New Generation Computing, 23(2005)339-356
Ohmsha, Ltd. and Springer

NEW GENERATION COMPUTING

Solving the Subset-Sum Problem by P Systems with Active Membranes

Mario J. PÉREZ JIMÉNEZ and Agustín RISCOS NÚÑEZ
Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. Ingeniería Informática, Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, España
{Mario.Perez, Agustin.Riscos-Nunez} @cs.us.es

Received 2 October 2003
Revised manuscript received 8 December 2004

Abstract We present the first membrane computing solution to the Subset-Sum problem using a family of deterministic P systems with active membranes. We do not use priority among rules, membrane dissolution nor cooperation; it suffices to control the electrical charges of the membranes and to introduce some counters. The number of steps of any computation is of the linear order (but it is necessary a polynomial-time of precomputed resources).

Keywords: Membrane Computing, Complexity Classes, Active Membranes, Subset-Sum Problem.

§1 Introduction

In Reference,[9] an unconventional distributed parallel model of computation is introduced, starting from the observation that the processes which take place in the complex structure of a living cell can be considered as computations. The devices of this model are called P systems. A detailed introduction to the area of P systems can be found at Reference,[11] and a survey and an updated bibliography can be found at the web address `http://psystems.disco.unimib.it`.

Up to now, P systems dealing with numerical problems have been rarely considered in the literature. The present paper is a first approach in this direction. Our claim is that, once we have seen the construction and verification of some families of P systems solving a few numerical **NP**–complete problems, we will be perhaps able to extract a common scheme that could be the basis for attacking new numerical problems.

We present here a membrane computing solution to the Subset-Sum problem, and we analyze it from the point of view of the complexity classes. A *com-*

plexity class for a model of computation is a collection of problems that can be solved (or languages that can be decided) by some devices of this model with *similar* computational resources.

In this paper we present a *polynomial complexity class* in cellular computing with membranes inspired by some ideas of Gh. Păun (Reference,[11] Section 7.1) discussed with some members of the Research Group on Natural Computing from the University of Seville. This class allows us to detect some intrinsic difficulties of the resolution of a problem in the model above mentioned.

The paper is organized as follows. In the next section we define recognizer P systems and also P systems with active membranes, which will be the models considered to study the complexity classes. Section 3 defines the polynomial complexity class $\mathbf{PMC}_{\mathcal{F}}$, associated with a collection $\mathcal{F}$ of P systems. In Sections 4 and 5 a cellular solution to the Subset-Sum problem is presented. Section 6 is devoted to study the computational complexity of this solution. The main results and conclusions are given in Sections 7 and 8, respectively.

We work in this paper with membrane systems using symbol–objects.

§2 Preliminaries

Recall that a decision problem, X, is a pair (I_X, θ_X) such that I_X is a language over a finite alphabet (whose elements are called *instances*) and θ_X is a total boolean function over I_X.

2.1 Recognizer P Systems

Definition 2.1
A P system with input is a tuple (Π, Σ, i_Π), where:

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

Definition 2.2
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 $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_{i_\Pi} \cup m, \ldots \mathcal{M}_p)$.

Remark 2.1
We will denote by I_Π the set of all inputs of the P system Π. That is, I_Π is a collection of multisets over Σ.

The computations of a P system with input a multiset m over Σ are defined in a natural way. The only novelty is that the initial configuration must be the initial configuration of the system associated with the input multiset m.

In the case of P systems with input and with external output, the computation is defined in a similar way, with a slight difference: in the configurations we will not work directly with the membrane structure μ but with another structure associated with it, including, in some sense, the environment.

Definition 2.3
Let $\mu = (V(\mu), E(\mu))$ be a membrane structure. The *membrane structure with environment* associated with μ is the rooted tree $Ext(\mu)$ such that: (a) the root of the tree is a new node that we will denote env; (b) the set of nodes is $V(\mu) \cup \{env\}$; and (c) the set of edges is $E(\mu) \cup \{\{env, skin\}\}$. The node env is called the *environment* of the structure μ.

Note that we have added a new node representing the environment which is only connected with the skin, while the original membrane structure remains unchanged. In this way, every configuration of the system contains information about the contents of the environment.

Definition 2.4
A *recognizer* P system is a P system with input (Π, Σ, i_Π), and with external output such that:

1. The working alphabet contains two distinguished elements: Yes, No.
2. All computations halt.
3. If $\mathcal{C}$ is a computation of Π, then either the object Yes or the object No (but not both) have to be sent out to the environment, and only in the last step of the computation.

Definition 2.5
We say that $\mathcal{C}$ is an *accepting* computation (respectively, *rejecting* computation) if the object Yes (respectively, No) appears in the environment associated with the corresponding halting configuration of $\mathcal{C}$.

2.2 P Systems with Active Membranes

In the basic P systems (chapter 3 from Reference[11]), the membrane structure is supposed to be fixed. A natural variant is to let the number of membranes decrease (by membrane dissolution) or increase (by membrane division). This idea was explored by Gh. Păun[10] and it gave rise to a new model: *P systems with active membranes*.

As it has been proved that transition P systems without membrane division are an universal model of computation, we can not increase their computational power. However, allowing membrane division produces a significant speed-up of computations (in Reference[12] it is shown that if $\mathbf{P} \neq \mathbf{NP}$, then a deterministic P system without membrane division is not able to solve any **NP**–complete problem in polynomial time). This speed-up can be specially relevant if we are dealing with a real world problem (see Reference[3] for an algorithm breaking DES).

Next, we are going to introduce the definition of P systems with active membranes. We consider only 2-division for elementary membranes, and we do not use cooperation nor priority among rules.

