Computing Efficiency in Membrane Systems

Claudio Zandron
DISCo - Università degli Studi di Milano-Bicocca
Viale Sarca 336/14, 20126 Milano, Italy
Email: claudio.zandron@unimib.it

***Abstract*—Membrane systems (or P systems) are a computational model inspired by the functioning of the cell, and based upon the notion of cellular membrane. In this paper, we give a survey of some main results concerning the model, to show its potential to approach various problems in the area of the theory of computation.**

Keywords-Natural Computing; Membrane systems; computational complexity.

I. Introduction

Membrane systems (also known as *P systems*) have been proposed by Gh. Paun in [14] as a parallel, nondeterministic, synchronous and distributed model of computation inspired by the structure and functioning of living cells. The model consists of a hierarchical structure composed by several membranes, embedded into a main membrane called the *skin*. Membranes defines *regions* that contain multisets of *objects* (represented by symbols of an alphabet) and *evolution rules*.

Using these rules, the objects evolve and are moved from a region to a neighboring one. The rules are applied using a maximally parallel and nondeterministic semantic: all objects which can evolve in a computation step must evolve; if different sets of rules can be applied in a computation step (in a maximal parallel way), then one of them is nondeterministically chosen.

A *computation* starts from an initial configuration of the system and terminates when no evolution rule can be applied. The result of a computation is the multiset of objects contained into an *output membrane*, or emitted from the skin membrane.

The model was investigated both under theoretical aspects as well as for applications to other disciplines. In particular, the formalism is suitable to model various biological systems, thanks to its features. For instance, it allows identification of separated compartmentalized spaces where different reactions can take place; it is characterized by an easy understandable writing of reactions; it can be easily simulated in a distributed and parallel computing architecture, by separating the computation carried on by each single compartment, and simply syncronizing the exchange of information at specific time-steps. Other types of application were also investigates to different disciplines, such as cryptography, approximate optimization, or even economy. For a systematic introduction to P systems, we refer the reader to [16] [17]; recent information can be found in the dedicated webpage [32].

P systems with active membranes is a variant of the basic model introduced in [15]: in this variant, membranes can be multiplied by dividing existing ones, and the objects are communicated according to electrical charges associated with the membranes. Such features allow the construction of an exponential workspace in linear time, which can then be used in parallel to attack computationally hard problems.

In this paper we give a survey of some main results concerning theoretical aspects of the model, to show its potential in approaching various problems in the area of the theory of computation. In Section 2, we recall formal definitions related to P systems with active membranes. In Section 3, we give a very brief summary of results concerning their computating power, while in Section 4 we recall some results related to computing efficiency. Section 5 is devoted to recall some results concerning space complexity for membrane systems. Finally Section 6 draws some conclusions and suggest a few topics for research investigation.

II. Definitions

In this section, we recall the basic definition of P systems with active membranes.

Definition 1: A *P system with active membranes* of initial degree $d \geq 1$ is a tuple $\Pi = (\Gamma, \Lambda, \mu, w_1, \ldots, w_d, R)$, where:

- Γ is an alphabet, a finite non-empty set of symbols, usually called *objects*;
- Λ is a finite set of labels for the membranes;
- μ is a membrane structure (i.e., a rooted *unordered* tree) consisting of d membranes enumerated by $1, \ldots, d$; each membrane is labeled by an element of Λ, not necessarily in a one-to-one way, and possesses an *electrical charge* (or polarization), that can be neutral (0), positive (+) or negative ($-$).
- $w_1, \ldots, w_d$ are strings over Γ, describing the initial multisets of objects placed in the d regions of μ;
- R is a finite set of rules.

The rules are of the following kinds:

- *Object evolution rules*, of the form $[a \to w]_h^\alpha$
  They can be applied if the membrane h has charge α and contains an occurrence of the object a; the object a is rewritten into the multiset w.
- *Send-in communication rules*, of the form $a\,[\;]_h^\alpha \to [b]_h^\beta$
  They can be applied to a membrane labeled by h, having charge α and if the external region contains an occurrence of the object a; the object a is sent into h becoming b and, simultaneously, the charge of h is changed to β.
- *Send-out communication rules*, of the form $[a]_h^\alpha \to [\;]_h^\beta\, b$
  They can be applied to a membrane labeled by h, having charge α and containing an occurrence of a; the object a is sent out from h to the outside region becoming b. Simultaneously, the charge of h is changed to β.

