Complexity aspects of polarizationless membrane systems

Alberto Leporati · Claudio Ferretti · Giancarlo Mauri · Mario J. Pérez-Jiménez · Claudio Zandron

Abstract We investigate polarizationless P systems with active membranes working in maximally parallel manner, which do not make use of evolution or communication rules, in order to find which features are sufficient to efficiently solve computationally hard problems. We show that such systems are able to solve the **PSPACE**-complete problem QUANTIFIED 3-SAT, provided that non-elementary membrane division is controlled by the presence of a (possibly non-elementary) membrane.

Keywords Membrane computing · Computational complexity · PSPACE complete problems

1 Introduction

Membrane systems (also known as *P systems*) have been introduced in Păun (2000a) as a parallel, nondeterministic, synchronous and distributed model of computation inspired by the structure and functioning of living cells. The basic model consists of a hierarchical

A. Leporati · C. Ferretti · G. Mauri · C. Zandron (✉)
Dipartimento di Informatica, Sistemistica e Comunicazione, Università di Milano–Bicocca,
Viale Sarca 336, I-20126 Milan, Italy
e-mail: zandron@disco.unimib.it

A. Leporati
e-mail: leporati@disco.unimib.it

C. Ferretti
e-mail: ferretti@disco.unimib.it

G. Mauri
e-mail: mauri@disco.unimib.it

M. J. Pérez-Jiménez
Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence,
University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
e-mail: marper@us.es

structure composed by several membranes, embedded into a main membrane called the *skin*. Membranes divide the Euclidean space into *regions*, that contain some *objects* (represented by symbols of an alphabet) and *evolution rules*. Using these rules, the objects may evolve and/or move from a region to a neighboring one. Usually, the rules are applied in a nondeterministic and maximally parallel way: all the objects that may evolve are forced to evolve. A *computation* starts from an initial configuration of the system and terminates when no evolution rule can be applied. The result of a computation is the multiset of objects contained into an *output membrane*, or emitted from the skin of the system. An interesting subclass of membrane systems is constituted by *recognizer* P systems, in which: (1) all computations halt, (2) only two possible outputs exist (usually named `yes` and `no`), and (3) the result produced by the system depends only upon its input, and is not influenced by the particular sequence of computation steps taken to produce it. For a systematic introduction to P systems we refer the reader to Păun (2002), whereas the latest information can be found in http://ppage.psystems.eu/.

Since the introduction of membrane systems, many investigations have been performed on their computational and complexity aspects. It is known that every deterministic Turing machine working in polynomial time can be simulated in polynomial time by a family of recognizer P systems using only basic rules, that is, evolution, communication, and rules involving dissolution (Pérez-Jiménez et al. 2004). On the other hand, if a decision problem is solvable in a polynomial time by a family of recognizer P systems (using only basic rules), then there exists a deterministic Turing machine solving it in a polynomial time (Zandron et al. 2000). As a consequence of these results, the class of all decision problems that can be solved in a polynomial time by this kind of P systems is equal to the standard complexity class **P** (Gutiérrez-Naranjo et al. 2006b). For that reason, recognizer P systems that build an exponential workspace (expressed in the number of objects) in a polynomial time cannot solve **NP**-complete problems in a polynomial time unless $\mathbf{P} = \mathbf{NP}$. Hence, under the assumption that $\mathbf{P} \neq \mathbf{NP}$, in order to efficiently solve computationally hard problems by means of P systems it seems necessary to be able to construct in a polynomial time an exponential workspace, expressed by the number of *membranes*. The computation models thus obtained, usually called *P systems with active membranes*, abstract the way of producing new biological membranes through the processes of *mitosis* (membrane division) and *autopoiesis* (membrane creation).

Membrane division has been successfully used many times in the literature to efficiently solve **NP**-complete problems. The first solutions were given constructing a P system associated with each instance of the problem, i.e., working in the so called *semi-uniform* setting. Actually, we say that this kind of solutions are *semi-uniform* if the following two conditions hold:

- There exists a deterministic Turing machine that, given any instance $\mathcal{I}$ of the problem Q under consideration, builds in a polynomial time (with respect to the *size* of $\mathcal{I}$) the P system $\Pi_{Q,\mathcal{I}}$ that processes the instance $\mathcal{I}$. We also say that the family $\{\Pi_{Q,\mathcal{I}}\}$ of P systems associated with all the instances of Q is *polynomially uniform by Turing machines*.
- The instance $\mathcal{I}$ of the problem has an affirmative answer if and only if every computation of the P system $\Pi_{Q,\mathcal{I}}$ associated with it is an *accepting* computation (we also say that the system $\Pi_{Q,\mathcal{I}}$ is *confluent*).

The first semi-uniform polynomial–time solutions of computationally hard decision problems were given by Păun (2000b, 2001), Zandron et al. (2000), Krishna et al. (1999),

and Obtulowicz (2001). In 2003, Sosik (2003) gave a semi-uniform polynomial-time solution to QSAT, a well known **PSPACE**-complete problem.

Another way to solve decision problems by means of P systems considers the possibility to have an input membrane in the systems, in which we can introduce objects before the system starts to work. In this case, all the instances of a decision problem having the same size (according to a prefixed polynomial time criterion) are processed by the same system; we call this an *uniform* solution. P systems with active membranes have also been successfully used to design uniform polynomial–time solutions to some well known **NP**-complete problems, such as SAT (Pérez-Jiménez et al. 2003), SUBSET SUM (Pérez-Jiménez and Riscos-Núñez 2005), KNAPSACK (Pérez-Jiménez and Riscos-Núñez 2004), PARTITION (Gutiérrez-Naranjo et al. 2005), and the COMMON ALGORITHMIC PROBLEM (Pérez-Jiménez and Romero-Campero 2005).

All the papers mentioned above deal with P systems whose membranes have three polarizations, use only division of elementary membranes (in Sosik (2003) also division rules for non-elementary membranes are permitted) and work in *maximally parallel* way, meaning that in each computation step the application of rules is maximal, i.e., no further rule can be applied in any region. The number of polarizations can be decreased to *two* without loss of efficiency and computational power, as shown in (Alhazov and Freund 2005).

