Some Open Problems Collected During 7th BWMC

Gheorghe Păun

Institute of Mathematics of the Romanian Academy
PO Box 1-764, 014700 Bucureşti, Romania
and
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
george.paun@imar.ro, gpaun@us.es

Summary. A few open problems and research topics collected during the 7th Brainstorming Week on Membrane Computing are briefly presented; further details can be found in the papers included in the volume.

1 Introduction

This is a short list of open problems and research topics which I have compiled during the Seventh Brainstorming Week on Membrane Computing (BWMC), Sevilla, 2-6 February 2009. Different from the previous years, when such lists (much more comprehensive) were circulated before the meeting, this time the problems were collected during and inspired by the presentations in Sevilla. Most of the problems directly refer to these presentations, hence for further details, in several cases for (preliminary) solutions, the reader should consult the present volume. This list is only a way to call attention to the respective ideas, not necessarily a presentation of the state-of-the-art of the research related to these problems.

Of course, the reader is assumed familiar with membrane computing, so that no preliminaries are provided. As usual, for basic references, the book [17], the handbook [21], and the membrane computing website [25] are suggested.

The problems are identified by letters, with no significance assigned to the ordering.

2 Open Problems

A. The following problem was considered in [5] and then in [3], as an extension to P systems of a problem well known for cellular automata (where it is called the "firing squad synchronization problem", FSSP for short): Find a class of cell-like P systems where all membranes except the skin one may contain given objects, input some objects in the skin region, and aim to reach a configuration where each region contains a designated object F, all regions get this object at the same time, and, moreover, the system stops at that step. (A more technical presentation of the problem can be found in the cited papers.)

I recall from [3] some lines from the end of the Introduction: "The synchronization problem as defined above was studied in [5] for two classes of P systems: transitional P systems and P systems with priorities and polarizations. In the first case, a non-deterministic solution to FSSP was presented and for the second case a deterministic solution was found. These solutions need a time $3h$ and $4n + 2h$ respectively, where n is the number of membranes of a P system and h is the depth of the membrane tree.

In this article we significantly improve the previous results in the non-deterministic case. In the deterministic case, another type of P system was considered and this permitted to improve the parameters. The new algorithms synchronize the corresponding P systems in $2h + 3$ and $3h + 3$ steps, respectively."

Of course, these results depends on the type of P systems used. In particular, in both papers one makes use of a powerful communication command, of a broadcasting type: $in!$ target indications can be associated with objects, with the meaning that the objects marked with $in!$ are sent to ALL membranes placed inside the membrane where the rule containing the command $in!$ was used. This is a very powerful feature, somewhat related to the considered issue itself, so that the natural question is whether or not one can get rid of $in!$. This will probably imply some cost, for instance, in terms of the time needed for synchronization, or – as A. Alhazov pointed out during BWMC – in the knowledge we assume to have (inside the system) about the starting systems (degree, depth, etc.).

A related topic is to consider synchronization issues for other classes of P systems: symport/antiport, active membranes, maybe also for spiking neural P systems (where the object F should be replaced with a specified number of spikes).

Actually, because of its interest (and presumably, of its difficulty, at least to define it), this can be formulated as a separate problem:

B. Define and investigate synchronization problems for spiking neural P systems (SN P systems).

C. Synchronization can suggest another problem, which has appeared from time to time in various contexts (for instance, when constructing SN P systems for simulating Boolean circuits, see [11]): reset a system to its initial configuration after completing a specified task. As in the case of synchronization, we need to consider various classes of P systems and, if the resetting is possible for them, we have to look for the time of doing this.

A possible way to address this problem is the following one: let us choose a class of P systems which halts the computation by sending a specified object to the environment (this is the case with the systems used in solving computationally hard problems in a feasible time; those systems are in general with active membranes, membrane creation, etc.), AND for which the synchronization problem can be solved; when a system halts and produces an object, say, T, we use this object in order to start the synchronization phase; when the synchronization is completed, the object F introduced in each membrane is used for restoring the contents of that membrane as in the beginning of the computation. Of course, there are several difficulties here: is it possible to work with systems whose membrane structure evolves during the computation? (I guess, not); it is easy to introduce the objects present in the initial configuration starting from F, but how the "garbage" objects, those remaining inside in the end of a computation, can be removed? (this needs some kind of context sensitivity, for instance, using F as a promotor of erasing rules, or other similar ideas).

