Journal of Membrane Computing (2021) 3:87–96
https://doi.org/10.1007/s41965-021-00074-2

REGULAR PAPER

# Alternative space definitions for P systems with active membranes

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

Received: 5 November 2020 / Accepted: 25 February 2021 / Published online: 16 April 2021

**Abstract**
The first definition of space complexity for P systems was based on a hypothetical real implementation by means of biochemical materials, and thus it assumes that every single object or membrane requires some constant physical space. This is equivalent to using a unary encoding to represent multiplicities for each object and membrane. A different approach can also be considered, having in mind an implementation of P systems in silico; in this case, the multiplicity of each object in each membrane can be stored using binary numbers, thus reducing the amount of needed space. In this paper, we give a formal definition for this alternative space complexity measure, we define the corresponding complexity classes and we compare such classes both with standard space complexity classes and with complexity classes defined in the framework of P systems considering the original definition of space.

**Keywords** Membrane Systems · Computational Complexity · Space Complexity

## 1 Introduction

P systems with active membranes have been introduced in [27], considering the idea of generating new membranes through the 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 [40], unless non-confluent systems are used [34]. In [35] 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 [6]. Relations with the classes **EXP** and **EXPSPACE** were investigated in [33].

A series of works then defined various complexity classes characterized by P systems that make use of different features. For instance, the works [12, 13] focused on the crucial role of membrane dissolution; polarizationless systems have been investigated in [4, 5, 11, 14]; constraints on membrane division [22] or on the depth of membrane structure [16] have been the subjects of other works, while [37, 38] focused on the role of cooperation.

More recently, other aspects have also been studied. In [1, 25] 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 [24] such kind of rules are applied in a different variant of P systems, having proteins on membranes. In [7, 10] solutions

✉ 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 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 Valerio 12/1, 34127 Trieste, Italy

for SAT are proposed which use different strategies than previously proposed solutions. Systems of a shallow depth are the subject of [17–19]. A recent survey on different strategies to approach computationally hard problems by P systems with active membranes can be found in [36].

A natural research topic that has been approached after the first works on time complexity concerns space complexity, a notion introduced for the first time in the framework of P systems in [29]. The definition was 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 [30, 31], as well as when only sublinear [32] or even constant amount of space [15] is available. A recent survey concerning results obtained by considering different bounds on space can be found in [39].

When defining space complexity for P systems, a different approach can also be considered, 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., [8, 9]), 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 this paper, we consider this option: we introduce a formal definition for this alternative space complexity measure, we define the corresponding complexity classes and we compare such classes both with standard space complexity classes defined for Turing machines and with complexity classes defined in the framework of P systems considering the original definition of space [29]. We will give results concerning the use of a constant, polynomial or exponential amount of space, respectively, trying to understand whether or not the classes of solvable problems differ.

The paper is organized as follows. In Section 2 we recall some definitions concerning P systems with active membranes and space requirements in P systems computations. In Section 3, we introduce a different definition for measuring space (which we call *binary space* to underline that information concerning objects is stored in binary) and we give some results following immediately from this definition. In Section 4 we compare the new binary space complexity classes with standard complexity classes and with space complexity classes for P systems based on the standard definition of space. Finally Section 5 draws some conclusions and presents some future research topics on this subject.

## 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* [28].

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

- $\Gamma$ is an alphabet, i.e., a finite non-empty set of symbols, usually called *objects*; in the following, we assume $\Gamma = \{O_1, O_2, \dots, O_n\}$;
- $\Lambda$ 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, labelled by elements of $\Lambda$ in a one-to-one way, defining regions (the space between a membrane and all membranes immediately inside it, if any);
- $w_{h_1}, \dots, w_{h_d}$, with $h_1, \dots, h_d \in \Lambda$, 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 labelled 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 labelled 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 labelled 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 labelled 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 labelled 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 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). However, to clarify the operational semantics, each computation step is conventionally described as a sequence of micro-steps as follows. First, all evolution rules are applied inside the elementary membranes, followed by all communication, dissolution and division rules involving the membranes themselves; this process is then repeated on the membranes containing them, and so on towards the root (outermost membrane). In other words, 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.

In order 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 [23].

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 to signal acceptance or rejection. If all the 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 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) 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 [26, Definition 7.1]). We say that $L$ belongs to the complexity class $\mathbf{MC}^\star_{\mathcal{D}}(f)$ 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 $\mathbf{PMC}^\star_{\mathcal{D}}$ 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 $\mathbf{NMC}^\star_{\mathcal{D}}(f)$ and $\mathbf{NPMC}^\star_{\mathcal{D}}$.

