t-1}\| + \|n_b - 3^{t-1}\|$	$R_{71}, R_{72}, R_{73}, R_{74}, R_{75}$
8	$\{\frac{1}{2}(3^n - 1)\|n \geqslant 1\}$	$\|n_b - \frac{1}{2}(3^t - 1)\|$	$R_{81}, R_{82}, R_{83}, R_{84}$

Table 5
Summarization of the MIEA work ('>' means better than). The acronyms are referred to Appendix A.

Methods	Refs.	MS	AIM	Rules	Problems	Compared results
MA	[48–50]	NMS	Tabu search	Transport	TSP	>GA,SA,TPSA,ACO,NN
	[51]	NMS	Brownian + GA	Transport	TSP	
	[52]	NMS	Brownian + GA	Transport	JSSP	
	[77]	NMS	GA	Transformation, communication	DNA sequence design	>MOEA, QCSEA
MA4MS, MA4MS-LS	[38]	NMS	GA	Transport	Min Storage problems	>greedy,Min,Max,MaxMinMax, MinMaxMin, MaxMinMaxMin, MinMaxMinMax, Best Fit
HBS	[97]	NMS	IBIAMC + SQP	Transformation, communication	Functions, GBSP	
BIPOA	[79]	NMS	GA	Transformation, communication	Functions, parameter estimation	>PSOPS,GA,RGA,HGA,SGA
BCMC	[72]	NMS	GA	Transformation with catalyst, communication	Functions	>MC,GA
QEPS	[88,90,86]	OLMS	QIEA	Transformation, communication	Knapsack problems	>bQIEAo,bQIEAm,bQIEAn, bQIEAcms, QEPS with NMS
	[43]	OLMS	QIEA	Transformation, communication	Radar emitter signals	>QIEA, Greedy algorithm
	[91]	OLMS	QIEA	Transformation, communication	SAT	>QIEA
MAPS	[42]	OLMS	QIEA + LS	Transformation, communication	Functions, digital filter design	>rQIEA,NQGA,GA
MOMA	[89]	OLMS	QIEA	Transformation, communication	Knapsack problems	>HLGA,NPGA,VEGA,NSGA, SPEA,QIEA
PSOPS	[99]	OLMS	PSO	Transformation, communication	Functions, radar emitter signals	>PSO
	[74]	OLMS	PSO	Transformation, communication	Controller design	>PSO,Z-N,SGA, dsDNA-MC
MQEPS	[94]	OLMS	QIEA	Transformation, communication	SAT, radar emitter signals	>QEPS,QIEA,Greedy algorithm
ACOPS	[83]	OLMS	ACO	Transformation, communication	TSP	>ACO, Nishida's MA
DEPS	[8]	OLMS	DE + Simplex method	Transformation, communication	Functions, radar emitter signals	>DE, Greedy algorithm
HPSOPS	[93]	OLMS	HPSOWM	Transformation, communication	BPoPS	>HPSOWM, GA
MDPS	[13]	OLMS	PSO	Transformation, communication	Functions	
MCCOP	[6]	OLMS	GA	Transformation, communication	Functions	>HCVEGA,DMS-PSO, SMES
BIAMC	[96]	OLMS	GA	Communication, transition, abstraction	GBSP	>IEA,OGA,OEGA,BLXGA, UEGA, GA
	[75]	OLMS	PSO + Neighborhood search	Communication, transition	MPOP	>Coello00,Coello02,QPSO,CA
HMEA	[76]	OLMS	PSO	Communication, transition	Functions	>CRGA,PSO,ABC
MA	[73]	OLMS	GA	Transformation, communication	Image Segmentation	>Otsu,GA-Otsu,P-Otsu, Ksw,GA-Ksw, P-Ksw
MCOA	[78]	OLMS	GA	Transformation, communication	Functions	
MQSFL	[17]	OLMS	QSFL	Transformation, communication	spectrum allocation	>GA,PSO,QGA,CSGC
MQPSO	[18]	OLMS	QPSO	Transformation, communication	spectrum allocation	
MA	[58]	OLMS	PSO	Transformation, communication	image segementation	>GA,PSO,ICM
MAQIS	[87]	OLMS	Distinct components of QIEA	Transformation, communication	Knapsack problems, image decomposition	>QEPS,QRG1,QRG2, QRG3,QRG4,QRG5
MA	[95]	HMS	GA	Transformation,	Functions	>SGA,DE,FADE

Table 5 (*continued*)

Methods	Refs.	MS	AIM	Rules	Problems	Compared results
				communication		
	[57]	HMS	DE	Transformation, communication	Image Segmentation	>PSO,GA
MCBPSO	[67]	HMS	PSO	Transport	Parameter selection of LS-SVM	>PSO, CPSO, PSOPDE, CBPSO
DNAMC	[26]	HMS	GA	Rewriting, splicing, uniport	Functions, controller design	GA, PSO, H.T.Toivonen, SA, CA, CHAT, IEA, OEGA, UEGA, TEGA, BLXGA, BOA, OGA
DMOAP	[27]	HMS	GA	Transformation, communication	controller design	
DEA	[81]	HMS	DE	Transformation, communication	Functions	
MOBIAMC	[98]	HMS	GA	Transformation, communication	Functions	>NSGA-II,SAEA,DNAMOGA
DAIMC	[80]	DMS	GA	Communication, export, import, assimilation, division	S-boxes	
RQEPS	[41]	DMS	rQIEA	Transformation, communication, mergence, division	Radar emitter signals	>QEPS, Greedy algorithm
MOMC	[39]	DMS	GA	Communication, division, dissolution	Functions	>SPEA2,PAES,NSGA-II
EMA	[40]	DMS	cellular automata & chaotic search	Communication, division, dissolution	Functions	>GA,PSO,DE
QEAM	[84]	DMS	QIEA + tabu search	Transformation, communication, division, mergence	SAT	>QIEA,QEPS
PMOA	[25]	DMS	GA	Communication, mergence, division	Functions	>FFGA,HLGA,NPGA,NSGA, SOEA,VEGA
TPS algorithm	[28]	SNS	GA	Transformation, communication	Functions, Simulated Moving Bed	>IMOEA,NSGA,NSGA2,SPEA, SPEA2
	[29]	SNS	GA	Transformation, communication	Functions, controller design	>FFGA,HLGA,NPGA,NSGA, SOEA,VEGA,SPEA
DETPS	[85]	SNS	Distinct DE variants	Transformation, communication	MPOP	HIHC,GAIS,GAINM,TLBO, M-ES,PESO,CDE,CoDE, ABC, (μ+λ)-ES,UPSO,CPSO, PSO-DE,ABCA,MDE,MA-MDE, rand/1,rand/2,best/1, best/2, CTB/1, MDE-IHS
PSMA	[92]	DNS	Distinct QIEA variants	Transformation, communication	Knapsack problems, DSR	>QIEA02,QIEA04,QIEA07, QIEA08,QEPS,ACS,HPSO, IIGA,VSHDE,ACO,SA + TS, MTS,PSO

3.1. Recent results on MIEAs

The investigations on various variants of MIEAs, as displayed in Table 5, show that this class of hybrid optimization approaches has good performance, however, since the first version of MIEAs was introduced in 2004, the question about the role of a P system in an MIEA is reconsidered. Here we attempt to explore additional insights into the MIEA performance by experimentally investigating the balance capability between population diversity and convergence.

In the experiment, quantum-inspired evolutionary algorithm based on P systems (QEPS) and its counterpart quantum-inspired evolutionary algorithm (QIEA), are used to solve a knapsack problem with 800 items. The description of the knapsack problem appears in Appendix B. Population diversity considers two measures: Hamming distance (D_{hbw}) between the best and worst phenotypic individuals (binary solutions) in a population and mean Hamming distance (D_{hm}) of all phenotypic individuals in a population. The convergence is evaluated by using two measures: the best genotypic individual (Q-bit individual) convergence (C_{qb}) and the best fitness convergence (C_{fb}). D_{hbw} and D_{hm} are described as

$$D_{hbw} = \sum_{i=1}^{m} (x_{bi} \oplus x_{wi}) \tag{1}$$

$$D_{hm} = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left\{ \sum_{k=1}^{m} (x_{ik} \oplus x_{jk}) \right\} \tag{2}$$

Table 6
Application-based reference classification. The acronyms are referred to Appendix A.

