On the Power of Dissolution in P Systems with Active Membranes

Miguel A. Gutiérrez–Naranjo, Mario J. Pérez–Jiménez, Agustín Riscos–Núñez, and Francisco J. Romero–Campero

Research Group on Natural Computing,
Department of Computer Science and Artificial Intelligence,
University of Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
{magutier, marper, ariscosn, fran}@us.es

Abstract. In this paper we study membrane dissolution rules in the framework of P systems with active membranes but without using electrical charges. More precisely, we prove that the polynomial computational complexity class associated with the class of recognizer P systems with active membranes, without polarizations and without dissolution coincides with the standard complexity class **P**. Furthermore, we demonstrate that if we consider dissolution rules, then the resulting complexity class contains the class **NP**.

1 Introduction

Membrane Computing is inspired by the structure and functioning of living cells, and it provides a new non–deterministic model of computation which starts from the assumption that the processes taking place in the compartmental structure of a living cell can be interpreted as computations. The devices of this model are called *P systems*.

Roughly speaking, a P system consists of a cell-like membrane structure, in the compartments of which one places multisets of objects which evolve according to given rules in a synchronous non–deterministic maximally parallel manner.

In this paper we work with P systems with active membranes. This model was introduced in [7], abstracting the way of obtaining new membranes through the process of *mitosis* (membrane division) and providing a tool able to construct an exponential workspace in linear time. In these devices membranes are considered to have polarizations, one of the "electrical charges" $0, -, +$, and several times the problem was formulated whether or not these polarizations are necessary in order to obtain polynomial time solutions to **NP**–complete problems. The last result is that from [1], where it is proved that two polarizations suffice.

In the literature, P systems with active membranes have been successfully used to design (uniform) solutions to well-known **NP**–complete problems, such as SAT [12], *Subset Sum* [9], *Knapsack* [10], *Bin Packing* [11], *Partition* [3], and the *Common Algorithmic Problem* [13].

The present paper can be considered as a contribution to the interesting problem of characterizing the tractability in terms of descriptional resources required in membrane systems.

Specifically, in the framework of recognizer P systems with membrane division but not using polarizations, we prove the following: (a) the class of problems which can be solved in a polynomial time by a family of such P systems *without dissolution* is equal to class **P**, and (b) the class of problems which can be solved in a polynomial time by a family of such P systems *with dissolution* contains the class **NP**. Hence, we show a surprising role of the –apparently "innocent"– operation of membrane dissolution, as it makes the difference between efficiency and non–efficiency for polarizationless P systems with membrane division.

The paper is organized as follows. In the next section some preliminary ideas about recognizer membrane systems and polynomial complexity classes are introduced. In Section 3 we present a characterization of the class **P** through the polynomial complexity class associated with recognizer P systems with active membranes, without polarization and without dissolution. In Section 4 we show that every **NP**–complete problem can be solved in a semi–uniform way by families of recognizer P systems using membrane dissolution rules and division for elementary and non–elementary membranes. Conclusions and some final remarks are given in Section 5.

2 Preliminaries

2.1 Recognizer P Systems

In the structure and functioning of a cell, biological *membranes* play an essential role. The cell is separated from its environment by means of a *skin membrane*, and it is internally compartmentalized by means of *internal membranes*.

The main *syntactic* ingredients of a cell–like membrane system (P system) are the *membrane structure*, the *multisets*, and the *evolution rules*.

- A *membrane structure* consists of several membranes arranged hierarchically inside a main membrane (the *skin*), and delimiting *regions* (the space in–between a membrane and the immediately inner membranes, if any). When a membrane has no membrane inside, it is called *elementary*. A membrane structure can be considered as a rooted tree, where the nodes are called *membranes*, the root is called *skin*, and the leaves are called *elementary membranes*.
- Regions defined by a membrane structure can contain objects, corresponding to chemical substances present in the compartments of a cell. These objects can be described by symbols or by strings of symbols, in such a way that *multisets of objects* are placed in the regions of the membrane structure.
- The objects can evolve according to given *evolution rules*, associated with the regions (hence, with the membranes).

The *semantics* of the cell–like membrane systems is defined through a non–deterministic and synchronous model (a global clock is assumed) as follows:

- A *configuration* of a cell–like membrane system consists of a membrane structure and a family of multisets of objects associated with each region of the

structure. At the beginning, there is a configuration called the *initial configuration* of the system.
- In each time unit a given configuration is transformed in another configuration by applying the evolution rules to the objects placed inside the regions of the configurations, in a non–deterministic, maximally parallel manner (the rules are chosen in a non–deterministic way, and in each region all objects that can evolve must do it). In this way, we get *transitions* from one configuration of the system to the next one.
- A *computation* of the system is a (finite or infinite) sequence of configurations such that each configuration –except the initial one– is obtained from the previous one by a transition.
- A computation which reaches a configuration where no more rules can be applied to the existing objects and membranes, is called a *halting computation.*
- The result of a halting computation is usually defined through the multiset associated with a specific output membrane (or the environment) in the final configuration.

In this paper we use membrane computing as a framework to address the resolution of decision problems. In order to solve this kind of problems and having in mind that solving them is equivalent to recognizing the language associated with them, we consider P systems as *language recognizer* devices.