By looking at the solutions described in the above papers, it seems clear that the usual framework of P systems with active membranes is too powerful from the traditional complexity point of view. Hence, it would be interesting to analyse which features allow P systems with active membranes, but without polarizations, to still get polynomial–time solutions to computationally hard problems, and what features, once removed, only allow us to obtain polynomial–time solutions to tractable problems, in the classical sense. In this direction, in Păun (2005) a conjecture was formulated by Păun about the computational power of polarizationless P systems with active membranes and working in the maximally parallel mode, stating that such systems can only solve decision problems that are in **P** (by using only elementary division). In Mauri et al. (2007), some partial answers were given. In particular, it was shown that it is possible to solve the **NP**-complete problem SAT by polarizationless P systems with active membranes which use evolution, communication, dissolution and membrane division rules. When dissolution of membranes is not allowed, then only problems in the complexity class **P** can be solved. In the same paper it was shown that the same results hold even if the rules are applied in the *minimally parallel* way (Ciobanu et al. 2007): in each region where at least one rule can be applied, at least one rule must be applied (if there is no conflict with the objects). In Zandron et al. (to appear) it was shown that polarizationless P systems with active membranes that use only division of non-elementary membranes and dissolution rules, working in the maximally parallel way, are able to solve in polynomial time the **NP**-complete problem 3-SAT. This result provides further partial answers to Păun's conjecture, establishing that neither evolution nor communication rules, and no electrical charges are needed to solve **NP**-complete problems, provided that we can use strong division rules for non-elementary membranes (as well as dissolution rules, otherwise we would fall in the case considered in Gutiérrez-Naranjo et al. (2006a)).

Continuing in this direction, in the present paper we show that the **PSPACE**-complete problem QUANTIFIED 3-SAT can be solved by recognizer P systems with active membranes, without making use of evolution rules, communication rules and polarizations, provided that the activation of strong division rules for non-elementary membranes is controlled by the presence of a membrane (an additional feature, not required in Zandron et al.

(to appear) to solve 3-SAT). This result is complementary to the one given in Alhazov and Pérez-Jiménez (2006), where an efficient uniform solution to QSAT is given by using also evolution rules.

The paper is organized as follows. In Sect. 2 and 3 we recall the definition of polarizationless recognizer P systems with active membranes, thus establishing our model of computation, and we recall the definition of the **PSPACE**-complete decision problem QUANTIFIED 3-SAT. In Sect. 4 we show how this problem can be solved by means of recognizer P systems with active membranes, without evolution and communication rules and without polarizations. In Sect. 5 we draw some conclusions and we give directions for further investigations.

2 Polarizationless recognizer P systems with active membranes

Usually, P systems with active membranes are defined in the literature with three electrical charges (also called *polarizations*) associated with the membranes (even thought two charges suffice, as proved in Alhazov and Freund (2005)) to control the application of the rules, which can be of the following types: *evolution rules*, by which single objects evolve to a multiset of objects, *communication rules*, by which an object is introduced in or expelled from a membrane, and possibly changed to another object while performing this operation, *dissolution rules*, by which a membrane is dissolved under the influence of an object, that can also be modified during this operation, and *membrane division rules* (both for elementary and non-elementary membranes, or only for elementary membranes). However, in this paper we will consider *polarizationless* P systems with active membranes, that is, P systems in which no electrical charge is associated with any membrane.

Formally, a P system with polarizationless active membranes of the initial degree $n \geq 1$ is a tuple of the form $\Pi = (\Gamma, H, \mu, w_1, ..., w_n, R, h_0)$, where:

1. Γ is the alphabet of objects;
2. H is a finite set of labels for membranes;
3. μ is a membrane structure, consisting of n membranes injectively labelled with elements of H;
4. $w_1, ..., w_n$ are strings over Γ, describing the multisets of objects placed in the n initial regions of μ;
5. R is a finite set of developmental rules, of the following forms:
   (a) $[a \rightarrow v]_h$, for $h \in H$, $a \in \Gamma$, $v \in \Gamma^*$ (object evolution rules);
   (b) $a[\ ]_h \rightarrow [b]_h$, for $h \in H$, $a, b \in \Gamma$ (in communication rules);
   (c) $[a]_h \rightarrow b[\ ]_h$, for $h \in H$, $a, b \in \Gamma$ (out communication rules);
   (d) $[a]_h \rightarrow b$, for $h \in H$, $a, b \in \Gamma$ (dissolution rules);
   (e) $[a]_h \rightarrow [b]_h[c]_h$, for $h \in H$, $a, b, c \in \Gamma$ (weak division rules for elementary or non-elementary membranes);
   (f) $h_0 \in H$ or $h_0 = env$ indicates the output region (in the latter case, usually h_0 does not appear in the description of the system).

We can also consider rules of the form $[[\ ]_{h_1}[\ ]_{h_2}]_{h_3} \rightarrow [[\ ]_{h_1}]_{h_3}[[\ ]_{h_2}]_{h_3}$, where h_1, h_2, h_3 are labels from H: if the membrane with label h_3 contains other membranes than those with labels h_1, h_2, these membranes and their contents are duplicated and placed in both new copies of the membrane h_3; all membranes and objects placed inside membranes h_1, h_2, as well as the objects from membrane h_3 placed outside membranes h_1 and h_2, are

reproduced in the new copies of membrane h_3. These rules are called *strong division rules for non-elementary membranes*. A further variant of these rules, that will be used in this paper, is obtained by requiring that the application of a strong division rule is subject to the presence of a specified membrane in h_3, which in turn may be elementary or non-elementary. We will call these last rules *controlled* strong division rules for non-elementary membranes.

As usual, a computation starts in the *initial configuration*, which is given by the membrane structure μ (where all the membranes have neutral polarizations) and the strings (multisets) w_1, ..., w_n of objects initially present in the n regions of μ. Using the *maximally parallel manner*, at each computation step (a global clock is assumed) in each region of the system we apply the rules in such a way that no further rule can be applied to the remaining objects or membranes. In each step, every object and every membrane can be involved in only one rule. The application of a maximal set of rules during a computation step produces a new configuration of the system. A *computation* is a sequence C_0, C_1, ... of configurations such that $\mu = C_0$ is an initial configuration, and for all $i \geq 1$ the configuration C_i is obtained from C_{i-1} by applying a maximal set of rules as described above. A computation *halts* if the system reaches an *halting* configuration, that is, a configuration in which no rule can be applied anywhere in the system. A halting computation provides a *result* given by the number of objects present in region h_0 at the end of the computation; this is a region of the system if $h_0 \in H$ (and in this case, for a computation to be successful, exactly one membrane with label h_0 should be present in the halting configuration), or it is the environment if $h_0 = env$. An infinite computation produces no result.

A *recognizer* P system with active membranes is obtained from the definition given above by assuming that the system halts on every computation and produces one of two possible outputs, that are usually denoted by `yes` and `no`. A further requirement is that the system is *confluent*, that is, for any given input configuration, all the computations that can start with such a configuration end by producing the same output. In this way, we can say that a recognizer P system with active membranes recognizes the language which is composed by the strings that encode the initial configurations that produce `yes` (or any another symbol specified by the system to represent it) as a result. By considering the trivial bijection existing between these languages and decision problems, we can also say that a recognizer P system solves the decision problem whose positive instances are associated with the initial configurations of the system that produce as output the symbol chosen to represent `yes`.

