Solving NP Complete Problems Using P-Systems with Active Membranes*

Claudio Zandron, Claudio Ferretti, Giancarlo Mauri

DISCO - Universitá di Milano-Bicocca - Italy
zandron@disco.unimib.it

Abstract. A recently introduced variant of P-systems considers membranes which can multiply by division. These systems use two types of division: division for elementary membranes (i.e. membranes not containing other membranes inside) and division for non-elementary membranes. In two recent papers it is shown how to solve the Satisfiability problem and the Hamiltonian Path problem (two well known NP complete problems) in linear time with respect to the input length, using both types of division. We show in this paper that P-systems with only division for elementary membranes suffice to solve these two problems in linear time. Is it possible to solve NP complete problems in polynomial time using P-systems without membrane division? We show, moreover, that (if $P \neq NP$) deterministic P-systems without membrane division are not able to solve NP complete problems in polynomial time.

1 Introduction

The P-systems were introduced in [5] as a class of distributed parallel computing devices of a biochemical type. In the basic model we have a membrane structure composed by several cell-membranes, hierarchically embedded in a main membrane called the skin membrane. The membranes delimit regions and can contain objects. The objects evolve according to given evolution rules associated with the regions. A rule can modify the objects and send them outside the membrane or to an inner membrane. Moreover, the membranes can be dissolved. When a membrane is dissolved, all the objects in this membrane remain free in the membrane placed immediately outside, while the evolution rules of the dissolved membrane are lost. The skin membrane is never dissolved. The evolution rules are applied in a maximally parallel manner: at each step, all the objects which can evolve should evolve.

A computation device is obtained: we start from an initial configuration and we let the system evolve. A computation halts when no further rule can be applied. The output consist in the string obtained by concatenating the objects which leave the system through the skin membrane, taken in the order they are sent out.

* This work has been supported by the Italian Ministry of University (MURST), under project "Unconventional Computational Models: Syntactic and Combinatorial Methods".

Many variants are considered in [1], [4], [5], [6], [7], [8] and [10]. In these papers, the membranes are used as separators and as channels of communication, but they participate to the process in a passive way. The whole process is regulated by the evolution rules.

In [3] one consider membrane systems where the membranes play an active role in the computation: the evolution rules are associated both with objects and the membrane; the communication of the objects through the membranes is performed with the direct participation of the membranes; and finally, the membranes can not only be dissolved, but they can multiply by division. There are two different types of division rules: division rules for elementary membranes (i.e. membranes not containing other membranes) and division rules for non-elementary membranes. In [3] and [2] it is shown how to solve the Satisfiability problem and the Hamiltonian Path problem respectively (two well known NP complete problems) in linear time with respect to the input length, using P-systems with active membranes. We show here how to solve these two problems in linear time, using P-systems with active membranes which only make use of division for elementary membranes.

Another interesting question relates to the needed of active membranes: what about the possibility of solving NP complete problems in polynomial time using P-systems without membrane division? In [3] and [9] one points out that P-systems without active membranes already possess a great amount of parallelism (all the rules are applied at the same time on all objects which can evolve) but it seems not to be sufficient to solve complex problems in polynomial time. In fact, we show here that this is the case: if $P \neq NP$, a deterministic P-system without membrane division is not able to solve a NP complete problem in polynomial time; every deterministic P-system without active membranes working in time t, can be simulated by a deterministic Turing Machine working in time polynomial in t.

2 P-systems with Active Membranes

In the following, we refer to [11] for elements of Formal Language Theory. A membrane structure is a construct consisting of several membranes placed in a unique membrane; this unique membrane is called a skin membrane. We identify a membrane structure with a string of correctly matching parentheses, placed in a unique pair of matching parentheses; each pair of matching parentheses corresponds to a membrane. The membranes can be marked with $+$, $-$ or 0, and this is interpreted as an electrical charge (0 is the neutral charge). A membrane identifies a region, delimited by it and the membrane immediately inside it. If we place multisets of objects in the region from a specified finite set V, we get a super-cell. A super-cell system (or P-system) is a super-cell provided with evolution rules for its objects.

