Journal of Membrane Computing (2022) 4:251–260
https://doi.org/10.1007/s41965-022-00106-5

REGULAR PAPER

# Evaluating space measures in P systems

**Artiom Alhazov[1] · Alberto Leporati[2] · Luca Manzoni[3] · Giancarlo Mauri[2] · Claudio Zandron[2]**

Received: 8 July 2022 / Accepted: 11 September 2022 / Published online: 10 October 2022


**Abstract**
P systems with active membranes are a variant of P systems where membranes can be created by division of existing membranes, thus creating an exponential amount of resources in a polynomial number of steps. Time and space complexity classes for active membrane systems have been introduced, to characterize classes of problems that can be solved by different membrane systems making use of different resources. In particular, space complexity classes introduced initially considered a hypothetical real implementation by means of biochemical materials, assuming that every single object or membrane requires some constant physical space (corresponding to unary notation). A different approach considered implementation of P systems in silico, allowing to store the multiplicity of each object in each membrane using binary numbers. In both cases, the elements contributing to the definition of the space required by a system (namely, the total number of membranes, the total number of objects, the types of different membranes, and the types of different objects) was considered as a whole. In this paper, we consider a different definition for space complexity classes in the framework of P systems, where each of the previous elements is considered independently. We review the principal results related to the solution of different computationally hard problems presented in the literature, highlighting the requirement of every single resource in each solution. A discussion concerning possible alternative solutions requiring different resources is presented.

**Keywords** Membrane systems · Computational complexity · Space complexity

## 1 Introduction

P systems with active membranes have been introduced in [29], considering the idea of generating new membranes through division of existing ones. The exponential amount of resources that can be obtained in this way, in a polynomial number of computation steps, naturally leads to the definition of new complexity classes to be compared with the standard ones.

Initially, the research activity focused on the investigation of time complexity. It was proved that, to go beyond the complexity class **P**, the creation of new membranes is a necessary feature to gain enough computation efficiency [44], unless non-confluent systems are used [37]. In [38], it was proved that P systems with active membranes can solve all problems in the class **PSPACE** in polynomial time, a result which is valid also for uniform systems, as proved in [7]. Relations with the classes **EXP** and **EXPSPACE** were investigated in [36].

A series of works then defined various complexity classes characterized by P systems that make use of different features. For instance, the works [13, 14] focused on the crucial

✉ Claudio Zandron
claudio.zandron@unimib.it

Artiom Alhazov
artiom@math.md

Alberto Leporati
alberto.leporati@unimib.it

Luca Manzoni
lmanzoni@units.it

Giancarlo Mauri
giancarlo.mauri@unimib.it

[1] Vladimir Andrunachievici Institute of Mathematics and Computer Science, Academiei 5, Chişinău MD-2028, Moldova

[2] Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo), Università degli Studi di Milano-Bicocca, Viale Sarca 336, 20126 Milan, Italy

[3] Dipartimento di Matematica e Geoscienze, Università degli Studi di Trieste, Via Alfonso Valerio 12/a, 24127 Trieste, Italy

role of membrane dissolution; polarizationless systems have been investigated in [5, 6, 12, 17]; constraints on membrane division [24] or on depth of membrane structure [19] have been the subjects of other papers, while [40, 41] focused on the role of cooperation.

More recently, other aspects have also been studied. In [1, 27], a different kind of membrane division, called separation (since objects are separated between new membranes, rather than duplicated) is considered in the framework of P systems with active membranes; in [26] such kind of rules are applied in a different variant of P systems, having proteins on membranes. In [8, 11], solutions for the SAT problem are proposed which use different strategies than previously proposed solutions. Systems of a shallow depth are the subject of [20–22]. A recent survey on different strategies to approach computationally hard problems by P systems with active membranes can be found in [39].

Besides time complexity, space complexity has been also considered. This notion was firstly introduced in the framework of P systems in [31], with a definition based on a hypothetical real implementation by means of biochemical materials such as cellular membranes and chemical molecules. Under this assumption, it was assumed that every single object or membrane requires some constant physical space, and this is equivalent to using a unary encoding to represent multiplicities. The relations between standard computational complexity classes and the space complexity classes defined in these terms have been studied, both when at least a linear amount of space is used [32, 33], as well as when only sublinear [35] or even constant amount of space [18] is available. A recent survey concerning results obtained by considering different bounds on space can be found in [43].

A different approach to define space complexity for P systems was considered in [2], focusing the definition of space on the simulative point of view. In fact, by considering an implementation of P systems in silico (like the ones in, e.g., [9, 10]), it is not strictly necessary to store information concerning every single object: the multiplicity of each object in each membrane can be stored using binary numbers, thus reducing the amount of needed space.

In both cases, the definition of the space required by a system was considered as a unique total measure obtained by considering all the elements contributing to it: the total number of membranes, the total number of objects, the types of different membranes, and the types of different objects.

In this paper, we introduce a different definition for space complexity classes in the framework of P systems, where each of the previous elements is considered independently, thus proposing a vector measurement of previously defined scalar measures. This allows to consider the amount of each element separately, thus highlighting the requirement of each of them in every solution considered, instead of a global value. We review the principal results present in the literature and we discuss possible alternative solutions, requiring different resources balances.

The paper is organized as follows. In Sect. 2 we recall some definitions concerning P systems with active membranes and space requirements in P systems computations. In Sect. 3, we introduce a definition of space to measure the contribution by each component, namely the total number of membranes, the total number of objects, the types of different membranes, and the types of different objects. Moreover, we survey some main results concerning complexity in the framework of P systems, highlighting the use of each single resource. Section 4 presents some conclusions and future research topics.

## 2 Basic definitions

In this section, we shortly recall some definitions that will be useful while reading the rest of the paper. For a complete introduction to P systems, we refer the reader to *The Oxford Handbook of Membrane Computing* [30].