Problems	Numeric	Combinatorial	Single-objective	Multi-objective
TSP		[48–51,83]	[48–51,83]	
Benchmark functions	[38,81,28,25,29,42,95,99,96, 39,40,8,13,6,78,72,98]	[97]	[38,81,42,95,99,96,40,8,97, 13,6,78,72]	[28,25,29,39,98]
Knapsack problems		[88–90,87,92,86]	[88,90,87,92,86]	[89]
Radar emitter signals	[43,99,41,94,8]		[43,99,41,94,8]	
Controller design	[29,26,27,74]		[26,74]	[29,27]
SAT		[91,94,84]	[91,94,84]	
Simulated Moving Bed	[28]			[28]
S-boxes		[80]	[80]	
JSSP		[52]	[52]	
Digital filter design	[42]		[42]	
Parameter selection of LS-SVM	[67]		[67]	
GBSP		[96,97]	[96,97]	
MPOP	[85,75]	[85,75]		
Image decomposition	[87]		[87]	
Image Segmentation	[57,58,73]			
	[57,58,73]			
DNA sequence design		[77]	[77]	
DSR	[92]		[92]	
BPoPS		[93]	[93]	
Parameter estimation of PEMFCM	[79]		[79]	
Spectrum allocation problem		[17,18]	[17,18]	

Table 7
Summarization of the ADMCM work ('>' means better than).

Methods	References	Main work	Compared results
ARSM	[69,70,68]	Developing artificial cell systems by using genetic programming	
POPSM	[62]	Optimizing the structure and parameters of stochastic P systems by using a NEA	
	[5]	Optimizing the structure and parameters of stochastic P systems by using a NEA	
	[19]	Optimizing P system model parameters by using evolutionary based search algorithms	>CMA-ES, DE, ODE,GA, VNS-ECs
	[64]	Adjusting parameters of a MC model by applying PSO	
ADPS	[14]	Evolving the 4^2 P system by using GA to select evolution rules	
	[30]	Evolving the n^2 P system by using QIEA to select evolution rules	>GA in [14]
	[71]	Evolving the n^2 and $a^{2^n}b^{3^n}$ P systems by using GA to select evolution rules	>GA in [14] and QIEA in [30]
	[45]	Evolving the 4^2 P system by using clonal selection algorithm to select evolution rules	
	[9]	Evolving a P system by using GA based on a trace	>GA in [14]
	[7]	Evolving five arithmetic P systems by using QIEA to select evolution rules	
	[54]	Evolving the 4^2 P system by using GA to tune membrane structures, initial objects and evolution rules	

where x_{bi} and x_{wi} are the ith bits in the best and worst phenotypic solutions, respectively; m is the number of bits in a phenotypic solution; n is the number of individuals in a population; the symbol '$\oplus$' is exclusive OR operator; x_{ik} and x_{jk} are the kth bits in the ith and jth phenotypic solutions, respectively. The values of D_{hbw} and D_{hm} vary between 0 and m. Larger values of D_{hbw} and D_{hm} indicate more varieties between the best and worst phenotypic individuals, and each pair of phenotypic individuals in a population, respectively.

Convergences C_{qb} and C_{fb} are depicted as

$$C_{qb} = \frac{1}{m}\sum_{j=1}^{m} \max\{|\alpha_{bj}|^2, |\beta_{bj}|^2\} \tag{3}$$

$$C_{fb} = \max_{i\in\{1,2,\ldots,n\}} f_i(x) \tag{4}$$

where $[\alpha_{bj}\beta_{bj}]^T$ is the jth Q-bit in the best genotypic individual corresponding to the best fitness in a population; m is the number of Q-bits in a Q-bit individual. $f_i(x)$ is the fitness of the ith individual. The best Q-bit individual convergence C_{qb} in a population is used to observe how much Q-bits approach 0 or 1 in the searching process. $0.5 \leqslant C_{qb} \leqslant 1$.

The changes of the two population diversities, D_{hbw} and D_{hm}, in the evolution are shown in Fig. 6. The changes of C_{qb} and C_{fb} are shown in Fig. 7. In these figures, the comparisons of QEPS and QIEA for the knapsack problem with 800 items are illustrated. Each subfigure in Figs. 6 and 7 provides results of 30 independent random runs (green solid lines for QEPS and cyan solid lines for QIEA) and the mean values over 30 runs (black bold solid lines for QEPS and black bold dash-dot lines for QIEA). The values of D_{hbw}, D_{hm}, C_{qb} and C_{fb} in Figs. 6 and 7 are obtained by setting population size to 20 and the maximal number 20,000 of function evaluations (NoFE) as the stopping condition.

From the results, shown in Figs. 6 and 7, regarding the comparison between population diversity and convergence of QEPS and QIEA, we can draw the following conclusions:

(i) The Hamming distance D_{hbw} between the best and worst binary individuals and the mean Hamming distance D_{hm} of all binary individuals show a clear picture of QEPS having a greater potential to preserve the population diversity than QIEA. More specifically, QEPS and QIEA have almost the same initial values and decreasing values for changes of D_{hbw} and D_{hm} with respect to the number of function evaluations (NoFE), but QEPS maintains a much higher level of population diversity than QIEA throughout the evolutionary process. The two subfigures in Fig. 6 also demonstrate that QIEA loses population diversity too fast and very quickly goes down to a small value close to 0 when NoFE reaches values around 10,000, which implies that the individuals in QIEA become nearly identical and therefore lose exploration capability. When NoFE increases to 20,000, QEPS still has one-fourth of initial values of D_{hbw} and D_{hm}.

(ii) It can be seen from the results shown in Fig. 7(a) that QIEA converges much faster in Q-bit space than QEPS and quickly arrives at the maximal value 1, which implies that no further improvement of solutions in QIEA can be gained at the second half of evolutionary processes. The drastic convergence easily makes QIEA trapped in local extrema and consequently a premature end of the evolutionary process appears. On the contrary, C_{qb} of QEPS go up much slower than those corresponding to QIEA with respect to NoFE and finally mount up to around 0.9 for values of NoFE in the region of 20,000, which suggests that the solutions can be further improved if more NoFE is provided.

(iii) In Fig. 7(b), QIEA has faster increase of C_{fb} than QEPS and then stays at a relatively flat level after a certain NoFE, while QEPS goes through a slower start than QIEA and then rapidly goes beyond QIEA and keeps an ascending trend. Thus QEPS obtains better solutions than QIEA. Additionally, it is worth noting, according to the results in Fig. 7(b), that QEPS has better performance than QIEA in terms of the consistency of the results obtained for 30 independent runs when mean values are considered. This suggests that QEPS has better robustness properties than QIEA.

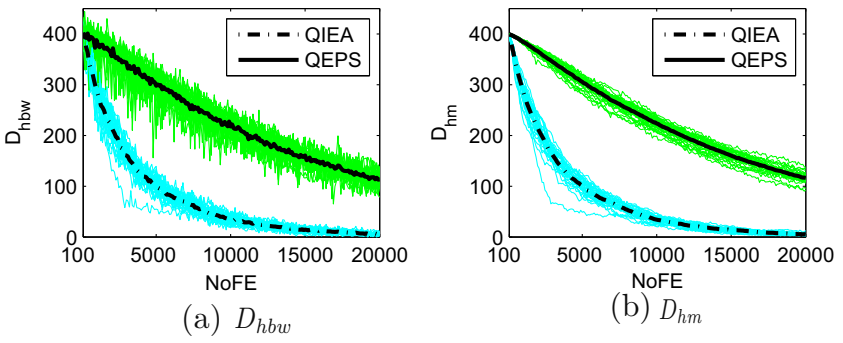

Fig. 6. Population diversity comparison of QEPS and QIEA.