A P-system with active membranes is a construct

$$\Pi = (V, T, H, \mu, w_1, \ldots, w_m, R),$$

where:

- $m \geq 1$;
- V is an alphabet (the total alphabet of the system);
- $T \subseteq V$ is the terminal alphabet;
- H is a finite set of labels for membranes;
- μ is a membrane structure, consisting of m membranes, labeled (not necessarily in a one-to-one manner) with elements of H; all membranes in μ are supposed to be neutral;
- $w_1, \ldots, w_m$ are strings over V, describing the multisets of objects placed in the m regions of μ;
- R is a finite set of developmental rules, of the following forms:
  (a) $[_h a \to v]_h^\alpha$, for $h \in H,\ a \in V,\ v \in V^*, \alpha \in \{+, -, 0\}$ (object evolution rules),
  (b) $a[_h]_h^{\alpha_1} \to [_h b]_h^{\alpha_2}$, where $a, b \in V,\ h \in H,\ \alpha_1, \alpha_2 \in \{+, -, 0\}$ (an object from the region immediately outside the membrane h is introduced in membrane h),
  (c) $[_h a]_h^{\alpha_1} \to [_h]_h^{\alpha_2} b$, for $h \in H,\ \alpha_1, \alpha_2 \in \{+, -, 0\},\ a, b \in V$ (an object is sent out from membrane h to the region immediately outside),
  (d) $[_h a]_h^\alpha \to b$, for $h \in H,\ \alpha \in \{+, -, 0\},\ a, b \in V$ (membrane h is dissolved),
  (e) $[_h a]_h^{\alpha_1} \to [_h b]_h^{\alpha_2} [_h c]_h^{\alpha_3}$, for $h \in H, \alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}, a, b, c \in V$ (division rules for elementary membranes)
  (f) $[_{h_0} [_{h_1}]_{h_1}^{\alpha_1} \ldots [_{h_k}]_{h_k}^{\alpha_1} [_{h_{k+1}}]_{h_{k+1}}^{\alpha_2} \ldots [_{h_n}]_{h_n}^{\alpha_2}]_{h_0}^{\alpha_0} \to$
  $\to [_{h_0} [_{h_1}]_{h_1}^{\alpha_3} \ldots [_{h_k}]_{h_k}^{\alpha_3}]_{h_0}^{\alpha_5} [_{h_0} [_{h_{k+1}}]_{h_{k+1}}^{\alpha_4} \ldots [_{h_n}]_{h_n}^{\alpha_4}]_{h_0}^{\alpha_6}$ for $k \geq 1, n \geq 1, h_i \in H, 0 \leq i \leq n$, and $\alpha_0, \ldots, \alpha_6 \in \{+, -, 0\}$ with $\{\alpha_1, \alpha_2\} = \{+, -\}$ (division rules for non-elementary membranes)

These rules are applied accordingly to the principles in [3].

1. All the rules are applied in parallel: in a step, the rules of type (a) (i.e. the rules that do not modify the membranes) are applied to all objects to which they can be applied; all other rules are applied to all membranes to which they can be applied; an object can be used by only one rule, nondeterministically chosen (there is no priority relation among rules), but any object which can evolve by a rule of any form, must evolve.
2. If a membrane is dissolved, then all the objects in its region are left free in the region immediately above it. Because all rules are associated with membranes, the rules of a dissolved membrane are no longer available at the next steps. The skin membrane is never dissolved.
3. All objects and membranes not specified in a rule and which do not evolve are passed unchanged to the next step.
4. If at the same time a membrane h is divided by a rule of type (e) and there are objects in this membrane which evolve by means of rules of type (a), then in the new copies of the membrane we introduce the result of the evolution; that is, we may suppose that first the evolution rules of type (a) are used, changing the objects, and then the division is produced, so that in the two

new membranes with label h we introduce copies of the changed objects. Of course, this process takes only one step. The same assertions apply to the division by means of a rule of type (f): always we assume that the rules are applied "from bottom-up", in one step, but first the rules of the innermost region and then level by level until the region of the skin membrane.

5. The rules associated with a membrane h are used for all copies of this membrane, irrespective whether or not the membrane is an initial one or it is obtained by division. At one step, a membrane h can be the subject of only one rule of types (b) – (f).
6. The skin membrane can never divide. As with any other membrane, the skin membrane can be "electrically charged".