Definition 1. *A* P system with input *is a tuple* (Π, Σ, i_Π)*, where: (a)* Π *is a P system with working alphabet* Γ*, with* p *membranes labelled with* $1, \ldots, p$*, and initial multisets* $\mathcal{M}_1, \ldots, \mathcal{M}_p$ *associated with them; (b)* Σ *is an (input) alphabet strictly contained in* Γ *and the initial multisets are over* $\Gamma - \Sigma$*; (c)* i_Π *is the label of a distinguished (input) membrane.*

The computations of a P system with input in the form of a multiset over Σ are defined in a natural way, but the initial configuration of (Π, Σ, i_Π) must be the initial configuration of the system Π to which we add the input multiset. More formally,

Definition 2. *Let* (Π, Σ, i_Π) *be a P system with input. Let* Γ *be the working alphabet of* Π*,* μ *the membrane structure, and* $\mathcal{M}_1, \ldots, \mathcal{M}_p$ *the initial multisets of* Π*. Let* m *be a multiset over* Σ*. The* initial configuration of (Π, Σ, i_Π) with input m *is* $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_{i_\Pi} \cup m, \ldots, \mathcal{M}_p)$.

Let (Π, Σ, i_Π) be a P system with input. Let Γ be the working alphabet of Π, μ the membrane structure, and $\mathcal{M}_1, \ldots, \mathcal{M}_p$ the initial multisets of Π. Let m be a multiset over Σ. Then we denote $\mathcal{M}_j^* = \{(a, j) : \ a \in \mathcal{M}_j\}$, for $1 \leq j \leq p$, and $m^* = \{(a, i_\Pi) : \ a \in m\}$.

Let us recall that a decision problem X is a pair (I_X, θ_X) where I_X is a language over a finite alphabet (its elements are called *instances*) and θ_X is a predicate (a total boolean function) over I_X.

Definition 3. *Let* $X = (I_X, \theta_X)$ *be a decision problem. Let* $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ *be a family of P systems with input. A* polynomial encoding *from* X *to* $\mathbf{\Pi}$ *is a pair* (cod, s) *of polynomial time computable functions over* I_X *such that for each*

instance $w \in I_X$, $s(w)$ *is a natural number and* $cod(w)$ *is an input multiset for the system* $\Pi(s(w))$.

Polynomial encodings are stable under polynomial time reductions [12]. More precisely, the following proposition holds.

Proposition 1. *Let* X_1, X_2 *be decision problems. Let* r *be a polynomial time reduction from* X_1 *to* X_2. *Let* (cod, s) *be a polynomial encoding from* X_2 *to* $\mathbf{\Pi}$. *Then* $(cod \circ r, s \circ r)$ *is a polynomial encoding from* X_1 *to* $\mathbf{\Pi}$.

Definition 4. *A* recognizer P system *is a P system with input and external output such that:*

1. *The working alphabet contains two distinguished elements* `yes` *and* `no`.
2. *All computations halt.*
3. *If* $\mathcal{C}$ *is a computation of the system, then either the object* `yes` *or the object* `no` *(but not both) must have been released into the environment, and only in the last step of the computation.*

In recognizer P systems, we say that a computation is an *accepting computation* (respectively, *rejecting computation*) if the object `yes` (respectively, `no`) appears in the environment associated with the corresponding halting configuration.

2.2 Recognizer P Systems with Active Membranes and Without Polarizations

A particularly interesting class of membrane systems are the systems with active membranes, where the membrane division can be used in order to solve computationally hard problems in polynomial or even linear time, by a space–time trade-off.

In this paper we work with a variant of P systems with active membranes that does not use polarizations.

Definition 5. *A P system with active membranes and without polarizations is a P system with* Γ *as working alphabet, with* H *as the finite set of labels for membranes, and where the rules are of the following forms:*

(a) $[\, a \rightarrow u \,]_h$ *for* $h \in H$, $a \in \Gamma$, $u \in \Gamma^*$. *This is an object evolution rule, associated with a membrane labelled with* h: *an object* $a \in \Gamma$ *belonging to that membrane evolves to a string* $u \in \Gamma^*$.

(b) $a\,[\ \ ]_h \rightarrow [\, b \,]_h$ *for* $h \in H$, $a, b \in \Gamma$. *An object from the region immediately outside a membrane labelled with* h *is introduced in this membrane, possibly transformed into another object.*

(c) $[\, a \,]_h \rightarrow b\,[\ \ ]_h$ *for* $h \in H$, $a, b \in \Gamma$. *An object is sent out from membrane labelled with* h *to the region immediately outside, possibly transformed into another object.*

(d) $[\, a \,]_h \rightarrow b$ *for* $h \in H$, $a, b \in \Gamma$: *A membrane labelled with* h *is dissolved in reaction with an object. The skin is never dissolved.*

(e) $[\,a\,]_h \rightarrow [\,b\,]_h\,[\,c\,]_h$ *for* $h \in H$, $a, b, c \in \Gamma$. *An elementary membrane can be divided into two membranes with the same label, possibly transforming some objects.*

These rules are applied according to the following principles:

- All the rules are applied in parallel and in a maximal manner. In one step, one object of a membrane can be used by only one rule (chosen in a non–deterministic way), but any object which can evolve by one rule of any form, must evolve.
- If at the same time a membrane labelled with 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 we suppose that first the evolution rules of type (a) are used, and then the division is produced. Of course, this process takes only one step.
- The rules associated with membranes labelled with h are used for all copies of this membrane. At one step, a membrane can be the subject of *only one* rule of types (b)-(e).

Let us note that in this framework we shall work without cooperation, without priorities, with cell division rules for elementary membranes, and without changing the labels of membranes. But we shall explicitly mention in each case whether we use dissolution or not.

We denote by $\mathcal{AM}^0_{-d}$ (respectively, $\mathcal{AM}^0_{+d}$) the class of all recognizer P systems with active membranes without polarizations and without using dissolution (respectively, using dissolution).

2.3 Polynomial Complexity Classes in Recognizer P Systems

Definition 6. *Let* $X = (I_X, \theta_X)$ *be a decision problem. Let* $\mathbf{\Pi} = (\Pi(w))_{w \in I_X}$ *be a family of recognizer membrane systems without input.*

- $\mathbf{\Pi}$ *is sound with regard to* X *if for each instance of the problem* $w \in I_X$, *if there exists an accepting computation of* $\Pi(w)$, *then* $\theta_X(w) = 1$.
- $\mathbf{\Pi}$ *is complete with regard to* X *if for each instance of the problem* $w \in I_X$, *if* $\theta_X(w) = 1$, *then every computation of* $\Pi(w)$ *is an accepting computation.*

These concepts can be extended to families of recognizer P systems with input membrane.

