Looking for Simple Common Schemes to Design Recognizer P Systems with Active Membranes That Solve Numerical Decision Problems

Carmen Graciani-Díaz and Agustín Riscos-Núñez

Dpto. Ciencias de la Computación e Inteligencia Artificial
{cgdiaz, ariscosn}@us.es

Abstract. Earlier solutions to decision problems by means of P systems used many counter objects to control the synchronization of different stages in a computation (usually as many counters as the stage must last in the worst case). In this paper we propose a way to replace those counters with some spacial objects for each stage. Furthermore, following the ideas presented in [1], in order to have a common scheme to attack numerical problems, all instances of a problem with the same size are solved by the same P system (which depends on the size) given an input which describes the corresponding instance of the problem. We illustrate these ideas with a cellular solution to the Subset-Sum problem.

1   Preliminaries

Since the introduction of P systems [3] a great amount of contributions in that field has been reported. In particular, many papers are devoted to solving decision or numerical NP–complete problems in polynomial time. In order to deal with such kind of problems, an exponential size workspace is generated (in the number of objects and the number of membranes). In this paper we deal with decision problems in the framework of P systems.

We 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 boolean function over I_X. For an instance u of the problem X, if $\theta_X(u) = 1$ (resp. $\theta_X(u) = 0$) the answer of the problem for that instance is Yes (resp. No).

In the general definition, P systems are *non-deterministic*. Therefore they do not seem to be a suitable tool to solve a decision problem. For that reason a condition that restricts, in a certain way, the non-determinism is demanded. More specifically, we will work with *confluent* systems (all computations with the same initial configuration produce the same answer).

When working with P systems with *external output*, the user can ignore the inner processes and take only into account the objects that the system expels to the environment. To know when a computation halts, it is demanded that some *halting indicator* is sent to the environment exactly in the last step.

These restrictions make more difficult the design of such systems. Earlier approaches in this area used counter objects to control the synchronization of

different stages in a computation. This kind of solutions need, therefore, extra objects and steps that are not necessary to obtain an answer but to control the procedure of obtaining it.

In this paper we want to show how this control can be obtained with only a few objects.

Furthermore, earlier solutions to NP-complete problems in polynomial time used to design one P system that solves one instance of the problem; therefore the system could not be used to solve any other instance of the problem, even if it was of the same size (see [8,2]). The introduction of P systems with *input* [6] gave rise to the design of families of systems, each of them able to solve all the instances of the problem of a given size.

Another goal of this paper is to present a solution of Subset-Sum problem with schemes of rules more uniform that depend only in the cardinality of the set.

The present work is a continuation of [4] and [1]. For this reason we have chosen the same problem and P system model: Subset-Sum problem and recognizer P systems with active membranes, respectively. The solution to Subset-Sum problem will illustrate also how the given schemes can be adapted for the new approach.

1.1 P Systems with Active Membranes

For a detailed description of a P system, $\Pi = (\Gamma, \mathsf{H}, \mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_p, \mathsf{R})$, with active membranes, we refer the reader to [2] and [7]. In what follows we briefly describe the rules of the model that will be used in next sections.

(a) $[_l\, \mathsf{a} \rightarrow \mathsf{v}\,]_l^\alpha$ (*evolution rules*), where $\mathsf{a} \in \Gamma$, $\mathsf{v} \in \Gamma^*$, $\alpha \in \{+, -, 0\}$, $\mathsf{l} \in \mathsf{H}$. Substitutes an object a by a multiset of objects v in a membrane with label l and charge α.

(b) $[_l\, \mathsf{a}\,]_l^\alpha \rightarrow \mathsf{b}\, [_l\,\,]_l^\beta$ (*communication rules*), where $\mathsf{a}, \mathsf{b} \in \Gamma$, $\alpha, \beta \in \{+, -, 0\}$ and $\mathsf{l} \in \mathsf{H}$. Sends out (to its father) an object a from a membrane with label l and charge α transformed into the object b. In addition, the charge of the membrane changes to β.

(c) $\mathsf{a}\, [_l\,\,]_l^\alpha \rightarrow [_l\, \mathsf{b}\,]_l^\beta$ (*communication rules*), where $\mathsf{a}, \mathsf{b} \in \Gamma$, $\alpha, \beta \in \{+, -, 0\}$ and $\mathsf{l} \in \mathsf{H}$. An object a enters in a membrane with label l and charge α (from its father) transformed into the object b. In addition, the charge of the membrane changes to β.

(d) $[_l\, \mathsf{a}\,]_l^\alpha \rightarrow [_l\, \mathsf{b}\,]_l^\beta\, [_l\, \mathsf{c}\,]_l^\gamma$ (*division rules*), where $\mathsf{a}, \mathsf{b}, \mathsf{c} \in \Gamma$, $\alpha, \beta, \gamma \in \{+, -, 0\}$ and $\mathsf{l} \in \mathsf{H}$. An object a divides a membrane with label l and charge α into two membranes with the same label and charges β and γ. In each of the new membranes the object a changes into objects b and c, respectively. This rule can only be applied to *elementary* membranes and never to the skin.