The membrane structure at a given time, together with all multisets of objects associated with the regions defined by the membrane structure, is the *configuration* of the system at that time. The *initial configuration* is $(\mu, w_1, \ldots, w_m)$. We can pass from a configuration to another one by using the rules in R, according to the principles previously described (we call this a *transition*). A *computation* is a sequence of transitions between configurations. A computation *halts* when there is no rule which can be applied to objects and membranes in the current configuration.

During the computation, objects can leave the skin membrane (with a rule of type *(c)*). The terminal symbols which leave the skin membrane are collected in the order of their expelling from the system, so a string is associated to a complete computation. The symbols not in T which leave the skin membrane and all symbols in T which remain in the system at the end of a halting computation are not considered in the generated strings. If a computation never stops, then it provides no output.

Moreover, we will need the extension introduced in [2], about the rules of types *(e)* and *(f)*: a membrane can be divided into finitely many membranes. This type of division is called *bounded membrane division.*

In the following, we do not make use of division for non-elementary membranes, so we give the following definition:

A *P-system with elementary active membranes* is a P-system with active membranes where the rules are of type (a) to (e) only (i.e. a P-system with active membranes not using rules of type (f)).

Finally, we say that a P-system (with or without active membranes) is *deterministic* if, in each moment, there is at most one possible transition. This means that there are no two different rules applicable at the same time in the same region on the same symbol.

More details and examples can be found in [5], [3] and [2].

3 Solving NP Complete Problems Using P-system with Elementary Active Membranes

In this section we show how to solve the NP complete problems proposed in [3] and [2] (Satisfiability and Hamiltonian Path) using P-systems with elementary

active membranes: although the systems in [3] and [2] do make use of division for non-elementary membranes, the time we need to execute the computation is of the same orders as theirs. In fact, the systems presented here solve Satisfiability and Hamiltonian Path in a time which is linear with respect to the dimension of the input; this time is comparable with the time needed by the systems in [3] and [2]. Moreover, the structure (i.e. the number of symbols, the number of rules and the number of membranes) of the systems we propose is similar to the structure of the systems proposed in [3] and [2].

Theorem 1. *The SAT problem can be solved in linear time (with respect to the number of variables and the number of clauses) by a P-system with elementary active membranes.*

Proof. Consider a boolean expression in conjunctive normal form

$$\alpha = C_1 \wedge C_2 \wedge ... \wedge C_m$$

with

$$C_i = y_{i,1} \vee y_{i,2} \vee ... \vee y_{i,p_i}$$

for some $m \geq 1$, $p_i \geq 1$, and $y_{i,j} \in \{x_k, \neg x_k | 1 \leq k \leq n\}$, for each $1 \leq i \leq m$, $1 \leq j \leq p_i$.

We build the P-system

$$\Pi = (V, T, H, \mu, \omega_0, \omega_1, R)$$

where

- $V = \{a_i, t_i, f_i | 1 \leq i \leq n\} \cup \{r_i | 0 \leq i \leq m\} \cup \{W_i | 1 \leq i \leq m+1\} \cup \{t\} \cup$ $\cup \{Z_i | 0 \leq i \leq n\}$
- $T = \{t\}$
- $H = \{0, 1\}$
- $\mu = [_0[_1]_1^0]_0^0$
- $\omega_0 = \lambda$
- $\omega_1 = a_1 a_2 ... a_n Z_0$

while the set R contains the following rules:
1. $[a_i]_1^0 \rightarrow [t_i]_1^0 [f_i]_1^0$, $1 \leq i \leq n$
We substitute one variable a_i in membrane 1 with two variables t_i and f_i. The membrane is divided in two membranes (the charge remains neutral for both membranes). In n steps we get all 2^n truth assignments for the n variables; each truth assignment is in a membrane labelled with 1.
2. $[Z_k \rightarrow Z_{k+1}]_1^0$, $0 \leq k \leq n-2$
3. $[Z_{n-1} \rightarrow Z_n W_1]_1^0$
4. $[Z_n]_1^0 \rightarrow []_1^+ \lambda$
We count n steps, the time needed to produce all the truth assignments. The step $n-1$ introduces the symbol W_1 used in the next steps, while the step n changes the charge of the membranes labelled with 1.