- *Dissolution rules*, of the form $[a]_h^\alpha \to b$
  They can be applied to a membrane labeled by h, having charge α and containing an occurrence of the object a; the membrane h is dissolved and its contents are left in the surrounding region unaltered, except that an occurrence of a becomes b.
- *Elementary division rules*, of the form $[a]_h^\alpha \to [b]_h^\beta\,[c]_h^\gamma$
  They can be applied to a membrane labeled by h, having charge α, containing an occurrence of the object a but having no other membrane inside (an *elementary membrane*); the membrane is divided into two membranes having label h and charge β and γ; the object a is replaced, respectively, by b and c while the other objects in the initial multiset are copied to both membranes.
- *Nonelementary division rules*, of the form

$$\big[[\,]_{h_1}^+ \cdots [\,]_{h_k}^+ [\,]_{h_{k+1}}^- \cdots [\,]_{h_n}^-\big]_h^\alpha \to$$

$$\big[[\,]_{h_1}^\delta \cdots [\,]_{h_k}^\delta\big]_h^\beta \big[[\,]_{h_{k+1}}^\epsilon \cdots [\,]_{h_n}^\epsilon\big]_h^\gamma$$

  They can be applied to a membrane labeled by h, having charge α, containing the positively charged membranes $h_1, \ldots, h_k$, the negatively charged membranes $h_{k+1}, \ldots, h_n$, and possibly some neutral membranes. The membrane h is divided into two copies having charge β and γ, respectively; the positively charged membranes $h_1, \ldots, h_k$ are placed inside the former membrane, their charge set to δ, while the negative ones are placed inside the latter membrane, their charges set to ϵ. Neutral membranes inside h are duplicated and placed inside both copies.

Each instantaneous configuration of a P system with active membranes is described by the current membrane structure, including the electrical charges, together with the multisets located in the corresponding regions. A computation step changes the current configuration according to the following set of principles:

- Each object and membrane can be subject to at most one rule per step, except for object evolution rules (inside each membrane any number of evolution rules can be applied simultaneously).
- The application of rules is *maximally parallel*: each object appearing on the left-hand side of evolution, communication, dissolution or elementary division rules must be subject to exactly one of them (unless the current charge of the membrane prohibits it). The same reasoning applies to each membrane that can be involved to communication, dissolution, elementary or nonelementary division rules. In other words, all possible rules that can be applied must be applied at each computation step; the only objects and membranes that do not evolve are those associated with no rule, or only to rules that are not applicable due to the electrical charges.
- When several conflicting rules can be applied at the same time, a nondeterministic choice is performed; this implies that, in general, multiple possible configurations can be reached after a computation step (e.g., consider two rules $a \to b$ and $a \to c$ in a region h; if an object a is present in that region, then it can nondeterministically produce either b or c, by using respectively the first or the second rule).
- While all the chosen rules are considered to be applied simultaneously during each computation step, they are logically applied in a bottom-up fashion: first, all evolution rules are applied to the elementary membranes, then all communication, dissolution and division rules; then the application proceeds towards the root of the membrane structure. In other words, each membrane evolves only after its internal configuration has been updated.
- The outermost membrane cannot be divided or dissolved, and any object sent out from it cannot re-enter the system again.

A *halting computation* of the P system Π is a finite sequence of configurations $\vec{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$, where $\mathcal{C}_0$ is the initial configuration, every $\mathcal{C}_{i+1}$ is reachable by $\mathcal{C}_i$ via a single computation step, and no rules can be applied anymore in $\mathcal{C}_k$. The result of a halting computation is the multiset of objects emitted from the skin during the whole computation. A *non-halting* computation $\vec{\mathcal{C}} = (\mathcal{C}_i : i \in \mathbb{N})$ consists of infinitely many configurations, again starting from the initial one and generated by successive computation steps, where the applicable rules are never exhausted. A non–halting computation produces no output.

P systems can also be used as *recognizers* (see, e.g., [3]) by employing two distinguished objects $\texttt{yes}$ and $\texttt{no}$; exactly one of these must be sent out from the outermost membrane during each computation, in order to signal acceptance or rejection respectively; we also assume that all computations are halting. If all computations starting from the same initial configuration are accepting, or all are rejecting, the P system is said to be *confluent*. If this is not necessarily the case, then we have a *non–confluent* P system, and the overall result is established as for nondeterministic Turing machines: it is acceptance iff an accepting computation exists. All P systems considered in this paper are confluent.