Needless to say that the resetting problem is of a particular interest for SN P systems, but this time I do not formulate this subproblem separately.

The reset problem can probably be addressed also from other points of view and in other contexts. An idea is to have somewhere/somewhow in the system a description of the system itself (a sort of "genome", in a "nucleus") and in the end of the computation to simply destroy the existing system, at the same time reconstructing it from its description, which amounts at resetting the initial system. Some related ideas, problems, and techniques are discussed, in a different set-up, in [6].

D. Let us continue with SN P systems, whose study is far from being exhausted, and which benefits of many suggestions coming from neurology, psychology, computer science. A problem suggested by Linqiang Pan and then preliminarily dealt with in [15] is the following: is it possible to work with SN P systems (in particular, having characterizations of Turing computable sets of numbers or recursively enumerable languages) which have neurons of a small number of types? By a "type" we can understand either only the set of rules present in a neuron (an SN P system whose neurons are of at most k such types is said to be in the kR-normal form), or we can consider both the set of rules and the number of spikes initially present in the neuron (and then we speak about SN P systems in the knR-normal form, if they have neurons of at most k such types).

In [15] it is proved that each recursively enumerable set of natural numbers can be generated by an SN P system (without forgetting rules, but using delays) in the $3R$-normal form when the number is obtained as the distance between spikes or when the number is given by the number of spikes from a specified neuron, but two types of neurons suffice in the accepting case. Slightly bigger values are obtained when we also consider the number of spikes initially present in a neuron for defining the "type" of a neuron: five and three, respectively.

A lot of open problems can now be formulated: First, we do not know whether or not these results can be improved. How these results extend to other classes of SN P systems? (Do the forgetting rules help? What about extended rules, about asynchronous SN P systems, systems with exhaustive use of rules, etc?) What about SN P systems which generate languages, or about universal SN P systems?

E. A problem about SN P systems which was formulated also earlier – see, e.g., problem **G** in [18] – concerns the possibility of dividing neurons, in general, of producing an exponential working space in a linear number of steps, in such a way to be able to solve computationally hard problems in a feasible time. Up to now, no idea how to divide neurons was discussed, although this operations was considered for cells in tissue-like P systems; see, e.g., [20]. What is really exciting is that a recent discovery of neuro-biology provides biological motivations for neuron division: there are so called neural stem cells which divide repeatedly, either without differentiating, hence producing further neural stem cells, or differentiating, so making possible the production of an exponential number of mature neurons; details can be found in [7].

In terms of SN P systems, we can easily capture the idea of having several types of neurons, for instance, using polarizations, as in P systems with active membranes. On the other hand, each neuron also has a label; if by division we obtain two new neurons with the same label, then polarizations are perhaps necessary; if we allow producing new neurons with labels different from the divided neuron, then the polarization can be included in the label (label changing can account also of polarizations).

However, the difficulties lie in other places. First, in SN P systems the contents of a neuron is described by the number of spikes it contains, and this number is usually controlled/sensed by means of a regular expression over alphabet $\{a\}$. This seems to be also the most natural idea for controlling the division, which leads to rules of the general form

$$[\ E]_i \rightarrow [\ E_1]_j[\ E_2]_k,$$

where E, E_1, E_2 are regular expressions over $\{a\}$, and i, j, k are labels of neurons (the labels of the produced neurons tell us which are the rules used in those neurons). By using such a rule, a neuron σ_i containing n spikes such that $a^n \in L(E)$ divides into two neurons, with labels j and k, containing n_1 and n_2 spikes, respectively, such that $n = n_1 + n_2$, and $a^{n_1} \in L(E_1),\ a^{n_2} \in L(E_2)$.

Now, the second important point/diffficulty appears: which are the synapses of the new neurons? There are two possibilities: to have the neurons places "in parallel" and to have them placed "serially/in cascade". In the first case, no connection exists between the two new neurons, but both of them inherit the links of the father neuron; in the second case, neuron σ_j inherits the synapses coming to σ_i, neuron σ_k inherits the synapses going out of σ_i, while a synapse (j, k) is also established. In order to distinguish the two interpretations, the rules are written as follows:

$$\begin{aligned}(a)\ \ & [\ E]_i \rightarrow [\ E_1]_j||[\ E_2]_k,\\ (b)\ \ & [\ E]_i \rightarrow [\ E_1]_j/[\ E_2]_k,\end{aligned}$$

and the semantics described above is illustrated in Figure 1.