We denote by $\mathcal{AM}^0$ the class of polarizationless recognizer P systems with active membranes, and we denote by $\mathcal{AM}^0(\alpha, \beta, \gamma, \delta)$, where $\alpha \in \{-d, +d\}$, $\beta \in \{-ne, +new, +nes, +necs\}$, $\gamma \in \{-ev, +ev\}$, and $\delta \in \{-comm, +comm\}$ the class of all recognizer P systems with polarizationless active membranes such that: (a) if $\alpha = +d$ (resp., $\alpha = -d$) then dissolution rules are permitted (resp., forbidden); (b) if $\beta \in \{+new, +nes\}$ (resp., $\beta = -ne$) then division rules for elementary and non-elementary membranes, weak or strong (resp., only division rules for elementary membranes) are permitted; if $\beta = +necs$ then *controlled* strong division rules for non-elementary membranes are permitted; (c) if $\gamma = +ev$ (resp., $\gamma = -ev$) then evolution rules are permitted (resp., forbidden); (d) if $\delta = +comm$ (resp., $\delta = -comm$) then communication rules are permitted (resp., forbidden).

The class of all decision problems which can be solved in uniform (resp., semi-uniform) way, and in polynomial time by a family $\mathcal{R}$ of recognizer membrane systems is denoted by $\mathbf{PMC}_{\mathcal{R}}$ (resp., $\mathbf{PMC}^*_{\mathcal{R}}$). The following inclusions directly follow from these definitions.

Proposition 1 *For all* $\alpha \in \{-d, +d\}$, $\beta \in \{-n\ e\ , +new, +nes, +necs\}$, $\gamma \in \{-ev, +ev\}$, $\delta \in \{-comm, +comm\}$ *and* $\varepsilon \in \{*, \lambda\}$:

1. $\mathbf{PMC}_{\mathcal{AM}^0(\alpha,\beta,\gamma,\delta)} \subseteq \mathbf{PMC}^*_{\mathcal{AM}^0(\alpha,\beta,\gamma,\delta)}$
2. $\mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(-d,\beta,\gamma,\delta)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(+d,\beta,\gamma,\delta)}$
3. $\mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,-ne,\gamma,\delta)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,+new,\gamma,\delta)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,+nes,\gamma,\delta)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,+necs,\gamma,\delta)}$
4. $\mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,\beta,-ev,\delta)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,\beta,+ev,\delta)}$
5. $\mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(,alpha,,beta,,gamma,-comm)} \subseteq \mathbf{PMC}^{\varepsilon}_{\mathcal{AM}^0(\alpha,\beta,\gamma,+comm)}$

where $\varepsilon = *$ (resp., $\varepsilon = \lambda$, the empty string) means that the complexity classes are associated with semi-uniform (resp., uniform) solutions.

Using this notation, Păun's conjecture (problem **F** in Păun 2005) can be restated as follows:

$$\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0(+d,-ne,+ev,+comm)} = \mathbf{PMC}^*_{\mathcal{AM}^0(+d,-ne,+ev,+comm)}$$

As stated in the Introduction, the results of Gutiérrez-Naranjo et al. (2006a and Mauri et al. (2007) proved the following theorem, considering a reachability problem (is the state in which the symbol `yes` is expelled to the environment reachable?) defined on the so called *dependency graph*. We refer the reader to Gutiérrez-Naranjo et al. (2006a) and Mauri et al. (2007) for the details.

Theorem 1 *For all* $\beta \in \{-ne, +new, +nes\}$

$$\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0(-d,\beta,+ev,+comm)} = \mathbf{PMC}^*_{\mathcal{AM}^0(-d,\beta,+ev,+comm)}$$

This result holds for systems working in the maximally parallel manner; in Mauri et al. (2007) also systems working with *minimal parallelism* were considered, but in this paper we will not address them.

3 The 3-SAT and QUANTIFIED 3-SAT decision problems

In the next section we present a semi-uniform family of recognizer P systems with active membranes that solves any instance of QUANTIFIED 3–SAT, without using polarizations or evolution and communication rules.

Let us start by recalling the **NP**-complete decision problem 3-SAT (Garey and Johnson 1998, p. 46,4). The instances of 3-SAT depend upon two parameters: the number n of variables, and the number m of 3-clauses. We recall that a *clause* is a disjunction of literals, occurrences of x_i or $\neg x_i$, built on a given set $X = \{x_1, x_2, ..., x_n\}$ of Boolean variables. A *3-clause* is a clause that contains exactly three literals. In what follows we will require that no repetitions of the same literal may occur in any clause. Without loss of generality we can also avoid the clauses in which both the literals x_i and $\neg x_i$, for any $1 \leq i \leq n$, occur. An *assignment* of the variables $x_1, x_2, ..., x_n$ is a mapping a: $X \rightarrow \{0,1\}$ that associates to each variable a truth value. The number of all possible assignments to the variables of X is 2^n. We say that an assignment *satisfies* the clause C if, assigned the truth values to all the variables which occur in C, the evaluation of C (considered as a Boolean formula) gives 1 (*true*) as a result.

We can now formally state the 3-SAT problem as follows.

Problem 1 NAME: 3-SAT.

– INSTANCE: a set $C = \{C_1, C_2, ..., C_m\}$ of 3-clauses, built on a finite set $\{x_1, x_2, ..., x_n\}$ of Boolean variables.
– QUESTION: is there an assignment of the variables $x_1, x_2, ..., x_n$ that satisfies all the clauses in C?

In what follows we will sometimes equivalently say that an instance of 3-SAT is a propositional formula $\gamma_{n,m} = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, expressed in the conjunctive normal form as a conjunction of m clauses, where each clause is a disjunction of three literals built using the Boolean variables $x_1, x_2, ..., x_n$. With a little abuse of notation, from now on we will denote by 3-SAT(n,m) the set of instances of 3-SAT which have n variables and m clauses.

The reason for which we are here interested into 3-SAT (rather than with the more generic problem SAT, see Garey and Johnson (1998, p. 39,4), where we put no upper bound on the number of literals that may appear in each clause) is that the number of possible 3-clauses which can be built using n Boolean variables is $2n \cdot (2n-2) \cdot (2n-4) = \Theta(n^3)$, a polynomial quantity with respect to n. This quantity is obtained by looking at a 3-clause as a triple, and observing that each component of the triple may contain one of the $2n$ possible literals, with the constraints that we do not allow the repetition of literals in the clauses, or the use of the same variable two or three times in a clause. On the other hand, an instance of SAT may have a number of clauses which is exponential in n, since for every $i \in \{1, 2, ..., n\}$ either variable x_i or its negation (or none of them) can appear in a clause, yielding to 3^n possible combinations.

Let us now turn our attention to QUANTIFIED 3-SAT, a special case of the QUANTIFIED BOOLEAN FORMULAS decision problem, which can be stated as follows.

Problem 2 NAME: QUANTIFIED BOOLEAN FORMULAS.