Definition 7. *Let* $X = (I_X, \theta_X)$ *be a decision problem. Let* $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ *be a family of recognizer P systems with input. Let* (cod, s) *be a polynomial encoding from* X *to* $\mathbf{\Pi}$.

- *We say that the family* $\mathbf{\Pi}$ *is sound with regard to* (X, cod, s) *if the following holds: for each instance of the problem* $w \in I_X$, *if there exists an accepting computation of* $\Pi(s(w))$ *with input* $cod(w)$, *then* $\theta_X(w) = 1$.

– *We say that the family* $\mathbf{\Pi}$ *is complete with regard to* (X, cod, s) *if the following holds: for each instance of the problem* $w \in I_X$*, if* $\theta_X(w) = 1$*, then every computation of* $\Pi(s(w))$ *with input* $cod(w)$ *is an accepting computation.*

The first results about *solvability* of **NP**–complete problems in polynomial time (even linear) by membrane systems were given by Gh. Păun [6], C. Zandron, C. Ferretti and G. Mauri [14], S.N. Krishna and R. Rama [4], and A. Obtulowicz [5] in the framework of P systems that lack an input membrane. Thus, the constructive proofs of such results need to design *one* system for *each* instance of the problem.

This method for solving problems provides a *specific purpose* algorithmic solution in the following sense: if we wanted to follow this approach for solving some decision problem in a laboratory, then the system constructed to solve a concrete instance would be useless when trying to solve another instance.

Now, we formalize these ideas in the following definition.

Definition 8. *Let* $\mathcal{R}$ *be a class of recognizer P systems without input membrane. A decision problem* $X = (I_X, \theta_X)$ *is solvable in polynomial time by a family,* $\mathbf{\Pi} = (\Pi(w))_{w \in I_X}$*, of P systems from* $\mathcal{R}$*, and we denote this by* $X \in \mathbf{PMC}^*_{\mathcal{R}}$*, if:*

– $\mathbf{\Pi}$ *is polynomially uniform by Turing machines, that is, there exists a deterministic Turing machine working in polynomial time which constructs the system* $\Pi(w)$ *from the instance* $w \in I_X$*.*
– $\mathbf{\Pi}$ *is polynomially bounded, that is, there exists a polynomial function* $p(n)$ *such that for each* $w \in I_X$*, all computations of* $\Pi(w)$ *halt in at most* $p(|w|)$ *steps.*
– $\mathbf{\Pi}$ *is sound and complete with regard to* X*.*

Next, we propose to solve a decision problem through a family of P systems constructed in polynomial time by a Turing machine, and verifying that each element of the family processes, in a specified sense, all the instances of *equivalent size*. We say that these solutions are *uniform solutions*.

Definition 9. *Let* $\mathcal{R}$ *be a class of recognizer P systems with input membrane. A decision problem* $X = (I_X, \theta_X)$ *is solvable in polynomial time by a family* $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$*, of P systems from* $\mathcal{R}$*, and we denote this by* $X \in \mathbf{PMC}_{\mathcal{R}}$*, if the following holds:*

– *The family* $\mathbf{\Pi}$ *is polynomially uniform by Turing machines.*
– *There exists a polynomial encoding* (cod, s) *from* I_X *to* $\mathbf{\Pi}$ *such that*
  - *The family* $\mathbf{\Pi}$ *is polynomially bounded with regard to* (X, cod, s)*; that is, there exists a polynomial function* p*, such that for each* $u \in I_X$ *every computation of* $\Pi(s(u))$ *with input* $cod(u)$ *is halting and, moreover, it performs at most* $p(|u|)$ *steps.*
  - *The family* $\mathbf{\Pi}$ *is sound and complete with regard to* (X, cod, s)*.*

It is easy to see that the classes $\mathbf{PMC}^*_{\mathcal{R}}$ and $\mathbf{PMC}_{\mathcal{R}}$ are closed under polynomial–time reduction and complement (see [8] for details).

3 Characterizing the Tractability by Recognizer P Systems with Active Membranes

Let Π be a recognizer P system with active membranes without polarizations and without dissolution. Let R be the set of rules associated with Π.

Each rule can be considered, in a certain sense, as a *dependency* between the object triggering the rule and the object or objects produced by its application.

We can consider a general pattern $(a,h) \to (a_1,h')(a_2,h')\dots(a_s,h')$, for rules of types $(a),(b),(c),(e)$, where:

- The rules of type (a) correspond to the case $h = h'$ and $s \geq 1$.
- The rules of type (b) correspond to the case $h = f(h')$ and $s = 1$.
- The rules of type (c) correspond to the case $h' = f(h)$ and $s = 1$.
- The rules of type (e) correspond to the case $h = h'$ and $s = 2$.

If h is the label of a membrane, then $f(h)$ denotes the label of the father of the membrane labelled with h. We adopt the convention that the father of the skin membrane is the environment (and we denote by *environment* the label associated with the environment of the system).

For example, let us consider a general rule $(a,h) \to (a_1,h')\dots(a_s,h')$. Then we can interpret that from the object a in membrane labelled with h we can *reach* the objects $a_1,\dots,a_s$ in membrane labelled with h'.

Next, we formalize these ideas in the following definition.