The use of these rules was already explored in [16], where `SAT` is solved in polynomial time by SN P systems having the possibility of dividing neurons. The

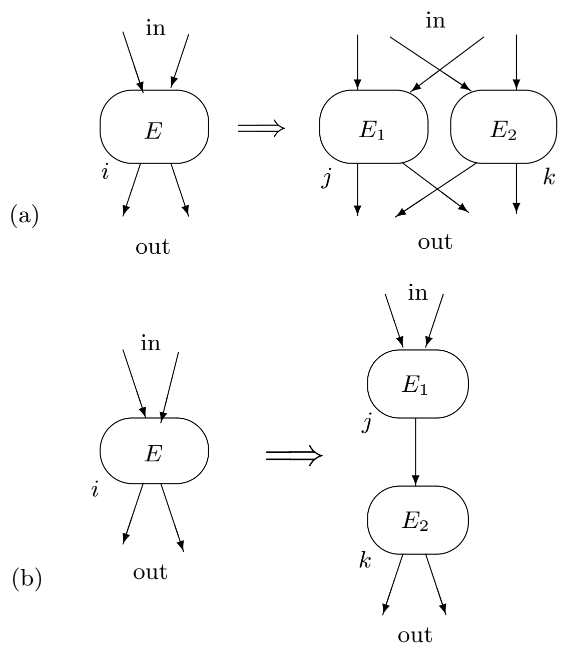

Fig. 1. Dividing neurons – two cases

solution is uniform. Is it possible to use only neuron division where the labels are not changed? (A more intensive use of the regular expressions controlling the division might be helpful, maybe also the use of polarizations, hence of suggestions coming from neural stem cells.) What about solving more difficult problems, for instance, `QSAT` (so that to cover **PSPACE**)?

In general, complexity investigations related to SN P systems need further efforts.

F. Both for stressing its interest and also because is related to the previous problem, I recall/reformulate here another problem from [18], problem **N**, about using pre-computed resources in solving hard problems by means of SN P systems. The motivation is clear, several papers have considered this idea, directly producing solutions to well-known problems, but still a more general approach is missing. The basic issue is to say which pre-computed (arbitrarily large) nets of neurons are allowed, how much information may be given "for free" (and how it can be evaluated/measured), how complexity classes should look like in this framework? A possible way to address some of these questions is to mix the idea of dividing neurons with that of pre-computed resources, and to build a net of neurons starting from a few given neurons, with a limited contents. This approach moves the pre-computation in the computation, with the important difference being that the time of the pre-computation (of "programming") is ignored as long as the produced system contains a limited quantity of information. (We return to the main problem: how to define and measure the information present in a system?) In the present volume there is a contribution related to this problem, namely [**?**], which address this issue starting from rules able to create neurons and synapses, hence to provide a dynamical SN P system.

G. Continuing with SN P systems, a problem which was vaguely formulate from time to time, but only orally, refers to a basic feature of the brain, the memory. How can this be captured in terms of SN P systems is an intriguing question. First, what means "memory"? In principle, the possibility to store some information for a certain time (remember that there is a short term and also a long term memory), and to use this information without losing it. For our systems, let us take the case of storing a number; we need a module of an SN P system where this number is "memorized" in such a way that in precise circumstances (e.g., at request, when a signal comes from outside the module), the number is "communicated" without "forgetting" it. In turn, the communication can be to only one destination or to several destinations. There are probably several ways to build such a module. The difficulty comes from the fact that if the number n is stored in the form of n spikes, "reading" these spikes would consume them, hence it is necessary to produce copies which in the end of the process reset the module. This is clearly possible in terms of SN P systems, what remains to do is to explicitly write the system. However, new questions appear related to the efficiency of the construction, in terms of time (after getting the request for the number n, how many steps are necessary in order to provide the information and to reset the module?), and also in terms of descriptional complexity (how many neurons and rules, how many spikes, how complex rules?). It is possible that a sort of "orthogonal" pair of ideas are useful: many spikes in a few neurons (n spikes in one neuron already is a way to store the number, what remains is to read and reset), or a few spikes in many neurons (a cycle of n neurons among which a single spike circulates, completing the cycle in n steps, is another "memory cell" which stores the number n; again, we need to read and reset, if possible, using only a few spikes). Another possible question is to build a reusable module, able to store several numbers: for a while (e.g., until a special signal) a number n_1 is stored, after that another number n_2, and so on.