**Definition 1** A *P system with active membranes* having initial degree $d \geq 1$ is a tuple $\Pi = (\Gamma, \Lambda, \mu, w_{h_1}, \ldots, w_{h_d}, R)$, where:

- $\Gamma$ is an alphabet, i.e., a finite non-empty set of symbols, usually called *objects*; in the following, we assume $\Gamma = \{O_1, O_2, \ldots, O_n\}$;
- $\Lambda$ is a finite set of labels for the membranes;
- $\mu$ is a membrane structure (i.e., a rooted *unordered* tree, usually represented by nested brackets) consisting of $d$ membranes, labeled by elements of $\Lambda$, defining regions (the space between a membrane and all membranes immediately inside it, if any);
- $w_{h_1}, \ldots, w_{h_d}$, with $h_1, \ldots, h_d \in \Lambda$, are strings over $\Gamma$ describing the initial multisets of objects placed in the $d$ regions of $\mu$;
- $R$ is a finite set of rules over $\Gamma$.

Membranes are polarized, that is, they have an attribute called *electrical charge*, which can be neutral (0), positive (+) or negative (−).

A P system can make a computation step by applying its rules to modify the membrane structure and/or the membrane content. The following types of rules can be used during the computation:

- *Object evolution rules*, of the form $[a \to w]_h^\alpha$ They can be applied inside a membrane labeled by $h$, having charge $\alpha$ and containing at least an occurrence of the object $a$; the copy of the object $a$ to which the rule is applied is

rewritten into the multiset $w$ (i.e., $a$ is removed from the multiset in $h$ and replaced by the objects in $w$).

- *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 $\alpha$ and such that the external region contains at least an occurrence of the object $a$; the copy of the object $a$ to which the rule is applied is sent into $h$ becoming $b$ and, simultaneously, the charge of $h$ is changed to $\beta$.
- *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 $\alpha$ and containing at least an occurrence of the object $a$; the copy of the object $a$ to which the rule is applied is sent out from $h$ to the outside region becoming $b$ and, simultaneously, the charge of $h$ is changed to $\beta$.
- *Dissolution rules*, of the form $[a]_h^\alpha \to b$ They can be applied to a membrane labeled by $h$, having charge $\alpha$ and containing at least an occurrence of the object $a$; the copy of the object $a$ to which the rule is applied is replaced by $b$, the membrane $h$ is dissolved and its contents are left in the surrounding region.
- *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 $\alpha$, containing at least an occurrence of the object $a$ but having no other membrane inside (in this case, the membrane is said to be *elementary*); the membrane is divided into two membranes having both label $h$ and charges $\beta$ and $\gamma$, respectively; the copy of the object $a$ to which the rule is applied is replaced, respectively, by $b$ and $c$ in the two new membranes, while the other objects in the initial multiset are copied to both membranes.
- *(Weak) Non-elementary division rules*, of the form $[a]_h^\alpha \to [b]_h^\beta\,[c]_h^\gamma$ These rules operate just like division for elementary membranes, but they can be applied to non-elementary membranes, containing membrane substructures and having a label $h$. Like the objects, the substructures inside the dividing membrane are replicated in the two new copies of it.

A configuration of a P system with active membranes is described by the current membrane structure (including the electrical charge of each membrane) and 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: this means that inside each membrane several evolution rules can be applied simultaneously, but each membrane can be involved only in a single communication, dissolution, or division rule per step.
- The application of rules is *maximally parallel*: each object appearing on the left-hand side of evolution, communication, dissolution or division rules must be subject to exactly one of them (unless the current charge of the membrane prohibits it, and according to the fact that a membrane can be involved in a single communication, dissolution, or division rule per step). The same principle applies to each membrane that can be involved in communication, dissolution, or division rules. In other words, 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 as a result of a computation step.
- In each computation step, all the chosen rules are applied simultaneously (in an atomic way). We stress the fact that the membranes evolve only after their internal configuration has been updated. For instance, before a membrane division occurs, all chosen object evolution rules must be applied inside it; in this way, the objects that are duplicated during the division are already the final ones.
- 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 $\Pi$ is a finite sequence of configurations $\mathcal{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$, 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 of $\Pi$ are applicable in $\mathcal{C}_k$. If this last condition is never reached (that is, in each configuration of the sequence there is at least one applicable rule), then a *non-halting* computation $\mathcal{C} = (\mathcal{C}_i : i \in \mathbb{N})$ is obtained, that consists of infinitely many configurations, again starting from the initial one and generated by successive computation steps.

P systems can be used as language *recognizers* by employing two distinguished objects *yes* and *no*; exactly one of these must be sent out from the outermost membrane, and only in the last step of each computation, to signal acceptance or rejection, respectively; we also assume that all computations are halting.

To solve decision problems (i.e., recognize languages over an alphabet $\Sigma$), we use *families* of recognizer P systems $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$. Each input $x$ is associated with a P system $\Pi_x$ in the family $\mathbf{\Pi}$ 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 [25].

These families of *recognizer* P systems can be used to solve decision problems as follows.

**Definition 2** Let $\Pi$ be a P system whose alphabet contains two distinct objects *yes* and *no*, such that every computation of $\Pi$ is halting and during each computation exactly one of the objects *yes*, *no* is sent out from the skin (in the last computation step) to signal acceptance or rejection. If all computations of $\Pi$ agree on the result, then $\Pi$ is said to be *confluent*; if this is not necessarily the case, then it is said to be *non-confluent* and the global result is acceptance if and only if there exists at least an accepting computation.

