Reaching efficiency through collaboration in membrane systems: Dissolution, polarization and cooperation

Luis Valencia-Cabrera, David Orellana-Martín, Miguel A. Martínez-del-Amor, Agustín Riscos-Núñez, Mario J. Pérez-Jiménez

Research Group of Natural Computing, Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, Sevilla, 41012, Spain

Abstract

From a computational complexity point of view, some syntactical ingredients play different roles depending on the kind of combination considered. Inspired by the fact that the passing of a chemical substance through a biological membrane is often done by an interaction with the membrane itself, systems with active membranes were considered. Several combinations of different ingredients have been used in order to know which kind of problems could they solve efficiently. In this paper, minimal cooperation with a minimal expression (the left-hand side of every object evolution rule has at most two objects and its right-hand side contains only one object) in object evolution rules is considered and a polynomial-time uniform solution to the SAT problem is presented. Consequently, a new way to tackle the P versus NP problem is provided.

1. Introduction

The first models in Membrane Computing were designed in such a manner that the number of membranes could not increase during a computation. They could only decrease by dissolving membranes as a result of applying some rules to the objects present in the system. However, in these systems an exponential workspace (expressed in terms of the number of objects) can be constructed in linear time, e.g. via evolution rules of the type $a \rightarrow (a^2, \text{here})$. Nevertheless, such a capability is not enough to efficiently solve NP-complete problems, unless $P = NP$ (see [6] for details).

It is well known that in ideal circumstances, a cell produces two identical copies by the processes of interphase and mitosis. First, the cell grows and makes a copy of its DNA (replication) and, finally, the cell separates its DNA into two sets and divides its cytoplasm, forming two new cells. Inspired from this mechanism, a new kind of rules was introduced in Membrane Computing allowing the proliferation of membranes by means of division rules. A membrane without any other membrane inside it (elementary membrane) can be divided by means of an interaction with an object from that membrane. In [12], cell-like P systems with active membranes are defined incorporating this ingredient. In such systems, each membrane has an electrical charge (positive, negative or neutral) associated with it, but the rules are non-cooperative and there are no priorities among rules. Besides, a non-elementary membrane (that is, a membrane with one or more membranes within it) with at least two inner membranes can also be divided. The skin cannot be divided.
Polarizationless P systems with active membranes and minimal cooperation in object evolution rules

We assume the reader is familiar with basic notions and terminology of membrane computing [13]. However, before going on let us briefly overview some notations that will be used throughout the paper. An alphabet \( \Gamma \) is a non-empty set whose elements are called symbols. A multiset over an alphabet \( \Gamma \) is an ordered pair \((\Gamma, f)\), where \( f \) is a mapping from \( \Gamma \) onto the set of natural numbers \( \mathbb{N} \). The support of a multiset \( m = (\Gamma, f) \) is defined as \( \text{supp}(m) = \{ x \in \Gamma | f(x) > 0 \} \), and its

\[ \text{supp}(m) = \{ x \in \Gamma | f(x) > 0 \}, \]

1 For more details about the uniformity of a solution, see [4].
size is $|m| = \Sigma_{x \in \text{supp}(m)} f(x)$. We denote by $M(\Gamma)$ the set of all multisets over $\Gamma$. If $m_1 = (\Gamma, f_1)$, $m_2 = (\Gamma, f_2)$ are multisets over $\Gamma$, then the union of $m_1$ and $m_2$, denoted by $m_1 + m_2$, is the multiset $(\Gamma, g)$, where $g(x) = f_1(x) + f_2(x)$ for each $x \in \Gamma$.

Next, minimal cooperation in object evolution rules is introduced in the framework of polarizationless P systems with active membranes. The term “minimal cooperation” is used in the following sense: the left-hand side of such rules consists of two symbols.

**Definition 2.1.** In the context of polarizationless P system with active membranes, several types of minimal cooperation in object evolution rules are defined as follows.

- **Minimal cooperation (mc):** object evolution rules are of the form $[u \rightarrow v]_h$, where $u, v \in M(\Gamma)$ such that $|u| \leq 2$, but at least one object evolution rule verifies $|u| = 2$.
- **Primary minimal cooperation (pmc):** object evolution rules are of the form $[u \rightarrow v]_h$, where $u, v \in M(\Gamma)$ and $1 \leq |u|, |v| \leq 2$, but at least one object evolution rule verifies $|u| = 2$.
- **Bounded minimal cooperation (bmc):** object evolution rules are of the form $[u \rightarrow v]_h$, where $u, v \in M(\Gamma)$ and $1 \leq |v| \leq |u| \leq 2$, but at least one object evolution rule verifies $|u| = 2$.
- **Minimal cooperation and minimal production (mcmp):** object evolution rules are of the forms $[a \rightarrow b]_h$ or $[a b \rightarrow c]_h$, where $a, b, c \in \Gamma$, but at least one object evolution rule is of the second type.

In these systems, send-in communication rules, send-out communication rules, dissolution rules and division rules are non-cooperative rules.

In polarizationless P systems with active membranes and minimal cooperation in object evolution rules, the rules are applied according to the same principles than in the “classical” P systems with active membranes (see [12], for details).