– INSTANCE: a well-formed Boolean formula $F = (Q_1x_1)(Q_2x_2) ... (Q_nx_n)E$, where E is a Boolean expression built on a finite set $\{x_1, x_2, ..., x_n\}$ of Boolean variables and each Q_i is either $\forall$ or $\exists$.
– QUESTION: is F true?

Precisely, QUANTIFIED 3-SAT is obtained when the Boolean formula E that appears in this definition is a conjunction of 3-clauses, that is, an instance of 3-SAT. Both QUANTIFIED BOOLEAN FORMULAS and QUANTIFIED 3-SAT are **PSPACE**-complete (Garey and Johnson 1998, pp. 261–262). For conciseness, in what follows we will denote by Q3SAT(n, m) the set of instances of QUANTIFIED 3-SAT in which $E \in$ 3-SAT (n, m).

4 Solving QUANTIFIED 3-SAT with dissolution and controlled strong division

In this section we propose a semi-uniform family $\{\Pi_{Q3SAT}(\gamma_{n,m})\}_{\gamma_{n,m}} \in Q3SAT(n, m)$ of polarizationless recognizer P systems with active membranes that solves the **SPACE**-complete decision problem QUANTIFIED 3-SAT by using only membrane dissolution rules and a form of controlled strong division rules for non-elementary membranes. Precisely, for every instance $\gamma_{n,m}$ of Q3SAT(n, m) we show how to build the system $\Pi_{Q3SAT}(\gamma_{n,m})$ that solves such an instance. In every such system, the symbol s will be chosen to represent the output `yes`.

Let $\gamma_{n,m} = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an instance of 3-SAT(n, m), built using the Boolean variables $x_1, x_2, ..., x_n$, and let $F = (Q_1x_1)(Q_2x_2) ... (Q_nx_n)\gamma_{n,m}$ be the instance of Q3SAT(n, m) obtained by applying in a prearranged way the universal and existential

quantifiers $\forall$ and $\exists$ to x_1, x_2, ..., x_n. The initial configuration of the recognizer P system $\Pi_{Q3SAT(\gamma_{n,\,m})}$ associated to such an instance (in the semi-uniform framework) is illustrated in Fig. 1. In this figure we have adopted the following abbreviation: by labelling a membrane with a sequence of comma-separated labels lab_1, lab_2, ..., lab_k, we denote k membranes nested one into the other—like the layers of an onion—where lab_1 is the inner membrane and lab_k the outer membrane. Hence the system depicted in Fig. 1 is composed by a skin that encloses a number of nested membranes (denoted by Qx_n, ..., Qx_1 in the figure) that depends on the given sequence Q_1, Q_2, ..., Q_n of quantifiers. Precisely, if $Q_i = \exists$ then we will have a single membrane associated with the quantified variable $\exists x_i$ (as illustrated on the right side of Fig. 6), whereas if $Q_i = \forall$ we will have two membranes associated with $\forall x_i$, labelled with x_i and x_i', one contained into the other (left side of Fig. 6). The total number of membranes represented by the sequence Qx_n, ..., Qx_1 in Fig. 1 will thus be comprised between n and $2n$, where the lower bound corresponds to $Q_1 = Q_2 = \cdots = Q_n = \exists$ and the upper bound corresponds to $Q_1 = Q_2 = \cdots = Q_n = \forall$. Moving towards the interior of the system, we have the m nested membranes C_1, ..., C_m which are associated with the clauses of $\gamma_{n,\,m}$. Membrane C_1 contains a membrane labelled with A, that will be used to generate all the possible assignments to x_1, x_2, ..., x_n. Membrane A contains the object x_1 (that represents the namesake variable) as well as n hierarchies of nested membranes. As depicted in Fig. 2, the notation $\boxed{x_i}\,{}^{k}_{d_i}$ that we have adopted in Fig. 1 indicates that symbol x_i is surrounded by k membranes, nested one into the other, all labelled by d_i. In this way, we can operate on membrane A through a rule which is activated by x_1 and, in the meanwhile, dissolve one membrane in each of the subsystems contained in A. After $m + 1$ steps x_2 emerges and activates another rule of A, and so on, until symbol s emerges and starts another phase of computation.

The computation of the system is composed by two phases: the *generation stage* and the *verification stage*. During the generation stage, 2^n copies of the subsystem enclosed by membrane C_m of the initial configuration depicted in Fig. 1 are produced, where in each copy membrane A contains an encoding of one of the possible assignments to x_1, x_2, ..., x_n. Each copy of the subsystem thus produced—that will be used to check whether the assignment it contains satisfies the Boolean expression $\gamma_{n,\,m}$—is enclosed into a hierarchy

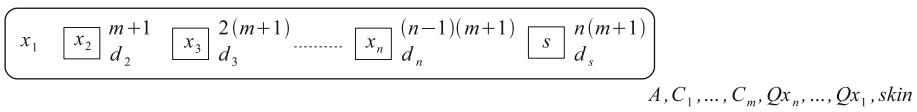
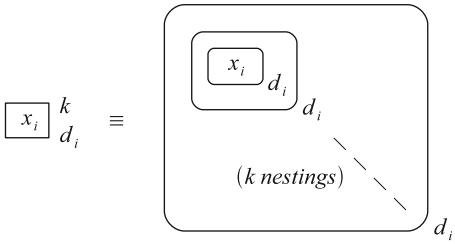


Fig. 1 Initial configuration of the system $\Pi_{Q3SAT(\gamma_{n,\,m})}$ that solves the instance $(Q_1x_1)(Q_2x_2)$ ... $(Q_nx_n)\gamma_{n,\,m}$ of Q3SAT(n, m). Only the objects and the membranes that occur in the region enclosed by membrane A are here detailed

Fig. 2 The hierarchies of nested membranes used in the system depicted in Figure 1 to perform the correct sequence of membrane divisions



constituted by the membranes which are associated to the quantified variables, as depicted in Figs. 4 and 5. The generation phase is performed by the following rules:

1. $[x_i]_A \rightarrow [t_i]_A [f_i]_A$ for all $i = 1, 2, \ldots, n$ $(dup\ A)$
2. $[[\ ]_A [\ ]_A]_{C_1} \rightarrow [[\ ]_A]_{C_1} [[\ ]_A]_{C_1}$ $(div\ C_1)$
3. $[[\ ]_{C_{j-1}} [\ ]_{C_{j-1}}]_{C_j} \rightarrow [[\ ]_{C_{j-1}}]_{C_j} [[\ ]_{C_{j-1}}]_{C_j}$ for all $j = 2, 3, \ldots, m$ $(div\ C_j)$
4. $[x_i]_{d_i} \rightarrow x_i$ for all $i = 1, 2, \ldots, n$
5. $[s]_{d_s} \rightarrow s$
6. $[s]_A \rightarrow s$