**Definition 3** Let $L \subseteq \Sigma^\star$ be a language, $\mathcal{D}$ a class of P systems (i.e., a set of P systems using a specific subset of features; in the following, we will consider some main classes: *AM*, active membranes—division for both elementary and non-elementary membranes, *NAM*, non active membranes—membranes cannot be divided, and *EAM*, elementary active membranes—only elementary membranes can be divided) and let $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ be a family of P systems, either confluent or non-confluent. We say that $\mathbf{\Pi}$ *decides* $L$ when, for each $x \in \Sigma^\star$, $x \in L$ if and only if $\Pi_x$ accepts.

Complexity classes for P systems are defined by imposing a uniformity condition on $\mathbf{\Pi}$ and restricting the amount of time or space available for deciding a language.

**Definition 4** Consider a language $L \subseteq \Sigma^\star$, a class of recognizer P systems $\mathcal{D}$, and let $f : \mathbb{N} \to \mathbb{N}$ be a proper complexity function (i.e., a "reasonable" one, see [28, Definition 7.1]). We say that $L$ belongs to the complexity class $\mathrm{MC}^\star_{\mathcal{D}}(f)$ if and only if there exists a family of confluent P systems $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding $L$ such that:

- $\mathbf{\Pi}$ is *semi-uniform*, i.e., there exists a deterministic Turing machine which, for each input $x \in \Sigma^\star$, constructs the P system $\Pi_x$ in polynomial time with respect to $|x|$;
- $\mathbf{\Pi}$ operates in time $f$, i.e., for each $x \in \Sigma^\star$, every computation of $\Pi_x$ halts within $f(|x|)$ steps.

In particular, a language $L \subseteq \Sigma^\star$ belongs to the complexity class $\mathrm{PMC}^\star_{\mathcal{D}}$ if and only if there exists a semi-uniform family of confluent P systems $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding $L$ in polynomial time.

The analogous complexity classes for non-confluent P systems are denoted by $\mathrm{NMC}^\star_{\mathcal{D}}(f)$ and $\mathrm{NPMC}^\star_{\mathcal{D}}$.

Another set of complexity classes is defined in terms of *uniform* families of recognizer P systems:

**Definition 5** Consider a language $L \subseteq \Sigma^\star$, a class of recognizer P systems $\mathcal{D}$, and let $f : \mathbb{N} \to \mathbb{N}$ be a proper complexity function. We say that $L$ belongs to the complexity class $\mathrm{MC}_{\mathcal{D}}(f)$ if and only if there exists a family of confluent P systems $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding $L$ such that:

- $\mathbf{\Pi}$ is *uniform*, i.e., for each $x \in \Sigma^\star$ deciding whether $x \in L$ is performed as follows: first, a polynomial-time deterministic Turing machine, given the length $n = |x|$ as a unary integer, constructs a P system $\Pi_n$ with a distinguished input membrane; then, another polynomial-time deterministic Turing machine computes an encoding of the string $x$ as a multiset $w_x$, which is finally added to the input membrane of $\Pi_n$, thus obtaining a P system $\Pi_x$ that accepts if and only if $x \in L$.
- $\mathbf{\Pi}$ operates in time $f$, i.e., for each $x \in \Sigma^\star$, every computation of $\Pi_x$ halts within $f(|x|)$ steps.

In particular, a language $L \subseteq \Sigma^\star$ belongs to the complexity class $\mathrm{PMC}_{\mathcal{D}}$ if and only if there exists a uniform family of confluent P systems $\mathbf{\Pi} = \{\Pi_x \mid x \in \Sigma^\star\} \subseteq \mathcal{D}$ deciding $L$ in polynomial time.

The analogous complexity classes for *non*-confluent P systems are denoted by $\mathrm{NMC}_{\mathcal{D}}(f)$ and $\mathrm{NPMC}_{\mathcal{D}}$.

As stated in the Introduction, the first definition of space complexity for P systems introduced in [31] considered a possible real implementation with biochemical materials, thus assuming that every single object and membrane requires some constant physical space. Such a definition (in the improved version from [23], taking into account also the space required by the labels for membranes and the alphabet of symbols) is the following:

**Definition 6** Considering a configuration $\mathcal{C}$ of a P system $\Pi$, its *size* $|\mathcal{C}|$ is the number of membranes in the current membrane structure multiplied by $\log|\Lambda|$, plus the total number of objects from $\Gamma$ they contain multiplied by $\log|\Gamma|$. If $\mathcal{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ is a computation of $\Pi$, then the *space required by* $\mathcal{C}$ is defined as

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

The *space required by* $\Pi$ itself is defined as the supremum of the space required by all computations of $\Pi$:

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

Finally, let $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ be a family of recognizer P systems, and let $s : \mathbb{N} \to \mathbb{N}$. We say that $\mathbf{\Pi}$ *operates within space bound* $s$ if and only if $|\Pi_x| \leq s(|x|)$ for each $x \in \Sigma^\star$.

Following what has been done for time complexity classes, we can define space complexity classes. By $\mathrm{MCSPACE}_{\mathcal{D}}(s(n))$ (resp. $\mathrm{MCSPACE}^\star_{\mathcal{D}}(s(n))$) we denote the class of languages which can be decided by uniform (resp.

semi-uniform) families, $\Pi$, of confluent P systems of type $\mathcal{D}$ (for example, when we refer to P systems with active membranes, we denote this by setting $\mathcal{D} = \mathcal{AM}$), where each $\Pi_x \in \mathbf{\Pi}$ operates within space bound $s(|x|)$.

In particular, the class of problems solvable in a polynomial space by uniform confluent systems is denoted by $\text{PMCSPACE}_{\mathcal{D}}$, and the class of problems solvable in an exponential space by uniform confluent systems is denoted by $\text{EXPMCSPACE}_{\mathcal{D}}$ (adding a star in case of semi–uniform classes). The corresponding classes for non-confluent systems are $\text{NPMCSPACE}_{\mathcal{D}}$ and $\text{NEXPMCSPACE}_{\mathcal{D}}$.