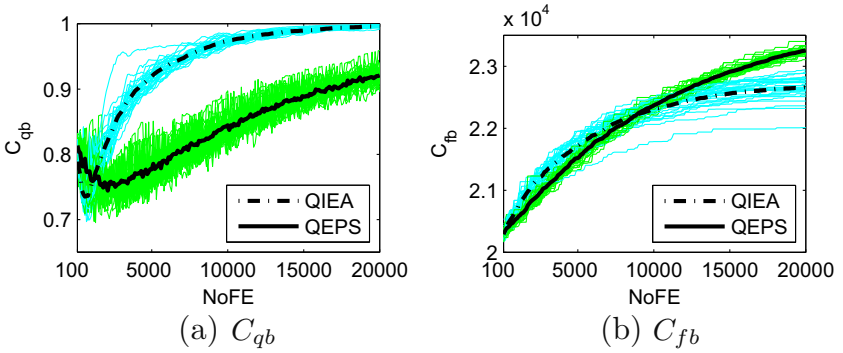

Fig. 7. Convergence comparison of QEPS and QIEA.

The convergence and population diversity are often conflicting features for population-based search methods. Rapid convergence usually results in a fast loss of population diversity, whereas better varieties of individuals produce more possibilities to improve solutions. The loss of population diversity means that the algorithm will fail to further explore the solution space. The diversity and convergence analysis above indicate that QEPS can achieve a better trade-off between convergence and diversity than QIEA, i.e., better balance between exploration and exploitation than QIEA. This better balance of these two essential features of any evolutionary approach is the principal explanation of the fact that QEPS achieves high quality solutions, better than QIEA if NoFE is large enough. For example, NoFE is greater than 10,000 for the knapsack problems with 800 items, which corresponds to 100 evolutionary generations.

3.2. Recent results on ADMCM

In [14], given a membrane structure and initial multisets, a redundant rule set $\Re_1$ with 18 evolution rules was constructed to design a P system for performing a simple mathematical problem 4^2 by using a genetic algorithm and finally one successful P system was obtained. In [30], given the same membrane structure, initial multisets and the redundant rule set $\Re_1$ as in [14], seven successful P systems were achieved by introducing a binary encoding technique; and a modified version, $\Re_2$, of $\Re_1$, was used to design a P system for performing n^2. $\Re_2$ also consists of 18 evolution rules and finally 12 successful P systems were gained. In [71], given a membrane structure and initial multisets, a redundant rule set $\Re_3$, differing from $\Re_2$, with 18 evolution rules was constructed to design a P system for calculating n^2 and finally two successful P systems were obtained; another redundant rule set $\Re_4$ with 18 evolution rules was applied to design a P system for computing $a^{2^n}b^{3^n}$. These studies seem to indicate that a specific redundant rule set might be constructed for fulfilling the design of P systems for a specific computational task, such as $4^2, n^2$ or $a^{2^n}b^{3^n}$. However, we can find out from these studies that there are some common rules in the four rule sets, $\Re_1, \Re_2, \Re_3$ and $\Re_4$. It is natural that we think of such a problem:

For a class of problems with some common goals – in this case it is about computing some integer values through known formulas, like power or exponential – we might have some models, they might have as expected some common set of rules, we do not know whether these models are complete, optimal or redundant and we aim to extract solutions, i.e., complete models which are also optimal.

This problem is important when we aim to solve similar problems and might have some algorithms for them, but it is not clear how good these are.

In what follows, we consider eight computational tasks, shown in Table 4, each of which has successful P systems, i.e., solutions, with a certain number of evolution rules. These solutions shall be non-minimal, maybe incomplete or redundant. The solutions are provided by non-halting systems and after n steps, or something related to n, we get a solution. We can combine their evolution rules into one redundant rule set $\Re$ to produce better solutions. This process lead to $\Re$ with 48 rules.

$$\Re = \left\{ \begin{array}{lll}
r_1 : [s]_2 \to bcx & r_2 : [s \to ab]_2 & r_3 : [s]_2 \to abc \\
r_4 : [s \to cx]_2 & r_5 : [s \to abc]_2 & r_6 : [s \to abx]_2 \\
r_7 : [s \to abxc]_2 & r_8 : [s \to a^2bx]_2 & r_9 : [s]_2 \to abcx \\
r_{10} : [a \to a^2]_2 & r_{11} : [a \to a^2x]_2 & r_{12} : [a]_2 \to [\,]\, a \\
r_{13} : [a]_2 \to [\,]\, b & r_{14} : [a \to ab]_2 & r_{15} : [b \to b^3]_2 \\
r_{16} : [b \to b^3x]_2 & r_{17} : [b \to ab^2]_2 & r_{18} : [b]_2 \to [\,]\, b \\
r_{19} : [b]_2 \to [b]c & r_{20} : [b]_2 \to [\,]\, a & r_{21} : [c]_2 \to c \\
r_{22} : [c \to ab]_2 & r_{23} : [x]_2 \to x & r_{24} : [x]_2 \to cx \\
r_{25} : [x]_2 \to [\,]\, abc & r_{26} : [x]_2 \to [x^2]c^2 & r_{27} : [x]_2 \to \lambda \\
r_{28} : [a \to b]_1 & r_{29} : [a \to bc]_1 & r_{30} : [a \to ab]_1 \\
r_{31} : [a \to cx]_1 & r_{32} : [a \to ac]_1 & r_{33} : [a \to bx]_1 \\
r_{34} : [a \to ax]_1 & r_{35} : [a \to abc]_1 & r_{36} : [a \to bcx]_1 \\
r_{37} : [b \to x]_1 & r_{38} : [b \to bc]_1 & r_{39} : [b \to a]_1 \\
r_{40} : [b \to bc^2]_1 & r_{41} : [b \to ab]_1 & r_{42} : [b \to bc^2x]_1 \\
r_{43} : [x \to bcx]_1 & r_{44} : [x \to ab]_1 & r_{45} : [x \to ac]_1 \\
r_{46} : [x \to a]_1 & r_{47} : [x \to xc]_1 & r_{48} : [x \to bc]_1 \\
r_{49} : [x \to bx]_1 & r_{50} : [x \to ax]_1 &
\end{array} \right\}$$

We consider a family of P systems $\Pi = (O, H, \mu, w_1, w_2, R, i_o)$, where $\mu = [[]_2]_1; w_1 = \emptyset$ and $w_2 = s; O = \{a, b, c, x, s\}$; $H = \{1, 2\}; i_o = 1; R$ is a subset of $\Re$ and is decided by using an optimization algorithm.

We use QIEA in [30] to design six deterministic and non-halting P systems for computing all even numbers, all odd numbers, the square of a natural number and three arithmetic operations, respectively, and two non-deterministic and halting P systems for generating a language and calculating an arithmetic operation. They are labeled as Nos. 1–8, as shown in Table 4,

where the evaluation function for each case is also presented. In this table, t is the current number of steps; n_a, n_b and n_c are the numbers of the objects a, b and c in the output membrane i_o, respectively. In the experiments, the population size, the number of Q-bits and the maximal number of iterations in QIEA are assigned as 30, 50 and 500, respectively. The independent runs are 100. The successful P systems obtained for each case are listed in Table 4, where $f(0) = 0$ and the P systems R_{ij} (i and j represent the index of the eight cases and the index of the successful P systems obtained for each case, respectively.) is detailedly shown in Appendix C.

4. Conclusions and future research lines

Both MC and EC are important branches of natural computing and bio-inspired computing. Their interplay is a promising and fertile field. Until now the interactions between MC and EC have produced two main research directions: MIEA and ADMCM. In addition, there are also some studies on using EC to solve the problems in MC. For example, in [37], a GA was applied to solve the broadcasting problem defined within a P system framework, which is regarded as NP-hard combinatorial optimization problems; in [56], the use of GAs to distribute membranes in processors. In this paper, we have presented a systematic review of the recent efforts to develop a theory of EMC. The concepts on EMC were clarified. A unified framework and a variety of topics in this area were discussed. Given the current increasing research interest in MIEA and ADMCM, we think it is worth discussing some future research directions in these areas that we envisage to further develop.

