Computing Efficiency in Membrane Systems and Their Variants

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. Some main variants are also recalled, as well as some main actual research directions.**

Keywords-Natural Computing; Membrane systems; computational complexity.

I. Introduction

The present paper is an extended version of [1], presented an the Thirteenth International Conference on Future Computational Technologies and Applications, April 18 - 22, 2021, in Porto, Portugal, and contains and introduction and some main result concerning membrane systems (as in the conference paper), as well as the description of some main variants of the basic model, and some main actual research directions.

Membrane systems (also known as *P systems*) have been proposed by Gh. Paun in [2] 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 synchronizing the exchange of information at specific time-steps. Other types of application were also investigates, such as cryptography, approximate optimization, and economy.

For a systematic introduction to P systems, we refer the reader to [3] [4]; recent information can be found in the dedicated webpage [5].

P systems with active membranes is a variant of the basic model introduced in [6]: 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 computing 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. In Section 6, we present some main variants of the basic model and, finally, Section 7 draws some conclusions and suggests 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

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

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

  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., [7]) by employing two distinguished objects `yes` and `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 [8].

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 [8] 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, or among themselves, also considering normal forms [9] for systems.

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 [3]:

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.

In order to show the functioning of the model, we summarize in the following the main ideas behind this result. Let us consider a register machine, a well-known computationally complete device [10], consisting of n registers $r_1, \ldots, r_n$, each one containing a non-negative integer. The input is placed in register r_1, while all the other registers are initially set to zero. The machine has a set of instructions (labelled by natural numbers) of two kinds: increment instruction $i : inc(r), j$ and conditional decrement instruction $i : dec(r), j, k$. A program counter (initially set to 1) define the instruction to execute.

An increment instruction increase by one the contents of register r, and set the program counter to j, while the result of a decrement instruction depends on the value of register r. If r is greater than zero, then it is decremented by one and the program counter is set to j; if r contains zero, then no decrement is applied and the program counter is set to k. The computation halts when the program counter reaches a value associated with a specific halting instruction (setting the program counter to zero), and the contents of r_1 is the output of the computation.

To simulate a register machine, we can use some symbols p_i, to store the actual value of the program counter (only one symbol of this type can be present at each time step, and a symbol r for each register r, used to store the value of the corresponding register (by considering the multiplicity of the symbol).

Using rewriting rules, an increment instruction of the register machine can be simulated by a membrane system by means of a single rewriting rule of the form $p_i \rightarrow p_j r$. In by applying such a rule, the actual program counter p_i is replaced by p_j, and the contents of register r is incremented by one by adding a symbol r.

A conditional decrement instruction can be simulated by the following rules. First, we apply $p_i \rightarrow p'_i d_r$; the symbol d_r will be used to check whether it is possible to delete an occurrence of r (if r is greater than zero) by means of the rule $d_r r \rightarrow d'_r$, while p'_i is replaced by p''_i. If the decrement is possible (i.e., there is at least one symbol r), then we have now the symbol d'_r, otherwise the symbol d_r is still present. The correct program counter can thus be set, by considering the only applicable rule between $p''_i d'_r \rightarrow p_j$ and $p''_i d_r \rightarrow p_k$.

A simpler form of cooperative rules can be defined by means of catalysts. A rewriting rule with catalyst is a rule of the form $CX \rightarrow Cw$, 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 [3] 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 [3] and [4].

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., [6] [11] [12] [13] [14] [15] [16]).

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: $PMC_{\mathcal{NAM}} \subseteq PMC_{\mathcal{EAM}} \subseteq PMC_{\mathcal{AM}}$

Moreover, it is easy to show that

Theorem 7: $P \subseteq PMC_{\mathcal{NAM}}$

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: $PMC_{\mathcal{NAM}} \subseteq P$

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 [14].

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: $PMC_{\mathcal{EAM}} \subseteq PSPACE$. Hence:

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

A stronger result was later proved [17]: 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 [18].

When division for non-elementary membranes is allowed, even harder problems can be solved, as expected. In a series of papers [19] [20] and [21] 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 [22].