5. $[t_i \to r_{h_{i,1}} ... r_{h_{i,j_i}}]_1^+$, $1 \leq i \leq n$ and the clauses $h_{i,1}$, ..., h_{i,j_i} contain x_i.
6. $[f_i \to r_{h_{i,1}} ... r_{h_{i,l_i}}]_1^+$, $1 \leq i \leq n$ and the clauses $h_{i,1}$, ..., h_{i,l_i} contain $\neg x_i$.
In one step, each symbol t_i is replaced with some symbols r_k, indicating the clauses satisfied if we set $x_i = true$; each symbol f_i is replaced with some symbols, indicating the clauses satisfied if we set $x_i = false$ (i.e. $\neg x_i = true$).

After this step, we start the "VERIFY STEPS": we have to verify if there is a membrane, labelled with 1, in which we get all symbols $r_1, r_2, ..., r_m$ (at least one symbol r_k for every k between 1 and m). In fact, this means that there is a truth assignment satisfying all the clauses. To do this, we apply (in a single step) the rules of type 7 in all membranes containing the symbol r_1; then, we apply (in a single step) the rules of types 8, 9 and 10, on all symbols which can evolve. We repeat this 2-steps process for each symbol $r_1, r_2, \ldots, r_m$, thus the whole process takes $2m$ steps.
7. $[r_1]_1^+ \to []_1^- r_1$
We first verify the presence of the symbol r_1 in membranes labelled with 1. Every membrane containing r_1 (i.e. every membrane with a truth assignment satisfying the first clause) sends this symbol outside (in membrane 0) and changes its charge from $+$ to $-$. The membranes not containing this symbol cannot further proceed their computation.
8. $r_1[]_1^- \to [r_0]_1^+$
9. $[r_k \to r_{k-1}]_1^-$, $2 \leq k \leq m$
10. $[W_i \to W_{i+1}]_1^-$, $1 \leq i \leq m$
The computation can only proceed in membranes with negative charge. In one step, each symbol W_i is replaced with a symbol W_{i+1} (the counter of the satisfied clauses is increased), each symbol r_k is replaced with r_{k-1} and the symbols r_1 in membrane 0 are sent back (as r_0) to the negative charged membranes labelled with 1, to change their charge from $-$ to $+$.

After applying rules 8, 9, and 10 (in a single step on all symbols and membranes which can evolve) we are ready to re-apply the rules of type 7. At this time, the symbols r_1 in membranes 1 are the symbols that was r_2 when the verify steps started (the index of each symbol r_k has been decreased). Thus the computation only proceed in membranes which contained both symbols r_1 and r_2 when the verify steps started. After the application of the rules of type 7 we can apply again the rules of type 8, 9 and 10 to prepare the membranes to verify the next clause.

In $2m$ steps we verify the existence of a membrane containing all symbols r_i. It is easy to see that if a membrane does not contain a symbol r_i, the computation in that membrane halts in less than $2m$ steps. As a consequence, that membrane will not contain the symbol W_{m+1}. The membranes containing this symbol after $2m$ steps are the membranes which contained all symbols r_i when the verify steps started, i.e. the membranes satisfying all the clauses.
11. $[W_{m+1}]_1^+ \to []_1^+ t$
If a membrane labelled with 1 executes all $2m$ verify steps, it contains the symbol W_{m+1}. Thus we send out to membrane 0 the symbol t, indicating that there is a truth assignment satisfying all clauses.

12. $[t]_0^0 \rightarrow []_0^+ t$
A symbol t is sent outside membrane 0 and the charge of this membrane is changed from 0 to $+$. No further computation is possible. Thus, we have to look at the output of membrane 0 after $n + 2m + 3$ steps. If we get the symbol t, it means that there is a truth assignment satisfying α; otherwise the formula is not satisfiable. □

As we have said at the beginning of this section, the time of computation of the system proposed in the previous proof is comparable with the time of computation of the system proposed by Paun in [3], and the same is true for the structure of the systems. In the next table we explicitly show this, by comparing the system presented here with the system proposed in [3]. We compare the systems under a computational complexity point of view as well as under a structural complexity point of view. With n and m we denote respectively the number of variables and the number of clauses of a generic instance of the SAT problem.