A different approach to define space complexity for P systems was introduced in [2], considering the information stored in the objects of the systems, and not the single objects themselves. Binary notation, instead of unary, was used to store the amount of objects in each region, with the following definition of *binary space*:

**Definition 7** Consider a configuration $\mathcal{C}$ of a P system $\Pi$. Let us denote by $h_1, h_2, ..., h_z$ the membranes of the current membrane structure (we stress the fact that $z$ can be smaller, equal, or greater than the initial number of membranes $d$, due to dissolution and duplication of membranes; we also stress the fact that we do not need to store unique IDs for membranes having the same label as we can, for example, indicate multisets of objects inside a string-like bracketed expression), and by $|O_{i,j}|$ the multiplicity of object $i$ within region $j$. The binary size $|\mathcal{C}|_B$ of a configuration $\mathcal{C}$ is defined as:

$$|\mathcal{C}|_B = z \cdot \log|\Lambda| + \sum_{j=1}^{z}\sum_{i=1}^{n}\Big(\lceil \log(|O_{i,j}|+1)\rceil + \log|\Gamma|\Big),$$

that is the number of membranes in the current membrane structure multiplied by $\log|\Lambda|$, plus the number of bits required to store the description of the multiset in each region.

If $\mathcal{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ is a computation of $\Pi$, then the *binary space* required by $\mathcal{C}$ is defined as

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

The *binary space* required by $\Pi$ itself is then obtained by computing the binary space required by all computations of $\Pi$ and taking the supremum:

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

Finally, let $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ be a family of recognizer P systems, and let $s : \mathbb{N} \to \mathbb{N}$. We say that $\mathbf{\Pi}$*operates within binary space bound* $s$ if and only if $|\Pi_x|_B \leq s(|x|)$ for each $x \in \Sigma^\star$.

Corresponding space complexity classes, that consider this different size measure, can be considered.

## 3 Considering separate feature contributions to space definition

In both cases of space definition considered in the previous Section, the amount of space required by a system was a single total measure obtained, in different ways, by considering all the elements contributing to it: the total number of membranes (denoted by *me*), the total number of objects (*ob*), the membrane types (*met*), and the object types (*obt*).

We introduce now a different definition of space in the framework of P systems, based on a vector measurement, instead of a scalar one, where each of the previous elements is considered independently, thus allowing to consider the contribution given by each element separately, and highlighting the requirement of each of them in every solution to complex computational problems considered in the literature.

The use of vector measurements has been successfully considered in many different domains. Just to cite some examples, it has been considered to define size for insertion–deletion systems [16], with the size measured by six numbers (insertion, left insertion context, right insertion context, size of deletion, left deletion context, right deletion context), or in splicing systems [15], using four numbers to consider the sizes of the recombination points in the strings involved in a splicing operation.

**Definition 8** Consider a P system $\Pi$ with active membranes, and a computation $\mathcal{C} = (\mathcal{C}_0, \ldots, \mathcal{C}_k)$ of $\Pi$. Let us denote the set of membrane labels of $\Pi$ by $\Lambda$, and the alphabet of objects by $\Gamma$. Moreover, let us denote by *MaxMe* the maximum number of membranes present at the same time in $\Pi$ during some steps of $\mathcal{C}$, and by *MaxOb* the maximum number of objects present at the same time in $\Pi$ during some steps of $\mathcal{C}$ (we stress the fact that the maximum number of membranes and objects do not necessarily appear in the same computation step).

We say that the computation $\mathcal{C}$ of $\Pi$ is bounded by $Space(me, ob, met, obt)$ if and only if $MaxMe \leq me$, $MaxOb \leq ob$, $|\Lambda| \leq met$, and $|\Gamma| \leq obt$.

The *space* required by $\Pi$ itself is then obtained by computing the corresponding space required by all computations of $\Pi$ and taking the supremum.

Finally, let $\mathbf{\Pi} = \{\Pi_x : x \in \Sigma^\star\}$ be a family of recognizer P systems. We say that $\mathbf{\Pi}$ *operates within Space(me, ob, met, obt)* if and only if each member $\Pi_x$ of the family operates within the above space.

Of course, complexity classes corresponding to this definition of space can be defined in a similar way as already done in the previous Section; as an example, by $\mathrm{MCSPACE}_{\mathcal{D}}(me, ob, met, obt)$ we denote the class of problems solved by P systems with active membranes of type $\mathcal{D}$ having features limited according to *me*, *ob*, *met*, *obt*.

Having defined the contribution of each element to the space requirements, we are ready to survey some main results present in the literature, to analyze the requirements in terms of single features required by each proposed solution.

The first results we analyze are from [31].

**Proposition 1** $\mathrm{P} \subseteq \mathrm{MCSPACE}^{*}_{NAM}(O(1))$, *and* $\mathrm{P} \subseteq \mathrm{MCSPACE}_{NAM}(O(1))$.

In this proposition, it is proved that semi-uniform systems with active membranes that do not use membrane division can solve all problems in P in constant space. In fact, all the work is done by the deterministic Turing machine used to build the system (a deterministic polynomial time uniformity condition is considered). The obtained system simply sends out a *yes* or *no* answer, in one step. Similar considerations remain valid also for the uniform case. As a consequence, in both cases we have $me = ob = mt = obt = 1$. Considering the corresponding proposition, we have:

**Proposition 1′** $\mathrm{P} \subseteq \mathrm{MCSPACE}^{*}_{NAM}(1, 1, 1, 1)$

The next proposition exploits the results from [44], in particular concerning the problems SAT with $n$ variables and $m$ clauses, and *Hamiltonian Path* over an undirected graph with $n$ nodes.