Definition 2.6
A P system with active membranes is a tuple $\Pi = (\Gamma, H, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_p, R)$, where:

1. $p \geq 1$, is the initial degree of the system;
2. Γ is the alphabet of symbol-objects;
3. H is a finite set of labels for membranes;
4. μ is a membrane structure, of p membranes, labelled (not necessarily in a one-to-one manner) by elements of H;
5. $\mathcal{M}_1, \ldots, \mathcal{M}_p$ are strings over Γ, describing the initial multisets of objects placed in the p regions of μ;
6. R is a finite set of evolution rules, of the following forms:

   (a) $[\, a \to \omega\,]_h^\alpha$ for $h \in H, \alpha \in \{+, -, 0\}$, $a \in \Gamma$, $\omega \in \Gamma^*$, *object evolution rules*: This is an object evolution rule, associated with a membrane labelled by h and depending on the polarity of that membrane, but not directly involving the membrane.

   (b) $a\,[\;]_h^{\alpha_1} \to [\, b\,]_h^{\alpha_2}$ for $h \in H$, $\alpha_1, \alpha_2 \in \{+, -, 0\}$, $a, b \in \Gamma$, *communication rules (send in rules)*: An object from the region immediately outside a membrane labelled by h is introduced in this membrane, possibly transformed into another object, and simultaneously, the polarity of the membrane can be changed.

   (c) $[\, a\,]_h^{\alpha_1} \to b\,[\;]_h^{\alpha_2}$ for $h \in H$, $\alpha_1, \alpha_2 \in \{+, -, 0\}$, $a, b \in \Gamma$, *communication rules (send out rules)*: An object is sent out from membrane labelled by h to the region immediately outside, possibly transformed into another object, and simultaneously, the polarity of the membrane can be changed.

   (d) $[\, a\,]_h^\alpha \to b$ for $h \in H$, $\alpha \in \{+, -, 0\}$, $a, b \in \Gamma$, *disolving rules*: A membrane labelled by h is dissolved in reaction with an object. The skin is never dissolved.

   (e) $[\, a\,]_h^{\alpha_1} \to [\, b\,]_h^{\alpha_2}\, [\, c\,]_h^{\alpha_3}$ for $h \in H$, $\alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}$, $a, b, c \in \Gamma$, *division rules for elementary membranes*: An elementary membrane can be divided into two membranes with the same label, possibly transforming some objects and their polarities.

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, should evolve.
- If a membrane is dissolved, its content (multiset and internal membranes) is left free in the surrounding region.
- 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 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 by h are used for all copies of this membrane. At one step, a membrane labelled by h can be the subject of *only one* rule of types (b)-(e).

Let us denote by $\mathcal{AM}$ the class of recognizer P systems with active membranes using 2-division for elementary membranes.

We would like to note that we are not using dissolution rules in the present paper.

§3 The Complexity Class $\mathrm{PMC}_{\mathcal{F}}$

The first results about "solvability" of **NP**–complete problems in polynomial time (even linear) by cellular computing systems with membranes were obtained using variants 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 (see for instance Reference[10] or Reference[12]).

If we wanted to perform such a solution of some decision problem in a laboratory, we will find a drawback on this approach: a system constructed to solve a concrete instance is useless when trying to solve another instance. This handicap can be easily overtaken if we consider a P system with input. Then, the same system could solve different instances of the problem, provided that the corresponding input multisets are introduced in the input membrane.

Therefore, when attacking a problem in the Cellular Computing framework, we will design P systems that are able to decide all the instances of "equivalent size", in certain sense.

Definition 3.1
Let L be a language, $\mathcal{F}$ a class of P systems with input, and $\mathbf{\Pi} = (\Pi(t))_{t\in\mathbf{N}}$ a family of P systems from $\mathcal{F}$. A polynomial encoding of L in $\mathbf{\Pi}$ is a pair (g, h) of polynomial-time computable functions, $g : L \to \bigcup_{t\in\mathbf{N}} I_{\Pi(t)}$, and $h : L \to \mathbf{N}$ such that for every $u \in L$ we have $g(u) \in I_{\Pi(h(u))}$.

Lemma 3.1
Let $L_1 \subseteq \Gamma_1^*$ and $L_2 \subseteq \Gamma_2^*$ be languages. Let $\mathcal{F}$ be a class of P systems with input, and $\mathbf{\Pi} = (\Pi(t))_{t\in\mathbf{N}}$ a family of P systems from $\mathcal{F}$. If $r : \Gamma_1^* \to \Gamma_2^*$ is a polynomial-time reduction from L_1 to L_2, and (g, h) is a polynomial encoding of L_2 in $\mathbf{\Pi}$, then $(g \circ r, h \circ r)$ is a polynomial encoding of L_1 in $\mathbf{\Pi}$.

Before going on, we need the following definition, based on the corresponding one given in Reference.[5]

Definition 3.2
Let $\mathcal{F}$ be a class of recognizer P systems. We say that a decision problem $X = (I_X, \theta_X)$ is *solvable in polynomial time* by a family of recognizer P systems

$\mathbf{\Pi} = (\Pi(t))_{t \in \mathbf{N}}$ from $\mathcal{F}$, and we write $X \in \mathbf{PMC}_{\mathcal{F}}$, if

- The family $\mathbf{\Pi}$ is $\mathcal{F}$ *consistent*; that is, $\forall t \in \mathbf{N}\ (\Pi(t) \in \mathcal{F})$.
- The family $\mathbf{\Pi}$ is *polynomially uniform by Turing machines*; that is, there exists a deterministic Turing machine constructing $\Pi(t)$ from t in polynomial time.
- There exists a polynomial encoding (g, h) from I_X to $\mathbf{\Pi}$ verifying:
  - $\mathbf{\Pi}$ is *polynomially bounded* with regard to (X, g, h); that is, there exists a polynomial function p, such that for each $u \in I_X$, every computation of $\Pi(h(u))$ with input $g(u)$ is halting and, moreover, it performs at most $p(|u|)$ steps.
  - $\mathbf{\Pi}$ is *sound* with regard to (X, g, h); that is, for each $u \in I_X$, if there exists an accepting computation of $\Pi(h(u))$ with input $g(u)$, then $\theta_X(u) = 1$.
  - $\mathbf{\Pi}$ is *complete* with regard to (X, g, h); that is, for each $u \in I_X$, if $\theta_X(u) = 1$, then every computation of $\Pi(h(u))$ with input $g(u)$ is an accepting one.