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 [23] 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:

- $P \subseteq MCSPACE_{\mathcal{NAM}}(O(1))$
- $NP \cup co-NP \subseteq EXPMCSPACE_{\mathcal{EAM}}$
- $PSPACE \subseteq EXPMCSPACE_{\mathcal{AM}}$

In [24] and [25] 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 [26]. 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 [27]. We refer the reader to [28] 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 [28] 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.

The first definition of space, described above, was based on a hypothetical implementation of P systems by means of chemicals and molecules. Nonetheless, a different approach can also be considered to define space complexity for P systems, by focusing the definition of space on the simulative point of view. In fact, various implementation of P systems in silico have been proposed (see, e.g., [29], [30]) where the multiplicity of each object in each membrane are stored by means of binary numbers. This, of course, reduces the amount of requested space, thus possibly leading to different results related to complexity classes defined on the basis of this definition of space.

Such problems have been recently investigated in [31]. The first results show that, for different considered systems, the computational classes defined on the basis of this new definition of space do not differ from the corresponding classes defined on the basis of the original space definition. In particular, classes for systems using at least a linear amount of space have been considered so far. It is conjectured that differences between classes defined considering the two definitions of space can be highlighted when sublinear space is considered, like in the case of logarithmic space: an important open problem is to find specific classes where this difference exists, thus proving that efficient storing of information concerning objects can be exploited in some cases. It would also be interesting to understand which features can be used/are necessary to obtain the same result.

VI. Variants of Membrane systems

Apart from the basic model, strictly based on the internal organization of the cell, many variants have been proposed in the literature, by considering different features of the cell, or different organizations and connection of the whole system. We recall in the following some of the main variants and their features; in particular, we will recall Tissue-like P systems and Spiking Neural P systems.

Other variants have been proposed and investigated, like *population P systems*, *P automata*, *P colonies*, *metabolic P systems*, *numerical P systems* or P systems incorporating *probabilistic* or *quantum* notions, or their application in various fields has been discussed. As an example, in [32] an overview

of research related to the design of robot controllers by means of some variants of membrane systems is presented - namely, numerical P systems, enzymatic numerical P systems, P colonies, and XP colonies - and their performances compared to traditional controllers for robot systems. In [33] the notion of energy associated to membranes has been considered, to define quantum-like systems.

We refer the interested reader to the webpage [5] and to the Handbook [4] as a starting point for the various variants.

A. Tissue P Systems

A variant appeared quite early was that of *Tissue P systems*, proposed in [34] to highlight role of inter-cellular communication, exchanging substances between adjacent cells. A specific introduction to this variant can be found in chapter 9 of [4].

Definition 6: A *tissue P system* is a structure $\Pi = (\Gamma, E, w_1, \ldots, w_d, R)$, where:

- Γ is an alphabet, i.e., a finite non-empty set of symbols, usually called *objects*;
- $E \subseteq \Gamma$ is the alphabet of objects initially located in the external environment, in *arbitrarily many* copies (i.e., they are never exhausted during the computation);
- $d \geq 1$ is the *degree* of the system, i.e., the initial number of cells;
- $w_1, \ldots, w_d$ are *finite* multisets over Γ, describing the initial contents of the d cells; here $1, \ldots, d$ are *labels* identifying the cells of the P systems, and 0 is the label of the external environment;
- R is a finite set of rules.

The rules of R are of the following types:

(a) *Communication rules*, denoted by $(h, u/v, k)$, where h and k are distinct labels (including the environment), and u and v are multisets over Γ (at least one of them nonempty): these rules are applicable if there exists a region with label h containing u as a submultiset and a region k containing v as a submultiset; the effect of the application of the rule is to exchange u and v between the two regions. If $h = 0$ (resp., $k = 0$) then the rule is denoted by $(\lambda, u/v, k)$ (resp., $(h, u/v, \lambda)$), and in that case we require multiset u (resp., v) to contain at least an object from $\Gamma - E$, i.e., an object with finite multiplicity, if v (resp., u) is empty

(b) *Division rules*, of the form $[a]_h \rightarrow [b]_h[c]_h$, where $h \neq 0$ is a cell label and $a, b, c \in \Gamma$: these rules can be applied to a cell with label h containing at least one copy of a; the effect of the application of the rule is to divide the cell into two cells, both with label h; the object a is replaced in the two cells by b and c, respectively, while the rest of the original multiset contained in h is replicated in both cells.