Rule 1 is used to generate the assignments: when the symbol x_i, for $i \in \{1, 2, ..., n\}$, occurs in membrane A then A divides; in one of the resulting copies the symbol x_i is rewritten to t_i, indicating the fact that we are assigning the value TRUE to the Boolean variable x_i. Similarly, in the other copy of A the symbol x_i is rewritten to f_i, indicating that the Boolean value FALSE is assigned to x_i. Rules 2 and 3 are strong division rules for non-elementary membranes: whenever a membrane C_j contains two membranes at their immediately inner level, it divides and each of the resulting copies contains one of the previous inner membranes. In order to control the order of application of division rules during the generation phase, only one symbol x_i occurs in membrane A every $m + 1$ computation steps. In this way we first divide membrane A, assigning the two Boolean values TRUE and FALSE to x_i as described above; then, rule 2 can be applied, thus duplicating membrane C_1. In the subsequent $m - 1$ computation steps, membranes C_2, C_3, ..., C_m are duplicated exactly in this order thanks to rules 3. Figure 4 depicts the first steps of this process for a formula $\exists x_1 \forall x_2 \gamma_{2,2}(x_1, x_2)$ containing $n = 2$ variables and $m = 2$ clauses (note that this example is conceived only for illustrative purposes, since at least three Boolean variables are needed to build valid 3-clauses and hence valid instances of QUANTIFIED 3-SAT).

The rules are applied in the maximal parallel manner. In particular, at every computation step one membrane labelled with d_i, for each $i \in \{1, 2, ..., n\}$ such that membrane d_i still occurs in the system, is dissolved. In this way, a symbol x_i emerges in membrane A just after the assignment to x_{i-1} and all the subsequent duplications of membranes C_1, C_2, ..., C_m have been performed. By using the same mechanism, symbol s emerges in membrane A after $n(m + 1)$ steps, that is, after all the assignments to x_1, x_2, ..., x_n and all the duplications of membranes C_1, C_2, ..., C_m have been performed. In practice, the construct composed by $n(m + 1)$ nested membranes, all labelled with d_s, together with the symbol s into the innermost membrane and the dissolution rule $[s]_{d_s} \rightarrow s$. implement a counter whose initial value is $n(m + 1)$ and which is decremented each time the dissolution rule is applied.

The same synchronization mechanism is also used to produce the nested hierarchy composed by the membranes associated with the quantified Boolean variables. Such a hierarchy is built using the following *controlled* strong division rules for non-elementary membranes:

7. $[[\ ]_{C_m} [\ ]_{C_m} [\ ]_{d_s}]_{x_n} \rightarrow [[\ ]_{C_m} [\ ]_{d_s}]_{x_n} [[\ ]_{C_m} [\ ]_{d_s}]_{x_n}$ $(div\ x_n)$
8. $[[\ ]_{x_i} [\ ]_{x_i} [\ ]_{d_s}]_{x_{i-1}} \rightarrow [[\ ]_{x_i} [\ ]_{d_s}]_{x_{i-1}} [[\ ]_{x_i} [\ ]_{d_s}]_{x_{i-1}}$ for all $i = 2, 3, \ldots, n$ $(div\ x_{i-1})$
9. $[[\ ]_{x_i} [\ ]_{x_i} [\ ]_{d_s}]_{x_{i'}} \rightarrow [[\ ]_{x_i} [\ ]_{d_s}]_{x_{i'}} [[\ ]_{x_i} [\ ]_{d_s}]_{x_{i'}}$ for all $i = 2, 3, \ldots, n$ $(div\ x_i')$

As stated in Sect. 2, these controlled rules constitute an even stronger form (with respect to rules 2 and 3) of division rules for non-elementary membranes. In fact, division occurs only if the membrane x_i (or x_i') to be splitted contains a predefined (in general, non-elementary) membrane d_s, besides the two non-elementary membranes to be distributed among the membranes produced by the division process. Also in this case we can use

membranes d_s as counters: we start with a structure $\boxed{s}_{d_s}$ having the desired initial number of nested layers, then at each computation step one layer is removed by means of rule 5, and when the symbol s emerges the division process terminates. By carefully calibrating the initial values of the counters contained into the membranes associated with the quantified variables we are able to build a hierarchy of membranes that compose a full binary tree: two copies of membranes x_2 (or x_2' , if variable x_2 is quantified by $\forall$) are contained into membrane x_1, two copies of x_3 (or x_3', if $\forall x_3$ occurs in the instance of QUANTIFIED 3-SAT we are solving) are contained into membrane x_2, and so on. This tree structure is used to simulate an exponential size Boolean circuit (see Figs. 3 and 5) whose inputs are bijectively associated with all the 2^n possible assignments. If we assign 1 to all the inputs that correspond to an assignment that satisfies $\gamma_{n,\,m}$, 0 to all the other inputs, and we evaluate the circuit, then the output is 1 if and only if the instance of QUANTIFIED 3-SAT we are considering is positive (that is, the corresponding formula is TRUE).

As an example, let us consider the quantified formula $\exists x_1 \forall x_2 (\neg x_1 \vee x_2)$ (note that, as stated above, this is not a valid instance of QUANTIFIED 3-SAT). This formula is true, as we can see by looking at the truth table reported in Fig. 3. On the right side of the same figure, the Boolean circuit that allows to determine the truth value of the formula is illustrated. The first layer of Boolean gates in the circuit checks the quantifier associated with x_n, the next layer performs the check on $x_{n\text{-}1}$ and so on, until we reach the output gate associated to x_1. Thus we have 2^i gates associated to the Boolean variable x_i, and this situation is reflected in the system produced by the generation stage, where we have 2^i subsystems associated to x_i. In Fig. 4 we can see the system Π_{Q3SAT} which is built to solve our example formula $\exists x_1 \forall x_2 (\neg x_1 \vee x_2)$, whereas in Fig. 5 we have the structure of the systems which can be used to solve any formula of the form $\exists x_1 \forall x_2 \exists x_3 \gamma(x_1, x_2, x_3)$, where the Boolean expression γ is composed of $m = 2$ clauses, built using $n = 3$ Boolean variables. The trick which is used to build the tree structure is to stop divisions in each of the membranes associated with the quantified variables at appropriate times. Precisely, the correct sequence of operations needed to build the system depicted in Fig. 5 is:

$$\begin{gathered} dup\ A,\ div\ C_1,\ div\ C_2,\ div\ x_3,\ div\ x_2,\ div\ x_2' \\ dup\ A,\ div\ C_1,\ div\ C_2,\ div\ x_3 \\ dup\ A,\ div\ C_1,\ div\ C_2 \end{gathered}$$

This sequence is obtained by making membrane A divide during the first step of computation, then after 6 steps, and then after further 4 steps. This is accomplished by