Remark 3.1
In the above definition we have imposed a *confluence* condition, in the following sense: for each instance $u \in I_X$, either every computation of $\Pi(h(u))$ with input $g(u)$ is an accepting computation, or every computation of $\Pi(h(u))$ with input $g(u)$ is a rejecting computation.

Remark 3.2
Note that, as a consequence of Definition 3.2, the complexity class $\mathbf{PMC}_{\mathcal{F}}$ is closed under complement.

Proposition 3.1
Let $\mathcal{F}$ be a class of recognizer P systems. Let X and Y be decision problems such that X is reducible to Y in polynomial time. If $Y \in \mathbf{PMC}_{\mathcal{F}}$, then $X \in \mathbf{PMC}_{\mathcal{F}}$.

That is, the complexity class $\mathbf{PMC}_{\mathcal{F}}$ is stable under polynomial-time reduction.

§4 A Linear Solution to the Subset-Sum Problem

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 $w(B) = k$.*

We will use a tuple $(n, (w_1, \ldots, w_n), k)$ 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. 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*: in each membrane it is checked whether or not the weight of its associated subset is exactly k. This stage cannot start in a membrane before the previous ones are over in that membrane.
- *Output stage*: when the previous stage has been completed in all membranes, the system sends out the answer to the environment.

Now we construct a family $\mathbf{\Pi}$ of P systems in $\mathcal{AM}$ (the class of recognizer P systems with active membranes using 2–division) solving the Subset-Sum problem in *linear* time.

First we consider the function h defined over the set of instances of the Subset-Sum problem, I_{SubS}, by $h(u) = ((n+k)(n+k+1)/2) + n$, with $u=(n,(w_1,\ldots,w_n),k) \in I_{SubS}$. The function $\langle m,n\rangle=((m+n)(m+n+1)/2)+m$ is polynomial, primitive recursive and bijective from $\mathbf{N}^2$ onto $\mathbf{N}$ (the inverse is also polynomial). Hence, h is polynomial-time computable.

For each $(n,k) \in \mathbf{N}^2$ we consider the P system $(\Pi(\langle n,k\rangle), \Sigma(n,k), i(n,k))$, where the input alphabet is $\Sigma(n,k) = \{x_1,\ldots,x_n\}$, the input membrane is $i(n,k) = e$ and $\Pi(\langle n,k\rangle) = (\Gamma(n,k), \{e,s\}, \mu, \mathcal{M}_s, \mathcal{M}_e, R)$ is defined as follows:

- Working alphabet: $\Gamma(n,k) = \{E_0,\ldots,E_n, Z_0,\ldots,Z_{2n+2k+2}, Q, Q_0,\ldots, Q_{2k+1}, x_0,\ldots,x_n, b_0, b, B_0, B, Yes, No, \#, d_+, d_-\}$.
- Membrane structure: $\mu = [_s\ [_e\ ]_e\ ]_s$.
- Initial multisets: $\mathcal{M}_s = Z_0$; $\mathcal{M}_e = E_0 B^k$.
- The set R of evolution rules consists of the following rules:

(a) $[_e E_i]^0_e \to [_e Q]^-_e [_e E_i]^+_e$, for $i = 0,\ldots,n$,
$[_e E_i]^+_e \to [_e E_{i+1}]^0_e [_e E_{i+1}]^+_e$, for $i = 0,\ldots,n-1$.

The goal of these rules is to generate one membrane for each subset of A. When an object E_i $(i < n)$ is present in a neutrally charged membrane, we pick the element a_i for its associated subset and divide the membrane. In the new membrane where Q appears, no further elements will be added to the subset, but the other new membrane must generate membranes for other possible subsets which are obtained by adding elements of index $i+1$ or greater.

(b) $[_e x_0 \to B_0]^0_e$; $[_e x_0 \to \epsilon]^+_e$; $[_e x_i \to x_{i-1}]^+_e$, for $i = 1,\ldots,n$.

In the beginning, objects x_j (with $1 \le j \le n$) encode the weights of the corresponding elements of A: for each a_j we have w_j copies of x_j. At the same time as we add elements to the subset associated with the membrane, these three rules calculate the weight of this subset.

(c) $[_e Q \to Q_0]^-_e$; $[_e B_0 \to b_0]^-_e$; $[_e B \to b]^-_e$.

The apparition of objects Q_0, b_0, and b will mark the beginning of the checking stage. The multiplicity of object b_0 encodes the weight of the associated subset, and the constant k is represented by the number of objects

b.

(d) $[_e b_0]_e^- \to [_e]_e^0 \#$; $\quad [_e b]_e^0 \to [_e]_e^- \#$.

We compare the number of occurrences of objects b and b_0 sending them out of the membrane alternatively, and changing the polarity of the membrane each time.

(e) $[_e Q_{2j} \to Q_{2j+1}]_e^-$, for $j = 0, \ldots, k$;
$[_e Q_{2j+1} \to Q_{2j+2}]_e^0$, for $j = 0, \ldots, k-1$.

Objects Q_i act as a counter for the checking stage, controlling the number of "checking loops" that take place.

(f) $[_e Q_{2k+1}]_e^- \to [_e]_e^0 Yes$; $\quad [_e Q_{2k+1}]_e^0 \to [_e]_e^0 \#$;
$[_e Q_{2j+1}]_e^- \to [_e]_e^- \#$, for $j = 0, \ldots, k-1$.

Finally, these rules use the information given by the counter to deal with the different checking results: the same number of objects b_0 and b, or excess of objects b_0, or lack of them.

(g) $[_s Z_i \to Z_{i+1}]_s^0$, for $i = 0, \ldots, 2n + 2k + 1$; $\quad [_s Z_{2n+2k+2} \to d_+ d_-]_s^0$.

There is another counter in the skin membrane, that waits until all membranes end their checking stage and then releases objects d_+ and d_- in the skin.