(i) *Interactions of MC and EC:* The interactions between MC and EC need to be further studied. On the one hand, a membrane system has a parallel-distributed framework and various evolution rules; all variants of membrane systems have a rigorous and sound theory attached to them. On the other hand, EC has been used to solve a broad spectrum of problems due to its relatively simple set of operations, easy to be understood and implemented, and amazingly good performance.
   - *EC state transition:* Is it beneficial to use MC transition concept to implement transitions of an EC algorithm? A state transition in EC may be considered as a configuration in the process of a P system evolution. In this respect, we can use the P system simulators *P-Lingua* [20] or *MeCoSim* [59] to simulate the evolutionary process of individuals performed by evolutionary operators such as crossover or mutation, or to mimic the behaviors of a colony of ants or a flock of birds.
   - *Solving MC problems:* So far we have seen the use of EC approaches to solving broadcasting problems and it is expected that they can be also applied for testing P systems in a way similar to testing software. Other types of problem like tile pasting systems, P systems self-assembly can also be addressed with such methods.

(ii) *Extensions and applications of MIEAs:* From the survey provided one can see that only a limited number of types of MIEAs have been considered, but a broad variety can be further investigated.
   - *Membrane structures:* Fixed membrane structures were mainly discussed in the literature. It is very natural to focus also on dynamic membrane structures of cell-like P systems with active membranes, population membrane systems with dynamic links, and also on network structures like in tissue-like membrane systems with various channels.
   - *Types of P systems:* Several types of cell-like P systems were involved in the existing MIEA work. The discussion of a tissue-like P system in the design of a MIEA has recently started and has shown an enormous research potential for developing more optimization approaches with good performance. More types of P systems might be applied to design approximate optimization algorithms to make full use of the characteristics of both MC models and EC. We have in mind models like cell-like membrane systems with active membranes, tissue-like membrane systems, population membrane systems, probabilistic/stochatic P systems, numerical P systems and enzyme numerical P systems.
   - *Evolution rules:* Until now, only several types of evolution rules, namely transformation, communication and division rules, have been considered. Actually the kinds of rules in P systems are rich. More rules, such as separation rules, bond making rules and dissolution rules, may be considered in a proper way to cater for various specific problems.
   - *Applications:* More real-world application problems, such as power system optimization, power system state estimation, software/hardware co-design, vehicle route plan, fault diagnosis, signal and image processing, can be solved by using MIEAs.
   - *Methods:* As usual, an EA is regarded as a subalgorithm placed inside a membrane. Actually this idea can be extended. A membrane structure can be used as a framework of the organization of several different types of evolutionary operators, as shown in [87], or several distinct kinds of evolutionary mechanisms, such as genetic algorithms, evolutionary programming, evolution strategy, differential evolution, particle swarm optimization, ant colony optimization and estimation of distribution algorithms. This is a synthesis at the level of algorithms. Further the flexible communication rules can be used at the level of genes, instead of individuals level. A deep performance analysis and evaluation of MIEAs to reveal the roles of P systems played in the hybrid optimization algorithms should be made.

(iii) *Future work on ADMCM*

- *POPSM optimization approach selection:* In POPSM, the choice of optimization approaches plays an important role in the automated design of cell models based on P systems for systems and synthetic biology. So the better meta-heuristic methods need to be discussed.
- *Design of P systems for simple problems:* The first step of ADPS is to design a class of P systems for solving completely specified and deterministically polynomial-time solvable problems, such as simple arithmetic operations, general sorting problems and some specific membrane systems with simple evolution rules like in the existing work. In this design, the three types of P systems, cell-like, tissue-like and neural-like P systems might be considered.
- *Design of P systems for computationally hard problems:* The next design of P systems will move from simple problems to completely specified but hard discrete or continuous problems, such as satisfiability problems and knapsack problems. In this step, more evolution rules like membrane division could be considered.
- *Design of P systems for real-world problems:* This is a very promising line of investigation as there are already many applications of P systems that can be further analysed. Some of them, like the models of various biological systems, are incomplete, others, like those utilized to specify robot controller, might be optimized or the model recallibrated.
- *Encoding techniques:* In ADPS, how to encode an individual P system with variable membrane structures, objects and evolution rules is the first key problem because this is the basis of constructing a population of candidate P systems and evaluating a P system. Except for the binary encoding technique, other methods are worth further exploring.
- *Evaluation methods:* As the object we aim to produce is a complex sP system with various components and interactions, it is not clear how to assess how good a candidate solution is. Some research for clarifying this aspect is worth considering.
- *Semantics:* This is an important aspect of this approach as it requires to assess how much of the behavior of a given P system should be considered. More from the semantics of a P system is considered, more complex the optimization process.
- *Verification:* Producing a solution implies to guarantee that this correct. In this respect some additional mechanisms, like formal verification, can be considered.

Acknowledgement

The authors are very grateful to the Editor-in-Chief, Professor Witold Pedrycz, the editors and the anonymous reviewers for their insightful recommendations and comments to improve this paper. The authors would also like to thank Xiaoli Huang for her work on the recent results of ADMCM and Chunxiu Liu for her work on the program of MIEA analysis.

Appendix A. Glossary

ABC	artificial bee colony
ACO	ant colony optimization
ACOPS	ACO based on P systems
ADMCM	automated design of membrane computing models
ADPS	automatic design of P systems
AIM	algorithm-in-membrane
BCMC	membrane computing optimization method based on the catalytic factor
ARSM	abstract rewriting systems on multisets
BIAMC	bio-inspired algorithm based on membrane computing
BIPOA	bio-inspired P systems based optimization algorithm
BPoPS	broadcasting problems of P systems
CRGA	changing range genetic algorithm
DAIMC	distributed approach inspired by membrane computing
DE	differential evolution
DEA	distributed evolutionary algorithm
DEPS	DE based on P systems
DETPS	hybrid approach combining DE with tissue P systems
DMOAP	dynamic multiobjective optimization algorithm inspired by P systems
DMS	dynamic membrane structure
DNAMC	algorithm based on DNA computing and membrane computing
DNS	dynamically network structure
DSR	distribution system reconfiguration
EA	evolutionary algorithm
EC	evolutionary computation

EMA	evolutionary membrane algorithm
EMC	evolutionary membrane computing
GA	genetic algorithm
GBSP	gasoline blending scheduling problems
HBS	hybrid optimization method combining IBIAMC with SQP
HMS	hybrid membrane structure
HMEA	hybrid membrane evolutionary algorithm
HPSOWM	hybrid particle swarm optimization with wavelet mutation
HPSOPS	a membrane algorithm based on HPSOWM and P systems
IBIAMC	improved bioinspired algorithm based on membrane computing
JSSP	job-shop scheduling problems
LS	local search
MA	membrane algorithm
MA4MS	membrane algorithm for min storage
MAPS	memetic algorithm based on P systems
MAQIS	MA with quantum-inspired subalgorithms
MC	membrane computing
MCBPSO	MC based PSO
MCCOP	constrained optimization evolutionary algorithm based on MC
MCOA	membrane computing based optimization algorithm
MDPS	monitored distributed P system
MIEA	membrane-inspired evolutionary algorithm
MOBIAMC	multiobjective bioinspired algorithm based on membrane computing
MOMA	multi-objective MA
MOMC	multi-objective optimization membrane computing
MPOP	manufacturing parameter optimization problems
MQEPS	modified QEPS
MQPSO	membrane quantum particle swarm optimization
MQSFL	membrane-inspired quantum shuffled frog leaping algorithm
MS	membrane structure
NEA	nested evolutionary algorithm
NN	neural network
NMS	nested membrane structure
NoFE	the number of function evaluations
NP	non-deterministic polynomial-time
NS	network structure
OLMS	one-level membrane structure
PEMFCM	proton exchange membrane fuel cell model
PMOA	P system based multi-objective optimization algorithm
POPSM	parameter optimization of P system models
PSPACE	polynomial space
PSO	particle swarm optimization
PSOPS	PSO based on P systems
PSMA	a population P system based membrane algorithm
QIEA	quantum-inspired evolutionary algorithm
QEPS	QIEA based on P systems
QEAM	QIEA based on P systems with active membranes
QPSO	quantum particle swarm optimization
QSFL	quantum shuffled frog leaping algorithm
RQEPS	real QEPS
SA	simulated annealing
SAT	satisfiability problem
SNS	statically network structure
SQP	sequential quadratic programming
TPS	tissue P system
TPSA	temperature parallel simulated annealing
TSP	traveling salesman problem

Appendix B. Description of a knapsack problem in Section 3.1

Knapsack problem, a well-known NP-hard combinatorial optimization problem, can be described as selecting from among various items those items that are most profitable, given that the knapsack has limited capacity. The knapsack problem is to select a subset from the given number of items so as to maximize the profit $f(x)$:

$$f(x) = \sum_{i=1}^{N} p_i x_i \tag{5}$$

subject to

$$\sum_{i=1}^{N} \omega_i x_i \leqslant C \tag{6}$$

where N is the number of items; p_i is the profit of the ith item; ω_i is the weight of the ith item; C is the capacity of the given knapsack; and x_i is 0 or 1.