H. The previous problem can be placed in a more general set-up, that of modeling other neurobiological issues in terms of SN P systems. A contribution in this respect is already included in the present volume, [13], where the sleep-awake passage is considered. Of course, the approach is somewhat metaphorical, as the distance between the physiological functioning of the brain and the formal structure and functioning of an SN P system is obvious, but still this illustrates the versatility and modeling power of SN P systems. Further biological details should be considered in order to have a model with some significance for the brain study (computer simulations will then be necessary, like in the case of other applications of P systems in modeling biological processes). However, also at this formal level there are several problems to consider. For instance, what happens if the sleeping period is shortened, e.g., because a signal comes from the environment? Can this lead to a "damage" of the system? In general, what about taking the environment into account? For instance, we can consider a larger system, where some modules sleep while other modules not; during the awake period it is natural to assume that the modules interact, but not when one of them is sleeping, excepting the case of an "emergency", when a sleeping module can be awakened at the request of a neighboring module. Several similar scenarios can be imagined, maybe also coupling the sleep-awake issue with the memory issue.

I. Let us now pass to general P systems, where still there are many unsettled problems and, what is more important for the health of membrane computing, there still appear new research directions. One of them was proposed and preliminarily explored in [23]: space complexity for P systems. It is somewhat surprising that this classic complexity issue was not considered so far, while the time complexity is already intensively investigated. We do not enter here into any detail – anyway, the details are moving fast in this area – but we refer to the mentioned paper.

J. Remaining in the framework of complexity, it is worth noting a sound recent result, the first one of this type, dealing with the relation between uniform and semi-uniform solutions to (computationally hard) problems. In [24] one produces a first example where semi-uniform constructions cover a strictly larger family than the uniform one. There also are classes of P systems for which the two approaches, uniform and semi-uniform, characterize the same family of problems – see, e.g., [22]. However, as also in [24] is stated, it is highly possible – and definitely of interest to find it – that the separation between uniform and semi-uniform families is strict for further classes of P systems.

K. Similarly short formulations (actually, only pointing out the problem and the main reference, always from the present volume) will also have the next problems. First, the idea of merging two "classic" topics in membrane computing: P systems with active membranes and P systems with string objects. Up to now only P systems with active membranes and with symbol objects were considered, but the necessity to mix the two ideas arose in [2] in the framework of a specific application. It is also natural to investigate this kind of systems as a goal per se, mainly from the complexity (descriptional and computational – and here both time and space) points of view.

L. A related general problem is suggested by the above paper [2]: looking for other "practical" problems, of known polynomial complexity, but which are so frequent and important that as efficient as possible solutions are desirable. The mentioned paper considers such a problem: search and updating of dictionaries. Many other problems can probably be addressed in terms of membrane computing, with good (theoretical) results. And, as it happens also in the case of [2], it is also possible that new classes of P systems will emerge in this context, as requested by the respective applications.

M. A problem several times mentioned, also addressed from various angles in several places concerns the general idea of "going backwards". Sometimes it is directly called computing backwards, as in [8], other times one speaks about dual P systems, as in [1], about reversible P systems, as in [12]. In the present volume, the reversibility is addressed again, in [4], but the question still deserves further research efforts.

N. I close this short list of research topics by mentioning the one proposed in [14] – the title is self-explanatory, so I do not enter into other details, but I mention that the issue seems rather promising, both theoretically (composition-decomposition, structural/descriptional complexity, etc.) and from the applications point of view.

Lists of open problems were presented in all previous brainstorming volumes – the reader can find them at [25]. There also exists an attempt to follow the research and to present the known solutions to some of these problems – see [19].

Acknowledgements

Work supported by Proyecto de Excelencia con investigador de recognoscida valia, de la Junta de Andalucia, grant P08 – TIC 04200.