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 $\mathbf{MC}_{\mathcal{D}}(f)$ 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 $\mathbf{PMC}_{\mathcal{D}}$ 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 $\mathbf{NMC}_{\mathcal{D}}(f)$ and $\mathbf{NPMC}_{\mathcal{D}}$.

As stated in the Introduction, the first definition of space complexity for P systems introduced in [29] 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 [20], 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 $\mathcal{C} = (\mathcal{C}_0, \dots, \mathcal{C}_k)$ is a computation of $\Pi$, then the *space required by* $\mathcal{C}$ is defined as

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

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 $\mathbf{MCSPACE}_{\mathcal{D}}(f(n))$ (resp. $\mathbf{MCSPACE}^\star_{\mathcal{D}}(f(n))$) we denote the class of languages which can be decided by uniform (resp. semi-uniform) families, $\mathbf{\Pi}$, 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 $f(|x|)$.

In particular, the class of problems solvable in polynomial space by uniform confluent systems is denoted by $\mathbf{PMCSPACE}_{\mathcal{D}}$, and the class of problems solvable in exponential space by uniform confluent systems is denoted by $\mathbf{EXPMCSPACE}_{\mathcal{D}}$ (adding a star in case of semi–uniform classes).

The corresponding classes for non-confluent systems are $\mathbf{NPMCSPACE}_{\mathcal{D}}$ and $\mathbf{NEXPMCSPACE}_{\mathcal{D}}$.

## 3 An alternative definition of space complexity for P systems

In this section, we first give a different definition of space complexity for P systems with active membranes. This definition considers the information stored in the objects of the systems, and not the single objects themselves. In other words, we store, using binary numbers, the multiplicity of each object in each membrane, thus reducing the amount of needed space with respect to the definition of space given in the previous section.

We will do this considering, for each region, a sequence of couples, describing how many occurrences of each object are present (only for objects having at least one occurrence in the region). As an example, considering an (ordered) alphabet $\Gamma = \{a, b, c, d\}$, a multiset $a^2, b^5, d^6$ can be described by the sequence of couples (010, 00), (101, 01), (110, 11) (where (010,00) corresponds to 2 occurrences of the first symbol in $\Gamma$, that is $a$, (101,01) to 5 occurrences of the second symbol $b$, etc.). Of course, different descriptions can also be considered: for instance, the bits describing the object can be avoided if we give, in order, the amount of each object, including objects having zero occurrences (sometimes this would allow to save space, but sometimes this would require more space, like in the case of sparse information - see, e.g., [21]). We leave as an open research topic the question whether or not different descriptions allow improvements in space usage.

We will refer to this definition of space by *binary space*, and we will add a symbol $B$ where appropriate, to distinguish between the definitions referring to this new measure and the definitions recalled in the previous section.

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

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

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:

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

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 binary space bound* $s$ if and only if $|\Pi_x|_B \le s(|x|)$ for each $x \in \Sigma^\star$.

We can thus define space complexity classes considering this newly introduced size measure, like we did in the previous section. By $\mathbf{MCBSPACE}_{\mathcal{D}}(f(n))$ (resp. $\mathbf{MCBSPACE}^*_{\mathcal{D}}(f(n))$) we denote the class of languages which can be decided by uniform (resp. semi–uniform) families, $\mathbf{\Pi}$, of confluent P systems of type $\mathcal{D}$, where each $\Pi_x \in \mathbf{\Pi}$ operates within space bound $f(|x|)$, considering this new definition of binary space. Similarly, we can define the usual complexity classes like we did in the previous section, simply adding a $B$ to underline the use of this new definition of space. For instance, the class of problems solvable by uniform (resp. semi–uniform) systems in polynomial binary space will be denoted by $\mathbf{PMCBSPACE}_{\mathcal{D}}$ (resp. $\mathbf{PMCBSPACE}^*_{\mathcal{D}}$).