We denote by $DAAM^0(\alpha, \beta, \gamma, \delta)$ the class of all recognizer polarizationless P systems with active membranes and division rules, where $\alpha, \beta, \gamma$ and $\delta$ are parameters associated with object evolution rules, communication rules, dissolution rules and division rules, respectively. The meaning of the parameters $\alpha, \beta, \gamma, \delta$ is the following:

- If $\alpha = -e$ (resp. $\alpha = +e$), object evolution rules are forbidden (resp. permitted).
- If $\alpha = mc$ (resp. $\alpha = pmc$, $\alpha = bmc$ or $\alpha = mcmp$), then minimal cooperation (primary minimal cooperation, bounded minimal cooperation or minimal cooperation and minimal production, respectively) in object evolution rules are permitted.
- If $\beta = -c$ (resp. $\beta = +c$) then communication rules are forbidden (resp. permitted).
- If $\gamma = -d$ (resp. $\gamma = +d$) then dissolution rules are forbidden (resp. permitted).
- If $\delta = -n$ (resp. $\delta = +n$) then division rules for only elementary membranes are permitted (resp. division rules for elementary and non-elementary membranes are permitted).

If separation rules are considered instead of division rules, the corresponding classes of recognizer membrane systems are denoted by $SAM^0(\alpha, \beta, \gamma, \delta)$.

Let us recall some interesting results expressed in these notations.

1. Families of systems from $DAAM^0(+e,+c,+d,+n)$ can solve PSPACE-complete problems in polynomial time and in a uniform way, that is, $PSPACE \subseteq PMC_{DAAM^0(+e,+c,+d,+n)}$ (see [4] for details). Moreover, in [17] and [16] the reverse inclusion was proved, so a stronger result is obtained in this framework: $PSPACE = PMC_{DAAM^0(+e,+c,+d,+n)}$.

2. Families of systems from $DAAM^0(+e,+c,-d,+n)$ can efficiently solve only problems in class P, that is, $PMC_{DAAM^0(+e,+c,-d,+n)} = P$ (see [7] for details).

3. Families of systems from $SAM^0(+e,+c,-d,+n)$ can efficiently solve only problems in class P, that is, $P_{SAM^0(+e,+c,-d,+n)} = P$ (see [19] for details).

4. Families of systems from $DAAM^0(bmc,+c,-d,-n)$ can solve NP-complete problems in polynomial time and in a uniform way, that is, $NP \cup co-NP \subseteq PMC_{DAAM^0(bmc,+c,-d,-n)}$ (see [18] for details).

5. Families of systems from $SAM^0(bmc,+c,-d,-n)$ can efficiently solve only problems in class P, that is, $PMC_{SAM^0(bmc,+c,-d,-n)} = P$ (see [20] for details).

6. Families of systems from $SAM^0(pmc,+c,-d,-n)$ can solve NP-complete problems in polynomial time and in a uniform way, that is, $NP \cup co-NP \subseteq PMC_{SAM^0(pmc,+c,-d,-n)}$ (see [18] for details).

Păun’s conjecture can be expressed as follows: $PMC_{DAAM^0(+e,+c,+d,+n)} = P$. It is a relevant open question.

From results 4 and 5, a new frontier of efficiency is deduced. In the framework of polarizationless P systems with active membranes not using dissolution rules, a new frontier of the efficiency is obtained. Specifically, when bounded minimal cooperation in object evolution rules is allowed in the previous computing framework, passing from separation rules to division rules amounts to passing from non-efficiency to efficiency.
Next, we try to find narrower frontiers of efficiency by showing that bounded minimal cooperation can be replaced by minimal cooperation and minimal production in object evolution rules. Let us recall that the application of this kind of rules implies that only a single object can be produced by the application of a rule.

3. On the efficiency of systems from $\mathcal{DAM}^0(mcmp, +c, −d, −n)$

In this section, we analyze what happens, from a computational complexity point of view, when minimal cooperation and minimal production in object evolution rules are considered instead of dissolution rules. The first work in this direction was addressed in [18]. The efficiency of such systems was proven when only object evolution rules such that the of length their left-hand sides are greater than or equal to the length of the corresponding right-hand side, and both lengths are at most two. Specifically, we show that the syntactical ingredient of minimal cooperation and minimal production in polarizationless P systems with active membranes (without dissolution and allowing only division for elementary membranes) is enough to solve computationally hard problems in an efficient way.

3.1. A uniform polynomial-time solution to the SAT problem

Next, a polynomial-time uniform solution to the SAT problem, a well known NP-complete problem [5] is provided by a family $\Pi = \{\Pi(t) | t \in \mathbb{N}\}$ of recognizer P systems from $\mathcal{DAM}^0(mcmp, +c, −d, −n)$.

Let us recall that the polynomial-time computable function (the Cantor pair function) $(n, p) = ((n + p)(n + p + 1)/2 + n$ is a primitive recursive and bijective function from $\mathbb{N} \times \mathbb{N}$ to $\mathbb{N}$. The family $\Pi = \{\Pi(t) | t \in \mathbb{N}\}$ is defined in such a manner that system $\Pi(t)$ will process any Boolean formula $\phi$ in conjunctive normal form (CNF) with $n$ variables and $p$ clauses, where $t = (n, p)$, provided that the appropriate input multiset $\text{cod}(\phi)$ is supplied to the system (through the corresponding input membrane), and will answer if there exists at least one truth assignment that makes true the input formula $\phi$, that is, $\Pi(t)$ will solve an instance of the SAT problem.