In the first case, $2^n$ membranes are generated, each containing one possible truth assignment to be checked. We have: $me = \mathcal{O}(2^n)$, $ob = \mathcal{O}((n * m) * (2^n))$, $met = 2$, and $obt = 4n + 2m + 4$.

The solution for the Hamiltonian Path problem tries all possible paths, to check if at least one of them satisfies the required conditions. In this case, we have $me = \mathcal{O}(n^n)$, $ob = \mathcal{O}(n * me) = \mathcal{O}(n^{n+1})$, $met = 2$, $obt = 6n + 4$.

In [38], it was shown how to exploit membrane division to solve the **PSPACE**-complete problem Satisfiability of Quantified Boolean Formulas (QBF), using semi-uniform P systems with active membranes:

**Proposition 2** $\mathbf{PSPACE} \subseteq \mathrm{EXPMCSPACE}^{*}_{AM}$

The solution makes use of an exponential number of membranes and objects: $me = \mathcal{O}(2^n)$, $ob = \mathcal{O}(2^n)$, $met = m + n + 2$, and $obt = 5n + m + 4$. By considering each single result, and the new space measure, we have:

**Proposition 2′** $\mathbf{NP} \subseteq \mathrm{MCSPACE}^{*}_{AM}(O(2^n), O(2^n), 2, 4n + 2m + 4)$
$\mathbf{NP} \subseteq \mathrm{MCSPACE}^{*}_{AM}(O(2^n), O(2^n), 2, 4n + 2m + 4)$

$$\mathbf{NP} \subseteq \mathrm{MCSPACE}^{*}_{AM}(O(n^n), O(n^n), 2, 6n + 4)$$

$$\mathbf{PSPACE} \subseteq \mathrm{MCSPACE}^{*}_{AM}(O(2^n), O(2^n), m + n + 2, 5n + m + 4))$$

In [4], the same last result was proved for uniform systems.

**Proposition 3** $\mathbf{PSPACE} \subseteq \mathrm{EXPMCSPACE}_{AM}$

An analysis of resources used for this solution shows that $me = \mathcal{O}(2^{2n}) = \mathcal{O}(4^n)$, $ob = \mathcal{O}(2^{2n}) = \mathcal{O}(4^n)$, $met = \mathcal{O}(m + n)$, and $obt = \mathcal{O}(m * n)$.

**Proposition 3′** $\mathbf{PSPACE} \subseteq \mathrm{MCSPACE}_{AM}(O(4^n), O(4^n), O(m + n), O(m * n))$

In [34], systems with limited power were considered; in particular, division rules for non-elementary membranes and dissolution rules were avoided. It was proved that such systems can solve all problems in the complexity class **PP** in polynomial time. In fact, a solution for the **PP**-complete problem SQRT3SAT was proposed.

**Proposition 4** $\mathbf{PP} \subseteq \mathbf{PMC}^{ndiss}_{EAM}$

The features of that solution are the following: $me = \mathcal{O}(2^n)$, $ob = \mathcal{O}(2^n)$, $met = 3$, and $obt = \mathcal{O}(n)$. Thus, we have

**Proposition 4′** $\mathbf{PP} \subseteq \mathbf{MCSPACE}^{ndiss}_{EAM}(O(2^n), O(2^n), 3, O(n))$

In [42], it was shown that a deterministic Turing machine working in polynomial space, with respect to the input length, can be efficiently simulated (both in terms of time and space) by a semi-uniform family of P systems with active membranes, using only communication rules.

**Proposition 5** *A Deterministic Turing machine M working in space s(n) and time t(n) can be simulated by a semi-uniform family of P systems in space* $\mathcal{O}(s(n))$ *and time* $\mathcal{O}(t(n))$

The main idea to prove this result was to store information bits by using polarizations associated with membranes, instead of objects inside them. As a consequence, the required amount of resources needed is very low: $me = s(n) + 2$, $ob = \mathcal{O}(n)$ initially, then 1, $met = 3$, $obt = 5 + 3 * |Q|$, where $Q$ is the set of states of the Turing machine $M$.

**Proposition 5′** *A Deterministic Turing machine M working in space s(n) and in time t(n) can be simulated by a semi-uniform family of P systems in space* $\mathrm{MCSPACE}^{*}_{AM}(s(n)+2, O(n), 3, 5+3*|Q|)$

In [3], similar results for uniform families of P systems with active membranes were proved, using a cubic slowdown and a quadratic space overheads:

**Proposition 6** *A deterministic Turing machine M working in space s(n) and time t(n) can be simulated by uniform confluent or non-confluent P systems within polynomial bounds for space and time.*

In both cases, we have the following: $me = \mathcal{O}(s(n))$, $ob = \mathcal{O}(s(n)^2)$, $met = 7$, and $obt = 3$.

**Proposition 6′** *A deterministic Turing machine M working in space s(n) and time t(n) can be simulated by uniform confluent or non-confluent P systems in space* $\mathbf{MCSPACE}^{*}_{AM}(O(s(n)), O(s(n)^2), 7, 3)$

We conclude this analysis by recalling a result from [33]. In this work, it was proved that recognizer P systems with active membranes that use polynomial space characterize the complexity class **PSPACE**. In particular, the result holds for both confluent and non-confluent systems, and independently of the use of membrane division rules.

This generic result allows to relate the number of computation steps to the maximum number of objects and the maximum number of membranes that can be obtained after those steps. In particular, let us consider a non-confluent P system $\Pi$ with active membranes, defined using a description of length $m$ made in any reasonable encoding (e.g., where the membrane structure is represented using strings of brackets), and where multisets are represented in unary. After $t$ steps of computation of $\Pi$, we have the following: $me = \mathcal{O}(2^{t*m+m*\log(m)})$, and $ob = \mathcal{O}(2^{t*t*m*\log(m)})$.

