Implementing ENPS by Means of GPUs for AI Applications

Manuel García–Quismondo and Mario J. Pérez–Jiménez

Research Group on Natural Computing
Dpt. of Computer Science and Artificial Intelligence, University of Sevilla
Avda. Reina Mercedes s/n. 41012 Sevilla, Spain
{mgarciaquismondo,marper}@us.es

Abstract. A P system represents a distributed and parallel computing model in which basic data structures are, for instance, multisets and strings. Enzymatic Numerical P Systems are a type of P systems whose basic data structures are sets of numerical variables. Separately, GPGPU is a novel technological paradigm which focuses on the development of tools for graphic cards to solve general purpose problems. This paper proposes an ENPS simulator based on GPUs and presents general concepts about its design and some future ideas and perspectives.

Keywords: numerical P systems, enzymatic numerical P systems, membrane controllers, distributed and parallel systems

1 Introduction

Membrane computing is a bio-inspired branch of natural computing, abstracting computing models from the structure and functioning of living cells and from the organization of cells in tissues or other higher order structures [15]. This branch of natural computing studies the design and properties of membrane systems or *P systems*. P systems are distributed and parallel computing models structured in compartments known as *membranes*. Membranes have associated basic data structures such as multisets, strings or numerical variables [16]. According to the way in which membranes are structured, there are several types of P systems. For instance, there exist cell-like P systems [15], tissue P systems [14] and spiking neural P systems [11], along with other types. In P systems, membranes and their associated data structures are processed by means of rewriting rules or *programs* associated to the cells, in order to perform sequences of configurations or *computations* [15][16].

P systems have been successfully applied in a wide range of domains [4]. For instance, they have been applied in microbiological modelling in order to model phenomena such as *quorum* sensing in *Vibrio fischeri* populations [20] and ecological modelling to predict the evolution of the bearded vulture [3] population in the Catalan Pyrenees, as well as image thresholding [6]. Such a versatility makes P systems a useful tool for gaining knowledge about a vast variety of different domains, thus providing promising approaches within the range of disciplines which composes the field of study of artificial intelligence.

A special type of P systems are enzymatic numerical P systems (ENPSs) [18]. ENPSs describe a deterministic, parallel model in which the basic data structures associated to membranes are numerical values which evolve by means of *programs* associated to the membranes [16]. In order for a program to be applied, a certain amount of a specific type of variable (enzyme) may be needed [18]. This model of computation has already been successfully applied to model robot controllers for obstacle avoidance, in which a robot needs to avoid obstacles situated in a closed circuit [19].

Separately, *GPGPU* is a novel technological discipline which consists of the application of graphic cards (GPUs) in order to execute parallel, distributed algorithms [22]. The basic idea is to take advantage of the parallel architecture of GPUs, traditionally used for graphics processing, in order to execute algorithms which can be performed in parallel, thus accelerating them.

In this paper, a GPU simulator for ENPSs is proposed. The parallel architecture of ENPSs makes the simulation of their computations a suitable task to be parallelized, thus expecting an acceleration in the simulation times if compared to the ones obtained by using sequential simulators.

This paper is structured as follows. Section 2 provides a quick introduction to ENPSs as a model of computation and discusses their applications in artificial intelligence. Section 3 provides a general overview of the current state-of-the-art about the results obtained by previous GPU simulators within the field of membrane computing. Finally, section 4 presents the conclusions obtained and proposes some directions for future work.

2 Enzymatic Numerical P Systems

Numerical P systems (NPSs) are a special kind of P systems in which numerical variables evolve from initial values by means of programs. Each program is composed of a production function and a distribution protocol. Each production function is a numerical function over a set of variables. If a variable appears on at least one production function, then it is consumed and its value

is set to 0. Each repartition protocol updates its values according to the result of the production function of its program and a coefficient associated to each variable [15]. A special kind of NPSs are enzymatic numerical P systems (ENPSs). Unlike NPSs, ENPSs describe a deterministic, parallel model of computation. ENPSs introduce the optional use of enzymes associated to programs. Thus, a program is applied only in the following cases: 1) The program does not have an associated enzyme. 2) The value of its associated enzyme is greater than the minimum of the values of the variables consumed by the program. All active programs in each membrane are executed in parallel. More information about ENPSs can be found in [18][19].

ENPSs have been successfully applied within the field of robotics. For instance, they have been used to model deterministic mobile robot controllers for obstacle avoidance. In this model, the speed of the two robot motors is set according to the values assigned to two variables of the system. Thus, the dynamical evolution of these variables describes the behavior of the robot through a closed circuit [19].

2.1 ENPSs and Artificial Intelligence

Mobile robot control problems, such as obstacle avoidance and odometric localizacion, can be considered as artificial intelligence problems. For instance, obstacle avoidance can be considered as a high-level planning problem [13]. In obstacle avoidance, the objective is to find a sequence of movements in a static or dynamical environment. The objective of this sequence is for robots which follow it to avoid crashing with any obstacles they might find in the environment. The input data is given as a series of sensor lectures obtained from the environment. This type of path planning problems arising from the field of robotics has already been attacked by using artificial intelligence techniques such as ant colony algorithms [9][8].