	Paun	*This*
Number of Steps of Computation	$2n + 2m + 1$	$n + 2m + 3$
Number of Symbols	$5n + m + 2$	$4n + 2m + 4$
Max Number of Membranes	$(m + 1) \times 2^n + 1$	$2^n + 1$
Depth	$m + 1$	1
Number of Rules	$3n + 2m + 2nm$	$4n + 2m + 4$

As one can see, for $n \geq 3$ the system we propose in this paper is slightly faster than the system in [3]. Another interesting aspect of the system we propose relates to the depth of the membrane structure: the depth of the system we propose is always one. This indicate a very simple membrane structure, composed by elementary membranes surrounded by the skin membrane.

Theorem 2. *The Undirected Hamiltonian Path Problem can be solved in linear time (with respect to the number of nodes in the given graph) by a P-system with elementary active membranes and bounded membrane division.*

Proof. Consider an undirected graph $G = (U, E)$ with n nodes, $n \geq 2$.
We build the P-system

$$\Pi = (V, T, H, \mu, \omega_0, \omega_1, R)$$

where

- $V = \{a_i, P_i | 1 \leq i \leq n\} \cup \{Z_i | 0 \leq i \leq 2n - 1\} \cup \{W_i | 0 \leq i \leq m + 1\} \cup$
  $\cup \{r_i, x_i | 1 \leq i \leq n + 1\} \cup \{t, R\}$
- $T = \{t\}$
- $H = \{0, 1\}$
- $\mu = [_0 [_1]_1^0]_0^0$
- $\omega_0 = \lambda$

– $\omega_1 = RZ_0$

while the set R contains the following rules:
1. $[R]_1^0 \to [a_1]_1^0 ... [a_n]_1^0$
We create n membranes labelled with 1, each containing a symbol a_i corresponding to a node in G.
2. $[Z_k \to Z_{k+1}]_1^0$, $0 \leq k \leq 2n-3$
3. $[Z_{2n-2} \to Z_{2n-1}C_1]_1^0$
4. $[Z_{2n-1}]_1^0 \to []_1^+ \lambda$
We count $2n$ steps, the time needed to produce all paths of length n.
5. $[a_i \to r_i P_i]_1^0$, $1 \leq i \leq n$
6. $[P_i]_1^0 \to [a_{j_1}]_1^0 ... [a_{j_k}]_1^0$, $1 \leq i \leq n$, $1 \leq j_h \leq n$ and (i, j_1), ..., (i, j_k) are edges of G.
In $2n$ steps, we create all paths of length n we can obtain starting from every node in G. At each step, we divide the membranes by replacing the symbol P_i with the symbols corresponding to the nodes directly connected to the node i. Note that the rule of type 4 changes the charge of the membranes labelled with 1 from 0 to +, thus after $2n$ steps, no rule of the previous types can be applied.
7. $[r_1]_1^+ \to []_1^- r_1$
8. $r_1[]_1^- \to [r_0]_1^+$
9. $[r_i \to r_{i-1}]_1^-$, $1 \leq i \leq m$
10. $[C_i \to C_{i+1}]_1^-$, $1 \leq i \leq m$
11. $[C_{m+1}]_1^+ \to []_1^+ t$
12. $[t]_0^0 \to []_0^+ t$
Once the charge of membranes labelled with 1 has become +, what we need to do is to verify the existence of a membrane containing the symbols $r_1, r_2, ..., r_n$. Such a membrane exists if and only if the graph G has a Hamiltonian path. To verify this, we can proceed as in the previous proof, using the rules of types 7 to 12. After $4n+2$ steps, we look at the output of membrane 0. If we get the symbol t then there is a Hamiltonian Path in G, otherwise no such path exists. □

As for the SAT problem, in the next table we compare the system presented here with the system presented in [2] to solve the Undirected Hamiltonian Path Problem. Given a graph G as the input for the Undirected Hamiltonian Path Problem, we denote with n the number of nodes of the graph.