Definition 10. *Let Π be a recognizer P system with active membranes without polarizations and without dissolution. Let R be the set of rules associated with Π. The dependency graph associated with Π is the directed graph $G_\Pi = (V_\Pi, E_\Pi)$ defined as follows:*

$V_\Pi = VL_\Pi \cup VR_\Pi$,

$$VL_\Pi = \{(a,h) \in \Gamma \times H : \exists u \in \Gamma^* \ ([a \to u]_h \in R) \vee$$
$$\exists b \in \Gamma \ ([a]_h \to [\ ]_h b \in R) \vee$$
$$\exists b \in \Gamma \ \exists h' \in H \ (h = f(h') \wedge a[\ ]_{h'} \to [b]_{h'} \in R) \vee$$
$$\exists b,c \in \Gamma \ ([a]_h \to [b]_h[c]_h \in R)\},$$

$$VR_\Pi = \{(b,h) \in \Gamma \times H : \exists a \in \Gamma \ \exists u \in \Gamma^* \ ([a \to u]_h \in R \wedge b \in alph(u)) \vee$$
$$\exists a \in \Gamma \ \exists h' \in H \ (h = f(h') \wedge [a]_{h'} \to [\ ]_{h'} b \in R) \vee$$
$$\exists a \in \Gamma \ (a[\ ]_h \to [b]_h \in R) \vee$$
$$\exists a,c \in \Gamma \ ([a]_h \to [b]_h[c]_h \in R)\},$$

$$E_\Pi = \{((a,h),(b,h')) : \exists u \in \Gamma^* \ ([a \to u]_h \in R \wedge b \in alph(u) \wedge h = h') \vee$$
$$([a]_h \to [\ ]_h b \in R \ \wedge \ h' = f(h)) \vee$$
$$(a[\ ]_{h'} \to [b]_{h'} \in R \ \wedge h \ = f(h')) \vee$$
$$\exists c \in \Gamma \ ([a]_h \to [b]_h[c]_h \in R \wedge h = h')\}.$$

Proposition 2. *Let Π be a recognizer P system with active membranes without polarizations and without dissolution. Let R be the set of rules associated with Π. There exists a Turing machine that constructs the dependency graph associated with Π, G_Π, in polynomial time (that is, in a time bounded by a polynomial function depending on the total number of rules and the maximum length of the rules).*

Proof. A deterministic algorithm that, given a P system Π with the set R of rules, constructs the corresponding dependency graph, is the following:

```
Input: Π (with R as its set of rules)
V_Π ← ∅;  E_Π ← ∅
for each rule r ∈ R of Π do
  if r = [a → u]_h ∧ alph(u) = {a_1, ..., a_s} then
    V_Π ← V_Π ∪ ⋃_{j=1}^{s} {(a,h),(a_j,h)};  E_Π ← E_Π ∪ ⋃_{j=1}^{s} {((a,h),(a_j,h))}
  if r = [a]_h → [ ]_h b then
    V_Π ← V_Π ∪ {(a,h),(b,f(h))};
    E_Π ← E_Π ∪ {((a,h),(b,f(h)))}
  if r = a[ ]_h → [b]_h then
    V_Π ← V_Π ∪ {(a,f(h)),(b,h)};
    E_Π ← E_Π ∪ {((a,f(h)),(b,h))}
  if r = [a]_h → [b]_h[c]_h then
    V_Π ← V_Π ∪ {(a,h),(b,h),(c,h)};
    E_Π ← E_Π ∪ {((a,h),(b,h)),((a,h),(c,h))}
```

The running time of this algorithm is bounded by $O(|R| \cdot q)$, where q is the value $\max\{length(r) : \ r \in R\}$.

Proposition 3. *Let $\Pi = (\Gamma, \Sigma, H, \mathcal{M}_1, \ldots, \mathcal{M}_p, R_1, \ldots, R_p, i_\Pi)$ be a recognizer P system with active membranes without polarizations and without dissolution. Let Δ_Π be defined as follows:*

$$\Delta_\Pi = \{(a,h) \in \Gamma \times H : \textit{there exists a path (within the dependency graph) from } (a,h) \textit{ to } (\texttt{yes}, environment)\}.$$

Then, there exists a Turing machine that constructs the set Δ_Π in polynomial time (that is, in a time bounded by a polynomial function depending on the total number of rules and the maximum length of the rules).

Proof. We can construct the set Δ_Π from Π as follows:

- We construct the dependency graph G_Π associated with Π.
- Then we consider the following algorithm:

```
Input: G_Π = (V_Π, E_Π)
  Δ_Π ← ∅
  for each (a, h) ∈ V_Π do
    if reachability (G_Π, (a, h), (yes, environment)) = yes then
      Δ_Π ← Δ_Π ∪ {(a, h)}
```

The running time of this algorithm is of the order $O(|V_\Pi| \cdot |V_\Pi|^2)$, hence[1] it is of the order $O(|\Gamma|^3 \cdot |H|^3)$.

Next, given a family of recognizer P systems solving a decision problem, we will characterize the acceptance of an instance of the problem, w, using the set $\Delta_{\Pi(s(w))}$ associated with the system $\Pi(s(w))$, that processes the given instance w. More precisely, the instance is accepted by the system if and only if there is an object in the initial configuration of the system $\Pi(s(w))$ with input $cod(w)$ such that there exists a path in the associated dependency graph starting from that object and reaching the object `yes` in the environment.

Proposition 4. *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}}$ be a family of recognizer P systems with input membrane solving X, according to Definition 9. Let (cod, s) be the polynomial encoding associated with that solution. Then, for each instance w of the problem X the following assertions are equivalent:*

(a) $\theta_X(w) = 1$ (that is, the answer to the problem is `yes` *for w).*

(b) $\Delta_{\Pi(s(w))} \cap ((cod(w))^* \cup \bigcup_{j=1}^{p} \mathcal{M}_j^*) \neq \emptyset$, where $\mathcal{M}_1, \ldots, \mathcal{M}_p$ are the initial multisets of the system $\Pi(s(w))$.