For each $n, p \in \mathbb{N}$, we consider the recognizer P system $\Pi((n, p)) = (\Gamma, \Sigma, H, \mu, M_1, M_2, R, i_{in}, i_{out})$ from $\mathcal{DAM}^0(mcmp, +c, −d, −n)$, defined as follows:

1. Working alphabet $\Gamma$:
   - $\{\text{yes}, \text{no}, \alpha, \beta, \beta', \beta'', \gamma, \gamma', \gamma'', \#\} \cup \{a_{i,k} | 1 \leq i \leq n, 1 \leq k \leq i\} \cup \{t_{i,k}, f_{i,k} | 1 \leq i \leq n, i \leq k \leq n + p - 1\} \cup \{\beta_k | 0 \leq k \leq n + 2p + 1\} \cup \{c_j | 1 \leq j \leq p\} \cup \{d_j | 2 \leq j \leq p\} \cup \{T_{i,k}, F_{i,k} | 1 \leq i \leq n, 0 \leq k \leq n - 1\} \cup \{T_i, F_i | 1 \leq i \leq n\} \cup \{x_{i,j,k}, \overline{x}_{i,j,k}, x_{i,j,0}^+ | 0 \leq i \leq n, 1 \leq j \leq p, 1 \leq k \leq n + p\}$

2. Input alphabet $\Sigma$: $\{x_{i,j,0}, \overline{x}_{i,j,0}, x_{i,j,0}^+ | 1 \leq i \leq n, 1 \leq j \leq p\}$

3. $H = \{1, 2\}$.
4. Membrane structure: $\mu = [1 \ 2]$, that is, $\mu = (V, E)$ where $V = \{1, 2\}$ and $E = \{(1, 2)\}$.
5. Initial multisets: $M_1 = \{\alpha, \beta_0\}$, $M_2 = \{a_{i,1}, T_{i,0}, F_{i,0}^\prime | 1 \leq i \leq n\}$.
6. The set of rules $R$ consists of the following rules:

   1.1. Rules for a general counter.
   - $\beta_k \rightarrow \beta_{k+1} [1]$, for $0 \leq k \leq n + 2p$
   - $\beta_{p+2p+1} \rightarrow \beta^+ [1]$

   1.2. Rules for an affirmative answer.
   - $\alpha \gamma \rightarrow \gamma' [1]$
   - $\gamma'' [1]$
   - $\gamma'' [1] \rightarrow \text{yes} [1]$

   1.3. Rules for a negative answer.
   - $\alpha \beta' \rightarrow \beta'' [1]$
   - $\beta'' [1] \rightarrow \text{no} [1]$

2.1. Rules to generate all truth assignments.
   - $a_{i,1} \rightarrow t_{i,1} [1]$
   - $a_{i,k} \rightarrow a_{i,k+1} [1]$, for $2 \leq i \leq n \land 1 \leq k \leq i - 1$

2.2. Rules to produce exactly $p$ copies of each truth assignment.
   - $t_{i,k} \rightarrow t_{i,k+1} [1]$
   - $f_{i,k} \rightarrow f_{i,k+1} [1]$
   - $T_{i,k} \rightarrow T_{i,k+1} [1]$
   - $F_{i,k} \rightarrow F_{i,k+1} [1]$
   - $T_{i,n-1} \rightarrow T_i [1]$
   - $F_{i,n-1} \rightarrow F_i [1]$
   - $1 \leq i \leq n$
Notice Let simplified conjunctive

4. A formal verification of the solution

We consider the polynomial encoding (cod, s) from SAT in \( \Pi \) defined as follows: let \( \varphi \) be a Boolean formula in conjunctive normal form (a conjunction of clauses such that each clause is the disjunction of one or more literals) and simplified (in each clause, literals are not repeated, and also none of the clauses contains both a literal and its negation). Let \( \text{Var}(\varphi) = \{x_1, \ldots, x_n\} \) be the set of propositional variables and \( \{C_1, \ldots, C_p\} \) the set of clauses of \( \varphi \). Let us assume that the number of variables and the number of clauses of the input formula \( \varphi \), are greater than or equal to 2. Then, we define \( s(\varphi) = (n, p) \) and

\[
\text{cod}(\varphi) = \{x_{i,j,0} \mid x_i \in C_j\} \cup \{\overline{x}_{i,j,0} \mid \neg x_i \in C_j\} \cup \{x^*_{i,j,0} \mid x_i \notin C_j, \neg x_i \notin C_j\}
\]

Notice that we can represent this multiset as a matrix, in such a way that the \( j \)-th row (\( 1 \leq j \leq p \)) encodes the \( j \)-th clause \( C_j \) of \( \varphi \). For instance, the formula \( \varphi = (x_1 + x_2 + \neg x_3)(\neg x_2 + x_4)(\neg x_2 + x_3 + \neg x_4) \) is encoded as follows:

\[
\text{cod}(\varphi) = \begin{pmatrix}
  x_{1,1,0} & x_{2,1,0} & \overline{x}_{3,1,0} & x^*_{4,1,0} \\
  x^*_{1,2,0} & \overline{x}_{2,2,0} & x^*_3_{2,2,0} & x_{4,2,0} \\
  x^*_{3,3,0} & \overline{x}_{2,3,0} & x_{3,3,0} & \overline{x}_{4,3,0}
\end{pmatrix}
\]