Once these notions have been defined, we are ready to state some results obtained by considering various complexity classes defined in terms of binary space. Just like it happens with the classes based on the original definition of space given in [29], some results follow immediately from the definitions (we denote a result that holds for both semi-uniform and uniform systems by [$\star$]):

**Proposition 1** *The following inclusions hold:*

$$\mathbf{PMCBSPACE}^{[\star]}_{\mathcal{D}} \subseteq \mathbf{EXPMCBSPACE}^{[\star]}_{\mathcal{D}}$$
$$\mathbf{NPMCBSPACE}^{[\star]}_{\mathcal{D}} \subseteq \mathbf{NEXPMCBSPACE}^{[\star]}_{\mathcal{D}}.$$

**Proposition 2** $\mathbf{MCBSPACE}^{[\star]}_{\mathcal{D}}(f) \subseteq \mathbf{NMCBSPACE}^{[\star]}_{\mathcal{D}}(f)$ *for each* $f : \mathbb{N} \to \mathbb{N}$, *and in particular*

$$\mathbf{PMCBSPACE}^{[\star]}_{\mathcal{D}} \subseteq \mathbf{NPMCBSPACE}^{[\star]}_{\mathcal{D}}$$
$$\mathbf{EXPMCBSPACE}^{[\star]}_{\mathcal{D}} \subseteq \mathbf{NEXPMCBSPACE}^{[\star]}_{\mathcal{D}}.$$

The results describing closure properties and providing an upper bound for time requirements of P systems operating in bounded binary space are still valid, too:

**Proposition 3** *The complexity classes* $\mathbf{PMCBSPACE}^{[\star]}_{\mathcal{D}}$, $\mathbf{NPMCBSPACE}^{[\star]}_{\mathcal{D}}$, $\mathbf{EXPMCBSPACE}^{[\star]}_{\mathcal{D}}$, *and* $\mathbf{NEXPMCBSPACE}^{[\star]}_{\mathcal{D}}$ *are all closed under polynomial-time reductions.*

***Proof*** Consider a language $L \in \mathbf{PMCBSPACE}^{\star}_{\mathcal{D}}$ and let $M$ be the Turing machine constructing the family $\mathbf{\Pi}$ that decides $L$. Let $L'$ be reducible to $L$ via a polynomial-time computable function $f$.

We can build a Turing machine $M'$ working as follows: on input $x$ of length $n$, $M'$ computes $f(x)$; then it behaves like $M$ on input $f(x)$, thus constructing $\Pi_{f(x)}$ (we stress the fact that, for the corresponding result concerning the uniform case, the construction of the P system involves two Turing machines, both operating in polynomial time; in this case, we simulate the composition of the two machines). Since $|f(x)|$ is bounded by a polynomial, $M'$ operates in polynomial time and $\Pi_{f(x)}$ in polynomial binary space; it follows that $\mathbf{\Pi}' = \{\Pi_{f(x)} \mid x \in \Sigma^\star\}$ is a polynomially semi-uniform family of P systems deciding $L'$ in polynomial binary space. Thus $L' \in \mathbf{PMCBSPACE}^{\star}_{\mathcal{D}}$.

The proofs for the three other classes and for the corresponding uniform classes are analogous. □

**Proposition 4** $\mathbf{MCBSPACE}^{[\star]}_{\mathcal{D}}(f)$ *is closed under complement for each function* $f : \mathbb{N} \to \mathbb{N}$.

***Proof*** By reversing the roles of objects *yes* and *no*, the complement of a language can be decided. □

**Proposition 5** *For each function* $f : \mathbb{N} \to \mathbb{N}$

$$\mathbf{MCBSPACE}^{[\star]}_{\mathcal{D}}(f(n)) \subseteq \mathbf{MC}^{[\star]}_{\mathcal{D}}\left(2^{O(f(n))}\right)$$
$$\mathbf{NMCBSPACE}^{[\star]}_{\mathcal{D}}(f(n)) \subseteq \mathbf{NMC}^{[\star]}_{\mathcal{D}}\left(2^{O(f(n))}\right).$$

***Proof*** Let $L \in \mathbf{MCBSPACE}^{\star}_{\mathcal{D}}(f(n))$ be decided by the semi-uniform family $\mathbf{\Pi}$ of recognizer P systems in binary space $f$; let $\Pi_x \in \mathbf{\Pi}$ with $|x| = n$ and let $\mathcal{C}$ be a configuration of $\Pi_x$.

The configuration $\mathcal{C}$ is described by the membrane structure and the objects inside it. The information concerning objects is stored using bits, as described above. The membrane structure can be stored directly using a bracketed expression. For $z$ membranes the binary space allocated requires $z \times log(|\Lambda|)$ bits; even by adding a constant number of bits for each bracket corresponding to each membrane, the space required is $O(z \times log(|\Lambda|))$. The binary space required by $\Pi_x$ is then $O\left(z \cdot \log|\Lambda| + \sum_{j=1}^{z}\sum_{i=1}^{n}\left(\lceil \log(|O_{i,j}| + 1)\rceil + \log|\Gamma|\right)\right) = O(f(n))$.