We conclude this section by summarizing in the following table the results we presented in the paper. For instance, the first result can be read as "deterministic $\mathrm{MCSPACE}^{*}_{(c)}(1,1,1,1) \supseteq P$, see [31]".

| D/C, U/SU | me | ob | met | obt | D | ⊇ class/problem | props | ref. |
|---|---|---|---|---|---|---|---|---|
| D,SU | 1 | 1 | 1 | 1 | (c) | $P$ | ConstTime | [31] |
| D,U | 1 | 1 | 1 | 2 | (c) | $P$ | ConstTime | [31] |
| SU | $2^n$ | $O(2^n)$ | 2 | $4n+2m+4$ | EAM | $NP \cup coNP$/SAT | | [44] |
| SU | $O(n^n)$ | $O(n^n)$ | 2 | $6n+4$ | EAM | $NP \cup coNP$/HPP | | [44] |
| SU | $O(2^n)$ | $O(2^n)$ | $m+n+2$ | $5n+m+4$ | AM | *PSPACE*/QSAT | | [38] |
| U | $O(4^n)$ | $O(4^n)$ | $O(m+n)$ | $O(m \times n)$ | AM | *PSPACE*/QSAT | | [4] |
| | $O(2^n)$ | $O(2^n)$ | 3 | $O(n)$ | (abce) | *PP*/SQRT-3SAT | PolTime | [34] |
| SU | $s(n)+2$ | $O(n) \cdots 1$ | 3 | $5+3*\|Q\|$ | AM | $DT(t(n))S(s(n))$ | | [42] |
| C,U | $O(s(n))$ | $O(s(n)^2)$ | 7 | 3 | AM | $DT(t(n))S(s(n))$ | | [3] |
| | $O(2^x)$, $x = t*m+m*\log(m)$ | $O(2^y)$, $y = t^2*m*\log(m)$ | | | AM | *PSPACE*/TM | time=t | [33] |

## 4 Conclusions

We introduced a definition for space complexity classes in the framework of P systems, where each element contributing to the definition of space used by the system (that is, the total number of membranes, the total number of objects, the types of different membranes, and the types of different objects) is considered independently. In this way, the contribution of each element in defining the total space required by the system to execute a computation can be highlighted independently.

Many different constructions by various authors have been presented in the literature to attack computationally hard problems. After presenting here some of them, we think it will be interesting to check the possibility to propose alternative solutions where the considered parameters are different (i.e., of a different asymptotic order) with respect to those already published. As an example, would it be possible to find solutions similar to those of Propositions 3 and 4 by using a constant amount of object types or of membrane types?

Moreover, we want to stress the fact that, under the usual uniformity condition considered, a polynomial time precomputing by a deterministic Turing machine is allowed and, in this case, membrane types and object types are forced to be at most polynomial. However, different uniformity conditions can also be considered, to highlight their impact on these features. For instance, how to factor the input for sublinear space complexity classes if we would like to have more object types with respect

to those allowed by the input size (maybe not all of them present since the beginning of the computation), but the uniformity condition forbids that?

It would also be useful to clarify how to proceed when the size of the problem instance is given by a few numbers like, e.g., the number of clauses and variables for SAT.

**Acknowledgements** This work was partially supported by Università degli Studi di Milano-Bicocca, Fondo di Ateneo per la Ricerca 2019, project 2019-ATE-0454. The first author acknowledges project 20.80009.5007.22 "Intelligent information systems for solving ill-structured problems, processing knowledge and big data" by the National Agency for Research and Development.

**Funding** Open access funding provided by Università degli Studi di Milano - Bicocca within the CRUI-CARE Agreement.

## Declarations

**Conflict of interest** On behalf of all authors, the corresponding author states that there is no conflict of interest.