We denote by \( \text{cod}_k(\varphi) \) the multiset \( \text{cod}(\varphi) \) when the third index of all objects is equal to \( k \). For instance:

\[
\text{cod}_3(\varphi) = \begin{pmatrix}
  x_{1,1,3} & x_{2,1,3} & \overline{x}_{3,1,3} & x^*_{4,1,3} \\
  x^*_{1,2,3} & \overline{x}_{2,2,3} & x^*_3_{2,2,3} & x_{4,2,3} \\
  x^*_{3,3,3} & \overline{x}_{2,3,3} & x_{3,3,3} & \overline{x}_{4,3,3}
\end{pmatrix}
\]

The Boolean formula \( \varphi \) will be processed by the system \( \Pi(s(\varphi)) \) with input multiset \( \text{cod}(\varphi) \). Next, we informally describe how that system works.

The solution proposed follows a brute force algorithm in the framework of recognizer P systems with active membranes, minimal cooperation and minimal production in object evolution rules. It consists of the following stages:

- **Generation stage**: using division rules, all truth assignments for the variables \( \{x_1, \ldots, x_n\} \) associated with \( \varphi \) are produced. Specifically, \( 2^n \) membranes labelled by 2 are generated, each of them encoding a truth assignment. This stage takes exactly \( n \) computation steps, \( n \) being the number of variables in \( \varphi \).
- **Production of enough copies for each truth assignment**: in this stage \( p \) copies of each truth assignment are produced to allow the checking of the literal associated with each variable in each clause. This stage takes exactly \( p \) computation steps.
- **First Checking stage**: checking whether or not each clause of the input formula \( \varphi \) is satisfied by the truth assignments generated in the previous stage, encoded by each membrane labelled by 2. This stage takes exactly one computation step.
• **Second Checking stage**: checking whether or not all clauses of the input formula $\varphi$ are satisfied by some truth assignment encoded by a membrane labelled by 2. This stage takes exactly $p - 1$ steps, $p$ being the number of clauses of $\varphi$.
• **Output stage**: the system sends to the environment the right answer according to the results of the previous stage. This stage takes exactly 4 steps.

4.1. Generation stage

At this stage, all the truth assignments for the variables associated with the Boolean formula $\varphi$ are generated, by applying division rules from 2.1 in membranes labelled by 2. In such manner at the $i$-th step, $1 \leq i \leq n$, of this stage, division rule is triggered by object $a_{i,i}$, producing two new membranes with all its remaining contents replicated in the new membranes labelled by 2. This stage ends when objects $t_{i,n}$, $f_{i,n}$, $1 \leq i \leq n$, have been generated.

**Proposition 1.** Let $C = (C_0, C_1, \ldots, C_q)$ be a computation of the system $\Pi(s(\varphi))$ with input multiset $\text{cod}(\varphi)$.

(a) For each $k$, $1 \leq k \leq n - 1$, at configuration $C_0$ we have $C_0(1) = (\alpha, \beta_k)$ (being $C_i(t)$ the contents of membrane $i$ at the moment $t$) and there are $2^k$ membranes labelled by 2 such that each of them contains: the set $\{a_{i+k+1} | k + 1 \leq i \leq n\}$ and the set $\text{cod}_i(\varphi)$; the multiset $\{T^p_{i,k}, F^p_{i,k} | 1 \leq i \leq n\}$; and a different subset $\{r_1, \ldots, r_k\}$, being $r \in \{t, f\}$.
(b) At configuration $C_n$ we have $C_n(1) = (\alpha, \beta_n)$ and there are $2^n$ membranes labelled by 2 such that each of them contains: the set $\text{cod}_n(\varphi)$; the multiset $\{T^p_{i,k}, F^p_{i,k} | 1 \leq i \leq n\}$; and a different subset $\{r_1, \ldots, r_n\}$, being $r \in \{t, f\}$.

**Proof.**

(a) By induction on $k$. The base case $k = 1$ follows bearing in mind that configuration $C_1$ is obtained from configuration $C_0$ by applying the rules $[\beta_0 \rightarrow \beta_1], \{a_{1,1} \rightarrow \{t_{1,1}\}, \{f_{1,1}\}, \{t_{1,0} \rightarrow t_{i,1}\}, \{f_{1,0} \rightarrow f_{i,1}\}, \{1 \leq i \leq n, [x_{i,j,0} \rightarrow x_{i,j,1}\}, [x_{i,j,0} \rightarrow x_{i,j,1}]_{1 \leq i \leq n, 1 \leq j \leq p}.$
Let us assume that the result holds for $k$, $1 \leq k < n - 1$. Let us see that the result also holds for $k + 1$.

On the one hand, at configuration $C_k$, we have $C_k(1) = (\alpha, \beta_k)$ and there are $2^k$ membranes labelled by 2 of each containing the set $\{a_{i+k+1} | k + 1 \leq i \leq n\}$ and the set $\text{cod}_i(\varphi)$; the multiset $\{T^p_{i,k}, F^p_{i,k} | 1 \leq i \leq n\}$; and a different subset $\{r_1, \ldots, r_k\}$, being $r \in \{t, f\}$.