[1] The Reachability Problem is the following: *given a (directed or undirected) graph, G, and two nodes a, b, determine whether or not the node b is reachable from a, that is, whether or not there exists a path in the graph from a to b.* It is easy to design an algorithm running in polynomial time solving this problem. For example, given a (directed or undirected) graph, G, and two nodes a, b, we consider a depth–first–search with source a, and we check if b is in the tree of the computation forest whose root is a. The total running time of this algorithm is $O(|V| + |E|)$, that is, in the worst case is quadratic in the number of nodes. Morover, this algorithm needs to store a linear number of items (it can be proved that there exists another polynomial time algorithm which uses $O(\log^2(|V|))$ space).

Proof. Let $w \in I_X$. Then $\theta_X(w) = 1$ if and only if there exists an accepting computation of the system $\Pi(s(w))$ with input multiset $cod(w)$. But this condition is equivalent to the following: in the initial configuration of $\Pi(s(w))$ with input multiset $cod(w)$ there exists at least one object $a \in \Gamma$ in a membrane labelled with h such that in the dependency graph the node $(\texttt{yes}, environment)$ is reachable from (a, h).

Hence, $\theta_X(w) = 1$ if and only if $\Delta_{\Pi(s(w))} \cap \mathcal{M}_j^* \neq \emptyset$ for some $j \in \{1, \ldots, p\}$, or $\Delta_{\Pi(s(w))} \cap (cod(w))^* \neq \emptyset$.

Theorem 1. $\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0_{-d}}$.

Proof. We have $\mathbf{P} \subseteq \mathbf{PMC}_{\mathcal{AM}^0_{-d}}$ because the class $\mathbf{PMC}_{\mathcal{AM}^0_{-d}}$ is closed under polynomial time reduction. Next, we show that $\mathbf{PMC}_{\mathcal{AM}^0_{-d}} \subseteq \mathbf{P}$. Let $X \in \mathbf{PMC}_{\mathcal{AM}^0_{-d}}$ and let $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbf{N}}$ be a family of recognizer P systems with input membrane solving X, according to Definition 9. Let (cod, s) be the polynomial encoding associated with that solution.

We consider the following deterministic algorithm:

```
Input: An instance w of X
  - Construct the system Π(s(w)) with input multiset cod(w).
  - Construct the dependency graph G_Π(s(w)) associated with Π(s(w)).
  - Construct the set Δ_Π(s(w)) as indicated in Proposition 3
    answer ← no; j ← 1
    while j ≤ p ∧ answer = no do
        if Δ_Π(s(w)) ∩ M_j^* ≠ ∅ then
            answer ← yes
        j ← j + 1
    endwhile
    if Δ_Π(s(w)) ∩ (cod(w))^* ≠ ∅ then
        answer ← yes
```

On one hand, the answer of this algorithm is `yes` if and only if there exists a pair (a, h) belonging to $\Delta_{\Pi(s(w))}$ such that the symbol a appears in the membrane labelled with h in the initial configuration (with input the multiset $cod(w)$).

On the other hand, a pair (a, h) belongs to $\Delta_{\Pi(s(w))}$ if and only if there exists a path from (a, h) to $(\texttt{yes}, environment)$, that is, if and only if we can obtain an accepting computation of $\Pi(s(w))$ with input $cod(w)$. Hence, the algorithm above described solves the problem X.

The cost to determine whether or not $\Delta_{\Pi(s(w))} \cap \mathcal{M}_j^* \neq \emptyset$ (or $\Delta_{\Pi(s(w))} \cap (cod(w))^* \neq \emptyset$) is of the order $O(|\Gamma|^2 \cdot |H|^2)$.

Hence, the running time of this algorithm can be bounded by $f(|w|) + O(|R| \cdot q) + O(p \cdot |\Gamma|^2 \cdot |H|^2)$, where f is the (total) cost of a polynomial encoding from X to $\mathbf{\Pi}$, R is the set of rules of $\Pi(s(w))$, H is the set of labels for membranes

of $\Pi(s(w))$, p is the number of (initial) membranes of $\Pi(s(w))$, and $q = \max\{length(r) : r \in R\}$. But from Definition 9 we have that all involved parameters are polynomials in $|w|$. That is, the algorithm is polynomial in the size $|w|$ of the input.

Now, we consider *division rules for non–elementary membranes*, that is, rules of the following form $[\,[\,]_{h_1}[\,]_{h_2}\,]_{h_0} \rightarrow [\,[\,]_{h_1}\,]_{h_0}\,[\,[\,]_{h_2}\,]_{h_0}$, where h_0, h_1, h_2 are labels: if the membrane with label h_0 contains other membranes than those with labels h_1, h_2, then such membranes and their contents are duplicated and placed in both new copies of the membrane h_0; all membranes and objects placed inside membranes h_1, h_2, as well as the objects from membrane h_0 placed outside membranes h_1 and h_2, are reproduced in the new copies of membrane h_0. We denote by $\mathcal{AM}^0_{-d,+ne}$ the class of all recognizer P systems with active membranes without polarization, without membrane dissolution rules, and using division rules for elementary and non–elementary membranes.

If $\Pi \in \mathcal{AM}^0_{-d,+ne}$, then we define the dependency graph associated with Π as the directed graph G_Π from Definition 10, that is, the division rules for non–elementary membranes do not add any node or edge to the dependency graph.

Then, the proof of Theorem 1 provides the following result:

Theorem 2. $\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0_{-d,+ne}}$.

Now, we study similar characterizations of $\mathbf{P}$ dealing with semi–uniform solutions in the framework of recognizer P systems with active membranes without polarizations and without dissolution.