In order to solve decision problems (i.e., decide languages), we use *families* of recognizer P systems $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$. Each input x is associated with a P system Π_x that decides the membership of x in the language $L \subseteq \Sigma^\star$ by accepting or rejecting. The mapping $x \mapsto \Pi_x$ must be efficiently computable for each input length [13].

Definition 2: A family of P systems $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ is said to be *(polynomial-time) uniform* if the mapping $x \mapsto \Pi_x$ can be computed by two deterministic polynomial-time Turing machines F (for "family") and E (for "encoding") as follows:

- The machine F, taking as input *the length n of x* in unary notation, constructs a P system Π_n, which is common for all inputs of length n, with a distinguished input membrane.
- The machine E, on input x, outputs a multiset w_x (an encoding of x).
- Finally, Π_x is simply Π_n with w_x added to the multiset placed inside its input membrane.

Definition 3: If the mapping $x \mapsto \Pi_x$ is computed by a *single* polynomial-time Turing machine, the family $\mathbf{\Pi}$ is said to be *semi-uniform*. In this case, inputs of the same size may be associated with P systems having possibly different membrane structures and rules.

Any explicit encoding of Π_x is allowed as output of the construction, as long as the number of membranes and objects represented by it does not exceed the length of the whole description, and the rules are listed one by one. This restriction is enforced to mimic a (hypothetical) realistic process of construction of the P system, where membranes and objects are placed in a constant amount during each construction step, and require actual physical space proportional to their number. Moreover, notice that uniformity condition can also be restricted to be computed in classes below $\mathbf{P}$, such as log–space Turing machines. We refer the reader to [13] for further details on the encoding of P systems.

Finally, we describe how time and space complexity for families of recognizer P systems are measured.

Definition 4: A uniform or semi–uniform family of P systems $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ is said to decide the language $L \subseteq \Sigma^\star$ (in symbols $L(\mathbf{\Pi}) = L$) in time $f\colon \mathbb{N} \to \mathbb{N}$ iff, for each $x \in \Sigma^\star$,

- the system Π_x accepts if $x \in L$, and rejects if $x \notin L$;
- each computation of Π_x halts within $f(|x|)$ computation steps.

Definition 5: Let $\mathcal{C}$ be a configuration of a P system Π. The *size* $|\mathcal{C}|$ *of* $\mathcal{C}$ is defined as the sum of the number of membranes in the current membrane structure and the total number of objects they contain. If $\vec{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ is a halting computation of Π, then the *space required by* $\vec{\mathcal{C}}$ is defined as

$$|\vec{\mathcal{C}}| = \max\{|\mathcal{C}_0|, \ldots, |\mathcal{C}_k|\}$$

or, in the case of a non-halting computation $\vec{\mathcal{C}} = (\mathcal{C}_i : i \in \mathbb{N})$,

$$|\vec{\mathcal{C}}| = \sup\{|\mathcal{C}_i| : i \in \mathbb{N}\}.$$

Non-halting computations might require an infinite amount of space (in symbols $|\vec{\mathcal{C}}| = \infty$). The *space required by* Π itself is then

$$|\Pi| = \sup\{|\vec{\mathcal{C}}| : \vec{\mathcal{C}} \text{ is a computation of } \Pi\}.$$

Notice that $|\Pi| = \infty$ occurs if either Π has a non-halting computation requiring infinite space, or Π has an infinite set of halting computations, such that for each bound $b \in \mathbb{N}$ there exists a computation requiring space larger than b.

III. Computing Power of Membrane Systems

The first studies of the model concerned its computing power: various types of Membrane systems have been compared with computing models like automata, Turing machines and register machines.

It is known that using a single membrane we can only generate the length sets of context–free languages, and the power cannot be extended by using an unlimited number of membranes. However, if we allow to dissolve membranes after the application of rewriting rules then the computing power is increased, when at least two membranes are used.