(c) *Separation rules*, of the form $[a]_h \rightarrow [\Gamma_1]h[\Gamma_2]h$, where $h \neq 0$ is a cell label, $a \in \Gamma$, and $\{\Gamma_1, \Gamma_2\}$ is a partition of Γ: these rules can be applied to a cell with label h containing at least one copy of a; the effect of the application of the rule is to separate the cell into two cells, both with label h; the object a is consumed, while the objects from Γ_1 in the original multiset contained in h are placed inside one of the cells, and those from Γ_2 in the other. All separation rules in R must share the same partition $\{\Gamma_1, \Gamma_2\}$ of Γ.

A *tissue P system with cell division* only uses communication and division rules, while a *tissue P system with cell separation* only uses communication and separation rules.

A *configuration* $\mathcal{C}$ of a tissue P system consists of a multiset over $\Gamma - E$ describing the objects appearing with finite multiplicity in the environment, and a multiset of pairs (h, w), where h is a cell label and w a finite multiset over Γ, describing the cells. A *computation step* changes the current configuration according to the following set of principles:

- Each object can be subject to at most one rule, and each cell can be subject to *any number* of communication rules or, *alternatively*, a *single* division or separation rule.
- The application of rules is *maximally parallel*: each region is subject to a maximal multiset of rules (i.e., no further rule can be applied).
- 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.

A *halting computation* $\vec{\mathcal{C}} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ of the tissue P system Π is a finite sequence of configurations, where $\mathcal{C}_0$ is the initial configuration, every $\mathcal{C}_{i+1}$ is reachable from $\mathcal{C}_i$ via a single computation step, and no rules are applicable in $\mathcal{C}_k$.

Tissue P systems can be used as language *recognisers* by employing two distinguished objects `yes` and `no`: we assume that all computations are halting, and that one of the objects `yes` or `no` (but not both) is released into the environment, and only in the last computation step, in order to signal acceptance or rejection, respectively. If all computations starting from the same initial configuration are accepting, or all are rejecting, the tissue P system is said to be *confluent*. A confluent P system is said to *accept* if its computations are accepting, and to *reject* otherwise. Confluent tissue P systems may be locally nondeterministic, as long as all computations agree on the final result. As a special case, tissue P systems whose initial configuration generates a single computation are called *deterministic*.

Different results concerning computational aspects related to this variant have been presented in the literature (e.g. [35], [36], [37], [38] [39]). Cell division rules and cell separation rules have been considered in order to attack computationally hard problems (e.g. [40],[41], [42], [43]), exploiting the same ideas presented above for P systems with active membranes. It was shown that NP–complete problems can be solved in polynomial time, also in this case. Nonetheless, for this kind of systems it is not possible to reach enough power to solve $PSPACE$-complete problems, due to the missing structured organization of membranes. In fact, in turned out that such a model solves exactly all problems in the complexity class $\mathbf{P}^{\#\mathbf{P}}$, that is the class of problems that are solved by Turing machines using an oracle for counting problems [44].

B. Spiking Neural P Systems

Another variant that has been widely investigated are *spiking neural* (SN) P systems [45], inspired by the structure and

functioning of spiking neural systems. In this kind of a variant, there is a single objects, called the *spike*, communicated through connections called *synapses* among *neurons*.

Formally, a Spiking Neural P system of degree $m \geq 1$ is a construct

$$\Pi = (O, \sigma_1, \sigma_2, \ldots, \sigma_m, syn, in, out)$$

where:

1) $O = \{a\}$ is the singleton alphabet (a is called *spike*);
2) $\sigma_1, \sigma_2, \ldots, \sigma_m$ are *neurons*, of the form $\sigma_i = (n_i, R_i)$, with $1 \leq i \leq m$, where:
   a) $n_i \geq 0$ is the *initial number of spikes* contained in σ_i;
   b) R_i is a finite set of *rules* of the following two forms:
      (1) $E/a^c \rightarrow a; d$, where E is a regular expression over a, and $c \geq 1$, $d \geq 0$ are integer numbers; if $E = a^c$, then it is usually written in the following simplified form: $a^c \rightarrow a; d$;
      (2) $a^s \rightarrow \lambda$, for $s \geq 1$, with the restriction that for each rule $E/a^c \rightarrow a; d$ of type (1) from R_i, we have $a^s \notin L(E)$ (where $L(E)$ denotes the regular language defined by E);
3) $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$, with $(i, i) \notin syn$ for $1 \leq i \leq m$, is the directed graph of *synapses* between neurons;
4) $in, out \in \{1, 2, \ldots, m\}$ indicate the *input* and the *output* neurons of Π.