On the other hand, configuration $C_{k+1}$ is obtained from configuration $C_k$ by applying the rules: $[\beta_k \rightarrow \beta_{k+1}], \{a_{k+1,1} \rightarrow \{t_{k+1,1}\}, \{f_{k+1,1}\}, \{t_{k+1,0} \rightarrow t_{i,k+1}\}, \{f_{k+1,0} \rightarrow f_{i,k+1}\}, \{1 \leq i \leq k, [a_{i+k+1} \rightarrow a_{i,k+2}\}, \{k + 2 \leq i \leq n, [T_{i,k} \rightarrow T_{i,k+1}, \{F_{i,k} \rightarrow F_{i,k+1}\}, \{1 \leq i \leq n, [x_{i,j,k} \rightarrow x_{i,j,k+1}\}, [x_{i,j,k} \rightarrow x_{i,j,k+1}]_{1 \leq i \leq n, 1 \leq j \leq p}.

Hence, the result holds for $k + 1$.

(b) By applying (a) to $k = n - 1$ at configuration $C_{n-1}$ we have $C_{n-1}(1) = (\alpha, \beta_{n-1})$ and there are $2^{n-1}$ membranes labelled by 2 of each containing: the object $a_{n,n}$ and the set $\text{cod}_{n-1}(\varphi)$; the multiset $\{T^p_{i,k}, F^p_{i,k} | 1 \leq i \leq n\}$; and a different subset $\{r_1, \ldots, r_{n-1}\}$, being $r \in \{t, f\}$.

Then, (b) follows noting that configuration $C_{n-1}$ is obtained from configuration $C_{n-1}$ by applying the rules: $[\beta_{n-1} \rightarrow \beta_n], \{a_{n,n} \rightarrow \{t_{n,n}\}, \{f_{n,n}\}, \{t_{n-1} \rightarrow t_{i,n}\}, \{f_{n-1} \rightarrow f_{i,n}\}, \{1 \leq i \leq n - 1, [x_{i,j,n-1} \rightarrow x_{i,j,n}\}, \{x_{i,n,n} \rightarrow x_{i,n,n}^*\}_{1 \leq i \leq n, 1 \leq j \leq p}$.

4.2. Producing enough copies for each truth assignment

At this stage, in each membrane labelled by 2, a sufficient number of copies from each truth assignment will be generated. Specifically, $p$ copies of each of them will be produced, where $p$ is the number of clauses of the input formula. Let us recall that in the initial configuration there are $p$ copies of $T_1, F_1, \ldots, T_n, F_n$. These copies are replicated in the $2^n$ membranes labelled by 2 produced by applying division rules where a copy of each truth assignment is produced. By using cooperation we use the values $t_i$ and $f_i$ of the truth assignment associated with each membrane labelled by 2 to remove a copy of the opposite value $F_i$ or $T_i$, respectively. This stage takes exactly $p$ steps.

**Proposition 2.** Let $C = (C_0, C_1, \ldots, C_q)$ be a computation of the system $\Pi(s(\varphi))$ with input multiset $\text{cod}(\varphi)$.

(a) For each $k$, $1 \leq k \leq p - 1$, at configuration $C_{n+k}$ we have $C_{n+k}(1) = (\alpha, \beta_{n+k})$ and there are $2^n$ membranes labelled by 2 such that each of them contains: the set $\text{cod}_{n+k}(\varphi)$; a different subset $\{r_{i,n+k+1}, \ldots, r_{i,n+k}\}$, being $r \in \{t, f\}$; and the corresponding multiset $\{R^p_{n+k} \} \ldots, R^p_{n+k}\}$ verifying the following: for each $k$, $1 \leq i \leq n$, if $r_{i,n+k} = t_{i,n+k}$ then $R_i = T_i$ and $\overline{R}_i = F_i$; if $r_{i,n+k} = f_{i,n+k}$ then $R_i = F_i$ and $\overline{R}_i = T_i$.

(b) At configuration $C_{n+p}$ we have $C_{n+p}(1) = (\alpha, \beta_{n+p})$ and there are $2^n$ membranes labelled by 2 such that each of them contains: $n$ copies of object $\#$; the set $\text{cod}_{n+p}(\varphi)$; and a different subset $\{R^p_1, \ldots, R^p_{n}\}$ being $R \in \{T, F\}$. 
Proof.

(a) By induction on \( k \). The base case \( k = 1 \) follows bearing in mind that configuration \( C_{n+1} \) is obtained from configuration \( C_n \) by applying the rules: \( [\beta_n \rightarrow \beta_{n+1}]_1 \), \([t_1, t_{i,n}] \rightarrow t_{i,n+1} \), \([f_i, t_{j,n}] \rightarrow f_{i,n+1} \), for \( 1 \leq i \leq n \). \([x_i, j,n] \rightarrow x_i, j,n+1 \), \([x_i, j,n] \rightarrow \tilde{x}_i, j,n+1 \), \([x_i, j,n] \rightarrow x_i, j,n+1 \), \( 1 \leq i \leq n, 1 \leq j \leq p \).