More formally, let us denote by $NOP_k(\delta)$ (resp. $NOP_k(n\delta)$) the family of natural numbers generated by P systems having k membranes and using (resp. not using) the dissolving membrane action. The following results can be stated [16]:

Theorem 1: $NOP_1(n\delta) = NOP_*(n\delta) = NCF$
$NCF = NOP_*(n\delta) \subset (NE0L \subseteq) NOP_2(\delta)$
$NOP_*(\delta)(\subseteq ET0L) \subset NCS$

Even when dissolving membrane action is allowed, universality cannot be reached. Some more ingredients can be considered to obtain such a result like, e.g., cooperative rules, catalysts, or priorities defining the order of rules application.

A rewriting rule $a \to w$ is said to be cooperative if a contains more than one symbol. This turned out to be a feature very powerful in the framework of membrane systems. In fact, when using such rules one membrane turns out to be sufficient to obtain the same power as Turing machines:

Theorem 2: $NOP_1(coop) = NOP_*(coop) = NRE$

where *coop* indicates the possibility to use cooperative rules.

A simpler form of cooperative rules can be defined by means of catalysts. A rewriting rule with catalyst is a rule of the form $CX \to Cw$, where C and X are symbols and w is a string. C is said to be a catalyst: it is needed to activate the rule, but it is not changed by it. Such a feature also allows to obtain universal systems, but only when priorities defining a partial order concerning the application of the rules is also used:

Theorem 3: $NOP_2(cat, pri) = NOP_*(cat, pri) = NRE$

One can also consider structured objects instead of atomic ones, by consider strings of symbols: in this case, the systems are called Rewriting P systems. Let us denote by RP_k the family of languages generated by Rewriting P systems using k membranes and context–free rewriting rules. The following theorem from [16] shows that using a single membrane only context–free languages can be obtained, but a structure with four membranes allow to obtain a strictly more powerful class.

Theorem 4: $RP_1(CF) = CF \subset RP_4(CF)$

Thus, it is evident from these results that the power of such systems can be improved (as expected) by exploiting membranes to define regions to keep separated specific subsets of rules and objects. Once again, universality cannot be obtained using only this basic set of ingredients, and more features must be considered.

Further details can be found in [16] and [17].

IV. Computing Efficiency of Membrane Systems

Another interesting feature that can be considered, and already described in Section 2, is the possibility to give an active role to membranes. P systems with active membranes allow to create new membranes during the computation by division of existing membranes. In this way, we can obtain a trade off between time and space resources that allows to solve NP–complete (or even harder) problems in polynomial time and exponential space (see, e.g., [15] [7] [8] [29] [31]).

Theorem 5: The SAT problem can be solved in linear time (with respect to the number of variables and the number of clauses) by a confluent P-system with active membranes using elementary membrane division only.

In fact, consider a boolean expression Φ in conjunctive normal form, with m clauses and n variables. We can build a P-system $\Pi = (\Gamma, \Lambda, \mu, w_1, w_2, R)$ having initial objects $a_1, a_2, \ldots, a_n$ in region 2 and such that R is defined to contain a polynomial number of rules (with respect to the size of the input formula) that operate as it follow.

By using the variables a_i and elementary membrane division rules, in $O(n)$ steps we generate 2^n copies of membrane 2, containing all possible truth assignments of the n variables of Φ.

Then, in $O(m)$ steps we verify if there is at least one membrane containing a truth assignment that satisfies all the m clauses of Φ. In this case, an object yes is sent out from the skin membrane; otherwise, an object no is sent out.

Let us denote by $PMC_{\mathcal{NAM}}$, $PMC_{\mathcal{EAM}}$, and $PMC_{\mathcal{AM}}$ the class of problems solved in a polynomial number of steps (with respect to the input length) by P systems with active membranes without membrane division, with division for elementary membranes only, and for both elementary and non–elementary membranes, respectively.

The following results can be obtained directly from definitions:

Theorem 6: $PMC_{\mathcal{N}AM} \subseteq PMC_{\mathcal{E}AM} \subseteq PMC_{\mathcal{A}M}$

Moreover, it is easy to show that

Theorem 7: $P \subseteq PMC_{\mathcal{N}AM}$

In fact, the "trick" is that the deterministic Turing machine deciding $L \in P$ is used to solve directly the problem in polynomial time. Then, we build a P system with a single membrane containing either an object YES, whenever an input $x \in L$ is given, or NO, otherwise. This requires polynomial time, and then the P system simply send out the object in a single computation step.