Since $\Pi_x$ is a recognizer P system, by definition every computation halts: then it must halt within $2^{O(f(n))}$ steps in order to avoid repeating a configuration.

The same argument, with only some small differences, also works in the non-confluent case. All possible computations halt, even if not necessarily agreeing on the answer. Due to non-confluence, each computation can also contain repeated configurations. Nonetheless, for each computation containing a repeated configuration, there exists an equivalent one obtained by removing cycles in the computation path.

The proof for the uniform classes is analogous. □

## 4 Comparison with standard computational complexity classes

In this section, we compare the standard computational complexity classes with the complexity classes defined in the framework of P systems working in binary space.

Most results are an immediate consequence of the results given in [29], simply considering that $\mathbf{MCSPACE}_{\mathcal{D}}(f(n)) \subseteq \mathbf{MCBSPACE}_{\mathcal{D}}(f(n))$.

Thus, recalling various results from [29], we have:

**Proposition 6** *Let us denote by* $\mathcal{EAM}$ *and* $\mathcal{AM}^0$ *the classes of P systems with active membranes using only elementary membrane division and without polarizations, respectively. The following results hold:*

$$\mathbf{NP} \cup \mathbf{coNP} \subseteq \mathbf{EXPMCSPACE}^{\star}_{\mathcal{EAM}} \subseteq \mathbf{EXPMCBSPACE}^{\star}_{\mathcal{EAM}}$$

$$\mathbf{PSPACE} \subseteq \mathbf{EXPMCSPACE}^{\star}_{\mathcal{AM}} \subseteq \mathbf{EXPMCBSPACE}^{\star}_{\mathcal{AM}}$$

$$\mathbf{PSPACE} \subseteq \mathbf{EXPMCSPACE}_{\mathcal{AM}} \subseteq \mathbf{EXPMCBSPACE}^{\star}_{\mathcal{AM}}$$

$$\mathbf{PSPACE} \subseteq \mathbf{EXPMCSPACE}^{[\star]}_{\mathcal{AM}^0} \subseteq \mathbf{EXPMCBSPACE}^{[\star]}_{\mathcal{AM}^0}$$

An interesting research topic concerns the classes for which the inclusion $\mathbf{MCSPACE}_{\mathcal{D}}(f(n)) \subseteq \mathbf{MCBSPACE}_{\mathcal{D}}(f(n))$ is proper and, considering the above inclusions, whether or not the same results can be obtained with stricter binary space classes, by exploiting the improved information storage related to objects with respect to the standard space definition.

By considering a constant amount of space, in the semi-uniform case, the following result holds:

**Theorem 7** $\mathbf{P} = \mathbf{MCSPACE}^{\star}_{\mathcal{AM}}(O(1)) = \mathbf{MCBSPACE}^{\star}_{\mathcal{AM}}(O(1))$.

***Proof*** The inclusion $\mathbf{P} \subseteq \mathbf{MCSPACE}^{\star}_{\mathcal{AM}}(O(1))$ follows immediately from the definition of semiuniform P systems. Consider a language $L$ in $\mathbf{P}$ and a string $x$; a deterministic Turing machine can create in polynomial time a P system having a single membrane and one single object *yes* or *no*, directly answering the question whether or not $x \in L$. The inclusion $\mathbf{MCSPACE}^{\star}_{\mathcal{AM}}(O(1)) \subseteq \mathbf{MCBSPACE}^{\star}_{\mathcal{AM}}(O(1))$ follows, as stated above, from the definition of binary space.

For the converse, we simply need to recall that a confluent semiuniform P system without membrane division can be simulated, in polynomial time, by a deterministic Turing machine, like it was shown in [40]. It is easy to see that the proof works both considering the standard space definition as well as the binary space definition for P systems. Even when the division of membranes is allowed, but the system can only use an amount of space that is limited to a constant, the total number of membranes is limited by a constant and, as a consequence, the total number of configurations is polynomially bounded. Hence, the same simulation is still valid. □

It follows that, for semiuniform systems, when we allow only a constant amount of space, the improved storage allowed by binary space does not lead to improved efficiency.