Proposition 5. *Let $X = (I_X, \theta_X)$ be a decision problem. Let $\mathbf{\Pi} = (\Pi(w))_{w \in I_X}$ be a family of recognizer P systems without input membrane solving X, according to Definition 8. Then, for each instance w of the problem X the following assertions are equivalent:*

(a) $\theta_X(w) = 1$ (that is, the answer to the problem is `yes` *for w).*

(b) $\Delta_{\Pi(w)} \cap (\bigcup_{j=1}^{p} \mathcal{M}_j^*) \neq \emptyset$, where $\mathcal{M}_1, \ldots, \mathcal{M}_p$ are the initial multisets of the system $\Pi(w)$.

Proof. Let $w \in I_X$. Then $\theta_X(w) = 1$ if and only if there exists an accepting computation of the system $\Pi(w)$. But this condition is equivalent to the following: in the initial configuration of $\Pi(w)$ there exists an object $a \in \Gamma$ in a membrane labelled with h such that in the dependency graph the node $(\texttt{yes}, environment)$ is reachable from (a, h).

Hence, $\theta_X(w) = 1$ if and only if $\Delta_{\Pi(w)} \cap \mathcal{M}_j^* \neq \emptyset$ for some $j \in \{1, \ldots, p\}$.

Theorem 3. $\mathbf{P} = \mathbf{PMC}^*_{\mathcal{AM}^0_{-d}}$.

Proof. The proof of this result is analogous to the proof of Theorem 1, taking into account that in this case we are dealing with a semi-uniform solution. That is, in the previous theorem an instance $w \in I_X$ was processed by the P system $\Pi(s(w))$ with input $cod(w)$, and now such instance is processed by $\Pi(w)$.

Bearing in mind that division rules for non–elementary membranes do not influence the construction of the dependency graph, we have:

Theorem 4. $\mathbf{P} = \mathbf{PMC}^{*}_{\mathcal{AM}^{0}_{-d,+ne}}$.

We can consider a three dimensional representation of the above theorems, where $+u$ (respectively, $-u$) stands for uniform (respectively, semi–uniform) solutions.

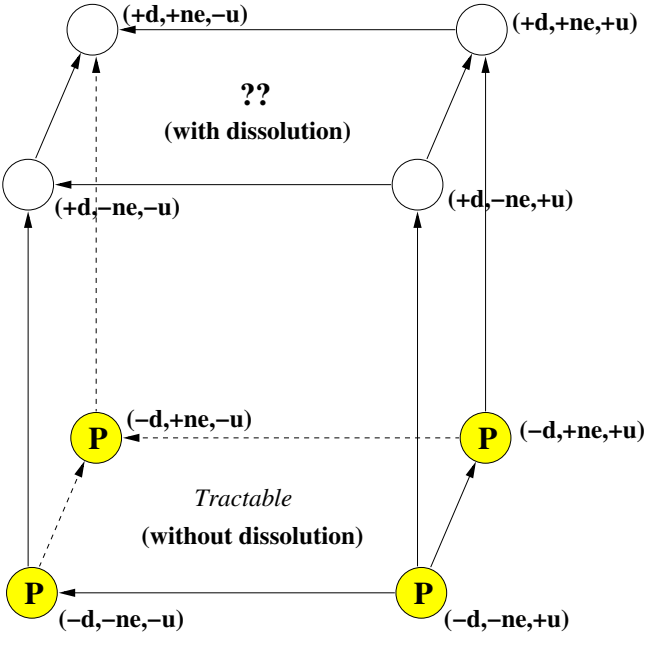

Fig. 1. Characterizations of **P** by P systems

What happens if we consider dissolution rules in the framework of recognizer P systems with active membranes and without polarizations? Will it be possible to solve **NP**–complete problems in that framework? In the next section we will affirmatively answer to this question.

4 Computational Efficiency Using Dissolution Rules

In this section we show that the class of decision problems solvable in polynomial time in a semi–uniform way by families of recognizer P systems with active membranes, without polarization, and using membrane dissolution rules and division rules for elementary and non–elementary membranes, contains the standard complexity class **NP**.

For that, we describe a family of such recognizer membrane systems which solves the Subset Sum problem in linear time and in a semi–uniform way.

The Subset Sum problem is the following one: *Given a finite set A, a weight function, $w : A \to \mathbf{N}$, and a constant $k \in \mathbf{N}$, determine whether or not there exists a subset $B \subseteq A$ such that $w(B) = k$.*

Proposition 6. *The Subset Sum problem belongs to the class* $\mathbf{PMC}^{*}_{\mathcal{AM}^{0}_{+d,+ne}}$.

Sketch of the Proof. We will use a tuple $u = (n, (w_1, \ldots, w_n), k)$ to represent an instance of the problem, where n stands for the size of $A = \{a_1, \ldots, a_n\}$, $w_i = w(a_i)$, and k is the constant given as input for the problem.

We propose here a solution to this problem based on a brute force algorithm implemented in the framework of P systems with active membranes, without polarizations, with dissolution, and using division for elementary and non–elementary membranes.

The idea of the design is better understood if we divide the solution to the problem into several stages:

- *Generation stage*: for every subset of A, a membrane is generated via membrane division.
- *Weight calculation stage*: in each membrane the weight of the associated subset is calculated. This stage will take place in parallel with the previous one.
- *Checking stage*: for each membrane it is checked whether or not the weight of its associated subset is exactly k. This stage cannot start before the previous ones are over.
- *Output stage*: when the previous stage has been completed in all membranes, the system sends out the answer to the environment.