Odometric localization is a widely used method for estimation of the momentary pose of a mobile robot with respect to its starting pose [12]. This estimation is affected by several error sources, such as imprecission in the mobile robot kinematic parameters and errors in the sensor lectures [1]. Thus, odometric localization entails an optimization problem, i.e., minimizing the global error in the pose estimation. As an optimization problem, odometric localization has been previously tackled by using well-known artificial intelligence paradigms, such as genetic algorithms [10] and artificial neural networks [7]. All in all, ENPs propose a new framework which can be applied in order to solve artificial intelligence problems arising from robotics [19].

2.2 Simulation of ENPSs

ENPSs describe a parallel model. Therefore, the huge computational power required by extensive models (for instance, those necessary for massive robot swarms and robots with complex sensor networks) accounts for the need for high performance computing platforms to simulate them. Besides, their parallel structure makes them appropriate to be simulated by means of parallel architectures such as GPUs, FPGAs and computer clusters.

3 Compute Unified Device Architecture (CUDA) Parallel Programming Model

Modern GPUs can physically contain up to 240 processor cores and 30,720 threads. All these threads are executed in parallel. Thus, modern GPUs define an architecture composed of a large number of parallel processing units with a certain degree of independency from each other [22]. In order to make the most of this massively parallel architecture, it is necessary to make use of programming languages specifically designed for these devices. Two of the main standards in GPGPU are OpenCL [21] and CUDA [23]. CUDA defines a parallel programming model which is an abstraction of the specific parallel device where the program is to be executed. The CUDA programming model developed by NVIDIA allows developers to write scalable parallel programs for GPUs using a straightforward extension of the C language (named *CUDA-C*). CUDA-C is a language designed to make the most of the GPGPU approach by enabling programmers to encode parallel applications to be run on GPUs [5]. That is, programmers are able to develop code to be executed on each GPU thread at the same time. This way they can take advantage of the GPU parallel architecture in order to obtain enormous speed-up if compared to sequential versions of the same code. More information about the CUDA programming model can be found on [23].

GPGPU and CUDA have been already successfully applied in order to simulate different kinds of P systems. To the best of our knowledge, they have been applied to simulate cell-like object-based P systems [5] and spiking neural P systems [2]. Their results include data which show noticeable speed-ups in comparison to their sequential counterparts. These results demonstrate the suitability of the GPGPU approach for simulating P systems in a parallel mode.

3.1 Design of the Simulator

The objective of the proposed ENPS GPU-based simulator is to fully simulate the behaviour of enzymatic numerical P systems, performing operations in parallel whenever possible. In order to do that, it is crucial to identify the operations susceptible for parallelization and write parallel kernels for them. This way the simulator can take advantage of the underlying parallel architecture. Thus, in each computational step, the simulator performs the following operations.

First, it checks each program in a different thread. This checking selects those programs which can be applied. Then, it assigns the production function of each applied programs to a thread. If there is at least one applicable program to consume a variable, then that variable is set to 0. Once all consumed variables have been cleared, each applicable production function is computed in parallel. Then, the contribution of each applicable distribution protocol to each variable is computed in parallel. Eventually, all contributions are added to the variables in order to update their values.

This simulator will be published under open source license. It can be used for simulating complex distributed processes modeled with enzymatic numerical P systems. Therefore, several robot behaviors can be simulated in parallel (for example, a robot could avoid obstacles, follow another robot or look for a target at the same time). The synchronization between several behaviors of one robot is done by the help of the enzyme variables which can be used as stop conditions [19]. Apart from simulating several behaviors for only one robot in parallel, the simulator could be used to simulate interaction and cooperation between several robots in complex distributed robotic systems.

3.2 Simulator Performance

All parallel parts of the algorithm are executed with a degree of parallelism of at least equal to the number of programs of the simulated model. The degree of parallelism can be even greater when the repartition protocol is applied. Hence, a theoretical acceleration of at least the number of programs of the model could be reached, if compared to the runtime of sequential simulators. Specifically, the simulator was tested by using an ENPS model of obstacle avoidance [19] as an example, along with other models. These models were simulated by using SNUPS [18]. Then, the resulting runtimes were compared with the GPU simulator runtimes, in order to get an approximate speed-up. In the specific case of the obstacle avoidance model, the total number of programs is 41 [19]. Hence, an acceleration of at least 41 is to be theoretically expected in this case, if compared to sequential ENPSs simulators.

4 Conclusions

In this paper, a GPU-based simulator for enzymatic numerical P systems is proposed. Enzymatic numerical P systems describe a parallel computing model with applications in artificial intelligence. This simulator might be suitable for large scale models which can be applied within the field of robotics.

The massively parallel environment provided by the GPUs is suitable for enzymatic numerical P systems simulations. However, it would be interesting to explore the possibility of scaling-up the currenly existing robot behaviors modeled with ENPSs and simulate them by means of GPU clusters or other parallel architectures (such as FPGAs or computer clusters). These systems might be applied to model the behavior of massive robot swarms and complex sensor networks.