	Krishna, Rama	*This*
Number of Steps of Computation	$3n+1$	$4n+2$
Number of Symbols	$4n+1$	$6n+4$
Max Number of Membranes	$(n+2) \times n \times (n-1)^{n-1}$	$2 \times n \times (n-1)^n$
Depth	$n+1$	1
Number of Rules	$14n+8$	$6n+5$

The system we propose to solve the Hamiltonian Path Problem is slightly slower with respect to the system proposed in [2], but the time remains linear

with respect to the input length. The depth of the system we propose is still one.

4 P-systems without Active Membranes

One interesting question, pointed out in [3] and [9], concerns the power of the operation of membrane division. Are we able to solve NP complete problems in polynomial time using P-systems without active membranes? In fact, the model without active membranes has a great amount of parallelism, but we do not know if it suffices to solve complex problems in an efficient way.

The systems presented in the previous section are deterministic (like the systems in [3] and [2]): at each step of computation there is at most one possible transition from a configuration to another one. We show now that, under the assumption of using deterministic P-systems, if $P \neq NP$ the amount of parallelism without membrane division is not enough to solve NP complete problems in polynomial time. Membrane division is necessary to get significant speed up of computation.

Theorem 3. *Consider a deterministic P-system, without membrane division and working in time t. We denote with A, B and C respectively the number of membranes, the number of symbols in V and the length of the rules (the number of symbol involved in the rule, on both left and right side) of the P-system. Moreover, we set $D = max\{C, B+2\}$. Such a system can be simulated by a Deterministic Turing Machine (DTM) in time $t' = O(A \times B \times D \times t \times log(A \times B \times C^t))$.*

Proof. Consider a P-system

$$P = (V, T, H, \mu, \omega_0, ..., \omega_k, R)$$

without membrane division working in time t.

We build a DTM $M = (V', K, S_0, \delta)$ with multiple tapes, which simulates P within a number of steps $O(A \times B \times D \times t \times log(A \times B \times C^t))$.

To simulate P with M, we have just to keep track of the quantity of each symbol $z \in V$ in each membrane. In fact, consider the application of a rule $[a \rightarrow bc]_i^\alpha$: all the symbols a in membrane i are substituted with two symbols, b and c. We can simulate this rule by adding to the quantities of symbols b and c, in membrane i, the quantity of symbols a and then by setting the last quantity to zero. In other words, the application of a rule in P corresponds to a modification of the quantities of the symbols involved in the rule of the specific membrane (and eventually, as we will see, in modifying the quantities of all symbols in a membrane, if a membrane placed immediately inside this one is dissolved by the rule). The modification of the quantities is done by adding the quantity of the symbol on the left side of the rule to each quantity of the symbols on the right side of the rule.

For the previous reasons, the DTM M has $2 \times (A \times B) + 3$ tapes:

- $A \times B$ "main" tapes, to keep track of the quantities of each symbol in each membrane (A membranes and B symbols) after every step.
- $A \times B$ "support" tapes, used to make partial sums.
- 1 "polarity" tape, used to keep track of the polarity of each membrane.
- 1 "structure" tape, used to keep track of the structure of membranes.
- 1 tape as output.

Every computation step of P can be simulated with 2 macro steps of M:

1. We simulate the application of a rule on each symbol in each membrane by modifying the quantity written on the tapes containing the partial sums. For example, consider a rule $[a \to bc]_1^0$, and let the strings written on the tapes corresponding to the symbols a, b and c in membrane 1 contain the quantity 100, 200 and 250. To simulate this rule, we have to put the value 0 in the first string, 300 in the second one and 350 in the third one. We do not write this quantity on the main tapes, because to simulate the application of another rule $[b \to de]_1^0$ we have to know that the quantity of symbols b when the computation step started was 200. For this reason we need support tapes. The rule to be applied (only one rule per symbol can be applied, because we consider deterministic P-system) is chosen accordingly to the polarity of the membrane; the polarity can be found in the "polarity" tape.

2. When the application of the rules has been executed on all objects, we copy the quantity of each symbol from the support tapes to the main tapes.