(h) $[_s d_+]_s^0 \to [_s]_s^+ d_+$; $\quad [_s d_- \to No]_s^+$; $\quad [_s Yes]_s^+ \to [_s]_s^0 Yes$; $\quad [_s No]_s^+ \to [_s]_s^0 No$.

The answering process is now activated: first the object d_+ acts as a query, changing the polarity of the skin membrane, and then any possible object Yes that may be present in the membrane has to be sent out (notice that there is no conflict because in this moment there are no objects No present in the skin, since the rule $d_- \to No$ needs a positive charge to be applied).

It is easy to prove that the above constructed P systems are deterministic. Moreover, we will prove in Section 6 that the family $\mathbf{\Pi} = (\Pi(t))_{t \in \mathbf{N}}$ solves the Subset-Sum problem in *linear* time.

§5 Comments about the Way the System Works

Let us start by discussing the subset generation method. We will introduce the concept of the subset *associated* with a membrane labelled by e in a recursive manner.

- The subset associated with the initial inner membrane is the empty set.
- If in any moment of the computation an object E_i appears in a neutrally charged membrane labelled by e, then we add the element a_i to the corresponding associated subset of that membrane (when the generation stage ends, the associated subset will not be modified anymore).

We will also refer to the membrane as associated with its corresponding subset.

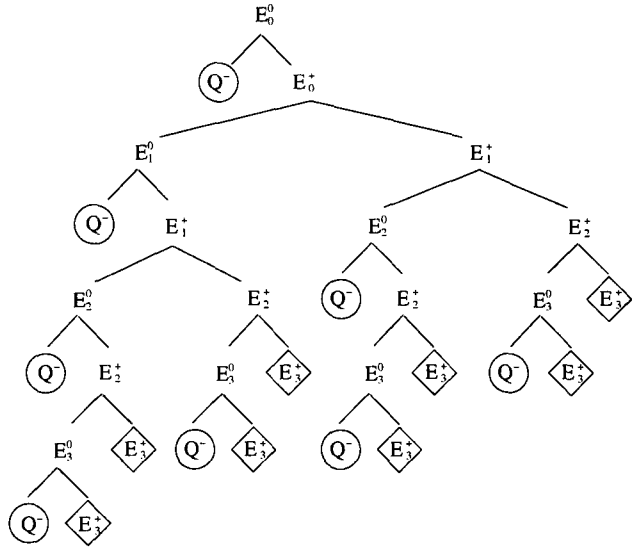

Fig. 1 Membrane Generation for $n = 3$

The objects that deal with the subset generation are objects E_i. Our goal is to generate a membrane for each subset, and we will do it in a sort of lexicographic order. That is, if an element a_j has already been added, then no elements $a_{j'}$ with $j' < j$ will be added in the future to the associated subset.

When applying a rule of the type $[_e E_i]_e^0 \rightarrow [_e Q]_e^- [_e E_i]_e^+$, the two new membranes inherit their father's associated subset. The membrane where object Q appears leaves the generation stage, and it will not divide anymore (such membranes appear in Fig. 1 marked with a circle). On the other hand, the positively charged membrane where object E_i appears will continue dividing, via the rule $[_e E_i]_e^+ \rightarrow [_e E_{i+1}]_e^0 [_e E_{i+1}]_e^+$, to generate membranes with different associated subsets, obtained as a result of adding objects of index $i+1$ or greater to the current subset. Note that if $i = n$, then the membrane gets blocked, because there are no more objects to add (we have marked that with a diamond).

Definition 5.1
We call *relevant membranes* associated with a configuration the membranes which contain the object Q_0 and have a negative charge.

The purpose of the membrane division that we carry on at the beginning of the process is to get a single relevant membrane for each subset of A. In total, this means 2^n different relevant membranes.

The weight stage takes place in parallel with the first one. Let us explain how this is done.

To calculate the weight of a subset we must sum the weights of all its elements. The system is designed in such a way to perform the sum of the elements' weights at the same time as the elements are added to the subset associated with

the membrane. This goes on until getting into a relevant membrane, and then the number of occurrences of object b_0 will encode exactly the weight of the associated subset.

In the next section we will prove that in a positively charged membrane $[_e]_e^+$ where the object E_i occurs, the multiplicity of object x_1 is equal to the weight of element $a_{i+1} \in A$. Thus, when in the next step we apply rules $[_e x_1 \to x_0]_e^+$, $[_e x_0 \to \epsilon]_e^+$ and $[_e E_i]_e^+ \to [_e E_{i+1}]_e^0 [_e E_{i+1}]_e^+$, we will get in each child membrane exactly $w(a_{i+1})$ occurrences of x_0.

In the case of the neutrally charged child we add the element a_{i+1} to the associated subset and in the next step we will add $w(a_{i+1})$ copies of B_0 to the membrane multiset. The situation is different for the positively charged child, where the element a_{i+1} will not be added to that membrane associated subset, nor to any of its descendants' subsets, so we "erase" the information $w(a_{i+1})$ storing instead the number $w(a_{i+2})$ as the number of occurrences of object x_0.

Next comes the checking stage. This stage begins in a membrane when it becomes relevant. This happens one step after the moment when object Q appears and the membrane gets negatively charged for the first time. In that step the objects B and B_0 become b and b_0, respectively, and the object Q produces Q_0 (as a result of applying the rules from (c)), so the membrane becomes relevant. We check whether there are exactly k objects b_0 comparing the number of objects b and b_0: they will be counted one by one alternatively, changing the membrane charge each time from negative to neutral and vice versa.

If the answer is affirmative, then after $2k$ steps of the checking stage we will not have any objects b or b_0 and the charge will be negative. The counter Q_j counts these $2k$ steps, and there are rules that take care of the cases when the answer is negative.

Finally, objects Z_j come into play for the response stage, counting $2n + 2k + 2$ steps from the beginning of the computation. Once these steps are made, we are sure that all membranes have finished their checking stage and so we just check if any of them said Yes and then we send the final answer outside.