This paper uses strongly correlated sets of unsorted data, i.e.,

$$\omega_i = uniformly\, random[1, \Omega]$$

$$p_i = \omega_i + \frac{1}{2}\Omega$$

where Ω is the upper bound of $\omega_i, i = 1, 2, \cdots, N$, and the average knapsack capacity C is applied.

$$C = \frac{1}{2}\sum_{i=1}^{N} \omega_i \tag{7}$$

Appendix C. Successful P systems obtained in Section 3.2

$$R_{11} = \left\{ \begin{array}{ll} r_6 : [s \to abx]_2 & r_{25} : [x]_2 \to []\ abc \\ r_{19} : [b]_2 \to [b]c & r_{32} : [a \to ac]_1 \end{array} \right\}$$

$$R_{12} = \left\{ \begin{array}{ll} r_7 : [s \to abxc]_2 & r_{27} : [x]_2 \to \lambda \\ r_{19} : [b]_2 \to [b]c & r_{40} : \left[b \to bc^2\right]_1 \end{array} \right\}$$

$$R_{13} = \left\{ \begin{array}{ll} r_7 : [s \to abxc]_2 & r_{25} : [x]_2 \to []\ abc \\ r_{21} : [c]_2 \to c & r_{32} : [a \to ac]_1 \end{array} \right\}$$

$$R_{14} = \left\{ \begin{array}{ll} r_4 : [s \to cx]_2 & r_{25} : [x]_2 \to []\ abc \\ r_{21} : [c]_2 \to c & r_{40} : \left[b \to bc^2\right]_1 \end{array} \right\}$$

$$R_{15} = \left\{ \begin{array}{ll} r_4 : [s \to cx]_2 & r_{25} : [x]_2 \to []\ abc \\ r_{21} : [c]_2 \to c & r_{32} : [a \to ac]_1 \\ r_{38} : [b \to bc]_1 & \end{array} \right\}$$

$$R_{16} = \left\{ \begin{array}{ll} r_7 : [s \to abxc]_2 & r_{27} : [x]_2 \to \lambda \\ r_{13} : [a]_2 \to []\ b & r_{38} : [b \to bc]_1 \\ r_{19} : [b]_2 \to [b]\ c & \end{array} \right\}$$

$$R_{21} = \left\{ \begin{array}{ll} r_9 : [s]_2 \to abcx & r_{46} : [x \to a]_1 \\ r_{40} : \left[b \to bc^2\right]_1 & \end{array} \right\}$$

$$R_{22} = \left\{ \begin{array}{ll} r_9 : [s]_2 \to abcx & r_{45} : [x \to ac]_1 \\ r_{35} : [a \to abc]_1 & \end{array} \right\}$$

$$R_{23} = \left\{ \begin{array}{ll} r_1 : [s]_2 \rightarrow bcx & r_{48} : [x \rightarrow bc]_1 \\ r_{38} : [b \rightarrow bc]_1 & \end{array} \right\}$$

$$R_{24} = \left\{ \begin{array}{ll} r_9 : [s]_2 \rightarrow abcx & r_{43} : [x \rightarrow bcx]_1 \\ r_{31} : [a \rightarrow cx]_1 & \end{array} \right\}$$

$$R_{25} = \left\{ r_4 : [s]_2 \rightarrow abc \quad r_{40} : \left[b \rightarrow bc^2 \right]_1 \right\}$$

$$R_{26} = \left\{ r_1 : [s]_2 \rightarrow bcx \quad r_{40} : \left[b \rightarrow bc^2 \right]_1 \right\}$$

$$R_{31} = \left\{ \begin{array}{ll} r_9 : [s]_2 \rightarrow abcx & r_{29} : [a \rightarrow bc]_1 \\ r_{40} : \left[b \rightarrow bc^2 \right]_1 & r_{50} : [x \rightarrow ax]_1 \end{array} \right\}$$

$$R_{32} = \left\{ \begin{array}{ll} r_9 : [s]_2 \rightarrow abcx & r_{32} : [a \rightarrow ac]_1 \\ r_{40} : \left[b \rightarrow bc^2 \right]_1 & r_{49} : [x \rightarrow bx]_1 \end{array} \right\}$$

$$R_{33} = \left\{ \begin{array}{ll} r_9 : [s]_2 \rightarrow abcx & r_{30} : [a \rightarrow ab]_1 \\ r_{40} : \left[b \rightarrow bc^2 \right]_1 & r_{47} : [x \rightarrow xc]_1 \end{array} \right\}$$

$$R_{34} = \left\{ \begin{array}{ll} r_9 : [s]_2 \rightarrow abcx & r_{31} : [a \rightarrow cx]_1 \\ r_{40} : \left[b \rightarrow bc^2 \right]_1 & r_{44} : [x \rightarrow ab]_1 \end{array} \right\}$$

$$R_{35} = \left\{ \begin{array}{ll} r_3 : [s]_2 \rightarrow abc & r_{40} : \left[b \rightarrow bc^2 \right]_1 \\ r_{35} : [a \rightarrow abc]_1 & \end{array} \right\}$$

$$R_{36} = \left\{ \begin{array}{ll} r_1 : [s]_2 \rightarrow bcx & r_{40} : \left[b \rightarrow bc^2 \right]_1 \\ r_{43} : [x \rightarrow bcx]_1 & \end{array} \right\}$$

$$R_{37} = \left\{ \begin{array}{ll} r_3 : [s]_2 \rightarrow abc & r_{42} : \left[b \rightarrow bc^2 x \right]_1 \\ r_{35} : [a \rightarrow abc]_1 & \end{array} \right\}$$

$$R_{38} = \left\{ \begin{array}{ll} r_9 : [s]_2 \rightarrow abcx & r_{40} : \left[b \rightarrow bc^2 \right]_1 \\ r_{43} : [x \rightarrow bcx]_1 & \end{array} \right\}$$

$$R_{39} = \left\{ \begin{array}{ll} r_3 : [s]_2 \rightarrow abc & r_{40} : \left[b \rightarrow bc^2 \right]_1 \\ r_{36} : [a \rightarrow bcx]_1 & r_{43} : [x \rightarrow bcx]_1 \end{array} \right\}$$

$$R_{41} = \left\{ \begin{array}{ll} r_5 : [s \rightarrow abc]_2 & r_{14} : [a \rightarrow ab]_2 \\ r_{19} : [b]_2 \rightarrow [b]c & \end{array} \right\}$$

$$R_{42} = \left\{ \begin{array}{ll} r_2 : [s \rightarrow ab]_2 & r_{14} : [a \rightarrow ab]_2 \\ r_{19} : [b]_2 \rightarrow [b]c & \end{array} \right\}$$

$$R_{43} = \left\{ \begin{array}{ll} r_4 : [s \rightarrow cx]_2 & r_{25} : [x]_2 \rightarrow [] \ abc \\ r_{38} : [b \rightarrow bc]_1 & r_{35} : [a \rightarrow abc]_1 \end{array} \right\}$$

$$R_{44} = \left\{ \begin{array}{ll} r_1 : [s]_2 \rightarrow bcx & r_{50} : [x \rightarrow ax]_1 \\ r_{39} : [b \rightarrow a]_1 & r_{32} : [a \rightarrow ac]_1 \end{array} \right\}$$