We have to repeat these two steps t times. To simulate a single step of computation of P, we have to simulate the application of at most $A \times B$ different rules in R (one rule for each object in each membrane). The number of sums to be executed for each rule in each membrane depends on the type of rule applied. Consider the four types of rules in R:

a)$[a \to x]_i^\alpha$ where $a \in V$ and $x \in V^*, |x| \leq C - 1$. To simulate such a rule, we have to execute at most C sums.

b)$a[]_i^\alpha \to [b]_i^\beta$, where $a, b \in V$. To simulate such a rule, we have to sum 1 to the quantity of the symbol b in membrane i and to sum -1 to the quantity of the symbol a in the membrane placed immediately outside membrane i (remember that, accordingly to the rules in [3], at each computation step only one symbol can enter a rule of type $(b), (c)$ or (d) in each membrane). If needed, we change the charge of the membrane in the "polarity" tape.

c)$[a]_i^\alpha \to []_i^\beta b$, where $a, b \in V$. To simulate such a rule on a symbol, we have to sum 1 to the quantity of the symbol b in the membrane placed immediately outside membrane i and to sum -1 to the quantity of the symbol a in membrane i. If needed, we change the charge of the membrane in the "polarity" tape. If i is the skin membrane, we have to write the symbol b on the output tape.

d)$[[a]_i^\alpha]_k^\beta \to [b]_k^\gamma$, where $a, b \in V$. To simulate such a rule, we have to sum 1 to the quantity of the symbol b in the membrane placed immediately outside membrane i, and to sum -1 to the quantity of the symbol a in membrane i. Then we have to sum the quantity of each symbol in membrane i to the quantity of the same symbol in membrane k. Thus, we have to execute $B + 2$ sums. After

these sums, we modify the structure of the membrane in the "structure" tape and we change the charge of the membrane k.

It is easy to see that the most expensive rules (in terms of time needed to simulate them) are those of type (a) and (d). For this reason we set $D = max\{C, B+2\}$. To simulate a single step of computation of P we need at most $A \times B \times D$ sums.

The time required by each sum depends, of course, on the number of digits of the numbers to sum. We can consider membrane system with a maximum number of initial symbols equal to $A \times B$ (i.e. every membrane contains at most one occurrence of each symbol in V). In fact, given a P-systems with an arbitrary finite number of occurrence of each symbol in each membrane, it is easy to build a P-systems with exactly one occurrence of each symbol in each membrane that requires a finite number of steps to reach that same configuration. Thus, after one step, the total quantity of symbols present in the system will be less than $A \times B \times C$. After the second step, this quantity will be less than $A \times B \times C^2$. After t steps, we get no more than $A \times B \times C^t$ symbols. This means that each sum will be made with number involving less than $log(A \times B \times C^t)$ digits.

The time required by a DTM to make a sum is linear in the number of digits of the number involved; every sum requires a time which is $O(log(A \times B \times C^t))$. To execute the sums for each symbols in each membrane, the time needed is $O(A \times B \times D \times log(A \times B \times C^t))$. After the execution of all the sums, we have to copy the obtained results from the support tapes to the main tapes; the time needed is $O(A \times B \times log(A \times B \times C^t))$. The total time requested by M to simulate a single step of P is $O(A \times B \times D \times log(A \times B \times C^t))$.

Of course, we have to simulate t steps of P, thus the total time needed is $O(t \times A \times B \times D \times log(A \times B \times C^t))$. □

If, given a problem, we could build for each instance of size n a P-systems without membrane division such that it gives the solution in time polynomial in n, then we could build a DTM that gives the same output in polynomial time too. From this follows:

Corollary 1. *If $P \neq NP$ then no NP complete problem can be solved in polynomial time, with respect to the input length, by a deterministic P-system without membrane division.*

As one can see, the time needed to simulate a deterministic P-system depends on the number of membranes of the P-system itself (as well as on the number of symbols in V and on the length of the rules). If we use P-systems without membrane division, the number of membrane remains the same (or is decreased, if we dissolve some membranes) during the whole computation process. This is not true, of course, for P-systems with membrane division. With this variant of P-systems, the number of membranes can grow exponentially: if we repeatedly divide A membranes, we can obtain, after t steps, a number of membranes equal to $A \times 2^t$. The simulation of such a system with a DTM (in the same way we have discussed above) would require an exponential number of strings, or an exponential amount of time equal to $O(t \times A \times 2^t \times B \times D \times log(A \times 2^t \times B \times C^t))$.