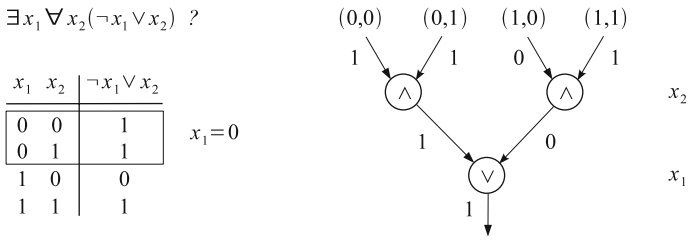


Fig. 3 Example of a quantified Boolean formula formed by one clause, built using two Boolean variables. On the left, its truth table is reported with an indication of the truth assignments that make the formula true. On the right, the tree which is used to check the satisfaction of the quantifiers $\forall$ and $\exists$ is depicted

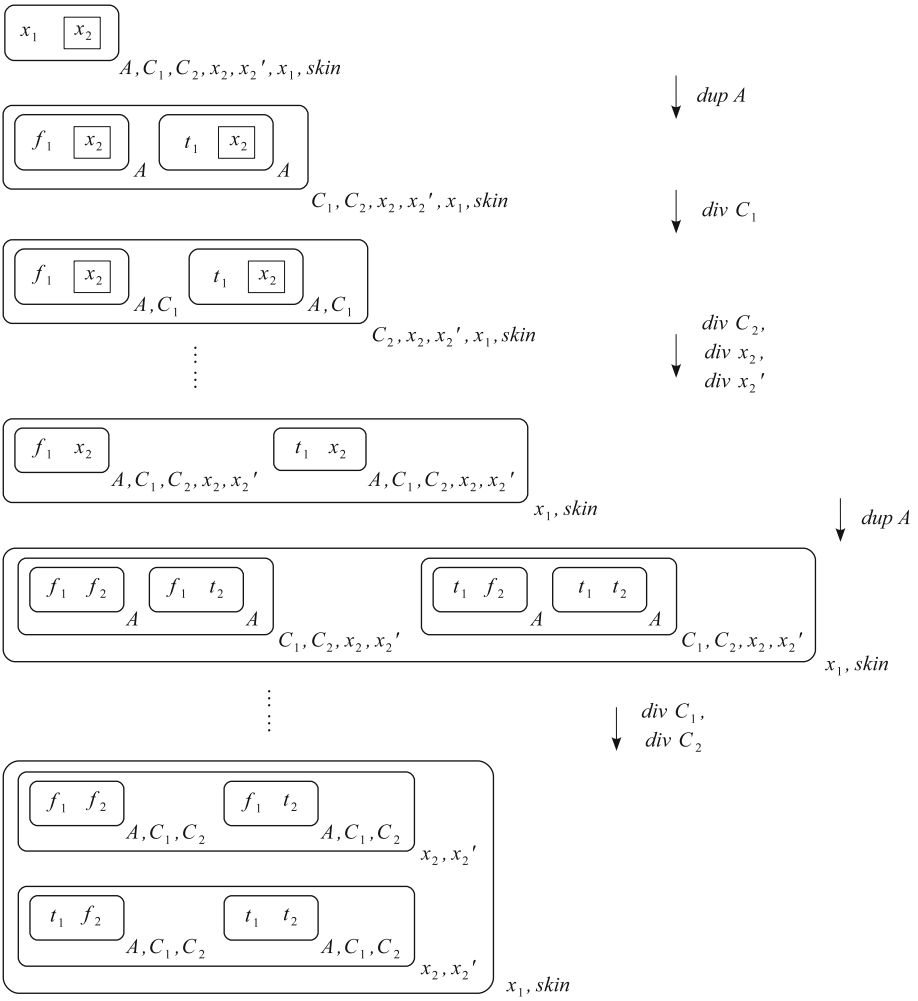


Fig. 4 Steps of the generation stage of a system designed to solve a formula composed of two clauses, built using two Boolean variables. For space reasons, we have used the abbreviations illustrated in Figs. 1 and 2 and we have omitted membranes d_s

initializing the system with seven layers of membranes d_2 and 11 layers of membranes d_3. In order to perform the two required divisions of membrane x_3 we will put in it (in the initial configuration of the system) a counter $\boxed{s}_{d_s}$ with 10 layers, and to make the division of membranes x_2 and x_2' stop after its first execution we will put in these membranes a counter having six layers.

When solving the generic instance $F = (Q_1 x_1)(Q_2 x_2) \cdots (Q_n x_n)\gamma_{n,m}$ of QUANTIFIED 3-SAT, the correct sequence of divisions is the following:

$$\begin{array}{c} dup\ A,\ div\ C_1,\ div\ C_2,\ \ldots,\ div\ C_m,\ div\ X_n,\ \ldots,\ div\ X_3,\ div\ X_2 \\ dup\ A,\ div\ C_1,\ div\ C_2,\ \ldots,\ div\ C_m,\ div\ X_n,\ \ldots,\ div\ X_3 \\ \vdots \\ dup\ A,\ div\ C_1,\ div\ C_2,\ \ldots,\ div\ C_m \end{array}$$