The rules of type (1) are called *firing rules*, while rules of type (2) are called *forgetting rules.*

If a neuron σ_i contains $k \geq c$ spikes, such that $a^k \in L(E)$, then the application of a spiking rule $E/a^c \rightarrow a; d \in R_i$ removes c spikes from σ_i (leaving $k - c$ spikes in the neuron), and prepares one spike to be sent to all neurons connected to σ_i by means of a synapse defined in syn. If the delay d is set to zero, then the spike is immediately emitted. On the contrary, the neuron waits d computation steps before sending out the spike. During the waiting time, the neuron is *closed*: it cannot receive new spikes (spikes sent to σ_i by other neurons are lost), and it cannot fire new rules. After d computation steps, the neuron sends the spike out and becomes open again, ready to receive spikes and to apply new rules.

To apply a *forgetting* rule $a^s \rightarrow \lambda$ from R_i, the neuron σ_i must contain *exactly* s spikes. In that case, all s spikes are removed from σ_i.

In each time unit, if a neuron σ_i can use one of its rules, then a rule from R_i must be used. If two or more rules can be applied in a neuron at the same time step, then only one of them is nondeterministically chosen to be applied. This means that in each neuron the rules are applied following a sequential semantics, while all neurons work in parallel.

A configuration of a Spiking Neural system is described by both the number of spikes present in each neuron and by the number of steps to wait until it becomes open (initially all neurons are open).

A *computation* starts in the initial configuration, with a positive integer number given in input to a specific *input neuron* as the number of time steps between the insertion of two spikes into that neuron, and proceeds by applying the rules in the neurons as described above. The output of the system is the number considered to be the number of time steps between the arrival of two spikes in a specific *output neuron*.

Spiking neural P systems have been widely investigated from a computational point of view ([46], [47], [48], [49], [50]), and many efforts are now concentrated on their application implementing artificial intelligence and deep learning strategies.

For instance, in [51] a Spiking Neural Network learning model is built, based on an evolutionary membrane algorithm, to solve the classical problem of supervised classification. The designed algorithm is able to automatically adjust the various learning parameters, by operating on the synaptic weight during the learning stage of the spiking neural model. The authors show that the proposed algorithm has competitive advantages, with respect to other experimental algorithms presented in the literature, in solving twelve supervised classification benchmark problems.

In [52] spiking neural P systems were used to solve the skeletonization problem: skeletonization consists in an image transformation, that simplifies the original image but preserving its topological properties. A parallel software simulating SN P systems was implemented on a Graphics Processors Units (GPU) architecture.

Fault diagnosis of power systems has been the subject of [53], where the problem is approached by means of an interval-valued fuzzy spiking neural P system: interval-valued fuzzy logic is used within a spiking neural P system, in order to deal with uncertainty. The paper shows that such a system can be used to accurately diagnose faulty sections in a power transmission networks.

VII. 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, as well as some references to the main variants of the basic model. For further reading on the subject, we refer the reader to the main volumes on the subject [3] [4].

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 [54], also containing some open problems worthing investigations.

Links to various paper concerning space complexity for membrane systems are available in [24] [25]; readers interested in results concerning sublinear space or even constant amount of space can refer to [55] and [56], respectively. A recent survey concerning results obtained by considering different bounds on space can be found in [57].

There are various research topics which deserve to be studied with respect to computing power and efficiency. In particular, precise characterizations of complexity classes obtained by considering systems using specific or minimal subset

of features, restriction on their functioning (like, e.g., communication in only one direction), and relations of such classes with standard complexity classes are currently active research topics [58] [59] [60] [61] are currently under investigations.