Another interesting result concerning the standard definition of space in the framework of P systems was presented in [30], and it focuses on the type of resources used. In particular, a solution for the **PSPACE**-complete problem Quantified 3SAT was given, for uniform systems using only communication rules (hence no evolution, membrane division and dissolution rules were used), thus proving the inclusion of **PSPACE** in this class. Once again, since the definition of binary space allows a more efficient allocation of space, the result is still valid:

**Proposition 8** *Let us denote by* $\mathcal{AM}$(-ev,+com,-dis,-div) *the class of P systems with active membranes using only communication rules (while rules for object evolution, dissolution, and division of membranes are not used). Then* $\mathbf{PSPACE} \subseteq \mathbf{PMCBSPACE}^{[\star]}_{\mathcal{AM}(\text{-ev,+com,-dis,-div})}$.

Once again, it would be interesting to understand whether or not the result remains valid for a smaller binary space class. In this case, the question can be answered negatively, by considering a result presented in [31]. In the article, it was shown that recognizer P systems with active membranes using polynomial space characterize the complexity class **PSPACE**. The result holds for both confluent and nonconfluent systems, and even in the case that non-elementary division is used. In particular, it was pointed out that such systems can be simulated by polynomial space Turing machines.

By considering the alternative definition for binary space, we can thus obtain the corresponding theorem:

**Theorem 9** *Let $\Pi$ be a nonconfluent P system with active membranes, running in binary space $S$. Then, it can be simulated by a deterministic Turing machine in space $O(S^2)$.*

***Proof*** We simulate $\Pi$ by means of a non-deterministic Turing machine $N$. The current configuration of $\Pi$ can be stored explicitly by $N$: the membrane structure is represented directly by using a bracketed expression, while multisets of objects inside each region are stored by means of tuples of integers encoded in binary. Of course, the same considerations we made in the proof of Proposition 5 hold also in this case.

For the simulation, we can use the same algorithm as in [31]: the space required by $N$ to store further information needed to carry on the simulation is then limited by $S$. It follows that the total amount of space required by $N$ is of the same order as the one required by $\Pi$, that is, $O(S)$.

Using Savitch's theorem [26], it is straightforward to see that $N$ (and thus $\Pi$) can be simulated by a deterministic Turing machine in space $O(S^2)$. □

It follows immediately from this theorem, from the results in [31], and from Proposition 8:

**Theorem 10** *Let $\mathcal{D}$ be a class of P systems with active membranes using at least communication rules. Then*

$$[\mathbf{N}]\mathbf{PMCBSPACE}^{[\star]}_{\mathcal{D}} = [\mathbf{N}]\mathbf{PMCSPACE}^{[\star]}_{\mathcal{D}} = \mathbf{PSPACE},$$

*where* $[\mathbf{N}]$ *denotes optional nonconfluence, and* $[\star]$ *optional semi-uniformity.*

Hence, even when a polynomial amount of space is used, the complexity classes defined on the basis of the definition of binary space coincide with the complexity classes defined in terms of the original definition of space (for systems using at least communication rules).

In [3] it was shown that exponential space Turing machines can be simulated by polynomially uniform exponential-space P systems with active membranes. In view of this result and of Theorem 9, and of the definition of binary space, we have the following:

**Theorem 11** *The following equivalences hold for an exponential amount of space:*

$$\begin{aligned}\mathbf{EXPSPACE} &= \mathbf{EXPMCBSPACE}_{\mathcal{AM}} \\ &= \mathbf{EXPMCBSPACE}^{\star}_{\mathcal{AM}} \\ &= \mathbf{NEXPMCBSPACE}^{\star}_{\mathcal{AM}}.\end{aligned}$$

*Proof* The following inclusions hold by definition:

$$\mathbf{EXPMCBSPACE}_{\mathcal{AM}} \subseteq \mathbf{EXPMCBSPACE}^{\star}_{\mathcal{AM}} \subseteq \mathbf{NEXPMCBSPACE}^{\star}_{\mathcal{AM}}.$$

For what concerns the inclusion $\mathbf{NEXPMCBSPACE}^{\star}_{\mathcal{AM}} \subseteq \mathbf{EXPSPACE}$, it is an immediate corollary of Theorem 9.