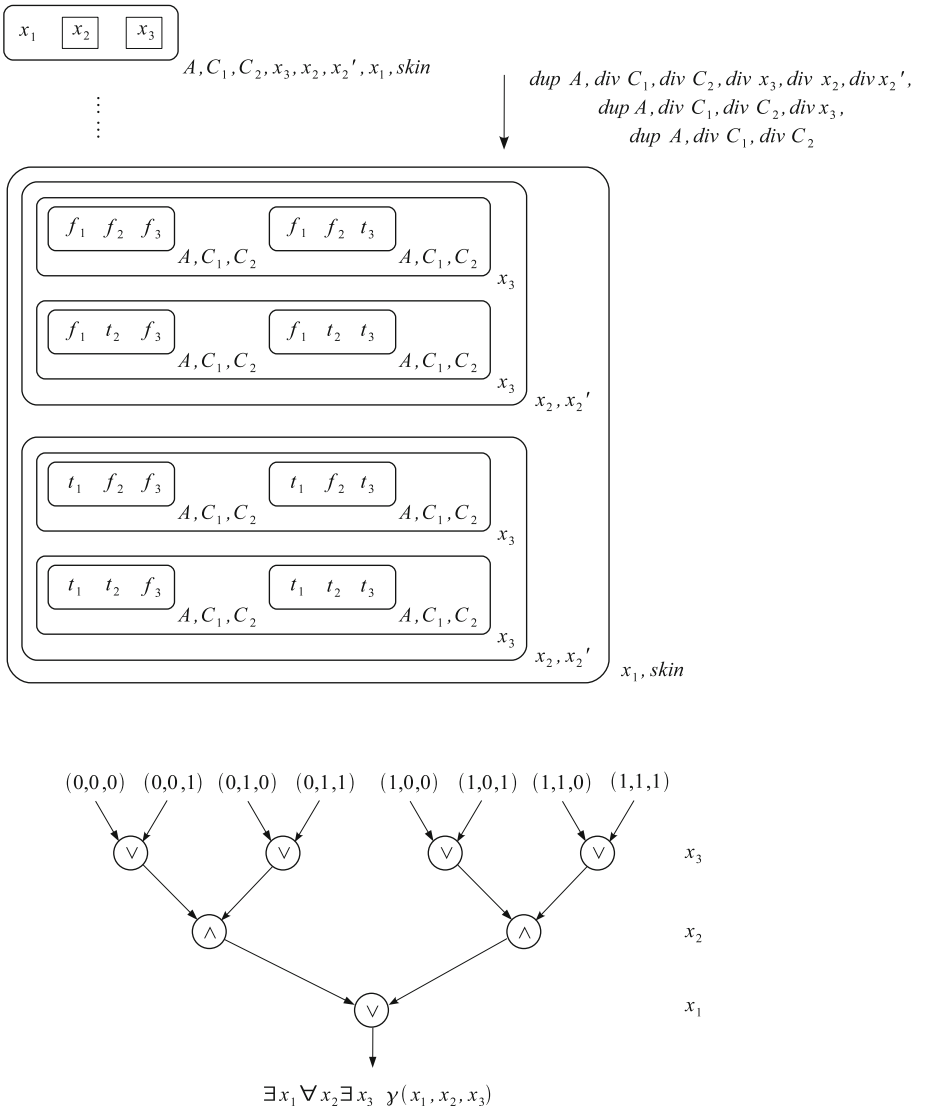


Fig. 5 First and last configuration of the generation stage of a system designed to work on a formula $\exists x_1 \forall x_2 \exists x_3 \gamma(x_1, x_2, x_3)$, where the Boolean expression γ has two clauses, built using three Boolean variables. In the lower part of the figure, the evaluation tree that corresponds to the derived membrane structure is illustrated

where "*div* X_i" denotes either the sequence *div* x_i, *div* x_i' (if $Q_i = \forall$ in F) or the operation *div* x_i (if $Q_i = \exists$). All the counters contained in the membranes associated with the quantified variables, as well as the number of layers of the hierarchies d_2, ..., d_n and the counter contained in membrane A, have to be initialized accordingly. The precise values to be assigned depend of course upon the quantifiers that occur in F.

The next phase of computation is the *verification* stage, which starts when the symbol s appears in A. All the copies of membrane A are dissolved by executing rule 6, so that all the

objects t_i and f_i that represent the truth values of x_1, x_2, ..., x_n can reach the corresponding membrane C_1 and activate its rules. These rules depend upon the Boolean expression $\gamma_{n,\,m}$ contained into the instance F of QUANTIFIED 3-SAT we are solving. For example, assume that the first clause of $\gamma_{n,m}$ is $C_1 = x_1 \vee \neg x_3 \vee x_4$. Then, membranes C_1 will contain the following dissolution rules:

$$[t_1]_{C_1} \rightarrow t_1$$
$$[f_3]_{C_1} \rightarrow f_3$$
$$[t_4]_{C_1} \rightarrow t_4$$

In this way, a membrane labelled with C_1 is dissolved if and only if at least one of the objects t_i and f_i that encode the assignment satisfy the clause. If no object satisfies the clause then the computation in that subsystem halts; on the contrary, if the assignment under consideration satisfies C_1 then by dissolving membrane C_1 the objects t_i and f_i that encode the assignments are released to membrane C_2. Then, the rules that correspond to clause C_2 are executed; if the assignment satisfies also C_2 then the corresponding membrane is dissolved and the computation continues in membrane C_3, otherwise membrane C_2 is not dissolved and the computation halts in that subsystem. If an assignment satisfies all the clauses of $\gamma_{n,m}$ then it will dissolve all the membranes C_1, C_2, ..., C_m, and the objects that represent the assignment will reach the membrane labelled with x_n, where the check on the quantifier associated with the Boolean variable x_n can start. This check is performed as described in Fig. 6. If $Q_i = \exists$, then to satisfy the constraint $\exists x_i$ it suffices that either the object t_i or the object f_i occurs into the membrane. If this is the case, then we dissolve the membrane and all the objects contained in it are released to the surrounding membrane, where the check on the quantifier associated with x_{i-1} can start. On the other hand, if $Q_i = \forall$ then we have two membranes: an inner membrane x_i and an outer membrane x_i'. The inner membrane is dissolved if and only if the object t_i occurs in it, whereas the outer membrane is dissolved if it contains the object f_i. Note that we are assuming that, when dissolving a membrane, also the rules contained in it are dissolved, whereas the objects are released to the surrounding membrane. Notice also that once a quantified variable has been checked, then it is never examined again. The check proceeds from the quantified variable x_n down to x_1, mimicking the evaluation of the full binary tree associated with the quantifiers contained in F. This process halts in those subsystems that do not contain the required objects, and halts in any case after checking the quantified variable x_1.

In conclusion, the instance F of QUANTIFIED 3-SAT solved by the system is positive (that is, TRUE) if and only if in the halting configuration (in which no rule can be applied) at least one copy of symbol s occurs in the region enclosed by the skin membrane.

As stated above, we have focused our attention on QUANTIFIED 3-SAT because in such a problem the number m of clauses is polynomial (at most cubic) in the number n of variables. Hence the size of the initial configuration of the system Π_{Q3SAT} is polynomial,

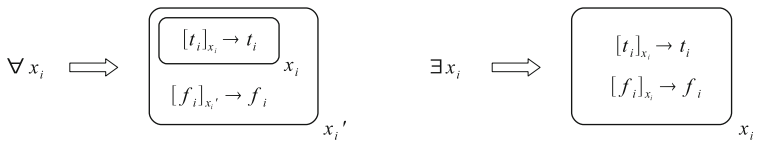


Fig. 6 A schema of the rules and membrane structures used to check the satisfaction of quantified variables. On the left, membranes labelled with x_i and x_i' are dissolved only if they contain both the objects t_i and f_i; on the right, one of these objects suffices to dissolve membrane x_i

and it is apparent that also the number of computation steps performed during both the generation and the verification stage is polynomial in n.

5 Conclusions and directions for further research

We have continued our investigations concerning the computational power of polarizationless recognizer P systems with active membranes, started in Mauri et al. (2007) and Zandron et al. (to appear). In particular, we have shown that the **PSPACE**-complete problem QUANTIFIED 3-SAT can be solved in a semi-uniform way by using recognizer P systems with active membranes, without making use of evolution rules, communication rules and polarizations (that is, electrical charges associated to the membranes), provided that the activation of strong division rules for non-elementary membranes is controlled by the presence of a specified membrane. It is an open problem whether the same result can be obtained by avoiding the control mechanism, like in the solution of 3-SAT given in Zandron et al. (to appear).