Other research directions concern the application of variants of P systems to real-life problems, as described in the previous section. Some actual research topics concern neovascularization lesion [62], Covid-19 pandemic [63], and the development of software tools to efficiently simulates P systems by exploiting the high level of parallelism offered by the use of modern GPUs [64].

Another interesting topic can be found in [65], where self-organization and self-repairing processes for membrane systems is presented and discussed (together with other types of bio-inspired systems), including self-organizing method based on optimization of cells placement and population P systems.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] C. Zandron, Computing Efficiency in Membrane Systems, Proc. of Future Computing Conference 2021 (H. Sato, The Univ. of Tokyo, ed.), The Thirteenth International Conference on Future Computational Technologies and Applications, April 18 - 22, 2021, Porto, Portugal, 8–13.

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

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

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

[5] The P systems Web page: http://ppage.psystems.eu/ (Retrieved 2022-05-30)

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

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

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

[9] 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

[10] M. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, 1967.

[11] 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)

[12] Z. Gazdag, G. Kolonits, A new method to simulate restricted variants of polarizationless P systems with active membranes, Journal of Membrane Computing, 1, 4, 2019, 251–261

[13] 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

[14] 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

[15] 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

[16] A.E. Porreca, G. Mauri, C. Zandron, Non-confluence in divisionless P systems with active membranes, Theoretical Computer Science, 411, 6, 2010, 878–887

[17] 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

[18] 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

[19] 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

[20] P. Sosik, The computational power of cell division in P systems: Beating down parallel computers?, Natural Computing, 2(3), 2003, 287–298

[21] A.E. Porreca, G. Mauri, C. Zandron, Complexity classes for membrane systems, RAIRO-Theor. Inform. and Applic. 40(2), 2006, 141–162

[22] 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

[23] 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

[24] 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

[25] 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

[26] 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

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

[28] 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

[29] J. Cecilia, J. Garcia, G. Guerrero, M. Martinez-del Amor, I. Perez-Hurtado, M.J. Perez-Jimenez, Simulating a P system based efficient solution to SAT by using GPUs, Journal of Logic and Algebraic Programming, 79, 6, 2010, 317–325

[30] M. Garcia-Quismondo, R. Gutierrez-Escudero, M. Martinez-del Amor, E. Orejuela-Pinedo, I. Perez-Hurtado, P-Lingua 2.0: A software framework for cell-like P systems, International Journal of Computers, Communications and Control, 4, 3, 2009, 234–243

[31] A. Alhazov, A. Leporati, L. Manzoni, G. Mauri, C. Zandron, Alternative space definitions for P systems with active membranes. Journal of Membrane Computing, 3, 2021, 87–96

[32] C. Buiu, A.G. Florea, Membrane computing models and robot controller design, current results and challenges, Journal of Membrane Computing, 1, 4, 2019, 262–269

[33] 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

[34] C. Martín-Vide, Gh. Păun, J. Pazos, A. Rodríguez-Patón, Tissue P systems, Theoretical Computer Science, 296, 2, 2003, 295–326

[35] R. Freund, A. Leporati, G. Mauri, A.E. Porreca, S. Verlan, C. Zandron, Flattening in (tissue) P systems, LNCS 8340, 2014, 173–188

[36] B. Song, X. Zeng, A. Rodriguez-Paton, Monodirectional tissue P systems with channel states, Information Sciences, 546, 2021, 206–219

[37] Y. Luo, Y. Zhao, C. Chen, Homeostasis tissue-like P systems, IEEE Transactions on NanoBioscience, 20(1), 2020, 126–136

[38] 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

[39] L. Valencia Cabrera, B. Song, Simulating Tissue P systems with promoters through MeCoSim and P-Lingua, Journal of membrane computing, 2 (2), 2019, 95–107

[40] R. Ceterchi, D. Orellana-Martin, G. Zhang, Division rules for tissue P systems inspired by space filling curves, Journal of Membrane Computing, 3(2), 2021, 105–115

[41] D. Diaz-Pernil, H.A. Christinal, M.A. Gutierrez-Naranjo, Solving the 3-COL problem by using tissue P systems without environment and proteins on cells, Information Sciences, 430, 2018, 240–246

[42] A. Leporati, C. Zandron, C. Ferretti, G. Mauri, Tissue P systems with small cell volume, Fundamenta Informaticae, 154, 1-4, 2017, 261–275