§6 Formal Verification

We want to prove that the family $\mathbf{\Pi}$ of P systems presented in this paper *solves* the Subset-Sum problem, according to the definition given in Section 3.

In this section we will show that all computations of the systems from the family $\mathbf{\Pi}$ halt, either the object Yes or the object No (but not both) is sent out to the environment in the last step of each computation, and the systems give the same output corresponding to the same input multiset.

Now we will check the second condition of Definition 3.2, i.e., whether the family of P systems we have defined is polynomially uniform by Turing machines. First of all, observe that the evolution rules of $\Pi(h(u))$ are defined in a recursive manner from the instance u, in particular from n and k. Let us list the necessary resources to build $\Pi(h(u))$ from $u \in I_{SubS}$, with $u = (n, (w_1, \ldots, w_n), k)$:

- size of the alphabet: $4n + 4k + 17 \in O(\langle n, k \rangle)$,
- number of membranes: $2 \in \Theta(1)$,

- $|\mathcal{M}_e| + |\mathcal{M}_s| = k + 2 \in \Theta(k)$,
- sum of rules' lengths: $35n + 27k + 110 \in O(\langle n, k \rangle)$.

So a Turing Machine can build $\Pi(h(u))$ in polynomial time with respect to $h(u)$. Next we consider a function $g : I_{SubS} \to \bigcup_{t \in \mathbf{N}} I_{\Pi(t)}$, defined by $g(u) = x_1^{w_1} \cdots x_n^{w_n}$.

Then, g is a polynomial-time computable function. Moreover, the pair (g, h) is a polynomial encoding of I_{SubS} in $\mathbf{\Pi}$ since for each $u \in I_{SubS}$ we have $g(u) \in I_{\Pi(h(u))}$.

We must still prove that every $\Pi(h(u))$ is polynomially bounded, sound, and complete with regard to $(SubS, g, h)$.

Let $u \in I_{SubS}$ be an instance of the Subset-Sum problem. We will analyze what happens in the computation of its associated P system to find the instant when it halts, or at least an upper bound.

First of all we will study the membrane division process. Note that a necessary and sufficient condition for a membrane to divide is that it is neutrally charged and an element E_j with $j \in \{0, 1, \ldots, n\}$ is inside it, or that the charge is positive and an element E_j with $j \in \{0, 1, \ldots, n-1\}$ is present. In the first case we get a membrane with negative charge, where the object E_j is replaced by Q, and another membrane with positive charge, where the object E_j remains unchanged (this membrane will divide again if $j < n$). In the second case we get one of the child membranes with neutral charge and the other one with positive charge. In both membranes the object E_j that caused the division disappears, and it is replaced by an object E_{j+1}. As the index set is finite, we cannot keep on increasing indexes indefinitely, so the subset generation stage (membrane division) is a finite process. Indeed, it can be easily proved that after $2n + 1$ steps no further membrane division will take place in the computation.

Let us examine now the weight calculation stage. This stage takes place during the process of adding elements to the associated subset of a membrane. If we follow a reasoning similar to the previous one, considering now the objects x_i and how they disappear or get their indices reduced in one unit because of the rules from (b), then we deduce that the second stage is also a finite process.

When a relevant membrane appears, the first two stages stop in that membrane and the checking stage begins. Let $D = \{a_{i_1}, \ldots, a_{i_r}\} \subseteq A$ be the subset associated with that membrane, and let $w_D = w(D)$. The multiset associated with the membrane will be $Q_0 b^k b_0^{w_D} x_1^{w_{i_r+1}} \cdots x_{n-i_r}^{w_n}$. Then, starting from that situation, rules $[_e b_0]_e^- \to [_e]_e^0 \#$ and $[_e b]_e^0 \to [_e]_e^- \#$ will be applied alternatively until we run out of some of the objects. Clearly, this is a finite cycle, because every time the cycle is completed a pair of objects from the membrane is removed, and we never create new ones.

At the same time, counter Q_i evolves, and it gives information about the step when the previous cycle stops. Depending on the counter index, we will know if there was an excess of objects b, or of objects b_0, or if there was exactly the same amount of them. Again, as we have a finite number of indexes for Q_i and it is always evolving, the number of steps of this stage is finite. Indeed, the maximal number of steps for this stage is $2k+2$, in the case when $w(A) \geq k$: first

we apply k times the loop "*first* $[_e b_0]_e^- \to [_e]_e^0\#$ *together with* $[_e Q_{2j} \to Q_{2j+1}]_e^-$ *and then* $[_e b]_e^0 \to [_e]_e^-\#$ *together with* $[_e Q_{2j+1} \to Q_{2j+2}]_e^0$ " ($2k$ steps); and after that 2 more steps are made, one for the rule $[_e Q_{2k} \to Q_{2k+1}]_e^-$ (maybe together with $[_e b_0]_e^- \to [_e]_e^0\#$) and another one for the rule $[_e Q_{2k+1}]_e^- \to [_e]_e^0 Yes$ (or maybe $[_e Q_{2k+1}]_e^0 \to [_e]_e^0\#$).

This leads us to the output stage. In this stage a new counter, Z_j, comes into play to guarantee that the answer is delivered only after all the inner processes are over. That is, the counter waits until the last membrane finishes its checking stage.

The relevant membrane associated with the total subset is the one that appears the last, because its generation stage is the longest. This membrane is the result of $2n+2$ steps: first we apply the loop "$[_e E_{i-1}]_e^0 \to [_e Q]_e^- [_e E_{i-1}]_e^+$; $[_e E_{i-1}]_e^+ \to [_e E_i]_e^0 [_e E_i]_e^+$" for $i = 1, \ldots, n$ ($2n$ steps), and then 2 extra steps $[_e E_n]_e^0 \to [_e Q]_e^- [_e E_n]_e^+$ and $[_e Q \to Q_0]_e^-$.