## References

1. Alhazov, A., Ishdorj, T. (2004). Membrane operations in P systems with active membranes. In: Păun, Gh., Riscos-Núñez, A., Romero-Jiménez, A., Sancho-Caparrini, F. (eds.) Second Brainstorming Week on Membrane Computing. pp. 37–44. No. 1/2004 in RGNC Reports, Fénix Editora
2. Alhazov, A., Leporati, A., Manzoni, L., Mauri, G., & Zandron, C. (2021). Alternative space definitions for P systems with active membranes. *Journal of Membrane Computing, 3*(2), 87–96. https://doi.org/10.1007/s41965-021-00074-2
3. Alhazov, A., Leporati, A., Mauri, G., Porreca, A. E., & Zandron, C. (2014). Space complexity equivalence of P systems with active membranes and Turing machines. *Theoretical Computer Science, 529*, 69–81. https://doi.org/10.1016/j.tcs.2013.11.015
4. Alhazov, A., Martín-Vide, C., Pan, L. (2003). Solving a PSPACE-complete problem by recognizing P systems with restricted active membranes. *Fundamenta Informaticae, 58*(2), 67–77.
5. Alhazov, A., & Pan, L. (2004). Polarizationless P systems with active membranes. *Grammars, 7*, 141–159.
6. Alhazov, A., Pan, L., & Păun, Gh. (2004). Trading polarizations for labels in P systems with active membranes. *Acta Informatica, 41*(2–3), 111–144. https://doi.org/10.1007/s00236-004-0153-z
7. Alhazov, A., Pérez-Jiménez, M.J. (2007). Uniform solution to QSAT using polarizationless active membranes. In: Durand-Lose, J., Margenstern, M. (eds.) Machines, Computations, and Universality, 5th International Conference, MCU 2007, Lecture Notes in Computer Science, vol. 4664, pp. 122–133. Springer, https://doi.org/10.1007/978-3-540-74593-8_11
8. Buño, K., & Adorna, H. (2020). Distributed computation of a kP system with active membranes for SAT using clause completion. *Journal of Membrane Computing, 2*(2), 108–120. https://doi.org/10.1007/s41965-020-00040-4
9. Cecilia, J., García, J., Guerrero, G., Martínez-del Amor, M., Pérez-Hurtado, I., & Pérez-Jiménez, M. (2010). Simulating a P system based efficient solution to SAT by using GPUs. *Journal of Logic and Algebraic Programming, 79*(6), 317–325.
10. García-Quismondo, M., Gutiérrez-Escudero, R., Martínez-del Amor, M.A., Orejuela-Pinedo, E., Pérez-Hurtado, I. (2009). P-Lingua 2.0: A software framework for cell-like P systems. International Journal of Computers, Communications & Control **4**(3), 234–243
11. Gazdag, Z., Kolonits, G. (2013). A new approach for solving SAT by P systems with active membranes. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, G. (eds.) Membrane Computing, 13th International Conference, CMC 2012. Lecture Notes in Computer Science, vol. 7762, pp. 195–207. Springer
12. Gazdag, Z., & Kolonits, G. (2019). A new method to simulate restricted variants of polarizationless P systems with active membranes. *Journal of Membrane Computing, 1*(4), 251–261.
13. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Campero, F.J. (2006). On the power of dissolution in P systems with active membranes. In: Freund, R., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing, 6th International Workshop, WMC 2005. Lecture Notes in Computer Science, vol. 3850, pp. 224–240. Springer, https://doi.org/10.1007/11603047
14. Gutiérrez-Naranjo, M. A., Pérez-Jiménez, M. J., Riscos-Nuñez, A., & Romero-Campero, F. J. (2006). Computational efficiency of dissolution rules in membrane systems. *International Journal of Computer Mathematics, 83*(7), 593–611. https://doi.org/10.1080/00207160601065413
15. Head, T. (1987). Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology, 49*(6), 737–759.
16. Kari, L., & Thierrin, G. (1996). Contextual insertion/deletion and computability. *Information and Computation, 131*(1), 47–61.
17. Leporati, A., Ferretti, C., Mauri, G., & Zandron, C. (2008). Complexity aspects of polarizationless membrane systems. *Natural Computing, 4*(8), 703–717.
18. Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2014). Constant-space P systems with active membranes. *Fundamenta Informaticae, 134*(1–2), 111–128. https://doi.org/10.3233/FI-2014-1094
19. Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2015). Membrane division, oracles, and the counting hierarchy. *Fundamenta Informaticae, 138*(1–2), 97–111. https://doi.org/10.3233/FI-2015-1201
20. Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2019). Characterizing PSPACE with shallow non-confluent P systems. *Journal of Membrane Computing, 1*(2), 75–84. https://doi.org/10.1007/s41965-019-00011-4
21. Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2020). Shallow laconic P systems can count. *Journal of Membrane Computing, 2*(1), 49–58. https://doi.org/10.1007/s41965-020-00032-4
22. Leporati, A., Manzoni, L., Mauri, G., Porreca, A. E., & Zandron, C. (2020). A Turing machine simulation by P systems without charges. *Journal of Membrane Computing, 2*(2), 71–79. https://doi.org/10.1007/s41965-020-00031-5