Let us assume that the result holds for \( k \), \( 1 \leq k < p - 1 \). Let us see that the result also holds for \( k + 1 \).

On the one hand, at configuration \( C_{n+k} \) we have \( C_{n+k}(1) = \{ \alpha, \beta_{n+k} \} \) and there are \( 2^n \) membranes labelled by 2 each of them containing the set \( \text{cod}_{n+k}(\phi) \); a different subset \( \{ t_1, t_{n+k} \}, \text{being } r \in \{ t, f \}; \) and the corresponding multiset \( \{ R_p, \tilde{R}_p, R_p, \tilde{R}_p \} \) verifying the following: for each \( 1 \leq i \leq n \), if \( t_{i,n+k} = t_{i,n+k} \) then \( R_i = T_i \) and \( \tilde{R}_i = F_i \); if \( t_{i,n+k} = f_{i,n+k} \) then \( R_i = F_i \) and \( \tilde{R}_i = T_i \).

On the other hand, configuration \( C_{n+k+1} \) is obtained from configuration \( C_{n+k} \) by applying the rules: \( [\beta_{n+k} \rightarrow \beta_{n+k+1}]_1 \), \([t_1, t_{n+k}] \rightarrow t_{i,n+k+1} \), \([f_i, t_{j,n+k}] \rightarrow f_{i,n+k+1} \), for \( 1 \leq i \leq n \). \([x_i, j,n+k] \rightarrow x_i, j,n+k+1 \), \([x_i, j,n+k] \rightarrow \tilde{x}_i, j,n+k+1 \), \([x_i, j,n+k] \rightarrow \tilde{x}_i, j,n+k+1 \), \( 1 \leq i \leq n, 1 \leq j \leq p \).

Hence, the result holds for \( k + 1 \).

(b) By applying (a) to \( k = p - 1 \), at configuration \( C_{n+p-1} \) we have \( C_{n+p-1}(1) = \{ \alpha, \beta_{n+p-1} \} \) and there are \( 2^n \) membranes labelled by 2 each of them containing the set \( \text{cod}_{n+p-1}(\phi) \); a different subset \( \{ t_1, t_{n+p-1} \}, \text{being } r \in \{ t, f \}; \) and the corresponding multiset \( \{ R_p, \tilde{R}_p, R_p, \tilde{R}_p \} \) verifying the following: for each \( 1 \leq i \leq n \), if \( t_{i,n+p-1} = t_{i,n+p-1} \) then \( R_i = T_i \) and \( \tilde{R}_i = F_i \); if \( t_{i,n+p-1} = f_{i,n+p-1} \) then \( R_i = F_i \) and \( \tilde{R}_i = T_i \).

Then, (b) follows noting that configuration \( C_{n+p} \) is obtained from configuration \( C_{n+p-1} \) by applying the rules: \( [\beta_{n+p} \rightarrow \beta_{n+p+1}]_1 \), \([t_1, t_{n+p}] \rightarrow t_{i,n+p} \), \([f_i, t_{j,n+p}] \rightarrow f_{i,n+p} \), \( 1 \leq i \leq n \). \([x_i, j,n+p] \rightarrow x_i, j,n+p+1 \), \([x_i, j,n+p] \rightarrow x_i, j,n+p+1 \), \([x_i, j,n+p] \rightarrow \tilde{x}_i, j,n+p+1 \), \([x_i, j,n+p] \rightarrow \tilde{x}_i, j,n+p+1 \), \( 1 \leq i \leq n, 1 \leq j \leq p \).

4.3. First checking stage

At this stage, we try to determine the clauses satisfied by the truth assignments encoded by each membrane labelled by 2. For that, rules from 2.4.2 will be applied in such a manner that an object \( c_j \) is produced if and only if the truth assignment encoded by that membrane makes true clause \( C_j \). This stage takes exactly one step.

Proposition 3. Let \( C = (C_0, C_1, \ldots, C_q) \) be a computation of the system \( \Pi(s, \phi) \) with input multiset \( \text{cod}(\phi) \). At configuration \( C_{n+p+1} \) we have \( C_{n+p+1}(1) = \{ \alpha, \beta_{n+p+1} \} \) and there are \( 2^n \) membranes labelled by 2 each of them contains \( t_j \) copies of object \( c_j \), for \( 1 \leq j \leq p \), if and only if the truth assignment encoded by that membrane makes true exactly \( t_j \) literals of clause \( C_j \), and \( np - (t_1 + \cdots + t_p) \) copies of object \( \alpha \).

Proof. It suffices to note that configuration \( C_{n+p+1} \) is obtained from configuration \( C_{n+p} \) by applying the rules: \( [\beta_{n+p} \rightarrow \beta_{n+p+1}]_1 \), \([t_1, t_{n+p}] \rightarrow t_{i,n+p} \), \([f_i, t_{j,n+p}] \rightarrow f_{i,n+p} \), \( 1 \leq i \leq n \). \([x_i, j,n+p] \rightarrow x_i, j,n+p+1 \), \([x_i, j,n+p] \rightarrow x_i, j,n+p+1 \), \([x_i, j,n+p] \rightarrow \tilde{x}_i, j,n+p+1 \), \([x_i, j,n+p] \rightarrow \tilde{x}_i, j,n+p+1 \), \( 1 \leq i \leq n, 1 \leq j \leq p \).