In the worst case, the checking stage in the membrane associated with the total subset will take $2k+2$ steps, so the counter Z_j must wait during $2n+2k+4$ steps. First it evolves from Z_0 up to $Z_{2n+2k+2}$ ($2n+2k+2$ steps), then $[_s Z_{2n+2k+2} \to d_- d_+]_s^0$ and $[_s d_+]_s^0 \to [_s]_s^+ d_+$ are applied (2 more steps). Thus the answer is sent out in the step $2n+2k+5$ if it is Yes, or in the next step if the answer is No. So, there exists a polynomial (in fact, *linear*) bound for the number of steps with respect to n and k.

We have still to show that the system is sound and complete with regard to $(SubS, g, h)$, that is, that the system outputs Yes if and only if the answer to the Subset-Sum problem for the given instance is affirmative.

First let us focus on the generation stage. Let $D = \{a_{i_1}, \ldots, a_{i_r}\}$ be a subset of A (with $i_1 < i_2 < \cdots < i_r$, and $r \leq n$); then, there must be an unique relevant membrane that encodes it. Indeed, here is the membrane sequence that leads to this membrane:

$$[_e E_0]_e^0 \Rightarrow [_e E_0]_e^+ \Rightarrow [_e E_1]_e^+ \Rightarrow \cdots \Rightarrow [_e E_{i_1-1}]_e^+ \Rightarrow [_e E_{i_1}]_e^0 \Rightarrow [_e E_{i_1}]_e^+ \Rightarrow \cdots \Rightarrow$$
$$\Rightarrow [_e E_{i_2-1}]_e^+ \Rightarrow [_e E_{i_2}]_e^0 \Rightarrow \cdots \Rightarrow [_e E_{i_r}]_e^0 \Rightarrow [_e Q]_e^- \Rightarrow [_e Q_0]_e^-.$$

We know that by definition the associated subset of a membrane encodes somehow its *history*, and so the sequence that we sketched above leads actually to a relevant membrane whose associated subset is D. We will now see that the number of occurrences of b_0 in that membrane is exactly $w(D)$.

Lemma 6.1
In a positively charged membrane labelled by e where the object E_i occurs, with $0 \leq i \leq n-1$, the multiplicity of object x_j is exactly the weight of element $a_{i+j} \in A$ for every j such that $1 \leq j \leq n-i$.

Proof
By induction on i.

For the base case, $i = 0$, we know that the multiset of the membrane e in the initial configuration is $E_0 B^k x_1^{w_1} \cdots x_n^{w_n}$. Then the rule $[_e E_0]_e^0 \to [_e Q]_e^- [_e E_0]_e^+$

is applied (it is in fact the only one that can be applied to these objects) and hence we get a positively charged child whose multiset is $E_0 B^k x_1^{w_1} \cdots x_n^{w_n}$. Then, for every j such that $1 \leq j \leq n-i$ the multiplicity of object x_j is exactly the weight of element $a_j \in A$, as we wanted to prove.

For the inductive step, let us suppose the result is true for $i < n-1$. Consider then a positively charged membrane labelled by e where the object E_{i+1} occurs. We will distinguish two cases:

First, let us suppose that the membrane is obtained as the result of applying the rule $[_e E_i]_e^+ \rightarrow [_e E_{i+1}]_e^0 [_e E_{i+1}]_e^+$. From the inductive hypothesis we know that the multiplicity of object x_j in the father membrane is exactly the weight of element $a_{i+j} \in A$ for every j such that $1 \leq j \leq n-i$. Let us note that also rules $[_e x_i \rightarrow x_{i-1}]_e^+$ for $i = 1, \ldots, n$ and $[_e x_0 \rightarrow \epsilon]_e^+$ have been applied, so the multiplicity of object x_j in our membrane is equal to the multiplicity of object x_{j+1} in the father membrane, for $1 \leq j \leq n$. Thus, from the inductive hypothesis we deduce that the multiplicity of object x_j is exactly the weight of element $a_{i+j+1} \in A$. This completes the proof.

The second case is getting our membrane via the use of the rule $[_e E_j]_e^0 \rightarrow [_e Q]_e^- [_e E_j]_e^+$, for $j = i+1$. Following a similar reasoning for the neutrally charged father membrane (since $[_e E_{i+1}]_e^0$ is obtained as a result of applying the rule $[_e E_i]_e^+ \rightarrow [_e E_{i+1}]_e^0 [_e E_{i+1}]_e^+$), we conclude that the multiplicity of object x_j in that membrane is exactly the weight of element $a_{i+j+1} \in A$, for $j = 1, \ldots, n-i-1$. Finally, the multiplicity of those objects x_j does not change in the last step, because the only rule we could apply at the same time with the division rule is $[_e x_0 \rightarrow B_0]_e^0$, but it does not affect objects x_j with $j \geq 1$. ■

Proposition 6.1
In a relevant membrane the multiplicity of the object b_0 is exactly the weight of the subset associated with that membrane.

Proof
Consider an arbitrary relevant membrane. Let us suppose that the associated subset is $D = \{a_{i_1}, \ldots, a_{i_r}\}$, with $i_1 < i_2 < \cdots < i_r$, and $r \leq n$. Then, as we said before, there is no other relevant membrane along the computation that encodes the same subset.

All the membranes in the sequence that leads to the membrane have a positive charge except one neutrally charged membrane, with an object E_{i_l} inside it, for each element $i_l \in D$.

Let us apply the previous lemma for $i = i_l - 1$ (with $1 \leq l \leq r$) and $j = 1$ to all the positively charged membranes where the objects $E_{i_l - 1}$ appear. We get that in each one of them the multiplicity of the object x_1 is exactly w_{i_l}, the weight of element $a_{i_l} \in A$. In the next step, rules from (b) are applied, so we deduce that, for $1 \leq l \leq r$, in the neutrally charged membrane where E_{i_l} occurs, the multiplicity of the object x_0 is w_{i_l}. Thus, in the following step w_{i_l} copies of object B_0 will be added to the associated multiset.