The opposite inclusion is also true:

Theorem 8: $PMC_{\mathcal{N}AM} \subseteq P$

In fact, a generic P system Π without membrane division can be simulated by a deterministic Turing machine M, with a polynomial slowdown, as proved in [29].

Since we have shown that the NP-complete problem SAT can be solved when elementary membrane division is allowed, then we can also state the following:

Theorem 9: $NP \subseteq PMC_{\mathcal{EAM}}$

From this result and from the closure properties for $PMC_{\mathcal{EAM}}$ it also follows:

Theorem 10: $coNP \subseteq PMC_{\mathcal{EAM}}$

P systems with elementary membrane division can be simulated by Deterministic Turing machines using polynomial space: $PMC_{\mathcal{E}AM} \subseteq PSPACE$. Hence:

Theorem 11: $NP \cup coNP \subseteq PMC_{\mathcal{E}AM} \subseteq PSPACE$

A stronger result was later proved [19]: a solution to the PP–complete problem SQRT–3SAT was obtained using P systems with active membranes and elementary membrane division. This proved that the class PP (Probabilistic Polynomial time: the class of decision problems solvable by a probabilistic Turing machine in polynomial time, with an error probability of less than 1/2 for all instances) is also included in $PMC_{\mathcal{EAM}}$. A characterization of the class $P^{\#P}$ was obtained in [10].

When division for non-elementary membranes is allowed, even harder problems can be solved, as expected. In a series of papers [2] [26] and [24] the following results were proved:

Theorem 12: $PSPACE \subseteq PMC_{\mathcal{AM}} \subseteq EXPTIME$

By limiting the nesting levels of membranes (and, as a consequence, the membrane division) to a constant depth, the problems in the class CH (Counting Hierarchy) can be solved, as proved in [23].

V. Space complexity of P–systems and polarization of membranes

In order to clarify relations between the amount of time and space needed to solve various classes of problems, in [18] a definition of space complexity for P systems has been introduced.

On the same line of what has been done for time complexity, we can define space complexity classes for Membrane systems. By $MCSPACE_T(f)$ we denote the class of languages decided by confluent recognizer P systems (of type T) within space $f(n)$. In particular, by $PMCSPACE_T (= MCSPACE_T^{[*]}(p(n)))$ we denote the class of languages decided by confluent recogniser P systems using at most a polynomial number of elements.

From the definitions and from the results we recalled in the previous section, it is easy to see that:

- $P \subseteq MCSPACE_{\mathcal{N}AM}(O(1))$
- $NP \cup co-NP \subseteq EXPMCSPACE_{\mathcal{E}AM}$
- $PSPACE \subseteq EXPMCSPACE_{\mathcal{A}M}$

In [20] and [21] it has been shown, respectively, that the $PSPACE$–complete problem Quantified–3SAT can be solved by P–systems with active membranes using a polynomial amount of space, and that such P systems can be simulated by Turing machines with only a polynomial increase in space requirements, thus giving a precise characterization of the class $PSPACE$ in terms of space complexity classes for membrane systems. A similar result to characterize the complexity class $EXPSPACE$ can be obtained by considering exponential space P systems, as showed in [1]. Thus, all types of Membrane systems with active membranes and both divisions for elementary and non-elementary membranes, and working in a polynomial space, exactly characterize $PSPACE$.

Investigation of classes of problems solved by P systems using sublinear space was also considered. In order to consider sublinear space, two distinct alphabets were considered in the definition of P systems: an *INPUT* alphabet and a *WORK* alphabet. Objects from the INPUT alphabet cannot be rewritten and do not contribute to the size of the configuration of a P system. The size of a configuration was defined as the sum of the number of membranes in the current membrane structure and the total number of working objects they contain; the space required by a computation is the maximum size among all configurations. Moreover, we need to define a uniformity condition for the families of P systems that is weaker than the usual **P** uniformity, to avoid the possibility to solve a problem directly by using the Turing machine that build the P systems we use to compute. We consider **DLOGTIME**-uniformity, defined on the basis of **DLOGTIME** Turing machines [12]. We refer the reader to [25] for formal definitions.