References

1. Antonelli, G., Chiaverini, S., Fusco, G.: A calibration method for odometry of mobile robots based on the least-square technique: Theory and experimental validation. *IEEE Transactions on Robotics*, 21, 5, 994-1004 (2005)
2. Cabarle, F., Adorna, H., Martínez-del-Amor, M.A.: A Spiking Neural P system simulator based on CUDA. *12th International Conference on Membrane Computing (CMC12)*, 23/08/2011-27/08/2011, Fontainebleau, France, 77-92 (2011)
3. Cardona, M., Colomer, M.A., Pérez-Jiménez, M.J., Sanuy, D., Margalida, A.: Modelling ecosystem using P systems: The bearded vulture, a case study. *Lecture notes in Computer Science*, 5391, 95-116 (2009)
4. Cecilia, J.M. García, J.M., Guerrero, G.D., Martínez-del-Amor, M.A, Pérez-Hurtado, I., Pérez-Jiménez, MJ.: Simulation of P systems with Active Membranes on CUDA. *Briefings in Bioinformatics*, 11, 3, 313-322 (2010)
5. Cecilia, J.M. García, J.M., Guerrero, G.D., Martínez-del-Amor, M.A, Pérez-Hurtado, I., Pérez-Jiménez, MJ.: Simulating a P system based efficient solution to SAT by using GPUs. *The Journal of Logic and Algebraic Programming*, 79, 317-325 (2010)
6. Christinal, H.A., Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Péerez-Jiménez, M.J.: Thresholding of 2D Images with Cell-like P Systems. *Romanian Journal of Information Science and Technology (ROMJIST)*, 13, 2, 131-140 (2010)
7. Conforth, M., Meng, Y.: An Artificial Neural Network Based Learning Method for Mobile Robot Localization. *Robotics Automation and Control*, InTech, Ch. 6, 2008
8. Dong, J., Liu, B., Peng, K., Yin, Y.: Robot Obstacle Avoidance based on an Improved Ant Colony Algorithm. *WRI Global Congress on Intelligent Systems (GCIS '09)*, 19/05/2009-21/05/2009, Xiamen, China, 103-106 (2009)

9. Du, R., Zhang, X., Chen, C., Guan, X.: Path Planning with Obstacle Avoidance in PEGs: Ant Colony Optimization Method. *International Conference on Cyber, Physical and Social Computing (CPSCom) 2010*, 18/12/2010-20/12/2010, Hangzhou, China, 768-773 (2010)
10. Gill, M.A.C, Zomaya, A.Y.: Genetic algorithms for robot control. *IEEE International Conference on Evolutionary Computation 1996*, 29/11/1996-01/12/1996, Perth, Australia, 462 (1996)
11. Ibarra, O., Pérez-Jimenez, M.J., Yokomori, T.: On spiking neural P systems. *Natural Computing*, 9, 2, 475-491 (2010)
12. Ivankjo, E., Komšić, I., Petrović, I.: Simple Off-Line Odometry Calibration of Differential Drive Mobile Robots. *Proceedings of 16th International Workshop on Robotics in Alpe-Adria-Danube Region - RAAD 2007*, 07/06/2007-09/06/2007, Ljubljana, Slovenija, 164-169 (2007)
13. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5, 1 (1986), 90-98
14. Pan, L., Pérez-Jiménez, M.J.: Computational complexity of tissue-like P systems. *Journal of Complexity*, 26, 3, 296-315 (2010)
15. Paun, Gh, Paun, R.: Membrane Computing and Economics: Numerical P Systems. *Fundamenta Informaticae*, 73, 1-2, 213-227 (2006)
16. Paun, Gh. Computing with membranes. *Journal of Computer and System Sciences*, 61, 1, 108-143 (2000)
17. Paun, Gh.: Membrane Computing. An Introduction. Springer, XI+419 (2002)
18. Pavel, A., Arsene, O., Buiu, C.: Enzymatic Numerical P Systems - A New Class of Membrane Computing Systems. *Proceedings 2010 IEEE Fifth International Conference on Bio-inspired Computing: Theories and Applications (BICTA 2010)*, 23-26/09/2010, Liverpool, UK, 1331-1336 (2010)
19. Pavel, A, Buiu, C.: Using enzymatic numerical P systems for modeling mobile robot controllers. *Natural Computing*, (in press)
20. Romero, F.J., Pérez-Jiménez, M.J.: A model of the Quorum Sensing System in Vibrio Fischeri using P systems. *Artificial Life*, 14, 1, 95-109 (2008)
21. Takizawa, H, Koyama, K., Sato, K. Komatsu, K., Kobayashi, H.: CheCL: Transparent Checkpointing and Process Migration of OpenCL Applications. *International Parallel and Distributed Processing Symposium (IPDPS11)*, 16/05/2011-20/05/2011, Anchorage, USA (2011)
22. *http://www.gpgpu.org*
23. *http://www.nvidia.com/object/cuda_home_new.html*