This holds for i_l with $1 \leq l \leq r$, that is, for all the elements of D. We

conclude then that in the negatively charged membrane where Q appears the multiplicity of object B_0 is the sum of the weights of the elements of D, that is, the weight of the subset D. Finally, as rules from (c) are applied in the last step of the sequence, the multiplicity of object b_0 in the relevant membrane is exactly $w(D)$, as we wanted to prove. ■

Let us focus now on the checking stage. Consider, for example, a relevant membrane with an associated subset $D = \{a_{i_1}, \ldots, a_{i_r}\}$, and let $w(D) = w_D$ be the weight of this subset. Then the multiset associated with the membrane is $Q_0 b_0^{w_D} b^k x_1^{w_{i_r+1}} \cdots x_{n-i_r}^{w_n}$.

Hence, there are w_D and k occurrences of objects b_0 and b, respectively. The object Q_0 is also present in the membrane and there may also be some objects x_j, $j > 0$, but they do not evolve in the remaining stages because their evolution rules require a positive charge, and the membrane will never again get this polarization.

We distinguish three different possibilities.

1. Suppose $w_D < k$. Since the membrane is negatively charged, in the first step we can only apply the rules $[_e b_0]_e^- \to [_e]_e^0 \#$ and $[_e Q_0 \to Q_1]_e^-$. Then, as the charge is neutral now, rules $[_e b]_e^0 \to [_e]_e^- \#$ and $[_e Q_1 \to Q_2]_e^0$ will be applied next. This completes the first loop of the "checking cycle".
   We keep on looping until we run out of b_0 (remember we are in case $w_D < k$), i.e., we get to a situation $Q_{2w_D} b^{k-w_D} x_1^{w_{i_r+1}} \cdots x_{n-i_r}^{w_n}$ with a negative charge. Then the rule $[_e Q_{2w_D} \to Q_{2w_D+1}]_e^-$ will be applied, but no object b_0 will make the charge change to neutral this time, so we must apply the "reject" rule $[_e Q_{2w_D+1}]_e^- \to [_e]_e^- \#$.
   No more rules will be applied in the membrane, because the remaining objects are b and x_j, and there are no evolution rules for them with a negative charge condition.
2. Suppose now that $w_D = k$. Then, after k iterations of the checking cycle we have no more objects b or b_0 left, and the counter is Q_{2k}. In the next step we will apply only one rule, $[_e Q_{2k} \to Q_{2k+1}]_e^-$, and this leads to the affirmative answer: $[_e Q_{2k+1}]_e^- \to [_e]_e^- Yes$.
   No more rules will be applied in the membrane, because the remaining objects are x_j, and the membrane has a negative charge.
3. Finally, if $w_D > k$, then there are more objects b_0 than b. So in this case the checking cycle will also go through k loops, but the situation after this will be $Q_{2k} b_0^{w_D-k} x_1^{w_{i_r+1}} \cdots x_{n-i_r}^{w_n}$. Then the rule $[_e Q_{2k} \to Q_{2k+1}]_e^-$ is applied together with $[_e b_0]_e^- \to [_e]_e^0 \#$ and after that the "reject" rule $[_e Q_{2k+1}]_e^0 \to [_e]_e^0 \#$ ends the stage, because the remaining objects are b_0 and x_j with $j > 0$, and there are no evolution rules for them working in a neutrally charged membrane.

Finally, let us see that the output stage is sound. It is important to notice that no answer will be sent out while the skin membrane remains of a neutral

charge. The object d_+ is subsequently necessary to get any output, and the object d_+ evolves from the counter Z_j. We know that the purpose of counter Z_j is to wait until all inner processes are over. We also know that the last relevant membrane generated in the computation is the one associated with the total subset, $D = A$, and it appears in the step $2n + 2$.

Its checking stage will take $2k + 2$ steps in the worst case (if $w_D \geq k$ holds), so we wait $2n + 2k + 2$ steps before releasing object d_+ via the rule $[_s Z_{2n+2k+2} \rightarrow d_+ d_-]^0_s$.

When d_+ is sent out, we are sure that all inner processes are over. It is time then to look for possible objects Yes present in the membrane. To do so, object d_+ gives the positive charge to the skin membrane when it leaves the system.

Then, the rule $[_s d_- \rightarrow No]^+_s$ will be applied, and if there are any objects Yes present (i.e., if the checking stage was successful in some of the membranes labelled by e), then the rule $[_s Yes]^+_s \rightarrow [_s]^0_s Yes$ will be applied and no further rule will be possible. In case that no objects Yes were present in that moment, this would mean that the problem for the instance we are considering has a negative solution. Indeed, if no object Yes is sent out as output, then the charge will still be positive in the next step, allowing the rule $[_s No]^+_s \rightarrow [_s]^0_s No$ to be applied.

We have proved that for every subset of A a single relevant membrane associated with it appears in the computation. We have also seen that in every relevant membrane the weight of the subset is correctly encoded and compared with k. We have also checked that the object Yes is sent out if and only if the checking stage was successful in at least one inner membrane. Hence, we have proved the soundness and completeness of the family $\mathbf{\Pi}$ of P systems that we have defined. Also, it can be deduced that the family $\mathbf{\Pi}$ is $\mathcal{AM}$-consistent.

§7 Main Results

From the discussion in the previous section, and according to Definition 3.2, we deduce the following result:

Theorem 7.1
$SubS \in \mathbf{PMC}_{\mathcal{AM}}$.

Although the next result is a corollary of Theorem 7.1, we formulate it as another theorem, in order to stress its relevance.

Theorem 7.2
$\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$.

Proof
It suffices to make the following observations: the Subset-Sum problem is **NP**–complete, $SubS \in \mathbf{PMC}_{\mathcal{AM}}$ and the class $\mathbf{PMC}_{\mathcal{AM}}$ is stable under polynomial-time reduction. ■

§8 Conclusion

In this paper we have presented a family of recognizer P systems solving the Subset-Sum problem in *linear time*. This has been done in the framework of complexity classes in cellular computing with membranes.

The design presented here can be adapted for other *numerical* **NP**–complete problems. For instance, this approach has been used to formalize a solution to the *Knapsack problem*.[8] More generally, we believe that from such solutions we can extract some common features for attacking other numerical problems in the future.