The efficient simulation of logarithmic space Turing machines (or other equivalent models) by employing standard techniques used in the papers previously cited seems not to work because of two main problems: we either need to use a polynomial number of working objects (thus violating the logarithmic space condition) or to use a polynomial number of rewriting rules (thus violating the uniformity condition). Nonetheless, it has been showed in [25] that such a simulation can be efficiently done by using membrane polarization both to communicate objects through membranes as well as to store some information:

Theorem 13: Each log–space deterministic Turing machine M can be simulated by a **DLOGTIME**-uniform family $\mathbf{\Pi}$ of confluent recognizer P systems with active membranes in logarithmic space.

An immediate corollary of Theorem 13 is that the class L (the class of problems solved by log–space Turing machines) is contained in the class of problems solved by **DLOGTIME**-uniform, log–space P systems with active membranes.

VI. Conclusions

We survey some results concerning P systems with active membranes, concerning both the computational aspects, as well as computational efficiency. Some results concerning space complexity of the model have also been recalled. For further reading on the subject, we refer the reader to the main volumes on the subject [16] [17].

A recent survey on different strategies to approach computationally hard problems (containing links to various research paper on the subject) by P systems with active membranes can be found in [27].

Links to various paper concerning space complexity for membrane systems are available in [20] [21]; readers interested in results concerning sublinear space or even constant amount of space can refer to [22] and [9], respectively. A recent survey concerning results obtained by considering different bounds on space can be found in [28].

There are various research topics which deserve to be investigated with respect to computing efficiency. In particular, precise characterizations of complexity classes obtained by considering systems using specific subset of features, as well as relations of such classes with standard complexity classes, both concerning time complexity and space complexity.

More general research directions actually under development concern the application of the model to describe different biological processes. As already pointed out, since Membrane systems are a bio-inspired computing model, it is natural to apply it to the description and simulation of complex bio-processes, to gain various kind of information on such processes.

Another interesting research directions concern the link of P systems with neural computing: a variant of neural-like P systems have been introduced in [6] under the name of Spiking P systems, and its application to the field of Artificial Intelligence and deep learning strategies is currently under development.

Acknowledgements

This work was partially supported by Università degli Studi di Milano-Bicocca, Fondo di Ateneo Quota Dipartimentale (FAQD-2019).

References

[1] A. Alhazov, A. Leporati, G. Mauri, A. E. Porreca, C. Zandron, Space complexity equivalence of P systems with active membranes and Turing machines, Theoretical Computer Science 529, 2014, 69—81

[2] A. Alhazov, C. Martin-Vide, L. Pan, Solving a PSPACE–complete problem by P–systems with restricted active membranes, Fund. Inf. 58, 2, 2003, 67–77

[3] E. Csuhaj-Varju, M. Oswald, Gy. Vaszil, P automata, Handbook of Membrane Computing, Gh. Paun et al. (Eds.), Oxford University Press, 2010, 144–167

[4] M. Gutierrez-Naranjo, M.J. Perez-Jimenez, P–systems with active membranes, without polarizations and without dissolution: a characterization of P, Unconv. Comp. 2005, C.S. Calude et al. (eds.), LNCS 3699, Springer, 2005, 105-116.

[5] M. Gutierrez-Naranjo, M. J. Perez-Jimenez, A. Riscos-Nunez, F. J. Romero-Campero, Characterizing tractability by cell-like membrane systems, in K.G. Subramanian et al. (Eds.), Formal Models, Languages and Applications, Ser. Mach. Percept. Artif. Intell., vol. 66, World Scientific, 2006, 137-–154

[6] M. Ionescu, Gh. Paun, T. Yokomori, Spiking neural P systems, Fundamenta Informaticae, 71, 2-3, 2006, 279–308

[7] S. N. Krishna, R. Rama, A variant of P-systems with active membranes: Solving NP-complete problems, Rom. J. of Inf. Sci. and Tech., 2, 4 (1999)

[8] A. Leporati, C. Zandron, M. A. Gutierrez-Naranjo, P systems with input in binary form, Int. J. of Found. of Comp. Sci., 17(1), 2006, 127–146

[9] A. Leporati, L. Manzoni, G. Mauri, A. E. Porreca, C. Zandron, Constant-space P systems with active membranes, Fundamenta Informaticae 134(1–2), 2014, 111–128

[10] A. Leporati, L. Manzoni, G. Mauri, A.E. Porreca, C. Zandron, Simulating elementary active membranes with an application to the P conjecture, LNCS 8961, Springer, 2014, 284–299