5 Conclusions

In [3] one points out that a question of a practical importance is to try to implement a P-system either in biochemical media or in electronic media. Moreover, it is underlined that it could be necessary to consider variants of P-systems which are more realistic. This paper goes in this direction, by showing how to solve two NP complete problems (Satisfiability and Hamiltonian Path) without using division for non-elementary membranes (which seems, of course, more complicated with respect to the division for elementary membranes). Moreover, in the same paper and in [9] the question arises if P-systems without membrane division can solve NP complete problems in polynomial time. We have shown that this is not the case: deterministic P-systems without active membranes cannot solve NP complete problems in polynomial time (unless $P = NP$). Every deterministic P-system working in polynomial time, with respect to the input length, can be simulated by a Deterministic Turing Machine working in polynomial time. This proves that P-systems which make use of the operation of membrane division effectively obtain a significant speed-up of computation.

We point out some problems worth further investigations. One problem, already pointed out in [9], is that of simulating P-systems with d-bounded division presented in [2] with P-systems with active membranes presented in [3] (where the division of a membrane always leads to two new membranes). Of course, we know that every NP problem can be reduced to SAT in polynomial time, so we know we can solve every NP problem with a 2-division P-system that works in polynomial time. Nevertheless, it would be interesting to find out if it is possible to simulate every k-division systems with a 2-division system without using the reduction between problems. In [9] one says that the main difficulty appears with the division for non-elementary membranes, but in the view of the results of this paper we need to simulate only division for elementary membranes. Moreover, it can be interesting to follow the direction of finding other ways to increasing the number of membranes; as we have seen, the basic model of P-system does not have sufficient parallelism to solve complex problem in an efficient way, thus we have to find other ways to get such power. Finally, we think another important topic is that of determinism and nondeterminism. An investigation of the power of Deterministic P-systems and Nondeterministic P-systems could be useful from both a mathematical point of view and for a practical implementation of a P-system. Many other open problems and research topics can be found in [9].

References

1. J. Dassow, Gh. Paun, On the power of membrane computing, J. Univ. Computer Sci., 5, 2 (1999), 33-49.
2. S. N. Krishna, R. Rama, A variant of P-systems with active membranes: Solving NP-complete problems, Romanian J. of Information Science and Technology, 2, 4 (1999).

3. Gh. Paun, P-systems with active membranes: attacking NP complete problems, submitted 1999 (see also CDMTCS Research report No. 102, 1999, Auckland Univ., New Zeland, www.cs.auckland.ac.nz/CDMTCS)
4. Gh Paun, Computing with membranes. An introduction, Bulletin of the EATCS, 67 (Febr. 1999), 139-152.
5. Gh. Paun, Computing with membranes, J. of Computer and System Sciences, in press (see also TUCS Research Report No 208, November 1998 http://www.tucs.fi).
6. Gh. Paun, Computing with membranes. A variant, J. of Computer and System Sciences, 11, 167-, 2000 (see also CDMTCS Report No. 0.98, 1999, of CS Department, Auckland Univ., New Zealand, www.cs.auckland.ac.nz/CDMTCS).
7. Gh. Paun, G.. Rozenberg, A. Salomaa, Membrane computing with external output, Fundamenta Informaticae, 41, 3, 2000 (see also TUCS Research Report No. 218, December 1998, http://www.tucs.fi).
8. Gh. Paun, S. Yu, On synchronization in P-systems, Fundamenta Informaticae, 38, 4 (1999), 397-410 (see also CS Department TR No 539, Univ. of Western Ontario, London, Ontario, 1999, www.csd.uwo.ca/faculty/syu/TR539.html).
9. Gh. Paun, Computing with membrane (P-systems): Twenty-six research topics, Research Report, 2000.
10. I. Petre, A normal form for P-systems, Bulletin of EATCS, 67 (Febr. 1999), 165-172.
11. G. Rozenberg, A. Salomaa, eds. , Handbook of Formal Languages, Springer-Verlag, Heidelberg, 1997.