Finally, the inclusion of **EXPSPACE** in $\mathbf{EXPMCSPACE}_{\mathcal{AM}}$ is proved in [3, Theorem 8]. Recalling that $\mathbf{EXPMCSPACE}_{\mathcal{AM}} \subseteq \mathbf{EXPMCBSPACE}_{\mathcal{AM}}$, it follows $\mathbf{EXPSPACE} \subseteq \mathbf{EXPMCBSPACE}_{\mathcal{AM}}$. □

Also in this case, considering binary space instead of the standard one does not result in improved efficiency. In fact, considering the theorem just proved and recalling [2, Corollary 1] proving the same results for classes with the original definition of space for P systems, we can prove that such classes are equal to **EXPSPACE**:

**Corollary 12**

$$\begin{aligned}\mathbf{EXPSPACE} = \mathbf{EXPMCSPACE}_{\mathcal{AM}} &= \mathbf{EXPMCSPACE}^{\star}_{\mathcal{AM}}\\ &= \mathbf{NEXPMCSPACE}^{\star}_{\mathcal{AM}} = \mathbf{EXPMCBSPACE}_{\mathcal{AM}}\\ &= \mathbf{EXPMCBSPACE}^{\star}_{\mathcal{AM}} = \mathbf{NEXPMCBSPACE}^{\star}_{\mathcal{AM}}\end{aligned}$$

.

## 5 Conclusions

We have proposed an alternative space complexity measure for P systems with active membranes, where the multiplicity of each object in each membrane is stored by using binary numbers. We have defined the corresponding complexity classes and we have compared some of them both with standard space complexity classes and with complexity classes defined in the framework of P systems considering the original definition of space [29].

It turned out that, for various considered systems, the computational classes defined on the basis of binary space do not differ from the corresponding classes defined on the basis of the original space definition for P systems. Among the various systems for which we proved such a result, we underline in particular that this is the case when we consider systems using all features of P systems with active membranes and a polynomial or exponential amount of space, as well as for semiuniform systems working in a constant space.

It would be interesting to find other classes for which the improved store efficiency obtained by considering binary space does not make any difference in computational efficiency, and to understand which features can be used/are necessary to obtain the same result. It also remains as an open problem to find, on the contrary, specific classes where this difference exists, thus proving that storing the information concerning objects in an efficient way can really be exploited in some cases. We conjecture, for instance, that this is the case for complexity classes defined by systems using a logarithmic amount of space.

Another possible research direction is to consider further variants of definition for space, and compare them with standard and binary space, or to consider the space required to describe the whole system executing the computation, that is including not only the data (object and membranes, in our case) but also the program (the rules, in our case).

**Acknowledgements** This work was partially supported by Università degli Studi di Milano-Bicocca, Fondo di Ateneo per la Ricerca 2018, project 2018-ATE-0527.

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

## Compliance with ethical standards

**Conflict of interest** Artiom Alhazov, Alberto Leporati, Luca Manzoni, Giancarlo Mauri and Claudio Zandron state 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., Mauri, G., Porreca, A.E., & Zandron, C. (2012). The computational power of exponential-space P systems with active membranes. In: Martínez-del-Amor, M.A., Păun, Gh., Pérez-Hurtado, I., Romero-Campero, F.J. (eds.) Tenth Brainstorming Week on Membrane Computing, Volume I. pp. 35–60. No. 1/2012 in RGNC Reports, Fénix Editora, http://www.gcn.us.es/icdmc2012_proceedings.
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., & Pan, L. (2004). Polarizationless P systems with active membranes. *Grammars*, *7*, 141–159.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. Leporati, A., Ferretti, C., Mauri, G., & Zandron, C. (2008). Complexity aspects of polarizationless membrane systems. *Natural Computing*, *4*(8), 703–717.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. Martinez del Amor, M., Orellana Martín, D., Cabarle, F.G., Pérez Jiménez, M.D., & Adorna, H.N. (2017). Sparse-matrix representation of spiking neural P systems for GPUs. In: Proceedings of BWMC 2017: 15th Brainstorming Week on Membrane Computing. pp. 161–170, https://idus.us.es/handle/11441/67895.
22. 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.
23. 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.
24. 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.
25. 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.
26. Papadimitriou, C. H. (1993). *Computational Complexity*. New York: Addison-Wesley.
27. Păun, Gh. (2001). P systems with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages and Combinatorics*, *6*(1), 75–90.
28. Păun, Gh, Rozenberg, G., & Salomaa, A. (Eds.). (2010). *The Oxford Handbook of Membrane Computing*. Oxford: Oxford University Press.
29. 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.
30. 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.
31. 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.
32. 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.
33. 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.
34. 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.
35. 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.
36. 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.
37. 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.
38. 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.
39. 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.
40. 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.