The solution presented here differs from other solutions to **NP**–complete problems given by C. Zandron,[12] Gh. Păun,[10] et al. in the following sense: a family of P systems with active membranes and with input is constructed, associated with the problem that is being solved, in such a way that *all* the instances of such problem that have the *same length* (according to a prefixed polynomial-time computable criterium) are processed by the *same* P system (to which an appropriate input, that depends on the concrete instance, is supplied). On the contrary, in the solutions presented by C. Zandron,[12] Gh. Păun,[10] et al. a *single* P system is associated with *each one* of the instances of the problem.

More precisely, suppose we solve an instance of a given problem. If we wanted to solve another instance of the *same length*, we would have to modify somehow the P system that was used for the previous instance. In the approach presented here, the changes needed are minimal, as they only affect to the input multiset, but in other solutions in the literature, the changes would affect more deeply to the P system, as the set of rules may be modified as well as the multisets associated with the membranes.

We are also interested in the generation of the family of P systems that solves the problem. It would be a great improvement to be able to generate the family using cellular tools (that is, another membrane system) and thus a linear bound for the generation of the family will be also possible.

Another issue related to the present paper is the computer simulation of P systems. An implementation *in silico* (in Prolog) for P systems with active membranes has been developed by the Research Group in Natural Computing from the University of Seville.[1,2] This simulation can help to debug some errors in the formal design and verification of P systems, and a feedback process also exists, as running simulations of already verified P systems can cause some bugs in the implementation to arise.

We conclude by proposing some open questions. In Reference[5] it is proven that $\mathbf{co}-\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$, so we can deduce that $\mathbf{NP} \cup \mathbf{co}-\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$.

1. Does $\mathbf{NP} \cup \mathbf{co}-\mathbf{NP} = \mathbf{PMC}_{\mathcal{AM}}$ hold?
2. Find a class $\mathcal{F}$ of recognizer membrane systems such that $\mathbf{NP} \cup \mathbf{co}-\mathbf{NP} = \mathbf{PMC}_{\mathcal{F}}$, or $\mathbf{PSPACE} = \mathbf{PMC}_{\mathcal{F}}$.
3. Let $\mathcal{F}^*$ be a collection of recognizer P systems with active membranes but *without polarizations*. Is it verified that $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{F}^*}$?

Acknowledgements

The authors gratefully 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) Cordón Franco, A., Gutiérrez Naranjo, M. A., Pérez Jiménez, M. J. and Riscos Núñez, A., "Cellular Solutions for Some Numerical NP-complete Problems: A Prolog Implementation," *Molecular Computational Models: Unconventional Approaches* (Gheorghe, M. ed), Idea Group, Inc., pp. 115-149, 2005.

2) Cordón Franco, A., Gutiérrez Naranjo, M. A., Pérez Jiménez, M. J. and Sancho Caparrini, F., "A Prolog Simulator for Deterministic P systems with Active membranes," *New Generation Computing*, *22*, pp. 349-363, 2004.

3) Narayanan, K. S., "Languages of P Systems: Computability and Complexity," Ph.D. Thesis, Indian Institute of Technology Madras.

4) Pérez Jiménez, M. J., Romero Jiménez, A. and Sancho Caparrini, F. "Decision P Systems and the $P \neq NP$ Conjecture," *Lecture Notes in Computer Science*, *2597*, pp. 390-401, 2003.

5) Pérez Jiménez, M. J., Romero Jiménez, A. and Sancho Caparrini, F., "Solving Validity Problem by Active Membranes with Input," *Brainstorming Week on Membrane Computing* (Cavaliere, M., Martín-Vide, C. and G. Păun, eds.), *Report GRLM 26/03*, pp. 279-290, 2003.

6) Pérez Jiménez, M. J., Romero Jiménez, A. and Sancho Caparrini, F., *Teoría de la Complejidad en Modelos de Computación Celular con Membranas* (Kronos, ed.), Sevilla, 2002.

7) Pérez Jiménez, M. J., Romero Jiménez, A. and Sancho Caparrini, F., "Complexity Classes in Cellular Computing with Membranes," *Natural Computing*, *2*, *3*, pp. 265-285, 2004.

8) Pérez-Jiménez, M. J., Riscos-Núñez, A., "A Linear Solution for the Knapsack Problem Using Active Membranes," *Membrane Computing* (Martín-Vide, C., Mauri, G., Păun, Gh., Rozenberg, G. and Salomaa, A. eds.), *Lecture Notes in Computer Science*, *2933*, pp. 250-268, 2004.

9) Păun, Gh., "Computing with Membranes," *Journal of Computer and System Sciences*, *61*, *1*, pp. 108-143, 2000.

10) Păun, Gh., "P Systems with Active Membranes: Attacking NP Complete Problems," *Journal of Automata, Languages and Combinatorics*, *6*, *1*, pp. 75-90, 2001.

11) Păun, Gh., *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.

12) Zandron, C., "A Model for Molecular Computing: Membrane Systems," Ph.D. Thesis, Università degli Studi di Milano.

Mario J. Pérez Jiménez: He is a Titular Professor of Department of Computer Science and Artificial Intelligence at University of Seville, where he is the head of the Research Group on Natural Computing. He has published 8 books of Mathematics and Computation, and more than 100 scientific articles in prestigious scientific journals. He is a member of European Molecular Computing Consortium, and has been an independent expert to the evaluation of NEST (New and Emergent Science and Technology) proposals under the Sixth Framework Programme of the European Community.

Agustín Riscos Núñez: He holds a research fellowship at the Departament of Computer Science and Artificial Intelligence area in the University of Seville since november 2001. He is a member of the Research Group on Natural Computing and also of European Molecular Computing Consortium. His main interests are complexity theory, membrane computing and other areas of Natural Computing, both at the theoretical level and the level of computer simulation. In last year he has actively collaborated with other members of the Research Group to several International Conferences and publishing several papers in international journals.