4.4. Second checking stage

At this stage, we try to determine whether some truth assignment encoded by a membrane labelled by 2 satisfies all clauses of the input formula. To that end, rules from 2.5 will be applied in such a manner that object \( d_j \) (\( 2 \leq j \leq p \)) is produced in a membrane labelled by 2 if and only if the truth assignment encoded by that membrane makes true the clauses \( C_1, \ldots, C_j \). Then, the input formula will be satisfied by the truth assignment encoded by a membrane labelled by 2 if and only if object \( d_p \) appears in that membrane. This stage takes exactly \( p - 1 \) computation steps.

Proposition 4. Let \( C = (C_0, C_1, \ldots, C_q) \) be a computation of the system \( \Pi(s, \phi) \) with input multiset \( \text{cod}(\phi) \).

(a) For each \( k \), \( 1 \leq k \leq p - 1 \), at configuration \( C_{n+p+1+k} \) we have \( C_{n+p+1+k}(1) = \{ \alpha, \beta_{n+p+1+k} \} \) and there are \( 2^n \) membranes labelled by 2 each of them contains an object \( d_1, d_2 \) if and only if the truth assignment encoded in that membrane, makes true clauses \( C_1, \ldots, C_{k+1} \).

(b) \( \phi \) is satisfiable if and only if at configuration \( C_{n+2p} \) there exists some membrane labelled by 2 which contains an object \( d_p \).

Proof.

(a) By induction on \( k \). For the base case \( k = 1 \) it suffices to note that configuration \( C_{n+p+2} \) is obtained from configuration \( C_{n+p+1} \) by applying the rules \( [\beta_{n+p+1} \rightarrow \beta_{n+p+2}]_1 \) and \([c_1d_2 \rightarrow d_2]_2 \).

Let us assume that the result holds for \( k \), \( 1 \leq k < p - 1 \). Then, at configuration \( C_{n+p+1+k} \) we have \( C_{n+p+1+k}(1) = \{ \alpha, \beta_{n+p+1+k} \} \) and there are \( 2^n \) membranes labelled by 2 each of them containing an object \( d_{k+1} \) if and only if the truth assignment encoded in that membrane, makes true clauses \( C_1, \ldots, C_{k+1} \).
4.5. Proof.

Proof. (b) In order to prove (b), let us note that the input formula \( \varphi \) is satisfiable if and only if there exists a truth assignment \( \sigma \) making true \( \varphi \), that is, making true the clauses \( C_1, \ldots, C_p \). From (a) we deduce that \( \varphi \) is satisfiable if and only at configuration \( C_{n+2p} \) there exists some membrane labelled by 2 which contains an object \( d_p \).

4. Output stage

The output phase starts at the \((n+2p+1)\)-th step, and takes exactly four steps.

- **Affirmative answer:** if the input formula \( \varphi \) is satisfiable then at least one of the truth assignments from a membrane with label 2 makes true all clauses. Thus, a copy of object \( d_p \) will appear in that membrane at configuration \( C_{n+2p} \).
  
  Then, by applying the rules \([d_p]_2 \rightarrow \varphi[ ]_2\) and \([\beta_{n+2p} \rightarrow \beta_{n+2p+1}]_1\), objects \( \gamma \) and \( \beta_{n+2p+1} \) are produced in the skin membrane. At the next step, by applying rules \([\alpha \gamma \rightarrow \varphi']_1\) and \([\beta_{n+2p+1} \rightarrow \beta']_1\), objects \( \gamma' \) and \( \beta' \) are produced in the skin membrane. At the step \( n+2p+3 \), by applying rule \([\gamma' \rightarrow \gamma'']_1\), object \( \gamma'' \) is produced in the skin membrane (let us notice that object \( \beta' \) cannot interact with \( \alpha \)). Finally, at the next step, by applying rule \([\gamma'' \rightarrow yes[ ]_1\), object \( yes \) is sent out to the environment. Hence, the computation halts and the answer of the computation is \( yes \).

- **Negative answer:** if the input formula \( \varphi \) is not satisfiable then none of the truth assignments encoded by a membrane with label 2 makes the formula \( \varphi \) true. Thus, object \( d_p \) does not appear in any membrane labelled by 2 in configuration \( C_{n+2p} \). At the step \( n+2p+1 \), only rule \([\beta_{n+2p} \rightarrow \beta_{n+2p+1}]_1\) is applicable to \( C_{n+2p} \). Then, \( C_{n+2p+1}(1) = \{\alpha, \beta_{n+2p+1}\} \).
  
  At the next step, by applying rule \([\beta_{n+2p+1} \rightarrow \beta']_1\) we have \( C_{n+2p+2}(1) = \{\alpha, \beta'\} \). At the step \( n+2p+3 \), rule \([\alpha \beta' \rightarrow \beta'']_1\) produces an object \( \beta'' \) in the skin membrane. Finally, at the last step, by applying rule \([\beta'' \rightarrow no[ ]_1\), an object \( no \) is released to the environment. Consequently, the computation halts and the answer of the computation is \( no \).