For each instance $u = (n, (w_1, \ldots, w_n), k)$ of the Subset Sum problem we consider the P system with active membranes, without polarization, without input membrane

$\Pi(u) = (\Gamma(u), H(u), \mu, \mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_{k+3}, R(u))$

defined as follows:

- Working alphabet:

$$\Gamma(u) = \{d_0, \ldots, d_{2n+1}, a_1, \ldots, a_n, e_1, \ldots, e_n\} \cup \{b, s, c, \underline{c}, z_0, \ldots, z_{2n+k+5}, \texttt{yes}, \texttt{no}\}.$$

- $H(u) = \{0, 1, 2, 3, \ldots, k+3\}$.
- Initial membrane structure: $\mu = [\,[\,[\,[\ldots[\,[\,]_0]_1 \ldots]_k]_{k+1}]_{k+2}]_{k+3}$.
- Initial multisets:
  $\mathcal{M}_0 = d_0$, $\mathcal{M}_{k+2} = z_0$ and $\mathcal{M}_i = \emptyset$, for every $i \in \{1, \ldots, k, k+1, k+3\}$.
- The set of evolution rules, $R(u)$, consists of the following rules:

(a) $[d_{2i} \to a_{i+1} d_{2i+1}]_0$ for $i \in \{0, \ldots, n\}$,
$[d_{2i+1} \to d_{2i+2}]_0$ for $i \in \{0, \ldots, n-1\}$.

The goal of the counter d_i is to control the apparition of an object a_j only in the odd steps. The importance of these objects will be explained in the next set of rules.

(b) $$\left.\begin{array}{l}[a_i]_0 \to [e_i]_0\,[b]_0\\ [e_i \to s^{w_i}]_0\end{array}\right\} \text{ for } i \in \{1,\ldots,n\}.$$

The object a_i triggers the rule for division of elementary membranes. After the division, in one membrane is placed an object e_i and in the other one an object b. The object b remains inactive whereas the object e_i evolves in the next step to as many copies of object s as the weight w_i.

(c) $$[\,[\,]_i\,]_i]_{i+1} \to [\,[\,]_i\,]_{i+1}\,[\,[\,]_i\,]_{i+1} \qquad \text{for } i \in \{1,\ldots,k\}.$$

This is the set of rules for the division of non-elementary membranes. These three first set of rules produce a membrane structure with 2^n branches. On each of the leaves of the tree we have a membrane with as many objects s as the weight of a possible subset, S, of A.

(d) $$[d_{2n+1}]_0 \to b,$$
$$[s]_i \to c \qquad \text{for } i \in \{1,\ldots,k+1\}.$$

When the generation stage has finished, the object d_{2n+1} dissolves the membrane with label 0. At this point, the elements s start to dissolve membranes. If there are enough objects s, all the membranes of the branch with labels $1,\ldots,k+1$ are dissolved. Otherwise, the branch remains inactive.

(e) $$[c \to \underline{c}]_{k+1}.$$

This is a waiting step and the key of the computation. If in a branch the encoded weight of the subset, w_S, is less than k, the branch becomes inactive. Otherwise all the membranes of the branch are dissolved until reaching the membrane with label $k+1$. If $w_S = k$ then in this membrane there are no objects s that dissolve it and the object $\underline{c}$ remains in the membrane. On the contrary, if $w_S > k$, then the membrane is dissolved in the same step in which $\underline{c}$ is produced and $\underline{c}$ goes to the membrane with label $k+2$.

(f) $$[z_i \to z_{i+1}]_{k+2} \qquad \text{for } i \in \{0,\ldots,2n+k+4\},$$
$$[\underline{c}]_{k+1} \to \texttt{yes},$$
$$[yes]_{k+2} \to \texttt{yes},$$
$$[z_{2n+k+5}]_{k+2} \to \texttt{no}.$$

If one of the subsets of A has weight k, then an object $\underline{c}$ appears in a membrane with label $k+1$. This object dissolves the membrane and sends an object `yes` to the membrane with label $k+2$. In this membrane we keep a counter z_i along the computation. If at some step an object $\underline{c}$ has sent an object `yes` to this membrane, this object will dissolve the membrane in the next step preventing that the object z_{2n+k+5} appears in the membrane. Otherwise, if the object $\underline{c}$ is never produced, then we eventually get an object z_{2n+k+5} in the membrane with label $k+2$. In the following step this membrane is dissolved and an element `no` is sent to the membrane with label $k+3$.

(g) $$[\texttt{no}]_{k+3} \to \texttt{no}\,[\,]_{k+3},$$
$$[\texttt{yes}]_{k+3} \to \texttt{yes}\,[\,]_{k+3}.$$

From above, we know that the membrane with label $k+3$ (recall that this is the skin membrane) is reached by one and only one of the objects `yes` or `no`. The rules in group (g) send that object to the environment in the last step of the computation. □

Theorem 5. $\mathbf{NP} \cup \mathbf{co\text{-}NP} \subseteq \mathbf{PMC}^{*}_{\mathcal{AM}^{0}_{+d,+ne}}$.

Proof. It suffices to remark that the Subset Sum problem is **NP**–complete, belonging to the class $\mathbf{PMC}^{*}_{\mathcal{AM}^{0}_{+d,+ne}}$, and this class is stable under polynomial-time reduction and closed under complement.

Remark 1. A. Alhazov et al. in [2] showed that $\mathbf{SAT} \in \mathbf{PMC}^{*}_{\mathcal{AM}^{0}_{+d,+ne}}$. Hence the result in Theorem 5 can also be deduced from this remark.

The following picture illustrates the results obtained in this paper.