$$R_{51} = \left\{ \begin{array}{ll} r_7 : [s \to abxc]_2 & r_{19} : [b]_2 \to [b]c \\ r_{23} : [x]_2 \to x & r_{40} : \left[b \to bc^2\right]_1 \\ r_{13} : [a]_2 \to []b & r_{49} : [x \to bx]_1 \end{array} \right\}$$

$$R_{52} = \left\{ \begin{array}{ll} r_7 : [s \to abxc]_2 & r_{40} : \left[b \to bc^2\right]_1 \\ r_{24} : [x]_2 \to cx & r_{49} : [x \to bx]_1 \\ r_{14} : [a \to ab]_2 & \end{array} \right\}$$

$$R_{53} = \left\{ \begin{array}{ll} r_8 : \left[s \to a^2bx\right]_2 & r_{19} : [b]_2 \to [b]c \\ r_{25} : [x]_2 \to [] \ abc & r_{35} : [a \to abc]_1 \\ r_{14} : [a \to ab]_2 & \end{array} \right\}$$

$$R_{61} = \left\{ \begin{array}{ll} r_7 : [s \to abxc]_2 & r_{26} : [x]_2 \to \left[x^2\right]c^2 \\ r_{19} : [b]_2 \to [b]c & r_{22} : [c \to ab]_2 \\ r_{14} : [a \to ab]_2 & \end{array} \right\}$$

$$R_{62} = \left\{ \begin{array}{ll} r_8 : \left[s \to a^2bx\right]_2 & r_{26} : [x]_2 \to \left[x^2\right]c^2 \\ r_{19} : [b]_2 \to [b]c & r_{35} : [a \to abc]_1 \\ r_{14} : [a]_2 \to [] \ a & r_{38} : [b \to bc]_1 \end{array} \right\}$$

$$R_{63} = \left\{ \begin{array}{ll} r_8 : \left[s \to a^2bx\right]_2 & r_{26} : [x]_2 \to \left[x^2\right]c^2 \\ r_{19} : [b]_2 \to [b]c & r_{14} : [a \to ab]_2 \end{array} \right\}$$

$$R_{71} = \left\{ \begin{array}{ll} r_2 : [s \to ab]_2 & r_{10} : \left[a \to a^2\right]_2 \\ r_{15} : \left[b \to b^3\right]_2 & r_{27} : [x]_2 \to \lambda \\ r_{11} : \left[a \to a^2x\right]_2 & \end{array} \right\}$$

$$R_{72} = \left\{ \begin{array}{ll} r_2 : [s \to ab]_2 & r_{10} : \left[a \to a^2\right]_2 \\ r_{15} : \left[b \to b^3\right]_2 & r_{24} : [x]_2 \to cx \\ r_{16} : \left[b \to b^3x\right]_2 & \end{array} \right\}$$

$$R_{73} = \left\{ \begin{array}{ll} r_5 : [s \to abc]_2 & r_{10} : \left[a \to a^2\right]_2 \\ r_{15} : \left[b \to b^3\right]_2 & r_{23} : [x]_2 \to x \\ r_{16} : \left[b \to b^3x\right]_2 & \end{array} \right\}$$

$$R_{74} = \left\{ \begin{array}{ll} r_2 : [s \to ab]_2 & r_{10} : \left[a \to a^2\right]_2 \\ r_{15} : \left[b \to b^3\right]_2 & r_{11} : \left[a \to a^2x\right]_2 \\ r_{16} : \left[b \to b^3x\right]_2 & r_{23} : [x]_2 \to x \end{array} \right\}$$

$$R_{75} = \left\{ \begin{array}{ll} r_5 : [s \to abc]_2 & r_{26} : [x]_2 \to \left[x^2\right]c^2 \\ r_{15} : \left[b \to b^3\right]_2 & r_{23} : [x]_2 \to \lambda \\ r_{11} : \left[a \to a^2x\right]_2 & \end{array} \right\}$$

$$R_{81} = \left\{ \begin{array}{ll} r_5 : [s \to abc]_2 & r_{14} : [a \to ab]_2 \\ r_{16} : \left[b \to b^3x\right]_2 & r_{23} : [x]_2 \to x \\ r_{26} : [x]_2 \to \left[x^2\right]c^2 & \end{array} \right\}$$

$$R_{82} = \left\{ \begin{array}{ll} r_2 : [s \to ab]_2 & r_{14} : [a \to ab]_2 \\ r_{16} : \left[b \to b^3 x\right]_2 & r_{27} : [x]_2 \to \lambda \\ r_{26} : [x]_2 \to \left[x^2\right] c^2 & \end{array} \right\}$$

$$R_{83} = \left\{ \begin{array}{ll} r_5 : [s \to abc]_2 & r_{14} : [a \to ab]_2 \\ r_{16} : \left[b \to b^3 x\right]_2 & r_{23} : [x]_2 \to x \\ r_{15} : \left[b \to b^3\right]_2 & \end{array} \right\}$$

$$R_{84} = \left\{ \begin{array}{ll} r_2 : [s \to ab]_2 & r_{14} : [a \to ab]_2 \\ r_{16} : \left[b \to b^3 x\right]_2 & r_{27} : [x]_2 \to \lambda \\ r_{15} : \left[b \to b^3\right]_2 & \end{array} \right\}$$