5. Main results

**Theorem 1.** \( \text{SAT} \in \text{PMC}_{\Pi_{\text{DAM}}^0(\text{mcmp}., +c, -d, -n)}^* \)

**Proof.** The family of P systems previously constructed verifies the following:

(a) Every system of the family \( \Pi \) belongs to \( \text{DAM}^0(\text{mcmp}., + c, - d, - n) \).

(b) The family \( \Pi \) is polynomially uniform by Turing machines because for each \( n, p \in \mathbb{N} \), the amount of resources needed to build \( \Pi(n, p) \) is of a polynomial order in \( n \) and \( p \):

- Size of the alphabet: \( 3np^2 + 3n^2p + 5np + \frac{3n^2}{2} + \frac{3n}{2} + 4p + 10 \in \Theta(max(np^2, n^2p)) \).
- Initial number of membranes: \( 2 \in \Theta(1) \).
- Initial number of objects in membranes: \( 2np + n + 2 \in \Theta(np) \).
- Number of rules: \( 3np^2 + 3n^2p + 8np + \frac{3n^2}{2} + \frac{3n}{2} + 3p + 6 \in \Theta(max(np^2, n^2p)) \).
- Maximal number of objects involved in any rule: \( 3 \in \Theta(1) \).

(c) The pair \((\text{cod}, s)\) of polynomial-time computable functions defined fulfill the following: for each input formula of the \( \text{SAT} \) problem, \( s(\varphi) \) is a natural number, \( \text{cod}(\varphi) \) is an input multiset of the system \( \Pi(s(\varphi)) \), and for each \( n \in \mathbb{N} \), \( s^{-1}(n) \) is a finite set.

(d) The family \( \Pi \) is polynomially bounded: indeed, for each input formula \( \varphi \) of the \( \text{SAT} \) problem, the deterministic P system \( \Pi(s(\varphi)) + \text{cod}(\varphi) \) takes exactly \( n+2p+4 \) steps, \( n \) being the number of variables of \( \varphi \) and \( p \) the number of clauses.

(e) The family \( \Pi \) is sound with regard to \((X, \text{cod}, s)\): indeed, for each input formula \( \varphi \), if the computation of \( \Pi(s(\varphi)) + \text{cod}(\varphi) \) is an accepting computation, then \( \varphi \) is satisfiable.

(f) The family \( \Pi \) is complete with regard to \((X, \text{cod}, s)\): indeed, for each input formula \( \varphi \) such that it is satisfiable, the computation of \( \Pi(s(\varphi)) + \text{cod}(\varphi) \) is an accepting computation.

Therefore, the family \( \Pi \) of P systems previously constructed solves the \( \text{SAT} \) problem in polynomial time in a uniform way. □

**Corollary 1.** \( \text{NP} \cup \text{co-NP} \subseteq \text{PMC}_{\Pi_{\text{DAM}}^0(\text{mcmp}., + c, - d, - n)}^* \)

**Proof.** It suffices to note that the \( \text{SAT} \) problem is an \( \text{NP} \)-complete problem, \( \text{SAT} \in \text{PMC}_{\Pi_{\text{DAM}}^0(\text{mcmp}., + c, - d, - n)}^* \), and class \( \text{PMC}_{\Pi_{\text{DAM}}^0(\text{mcmp}., + c, - d, - n)} \) is closed under polynomial-time reduction and under complement. □

**Corollary 2.** \( \text{P} = \text{PMC}_{\Pi_{\text{SAM}}^0(\text{mcmp}, + c, - d, - n)}^* \)
Proof. It suffices to notice that $P = PMC_{S,A^k \forall p(bmc_+-\ldots-n)}$ and each rule using minimal cooperation and minimal production is also a rule using bounded minimal cooperation, and realizing that the class is closed under polynomial-time reduction. For the reverse inclusion, we only need to keep in mind the Sevilla theorem to see that we can simulate any Deterministic Turing Machine with this kind of membrane systems. \hfill \Box

6. Conclusions

Limitations of polarizationless P systems with active membranes not using dissolution rules, with respect to efficiency, are well known. In this paper, the computational efficiency of such kind of P systems using only division rules for elementary membranes is studied in the case that a very restrictive cooperation in object evolution rules is considered. Specifically, the left-hand side of the rules consists of at most two objects and each such rule only can produce a single object. The efficiency of these systems is shown, improving a result concerning object evolution rules with minimal cooperation, where the length of their right-hand side is less than or equal to the corresponding left-hand side.

It is worth pointing out that the situation is completely different when division rules is replaced by separation rules; that is, when in the mechanism of producing an exponential number of membranes in linear time, distribution of objects is considered instead of the replication of objects. In this case, only problems in class $P$ can be efficiently solved by families of polarizationless P systems with active membranes which use minimal cooperation and minimal production in object evolution rules. Consequently, new frontiers of efficiency are obtained.

These results confirm the strength of the replication with respect to the distribution of objects, from an efficiency point of view, and the irrelevant role played by dissolution when minimal cooperation is considered.

As future work, we propose to study polarizationless P systems with active membranes when cooperation in communication rules is considered instead of cooperation in object evolution rules. It seems that, in this case, division rules for non-elementary membranes can play a relevant role from a computational complexity point of view.

Acknowledgements

This work was partially supported by Grant numbers 61472328 and 61320106005 of the National Natural Science Foundation of China.

References