[11] A. Leporati, G. Mauri, C. Zandron, Quantum Sequential P Systems with Unit Rules and Energy Assigned to Membranes, in R. Freund et al (eds.), Membrane Computing, 6th Int. Work., WMC 2005, Vienna, Austria, LNCS 3850, Springer, 2006, 310–325

[12] D.A. Mix Barrington, N. Immerman, H. Straubing, On uniformity within NC^1. Journal of Computer and System Sciences 41(3), 1990, 274–306

[13] N. Murphy, D. Woods, The computational power of membrane systems under tight uniformity conditions, Natural Computing 10(1), 2011, 613–632

[14] Gh. Păun, Computing with membranes, J. of Computer and System Sciences, 61(1), 2000, 108–143

[15] Gh. Păun, P systems with active membranes: Attacking NP-complete problems, J. of Automata, Languages and Combinatorics 6(1), 2001, 75–90

[16] Gh. Păun, Membrane Computing. An Introduction, Springer, Berlin, 2002

[17] Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Handbook of Membrane Computing, Oxford University Press, 2010

[18] A. E. Porreca, A. Leporati, G. Mauri, C. Zandron, Introducing a space complexity measure for P systems, Int. J. of Comp. Comm. and Control, 4(3), 2009, 301–310

[19] A. E. Porreca, A. Leporati, G. Mauri, C. Zandron, P systems with Elementary Active Membranes: Beyond NP and coNP, in Gheorghe M. et al. (eds.), CMC 2010, Jena, Germany, August 2010, LNCS 6501, Springer, 2011, 338–347

[20] A. E. Porreca, A. Leporati, G. Mauri, C. Zandron, P Systems with Active Membranes: Trading Time for Space, Natural Computing 10(1), 2011, 167–182

[21] A. E. Porreca, A. Leporati, G. Mauri, C. Zandron, P systems with active membranes working in polynomial space, Int. J. Found. Comp. Sc., 22(1), 2011, 65–73

[22] A. E. Porreca, A. Leporati, G. Mauri, C. Zandron, Sublinear-space P systems with active membranes. In E. Csuhaj-Varjú, M. Gheorghe, G. Rozenberg, A. Salomaa, G. Vaszil, G. (eds.) Membrane Computing, 13th International Conference, CMC 2012, LNCS 7762, 2013, 342–357

[23] A. E. Porreca, L. Manzoni, A. Leporati, G. Mauri, C. Zandron, Membrane division, oracles, and the counting hierarchy, Fundamenta Informaticae, 138, 1-2, 2015, 97–111

[24] A. E. Porreca, G. Mauri, C. Zandron, Complexity classes for membrane systems, RAIRO-Theor. Inform. and Applic. 40(2), 2006, 141–162
[25] A. E. Porreca, C. Zandron, A. Leporati, G. Mauri, Sublinear Space P systems with Active Membranes, Membrane Computing: 13th International Conference, LNCS, CMC 2012, Springer, Berlin, 2013, 342–357
[26] P. Sosik, The computational power of cell division in P systems: Beating down parallel computers?, Natural Computing, 2(3), 2003, 287–298
[27] P. Sosik, P systems attacking hard problems beyond NP: a survey. Journal of Membrane Computing 1(3), 2019, 198—208
[28] C. Zandron, Bounding the space in P systems with active membranes, Journal of Membrane Computing 2(2), 2020, 137–145
[29] C. Zandron, C. Ferretti, G. Mauri, Solving NP-complete problems using P systems with active membranes, In I. Antoniou, C.S. Calude, M.J. Dinneen, eds., Unconventional Models of Computation, Springer–Verlag, London, 2000, 289–301
[30] C. Zandron, C. Ferretti, G. Mauri, Two Normal Forms for Rewriting P systems, in Machines, Computations and Universality, Proc. of 3rd Int. Conf. MCU 2001, LNCS 2055, Springer-Verlag, 2001, 153–164
[31] C. Zandron, A. Leporati, C. Ferretti, G. Mauri, M. J. Pérez-Jiménez, On the Computational Efficiency of Polarizationless Recognizer P Systems with Strong Division and Dissolution, Fundamenta Informaticae, 87(1), 2008, 79-91
[32] The P systems Web page: http://ppage.psystems.eu/