References

[1] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press, Oxford, UK, 1996.
[2] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (2011) 4135–4151.
[3] C. Blum, A. Roli, Hybrid metaheuristics: an introduction, in: C. Blum, M.J.B. Aguilera, A. Roli, M. Sampels (Eds.), Hybrid Metaheuristics, Studies in Computational Intelligence, vol. 114, Springer, 2008, pp. 1–30.
[4] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. 237 (2013) 82–117.
[5] H. Cao, F.J. Romero-Campero, S. Heeb, M. Cámara, N. Krasnogor, Evolving cell models for systems and synthetic biology, Syst. Synth. Biol. 4 (2010) 55–84.
[6] H. Chen, J. Lu, A constrained optimization evolutionary algorithm based on membrane computing, J. Digital Inf. Manage. 10 (2012) 121–125.
[7] Y. Chen, G. Zhang, T.Wang, X. Huang, Automatic design of a P system for basic arithmetic operations, Chin. J. Electron. (English J.) 23 (2013) 302–304.
[8] J. Cheng, G. Zhang, X. Zeng, A novel membrane algorithm based on differential evolution for numerical optimization, Int. J. Unconventional Comput. 7 (2011) 159–183.
[9] A. Ciobanu, F. Ipate, Creating a P system from a trace using genetic algorithms, in: Proceedings of the Twelfth International Conference on Membrane Computing, 2011, pp. 479–483.
[10] G. Ciobanu, M.J. Pérez-Jiménez, G. Păun (Eds.), Applications of membrane computing, Natural Computing Series, Springer, 2006.
[11] M. Dorigo, T. Stützle, Ant Colony Optimization, Bradford Company, Scituate, MA, USA, 2004.
[12] A.E. Eiben, J. Smith, Introduction to Evolutionary Computing, Springer-Verlag, 2003.
[13] S. Elias, V. Gokul, K. Krithivasan, M. Gheorghe, G. Zhang, A variant of the distributed P system for real time cross layer optimization, J. Universal Comput. Sci. 18 (2012) 1760–1781.
[14] G. Escuela, M.A. Gutiérrez-Naranjo, An application of genetic algorithms to membrane computing, in: Proceedings of the Eighth Brainstorming Week on Membrane Computing, 2010, pp. 101–108.
[15] L. Fogel, A. Owens, M. Walsh, Artificial Intelligence through Simulated Evolution, Wiley, Chichester, WS, UK, 1966.
[16] N.P. Frisco, M. Gheorghe, M.J. Pérez-Jiménez (Eds.), Applications of Membrane Computing in Systems and Synthetic Biology, Springer, 2014.
[17] H. Gao, J. Cao, Membrane-inspired quantum shuffled frog leaping algorithm for spectrum allocation, J. Syst. Eng. Electron. 23 (2012) 679–688.
[18] H. Gao, J. Cao, Y. Zhao, Membrane quantum particle swarm optimisation for cognitive radio spectrum allocation, Int. J. Comput. Appl. Technol. 43 (2012) 359–365.
[19] C. García-Martínez, C. Lima, J. Twycross, N. Krasnogor, M. Lozano, P system model optimisation by means of evolutionary based search algorithms, in: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, 2010, pp. 187–194.
[20] M. García-Quismondo, R. Gutiérrez-Escudero, M.A. Martinez-del-Amor, E. Orejuela-Pinedo, I. Pérez-Hurtado, P-Lingua 2.0: a software framework for cell-like P systems, Int. J. Comput. Commun. Control 4 (2009) 234–243.
[21] M. Gheorghe, G. Păun, M.J. Pérez-Jiménez, G. Rozenberg, Research frontiers of membrane computing: open problems and research topics, Int. J. Found. Comput. Sci. 24 (2013) 547–623.
[22] F. Glover, Tabu search–part I, INFORMS J. Comput. 1 (1989) 190–206.
[23] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, 1st ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
[24] J.H. Holland, Adaptation in Natural and Artificial Systems, second ed., University of Michigan Press, Ann Arbor, MI, 1975. 1992.
[25] L. Huang, X. He, N. Wang, Y. Xie, P systems based multi-objective optimization algorithm, Progr. Nat. Sci. 17 (2007) 458–465.
[26] L. Huang, I.H. Suh, Controller design for a marine diesel engine using membrane computing, Int. J. Innovative Comput. Inf. Control 5 (2009) 899–912.
[27] L. Huang, I.H. Suh, A. Abraham, Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants, Inf. Sci. 181 (2011) 2370–2391.
[28] L. Huang, L. Sun, N. Wang, X. Jin, Multiobjective optimization of simulated moving bed by a kind of tissue P system, Chin. J. Chem. Eng. 15 (2007) 683–690.
[29] L. Huang, N. Wang, J.H. Zhao, Multiobjective optimization for controller design, Acta Automatica Sinica 34 (2008) 472–477.
[30] X. Huang, G. Zhang, H. Rong, F. Ipate, Evolutionary design of a simple membrane system, in: M. Gheorghe, G. Păun, G. Rozenberg, A. Salomaa, S. Verlan (Eds.), Membrane Computing, Lecture Notes in Computer Science, vol. 7184, 2012, pp. 203–214.
[31] H. Iba, N. Noman, New Frontier in Evolutionary Algorithms: Theory and Applications, Imperial College Press, 2011.
[32] J. Kennedy, R.C. Eberhart, Swarm Intelligence, Morgan Kaufman Publishers Inc., San Francisco, CA, USA, 2001.
[33] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[34] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA, 1992.
[35] J.R. Koza, Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press, Cambridge Massachusetts, 1994.
[36] W.B. Langdon, R. Poli, Foundations of Genetic Programming, Springer-Verlag, 2002.
[37] R. Lefticaru, F. Ipate, M. Gheorghe, G. Zhang, Tuning P systems for solving the broadcasting problem, in: G. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, G. Rozenberg, A. Salomaa, (Eds.), Membrane Computing, Lecture Notes in Computer Science, vol. 5957, 2009, pp. 354–370.
[38] A. Leporati, D. Pagani, A membrane algorithm for the min storage problem, in: H.J. Hoogeboom, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, Lecture Notes in Computer Science, vol. 4361 2006, pp. 443–462.

[39] C. Liu, M. Han, X.Z. Wang, A multi-objective evolutionary algorithm based on membrane systems, in: Proceedings of the Fourth International Workshop on Advanced Computational Intelligence, 2011, pp. 103–109.
[40] C. Liu, M. Han, X.Z. Wang, A novel evolutionary membrane algorithm for global numerical optimization, in: Proceedings of the Third International Conference on Intelligent Control and Information Processing, 2012, pp. 727–732.
[41] C. Liu, G. Zhang, H. Liu, M. Gheorghe, F. Ipate, An improved membrane algorithm for solving time-frequency atom decomposition, in: G. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, G. Rozenberg, A. Salomaa, (Eds.), Membrane Computing, Lecture Notes in Computer Science, vol. 5957, 2010, pp. 371–384.
[42] C. Liu, G. Zhang, X. Zhang, H. Liu, A memetic algorithm based on P systems for IIR digital filter design, in: Proceedings of the Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, 2009, pp. 330–334.
[43] C. Liu, G. Zhang, Y. Zhu, C. Fang, H. Liu, A quantum-inspired evolutionary algorithm based on P systems for radar emitter signals, in: Proceedings of the Fourth International Conference on Bio-Inspired Computing: Theories and Applications, 2009, pp. 24–28.
[44] T.M. Mitchell, Machine Learning, McGraw-Hill, New York, NY, 1997.
[45] E. Nabil, A. Badr, I. Farag, A P system design using clonal selection algorithm, Studia University Babes–Bolyai, Informatica LVI (2011) 11–24.
[46] N.J. Nilsson, Principles of Artificial Intelligence, Morgan Kaufman Publishers Inc., San Francisco, CA, USA, 1980.
[47] N.J. Nilsson, Artificial Intelligence: A New Synthesis, Morgan Kaufman Publishers Inc., San Francisco, CA, USA, 1998.
[48] T.Y. Nishida, An application of P systems: a new algorithm for NP-complete optimization problems, in: N. Callaos, W. Lesso, B. Sanchez (Eds.), Proceedings of the 8th World Multi-Conference on Systems, Cybernetics and Informatics, vol. V, 2004, pp. 109–112.
[49] T.Y. Nishida, Membrane algorithms, in: R. Freund, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, Lecture Notes in Computer Science, vol. 3850, 2006, pp. 55–66.
[50] T.Y. Nishida, Membrane algorithms: approximate algorithms for NP-complete optimization problems, in: G. Ciobanu, G. Păun, M.J. Pérez-Jiménez, (Eds.), Applications of Membrane Computing, Natural Computing Series, 2006b, pp. 303–314.
[51] T.Y. Nishida, Membrane algorithm with brownian subalgorithm and genetic subalgorithm, Int. J. Found. Comput. Sci. 18 (2007) 1353–1360.
[52] T.Y. Nishida, T. Shiotani, Y. Takahashi, Membrane algorithm solving job-shop scheduling problems, in: Proceedings of the 9th International Workshop on Membrane Computing, 2008, pp. 363–370.
[53] A. Olsson, Particle Swarm Optimization: Theory, Techniques and Applications, Engineering Tools, Techniques and Tables, Nova Science Publishers, Incorporated, 2011.
[54] Z. Ou, G. Zhang, T. Wang, X. Huang, Automatic design of cell-like P systems through tuning membrane structures, initial objects and evolution rules, Int. J. Unconventional Comput. 9 (2013) 425–443.
[55] G. Păun, G. Rozenberg, A. Salomaa, The Oxford Handbook of Membrane Computing, Oxford University Press, Inc., New York, NY, USA, 2010.
[56] M.Á. Peña, J. Castellanos, Membranes distribution using genetic algorithms, Int. J. Inf. Theories Appl. 18 (2011) 248–257.
[57] H. Peng, J. Shao, B. Li, J. Wang, M.J. Pérez-Jiménez, Y. Jiang, Y. Yang, Image thresholding with cell-like P systems, in: Proceedings of the Tenth Brainstorming Week on Membrane Computing, 2012, pp. 75–87.
[58] H. Peng, J. Wang, M.J. Pérez-Jiménez, P. Shi, A novel image thresholding method based on membrane computing and fuzzy entropy, J. Intell. Fuzzy Syst. 24 (2013) 229–237.
[59] I. Pérez-Hurtado, L. Valencia-Cabrera, M.J. Pérez-Jiménez, M.A. Colomer, A. Riscos-Núñez, Mecosim: A general purpose software tool for simulating biological phenomena by means of P systems, in: Proceedings of International Conference on Bio-Inspired Computing: Theories and Applications, 2010, pp. 637–643.
[60] K. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
[61] G. Păun, M.J. Pérez-Jiménez, Membrane computing: brief introduction, recent results and applications, BioSystems 85 (2006) 11–22.
[62] F.J. Romero-Campero, H. Cao, M. Cámara, N. Krasnogor, Structure and parameter estimation for cell systems biology models, in: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, 2008, pp. 331–338.
[63] G. Rozenberg, T. Bäck, J.N. Kok (Eds.), Handbook of Natural Computing, Springer, 2012.
[64] V. Sarpe, A. Esmaeili, I. Yazdanbod, T. Kubik, M. Richter, C. Jacob, Parametric evolution of a bacterial signalling system formalized by membrane computing, in: Proceedings of IEEE Congress on Evolutionary Computation, 2010, pp. 1–8.
[65] H. Schwefel (Ed.), Evolution and Optimum Seeking, A Wiley-Interscience Publication, Wiley, New York, 1995.
[66] E. Smirnov, Conjunctive and disjunctive version spaces with instance-based boundary sets, in: SIKS Dissertation Series, Shaker Publishing, 2001.
[67] Y. Sun, L. Zhang, X. Gu, Membrane computing based particle swarm optimization algorithm and its application, in: Proceedings of the Fifth International Conference on Bio-Inspired Computing: Theories and Applications, 2010, pp. 631–636.
[68] Y. Suzuki, Y. Fujiwara, J. Takabayashi, H. Tanaka, Artificial life applications of a class of P systems: Abstract rewriting systems on multisets, in: C. Calude, G. Păun, G. Rozenberg, A. Salomaa, (Eds.), Multiset Processing, Mathematical, Computer Science, and Molecular Computing Points of View, Lecture Notes in Computer Science, vol. 2235, 2001, pp. 299–346.
[69] Y. Suzuki, H. Tanaka, Chemical evolution among artificial proto-cells, in: Artificial Life VII: Proceedings of the Seventh International Conference on Artificial Life, 2000, pp. 54–63.
[70] Y. Suzuki, H. Tanaka, Computational living systems based on an abstract chemical system, in: Proceedings of the 2000 Congress on Evolutionary Computation, 2000, pp. 1369–1376.
[71] C. Tudose, R. Lefticaru, F. Ipate, Using genetic algorithms and model checking for P systems automatic design, in: D.A. Pelta, N. Krasnogor, D. Dumitrescu, C. Chira, R. Lung, (Eds.), Nature Inspired Cooperative Strategies for Optimization, Studies in Computational Intelligence, vol. 387, 2011, pp. 285–302.
[72] F. Wang, Y. Huang, M. Shi, S. Wu, Membrane computing optimization method based on catalytic factor, in: H. Zhang, A. Hussain, D. Liu, Z. Wang, (Eds.), Advances in Brain Inspired Cognitive Systems, Lecture Notes in Artificial Intelligence, vol. 7366, 2012, pp. 129–137.
[73] H. Wang, H. Peng, J. Shao, T. Wang, A thresholding method based on P systems for image segmentation, ICIC Express Lett. 6 (2012) 221–227.
[74] T. Wang, J. Wang, H. Peng, M. Tu, Optimization of PID controller parameters based on PSOPS algorithm, ICIC Express Lett. 6 (2012) 273–280.
[75] J. Xiao, Y. Huang, Z. Cheng, A bio-inspired algorithm based on membrane computing for engineering design problem, Int. J. Comput. Sci. Issues 10 (2013) 580–588.
[76] J. Xiao, Y. Huang, Z. Cheng, J. He, Y. Niu, A hybrid membrane evolutionary algorithm for solving constrained optimization problems, Optik 125 (2014) 897–902.
[77] J. Xiao, X. Zhang, J. Xu, A membrane evolutionary algorithm for DNA sequence design in DNA computing, Chin. Sci. Bull. 57 (2012) 698–706.
[78] J. Xing, H. Yang, An optimization algorithm based on evolution rules on cellular system, in: Z.L. et al., (Eds.), Computational Intelligence and Intelligent Systems, Communications in Computer and Information Science, vol. 316, 2012, pp. 314–320.
[79] S. Yang, N. Wang, A novel P systems based optimization algorithm for parameter estimation of proton exchange membrane fuel cell model, Int. J. Hydrogen Energy 37 (2012) 8465–8476.
[80] X. Yin, L. Qiu, H. Zhang, A distributed approach inspired by membrane computing for optimizing bijective S-boxes, in: Proceedings of the 27th Chinese Control Conference, 2008, pp. 60–64.
[81] D. Zaharie, G. Ciobanu, Distributed evolutionary algorithms inspired by membranes in solving continuous optimization problems, in: H.J. Hoogeboom, G. Păun, G. Rozenberg, A. Salomaa, (Eds.), Membrane Computing, Lecture Notes in Computer Science, vol. 4361, 2006, pp. 536–553.
[82] G. Zhang, Quantum-inspired evolutionary algorithms: a survey and empirical study, J. Heuristics 17 (2011) 303–351.
[83] G. Zhang, J. Cheng, M. Gheorghe, A membrane-inspired approximate algorithm for traveling salesman problems, Romanian J. Inf. Sci. Technol. 14 (2011) 3–19.