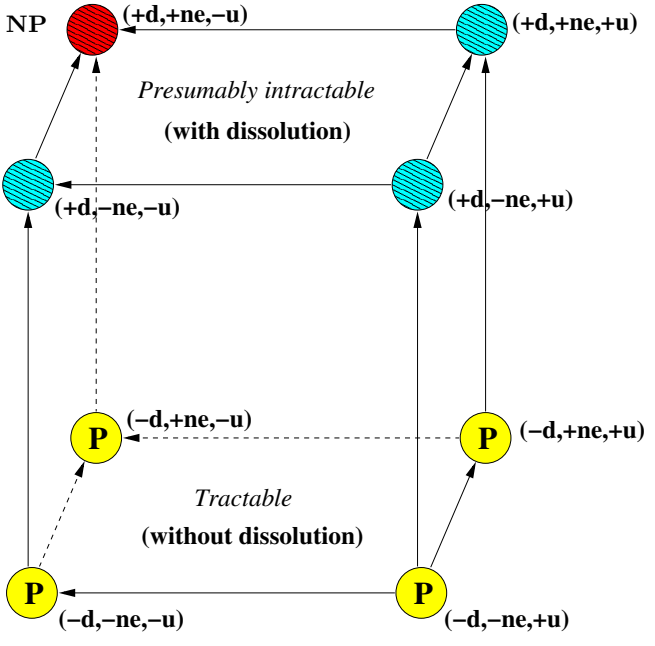

Fig. 2. A borderline between the tractability and the (presumable) intractability

5 Conclusions

A conjecture known in the membrane computing area under the name of the P–conjecture asserts that the polynomial–time solvability by deterministic Turing

machines is equivalent to the polynomial–time solvability by recognizer P systems with active membranes and without polarizations, that is, that conjecture can be expressed by the equality $\mathbf{P} = \mathbf{PMC}_{\mathcal{AM}^0}$, where $\mathcal{AM}^0$ is the class of all recognizer P systems with active membranes and without polarization.

In this paper we provide a *partial affirmative* answer to the P–conjecture in the case that the P systems from $\mathcal{AM}^0$ do not use dissolution rules. Besides, a *partial negative* answer to the P–conjecture is given when we use semi–uniform solutions and membrane division rules for elementary and non–elementary membranes (and suppossing that $\mathbf{P} \neq \mathbf{NP}$).

We have used the concept of *dependency graph* that initially was defined to help to design strategies that allow to choose short computations of recognizer membrane systems. In this paper we work with dependency graphs associated with a variant of recognizer P systems with active membranes. In this way we are able to characterize accepting computations of these systems through the reachability of a distinguished node of the graph from other nodes associated with the initial configuration.

We have shown that it is possible to solve in polynomial time and in a uniform way through recognizer P systems with active membranes without polarizations and without dissolution *only* problems which are tractable in the standard sense. Morover, if in this framework we consider membrane dissolution rules, then we can solve **NP**–complete problems in polynomial time, in a semi–uniform way and using division for elementary and non–elementary membranes.

Acknowledgement

The authors wish to acknowledge the support of the project TIC2002-04220-C03-01 of the Ministerio de Ciencia y Tecnología of Spain, cofinanced by FEDER funds.

References

1. A. Alhazov, R. Freund, Gh. Păun: P systems with active membranes and two polarizations. *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini, eds.), Report RGNC 01/04, 2004, 20–35.
2. A. Alhazov, L. Pan, Gh. Păun: Trading polarizations for labels in P systems with active membranes. *Acta Informaticae*, 41, 2-3 (2004), 111-144.
3. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez: A fast P system for finding a balanced 2-partition. *Soft Computing*, 9, 9(2005), 673–678.
4. S.N. Krishna, R. Rama: A variant of P systems with active membranes: Solving NP–complete problems. *Romanian Journal of Information Science and Technology*, 2, 4 (1999), 357–367.
5. A. Obtulowicz: Deterministic P systems for solving SAT problem. *Romanian Journal of Information Science and Technology*, 4, 1–2 (2001), 551–558.
6. Gh. Păun: P systems with active membranes: Attacking **NP**–complete problems. *Journal of Automata, Languages and Combinatorics*, 6, 1 (2001), 75–90.

7. Gh. Păun: Computing with membranes: Attacking **NP**–complete problems. In *Unconventional Models of Computation, UMC'2K* (I. Antoniou, C. Calude, M.J. Dinneen, eds.), Springer–Verlag, 2000, 94–115.
8. M.J. Pérez–Jiménez: An approach to computational complexity in Membrane Computing. In *Membrane Computing, 5th International Workshop, WMC5, Revised Selected and Invited Papers* (G. Mauri, Gh. Păun, M. J. Pérez-Jiménez, Gr. Rozenberg, A. Salomaa, eds.), LNCS 3365 (2005), 85-109.
9. M.J. Pérez-Jiménez, A. Riscos-Núñez: Solving the Subset-Sum problem by active membranes. *New Generation Computing*, 23, 4(2005), 367–384.
10. M.J. Pérez-Jiménez, A. Riscos-Núñez: A linear–time solution to the Knapsack problem using P systems with active membranes. In *Membrane Computing* (C. Martín-Vide, Gh. Păun, G. Rozenberg, A. Salomaa, eds.), LNCS 2933 (2004), 250–268.
11. M.J. Pérez-Jiménez, F.J. Romero-Campero: Solving the Bin Packing problem by recognizer P systems with active membranes. *Proceedings of the Second Brainstorming Week on Membrane Computing* (Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini, eds.), Report RGNC 01/04, University of Seville, 2004, 414–430.
12. M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini: A polynomial complexity class in P systems using membrane division. *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems, DCFS 2003* (E. Csuhaj-Varjú, C. Kintala, D. Wotschke, G. Vaszil, eds.), 2003, 284-294.
13. M.J. Pérez-Jiménez, F.J. Romero–Campero: Attacking the Common Algorithmic Problem by recognizer P systems. In *Machines, Computations and Universality, MCU'2004, Saint Petesburg, Russia, September 2004, Revised Selected Papers* (M. Margenstern, ed.), LNCS 3354 (2005), 304-315.
14. C. Zandron, C. Ferreti, G. Mauri: Solving NP-complete problems using P systems with active membranes. In *Unconventional Models of Computation, UMC'2K* (I. Antoniou, C. Calude, M.J. Dinneen, eds.), Springer–Verlag, 2000, 289–301.