It is clear from the construction of the system that a similar controlled behavior may be obtained by using the presence of objects instead of membranes. For example, rule 7 given above could be replaced with the following:

7'. $[[\ ]_{C_m}[\ ]_{C_m} s]_{x_n} \rightarrow [[\ ]_{C_m} s]_{x_n}[[\ ]_{C_m} s]_{x_n}$

where the presence of object s activates the rule. However, we think that the mechanism adopted in this paper (making use of membrane d_s to control the division process) could be less powerful, since at a certain point during the computation the membranes used to control the divisions disappear from the system. In a sense, it is like using a polarized system, forcing it to use its electrical charges only for a limited number of times. Hence, we believe that a deeper comparison between the computational power of polarized systems with the polarizationless recognizer P systems here described is a research direction of a clear interest.

Finally, in this paper we illustrated a *semi-uniform* solution to QUANTIFIED 3-SAT. It is an open question whether a *uniform* solution is possible, by encoding the instances of QUANTIFIED 3-SAT in an appropriate way.

Acknowledgments The authors wish to thank the anonymous referees for their observations, that allowed to improve a previous version of this paper.The work of the fourth author was supported by project TIN2006–13425 of the Ministerio de Educación y Ciencia of Spain, co-financed by FEDER funds, and by the project of Excellence TIC–581 of the Junta de Andalucía. The work of the other authors was partially supported by MIUR under the project "Azioni Integrate Italia–Spagna—Theory and Practice of Membrane Computing".

References

Alhazov A, Freund R (2005) On efficiency of P systems with active membranes and two polarizations. In: Mauri G, Păun Gh, Pérez-Jiménez MJ, Rozenberg G, Salomaa A (eds) Membrane computing, 5th international workshop, WMC 2004. Revised selected and invited papers, LNCS 3365, Springer-Verlag, pp 81–94

Alhazov A, Pérez-Jiménez MJ (2006) Uniform solution to QSAT using polarizationless active membranes. In: Gutiérrez MA, Păun Gh, Riscos-Núñez A, Romero-Campero FJ (eds) Proceedings of the Fourth Brainstorming Week on Membrane Computing, Volume I. Fénix Editora, Sevilla, pp 29–40

Ciobanu G, Pan L, Păun Gh, Pérez-Jiménez MJ (2007) P systems with minimal parallelism. Theoretical Computer Science 378(1):117–130

Garey MR, Johnson DS (1979) Computers and intractability. A guide to the theory on **NP**-completeness. W.H. Freeman and Company

Gutiérrez-Naranjo MA, Pérez-Jiménez MJ, Riscos-Núñez A (2005) A fast P system for finding a balanced 2-partition. Soft Computing 9(9):673–678

Gutiérrez-Naranjo MA, Pérez-Jiménez MJ, Riscos-Núñez A, Romero-Campero FJ (2006a) 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, revised selected and invited papers, LNCS 3850, Springer-Verlag, pp 224–240.

Gutiérrez-Naranjo MA, Pérez-Jiménez MJ, Riscos-Núñez A, Romero-Campero FJ, Romero-Jiménez A (2006b) Characterizing tractability by cell-like membrane systems. In: Subramanian KG, Rangarajan K, Mukund M (eds) Formal models, languages and applications. World scientific, Series in machine perception and artificial intelligence, 66:137–154

Krishna SN, Rama R (1999) A variant of P systems with active membranes: Solving **NP**-complete problems. Romanian Journal of Information Science and Technology 2(4):357–367

Mauri G, Pérez-Jiménez MJ, Zandron C (2007) On a Păun's conjecture in membrane systems. In: Mira J, Álvarez JR (eds) Second international work-conference on the interplay between natural and artificial computation, IWINAC 2007, LNCS 4527, Springer-Verlag, pp 180-192

Obtulowicz A (2001) Deterministic P systems for solving SAT problem. Romanian Journal of Information Science and Technology 4(1–2):551–558

Păun Gh (2000a) Computing with membranes. J Comput Syst Sci 1(61):108–143. See also Turku centre for computer science – TUCS report no. 208, 1998. Available at http://www.tucs.fi/ Publications/techreports/TR208.php

Păun Gh (2000b) Computing with membranes: attacking **NP**-complete problems. In: Antoniou I, Calude C, Dinneen MJ (eds) Unconventional models of computation. Springer-Verlag, pp 94–115

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

Păun Gh (2002) Membrane Computing. An Introduction. Springer–Verlag, NewYork

Păun Gh (2005) Further twenty six open problems in membrane computing. In: Gutiérrez-Naranjo MA, Riscos-Núñez A, Romero-Campero FJ, Sburlan D (eds) Proceedings of the Third Brainstorming Week on Membrane Computing. Fénix Editora, Sevilla, pp 249–262

Pérez-Jiménez MJ, Riscos-Núñez A (2004) A linear-time solution to the KNAPSACK problem using P systems with active membranes. In: Martín-Vide C, Păun Gh, Rozenberg G, Salomaa A (eds) Membrane Computing, International Workshop, WMC 2003. Revised Papers, LNCS 2933, Springer–Verlag, pp 250–268

Pérez-Jiménez MJ, Riscos-Núñez (2005) Solving the SUBSET SUM problem by active membranes. New Generation Computing 23(4):367–384

Pérez-Jiménez MJ, Romero-Campero FJ (2005) Attacking the COMMON ALGORITHMIC PROBLEM by recognizer P systems. In: Margenstern M (eds) Machines, Computations and Universality, LNCS 3354, Springer–Verlag, pp 304–315

Pérez-Jiménez MJ, Romero-Jiménez A, Sancho-Caparrini F (2003) A polynomial complexity class in P systems using membrane division. In: Csuhaj-Varjú E, Kintala C, Wotschke D, Vaszil G (eds) Proceedings of the 5th workshop on descriptional complexity of formal systems, DCFS 2003, computer and automation research institute of the Hungarian Academy of Sciences, Budapest, pp 284–294

Pérez-Jiménez MJ, Romero-Jiménez A, Sancho-Caparrini F (2004) The **P** versus **NP** problem through cellular computing with membranes. In: Jonoska N, Paun Gh, Rozenberg Gr (eds) Aspects of Molecular Computing, LNCS 2950, Springer-Verlag, pp 338–352

Sosik P (2003) The computational power of cell division. Natural Computing 2(3):287–298

The P systems (2000) Web page: http://ppage.psystems.eu/

Zandron C, Ferretti C, Mauri G (2000) Solving **NP**-complete problems using P systems with active membranes. In: Antoniou I, Calude C, Dinneen MJ (eds) Unconventional Models of Computation. Springer–Verlag, pp 289–301

Zandron C, Leporati A, Mauri G, Ferretti C, Pérez-Jiménez J On the computational efficiency of polarizationless recognizer P systems with strong division and dissolution. *Fundamenta Informaticae* (to appear)