23. Leporati, A., Mauri, G., Porreca, A.E., Zandron, C. (2014). A gap in the space hierarchy of P systems with active membranes. Journal of Automata, Languages and Combinatorics **19**(1–4), 173–184, http://theo.cs.ovgu.de/jalc/search/j19_i.html. Accessed 29 Sept 2022
24. Murphy, N., Woods, D. (2007). Active membrane systems without charges and using only symmetric elementary division characterise P. In: Eleftherakis, G., Kefalas, P., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing, 8th International Workshop, WMC 2007. Lecture Notes in Computer Science, vol. 4860, pp. 367–384, https://doi.org/10.1007/978-3-540-77312-2_23
25. Murphy, N., & Woods, D. (2011). The computational power of membrane systems under tight uniformity conditions. *Natural Computing, 10*(1), 613–632. https://doi.org/10.1007/s11047-010-9244-7
26. Orellana-Martín, D., Valencia-Cabrera, L., Riscos-Núñez, A., Pérez-Jiménez, M.J. (2019). P systems with proteins: a new frontier when membrane division disappears. Journal of Membrane Computing **1**(1), 29–39, https://doi.org/10.1007/s41965-018-00003-w
27. Pan, L., Alhazov, A., & Ishdorj, T. O. (2005). Further remarks on P systems with active membranes, separation, merging, and release rules. *Soft Computing, 9*(9), 686–690. https://doi.org/10.1007/s00500-004-0399-y
28. Papadimitriou, C.H. (1993). Computational Complexity. Addison-Wesley
29. Păun, Gh. (2001). P systems with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages and Combinatorics, 6*(1), 75–90.
30. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
31. Porreca, A.E., Leporati, A., Mauri, G., Zandron, C. (2009). Introducing a space complexity measure for P systems. International Journal of Computers, Communications & Control **4**(3), 301–310, http://univagora.ro/jour/index.php/ijccc/article/view/2779. Accessed 29 Sept 2022
32. Porreca, A. E., Leporati, A., Mauri, G., & Zandron, C. (2011). P systems with active membranes: Trading time for space. *Natural Computing, 10*(1), 167–182. https://doi.org/10.1007/s11047-010-9189-x
33. Porreca, A. E., Leporati, A., Mauri, G., & Zandron, C. (2011). P systems with active membranes working in polynomial space. *International Journal of Foundations of Computer Science, 22*(1), 65–73. https://doi.org/10.1142/S0129054111007836
34. Porreca, A.E., Leporati, A., Mauri, G., Zandron, C. (2011). P systems with elementary active membranes: Beyond NP and coNP. In: Gheorghe, M., Hinze, T., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing, 11th International Conference, CMC 2010. Lecture Notes in Computer Science, vol. 6501, pp. 338–347. Springer, https://doi.org/10.1007/978-3-642-18123-8_26
35. Porreca, A.E., Leporati, A., Mauri, G., Zandron, C. (2013). Sublinear-space P systems with active membranes. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, G. (eds.) Membrane Computing, 13th International Conference, CMC 2012. Lecture Notes in Computer Science, vol. 7762, pp. 342–357. Springer, https://doi.org/10.1007/978-3-642-36751-9_23
36. Porreca, A. E., Mauri, G., & Zandron, C. (2006). Complexity classes for membrane systems. *RAIRO Theoretical Informatics and Applications, 40*(2), 141–162. https://doi.org/10.1051/ita:2006001
37. Porreca, A. E., Mauri, G., & Zandron, C. (2010). Non-confluence in divisionless P systems with active membranes. *Theoretical Computer Science, 411*(6), 878–887. https://doi.org/10.1016/j.tcs.2009.07.032
38. Sosík, P. (2003). The computational power of cell division in P systems: Beating down parallel computers? *Natural Computing, 2*(3), 287–298. https://doi.org/10.1023/A:1025401325428
39. Sosík, P. (2019). P systems attacking hard problems beyond NP: A survey. *Journal of Membrane Computing, 1*(3), 198–208. https://doi.org/10.1007/s41965-019-00017-y.
40. Valencia-Cabrera, L., Orellana-Martín, D., Martínez-del-Amor, M. A., Riscos-Núñez, A., & Pérez-Jiménez, M. J. (2017). Computational efficiency of minimal cooperation and distribution in polarizationless P systems with active membranes. *Fundamenta Informaticae, 153*(1–2), 147–172. https://doi.org/10.3233/FI-2017-1535
41. Valencia-Cabrera, L., Orellana-Martín, D., Martínez-del-Amor, M. A., Riscos-Núñez, A., & Pérez-Jiménez, M. J. (2017). Reaching efficiency through collaboration in membrane systems: Dissolution, polarization and cooperation. *Theoretical Computer Science, 701*, 226–234. https://doi.org/10.1016/j.tcs.2017.04.015
42. Valsecchi, A., Porreca, A.E., Leporati, A., Mauri, G., Zandron, C. (2010). An efficient simulation of polynomial-space Turing machines by P systems with active membranes. In: Păun, Gh., Pérez-Jiménez, M.J., Riscos-Núñez, A., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing, 10th International Workshop, WMC 2009, Lecture Notes in Computer Science, vol. 6501, pp. 461–478. Springer, https://doi.org/10.1007/978-3-642-11467-0_31
43. Zandron, C. (2020). Bounding the space in P systems with active membranes. *Journal of Membrane Computing, 2*(2), 137–145. https://doi.org/10.1007/s41965-020-00039-x
44. Zandron, C., Ferretti, C., Mauri, G. (2001). Solving NP-complete problems using P systems with active membranes. In: Antoniou, I., Calude, C.S., Dinneen, M.J. (eds.) Unconventional Models of Computation, UMC'2K, Proc. Second Int. Conference, pp. 289–301. Springer, https://doi.org/10.1007/978-1-4471-0313-4_21




**Artiom Alhazov** is a principal researcher at the Vladimir Andrunachievici Institute of Mathematics and Computer Science. He has achieved over 300 publications including over 60 ones in journals, together with over 60 coauthors. By the time of publication of this paper, Google Scholar reported his h-index was 25. He got his PhD thesis in Spain in 2006 and, after some postdoc positions in Finland, Japan, and Italy, he defended his Habilitation thesis in Moldova in 2013. His main research visits have been to Austria, China, and France. His research interests are centered in Theoretical Computer Science, including but not limited to descriptional complexity parameters of small computationally universal systems from a wide variety of formal models of parallel distributed processing of strings and multisets, giving special focus on membrane systems.

**Alberto Leporati** PhD, is an Associate Professor at the University of Milano-Bicocca, at the Department of Informatics, Systems and Communication. His research activity concerns the theory of computational complexity. In particular, he studies the computational power of models of computation which are inspired by the working of living cells (Membrane Computing) and the laws of quantum mechanics (Quantum Computing). On these subjects, he published more than 100 papers on international journals and in peer-reviewed proceedings of international conferences. He is also a member of the Steering Committee for the CMC and ACMC international conference series, and he serves as Vice President of the International Membrane Computing Society.

**Luca Manzoni** is an associate professor at the University of Trieste, Italy. He obtained his PhD in Computer Science at the University of Milano-Bicocca in 2013. In 2012, he obtained a JSPS postdoctoral fellowship. In 2017, he obtained an award as the best young postdoc in Computer Science and Mathematics at the University of Milano-Bicocca. He has published more than 80 papers in international journal, conferences, and workshops. His interests are in the areas of natural computing models, like P systems, reactions systems, and cellular automata, and in the area of evolutionary computation, and genetic programming in particular.

**Giancarlo Mauri** is professor emeritus of Computer Science at the University of Milano-Bicocca. His research interests include natural computing and unconventional computing models, in particular membrane systems and splicing systems; bioinformatics, in particular algorithms for NGS data analysis; computational systems biology, in particular stochastic modeling and simulation of biological systems and processes. On these subjects, he published about 500 scientific papers in international journals, contributed volumes and conference proceedings. He is or has been member of the steering committees of the International Conferences on DNA Computing, Membrane Computing, Unconventional Computing and Natural Computing, Developments in Language Theory, and of the International workshop on Cellular Automata for Research and Industry.

**Claudio Zandron** got the PhD in Computer Science from the University of Milan in 2002. Since 2006, he is associate professor at the Department of Informatics, Systems and Communication of the University of Milano-Bicocca, Italy. His research interests concern the areas of formal languages, molecular computing models, DNA computing, Membrane Computing, and computational complexity.