[43] Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, Tissue P systems with cell division, International Journal of Computers, Communications & Control, 3, 3, 2008, 295–303

[44] A. Leporati, L. Manzoni, G. Mauri, A.E. Porreca, C. Zandron, Characterising the complexity of tissue P systems with fission rules, Journal of Computer and System Sciences, 90, 2017, 115–128

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

[46] Y. Jiang, Y. Su, F. Luo, An improved universal spiking neural P system with generalized use of rules, Journal of Membrane Computing, 1, 2019, 270–278. https://doi.org/10.1007

[47] L. Pan, Gh. Păun, B. Song, Flat maximal parallelism in P systems with promoters, Theoretical Computer Science, 623, 2016, 83–91

[48] T. Wu, S. Jiang, Spiking neural P systems with a flat maximally parallel use of rules, Journal of Membrane Computing, 3, 2021, 221-231

[49] X. Zhang, B. Wang, L. Pan, Spiking Neural P Systems with a Generalized Use of Rules, Neural Computation, 26, 2014, 2925-2943

[50] N. Zhou, H. Peng, J. Wang, Q. Yang, X. Luo Computational completeness of spiking neural P systems with inhibitory rules for generating string languages, Theoretical Computer Science, to appear

[51] C. Liu, W. Shen, L. Zhang, Y. Du, Z. Yuan, Spike Neural Network Learning Algorithm Based on an Evolutionary Membrane Algorithm, IEEE Access, 9, 2021, 17071–17082

[52] D. Diaz-Pernil, H.A. Peña-Cantillana, M.A. Gutierrez-Naranjo, A parallel algorithm for skeletonizing images by using spiking neural P systems, Neurocomputing, 115, 4, 2013, 81–91

[53] J. Wang, H. Peng, W. Yu, J. Ming, M.J., Perez-Jimenez, C. Tao, X. Huang, Interval-valued fuzzy spiking neural P systems for fault diagnosis of power transmission networks, Engineering Applications of Artificial Intelligence, 82, 2019, 102–109

[54] P. Sosik, P systems attacking hard problems beyond NP: a survey. Journal of Membrane Computing 1(3), 2019, 198—208

[55] 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

[56] 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

[57] C. Zandron, Bounding the space in P systems with active membranes, Journal of Membrane Computing 2(2), 2020, 137–145

[58] K.C. Buño, F.G. Cabarle, M.D. Calabia, H.N. Adorna, Solving the N-Queens problem using dP systems with active membranes, Theoretical Computer Science, 736, 2018, 1–14

[59] A. Leporati, L. Manzoni, G. Mauri, A.E. Porreca, C. Zandron, Monodirectional P systems, Natural Computing, 15, 4, 2016, 551–564

[60] A. Leporati, L. Manzoni, G. Mauri, A.E. Porreca, C. Zandron, Characterizing PSPACE with shallow non-confluent P systems, Journal of Membrane Computing, 1, 2, 2019, 75–84

[61] Y. Luo, H. Tan, Y. Zhang, Y. Jiang, The computational power of timed P systems with active membranes using promoters, Mathematical Structures in Computer Science, 29, 5, 2019, 663-680

[62] J. Xue, A. Camino, S.T. Bailey, X. Liu, D. Li, Y. Jia, Automatic quantification of choroidal neovascularization lesion area on OCT angiography based on density cell-like P systems with active membranes, Biomedical optics express, 9,7, 2018, 3208-3219

[63] F. Baquero, M. Campos, C. Llorens, J.M. Sempere, P Systems in the Time of COVID-19, Journal of Membrane Computing, 3, 4, 2021, 246–257

[64] I. Perez-Hurtado, L. Valencia-Cabrera, A. Riscos-Nunez, M.J. Perez-Jimenez, Design of Specific P Systems Simulators on GPUs, In Membrane Computing: 19th International Conference, CMC 2018, Dresden, Germany, September 4–7, 2018, Revised Selected Papers, 11399, 2019, 202-207

[65] L. Duo, L. Xiubin, Z. Qingqi, Q. Yanling, L. Yue, Achievements, challenges, and developing directions of bio-inspired self-repairing technology, Microelectronics Journal, 111, 2021, 105044