[84] G. Zhang, J. Cheng, M. Gheorghe, F. Ipate, X. Wang, QEAM: an approximate algorithm using P systems with active membranes, Int. J. Comput. Commun. Control (2014) (in press).
[85] G. Zhang, J. Cheng, M. Gheorghe, Q. Meng, A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems, Appl. Soft Comput. 13 (2013) 1528–1542.
[86] G. Zhang, M. Gheorghe, J. Cheng, Dynamic behavior analysis of membrane-inspired evolutionary algorithms, Int. J. Comput. Commun. Control 9 (2014) 227–242.
[87] G. Zhang, M. Gheorghe, Y. Li, A membrane algorithm with quantum-inspired subalgorithms and its application to image processing, Nat. Comput. 11 (2012) 701–717.
[88] G. Zhang, M. Gheorghe, C.Z. Wu, A quantum-inspired evolutionary algorithm based on P systems for knapsack problem, Fundam. Inf. 87 (2008) 93–116.
[89] G. Zhang, Y. Li, M. Gheorghe, A multi-objective membrane algorithm for knapsack problems, in: Proceedings of the Fifth International Conference on Bio-Inspired Computing: Theories and Applications, 2010, pp. 604–609.
[90] G. Zhang, C. Liu, M. Gheorghe, Diversity and convergence analysis of membrane algorithms, in: Proceedings of the Fifth International Conference on Bio-Inspired Computing: Theories and Applications, 2010, pp. 596–603.
[91] G. Zhang, C. Liu, M. Gheorghe, F. Ipate, Solving satisfiability problems with membrane algorithm, in: Proceedings of the Fourth International Conference on Bio-Inspired Computing: Theories and Applications, 2009, pp. 29–36.
[92] G. Zhang, H. Rong, J. Cheng, Y. Qin, A population-membrane-systeminspired evolutionary algorithm for distribution network reconfiguration, Chin. J. Electron. (English J.) (2014) (in press).
[93] G. Zhang, F. Zhou, X. Huang, J. Cheng, M. Gheorghe, F. Ipate, R. Lefticaru, A novel membrane algorithm based on particle swarm optimization for solving broadcasting problems, J. Universal Comput. Sci. 18 (2012) 1821–1841.
[94] G.X. Zhang, C.X. Liu, H.N. Rong, Analyzing radar emitter signals with membrane algorithms, Math. Comput. Modell. 52 (2010) 1997–2010.
[95] Y. Zhang, L. Huang, A variant of P systems for optimization, Neurocomputing 72 (2009) 1355–1360.
[96] J. Zhao, N. Wang, A bio-inspired algorithm based on membrane computing and its application to gasoline blending scheduling, Comput. Chem. Eng. 35 (2011) 272–283.
[97] J. Zhao, N. Wang, Hybrid optimization method based on membrane computing, Ind. Eng. Chem. Res. 50 (2011) 1691–1704.
[98] J. Zhao, N. Wang, P. Zhou, Multiobjective bio-inspired algorithm based on membrane computing, in: Proceedings of International Conference on Computer Science and Information Processing, 2012, pp. 473–477.
[99] F. Zhou, G. Zhang, H. Rong, M. Gheorghe, J. Cheng, F. Ipate, R. Lefticaru, A particle swarm optimization based on P systems, in: Proceedings of the 6th International Conference on Natural Computation, 2010, pp. 3003–3007.