References

1. O. Agrigoroaiei, G. Ciobanu: Dual P systems. *Membrane Computing - 9th International Workshop*, LNCS 5391, 2009, 95–107.
2. A. Alhazov, S. Cojocaru, L. Malahova, Y. Rogozhin: Dictionary search and update by P systems with string-objects and active membranes. In the present volume.
3. A. Alhazov, M. Margenstern, S. Verlan: Fast synchronization in P systems. *Membrane Computing, 9th Intern. Workshop. Edinburgh, UK, July 2008* (D.W. Corne et al eds.), LNCS 5391, Springer, Berlin, 2009, 118–128.
4. A. Alhazov, K. Morita: A short note on reversibility in P systems. In the present volume.
5. F. Bernardini, M. Gheorghe, M. Margenstern, S. Verlan: How to synchronize the activity of all components of a P system? *Prof. Intern. Workshop Automata for Cellular and Molecular Computing* (G. Vaszil, ed.), Budapest, Hungary, 2007, 11-22.
6. E. Csuhaj-Varju, A. Di Nola, Gh. Păun, M.J. Pérez-Jiménez, G. Vaszil: Editing configurations of P systems, *Third Brainstorming Week on Membrane Computing*, Sevilla, 2005, RGNC Report 01/2005, 131–155, and *Fundamenta Informaticae*, 82, 1-2 (2008), 29–46.
7. R. Galli, A. Gritti, L. Bonfanti, A.L. Vescovi: Neural stem cells: an overview. *Circulation Research*, 92 (2003), 598–608.

8. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez: Computing backwards with P systems. In the present volume.
9. O.H. Ibarra, A. Păun, Gh. Păun, A. Rodriguez-Patón, P. Sosik, S. Woodworth: Normal forms for spiking neural P systems. *Theoretical Computer Sci.*, 372, 2-3 (2007), 196–217.
10. M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems. *Fundamenta Informaticae*, 71, 2-3 (2006), 279–308.
11. M. Ionescu, D. Sburlan: Some applications of spiking neural P systems. *Proc. WMC8*, Thessaloniki, June 2007, 383–394, and *Computing and Informatics*, 27 (2008), 515–528.
12. A. Leporati, C. Zandron, G. Mauri: Reversible P systems to simulate Fredkin circuits. *Fundam. Inform.*, 74, 4 (2006), 529–548.
13. J.M. Mingo: Sleep-awake switch with spiking neural P systems: A basic proposal and new issues. In the present volume.
14. R. Nicolescu, M.J. Dinneen, Y.-B. Kim: Structured modeling with hyperdag P systems: Part A. In the present volume.
15. L. Pan, Gh. Păun: New normal forms for spiking neural P systems. In the present volume.
16. L. Pan, Gh. Păun, M.J. Pérez-Jiménez: Spiking neural P systems with neuron division and budding: In the present volume.
17. Gh. Păun: *Membrane Computing. An Introduction.* Springer, Berlin, 2002.
18. Gh. Păun: Twenty six research topics about spiking neural P systems. *Fiftth Brainstorming Week on Membrane Computing* (M.A. Gutiérrez-Naranjo, et al., eds.), Sevilla, January 29-February 2, 2007, Fenix Editora, Sevilla, 2007, 263–280.
19. Gh. Păun: Tracing some open problems in membrane computing, *Romanian J. Information Sci. and Technology*, 10, 4 (2007), 303–314.
20. Gh. Păun, M. Pérez-Jiménez, A. Riscos-Núñez: Tissue P systems with cell division. *Second Brainstorming Week on Membrane Computing* (A. Riscos-Núñez et al. eds.), Technical Report 01/04 of RGNC, Sevilla University, Spain, 2004, 380–386, and *Intern. J. Computers. Comm. Control*, 3, 3 (2008), 295–303.
21. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *Handbook of Membrane Computing.* Oxford University Press, 2009.
22. M.J. Pérez-Jiménez, A. Riscos-Núñez, Á. Romero-Jiménez, D. Woods: Complexity – Membrane division, membrane creation. In [21], in press.
23. A.E. Porreca, A. Leporati, G. Mauri, C. Zandron: Introducing a space complexity measure for P systems. In the present volume.
24. D. Woods, N. Murphy:
25. The P Systems Website: `http://ppage.psystems.eu`.