Rules of type (a) are applied as usual in the framework of P systems, that is, in a maximally parallel way. However, only one rule among the remaining types (b)–(d) can be applied to a membrane. The application of the rules is supposed to occur simultaneously (if division must take place in a membrane consider that the objects present in that membrane evolve previously). For a precise definition we refer the reader to [2] and [7].

1.2 P Systems with Input

A variant of P systems arises when considering the possibility of admitting external information before a computation starts.

A P system of degree p *with input* is a tuple (Π, Σ, i_Π) where:

- Π is a P system of degree p.
- Σ is an *input* alphabet strictly contained in the *work* alphabet, Γ.
- All the initial multisets are over the alphabet $\Gamma - \Sigma$.
- i_Π is a label that distinguishes the *input* membrane.

In a P system of degree p, with initial multisets $\mathcal{M}_1, \ldots, \mathcal{M}_p$, given a multiset of objects m over the input alphabet, the *initial configuration with input* m is the tuple $(\mu_\Pi, \mathcal{M}_1, \ldots, \mathcal{M}_{i_\Pi} \cup m, \ldots, \mathcal{M}_p)$. Let us denote by I_Π the set of all the possible input multisets.

1.3 P Systems with External Output

In this variant the environment collects the output of the computations, instead of an inner membrane.

There will be some special objects called *halting indicators*. A P system with external output is *valid* if no computation sends any halting indicator to the environment except in the last step. And that must only occurs if the computation is a halting one.

1.4 Language Recognizer P Systems

A *language recognizer* P system is a P system with input and external output such that the working alphabet contains two halting indicators yes and no. A language recognizer P system is *valid* and all its computations halt. If the object is yes (resp. no) we say that the computation is an *accepting* (resp. *rejecting*) one.

We say that $\{\Pi_n\}_{n \in \mathbb{N}}$ is a family of language recognizer P systems that solves, in polynomial time, a decision problem (I_X, θ_X) if it verifies the following properties:

- All the P systems in the family are language recognizers.
- There exists a deterministic Turing machine that constructs each member of the family, Π_n, from n in polynomial time.
- There exists a polynomial encoding for the set of instances, I_X, into the family of P systems, Π (that is, a pair of polynomial time computable functions (cod, s) where $\text{cod}: I_X \to \bigcup_{n \in \mathbb{N}} I_{\Pi_n}$ and $s: I_X \to \mathbb{N}$ verifying $\text{cod}(u) \in I_{\Pi_{s(u)}}$ for all $u \in I_X$) such that:
    - A polynomial function, p, exists so that for each $u \in I_X$ all the computations of the system $\Pi_{s(u)}$ with input $\text{cod}(u)$ halt at most in $p(|u|)$ steps.

- For each $u \in I_X$, if there exists an accepting computation in the system $\Pi_{s(u)}$ with input $\mathsf{cod}(u)$, then $\theta_X(u) = 1$. It is said then that the family is *sound*.
- For each $u \in I_X$, if $\theta_X(u) = 1$, then every computation in the system $\Pi_{s(u)}$ with input $\mathsf{cod}(u)$ is an accepting one. It is said then that the family is *complete*.

The resolution of an instance $u \in I_X$ by a family of P systems Π consist of two stages: during the first one (usually called pre-computation stage) we calculate $\mathsf{s}(u)$, $\mathsf{cod}(u)$ and $\Pi_{s(u)}$; during the second stage the P system $\Pi_{s(u)}$ with input $\mathsf{cod}(u)$ carries out its computation.

2 A Solution to the Subset-Sum Problem

We illustrate the previous discussion with a solution to Subset-Sum problem that can be stated as follows:

Given a finite set $\mathsf{A} = \{\mathsf{a}_1, \ldots, \mathsf{a}_n\}$, a weight function $\omega \colon \mathsf{A} \to \mathbb{N}$ such that $\omega(\mathsf{a}_i) = \omega_i$ for $i = 1, \ldots, n$, and a constant $k \in \mathbb{N}$, determine whether or not there exists a subset $\mathsf{D} \subseteq \mathsf{A}$ such that $\omega(\mathsf{D}) = k$.

The proposed solution is based on the one given at [4], and is divided into several stages:

- *Generation stage:* Elementary membrane divisions are carried out until obtaining a membrane associated with each subset of A.
- *Calculating 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 verified if the weight of the associated subset is equal to the constant k. This stage begins in each membrane after the previous ones are over.
- *Output stage:* When the previous stage has been completed in all membranes, the system sends the corresponding answer to the environment and the computation halts.

For each $n \in \mathbb{N}$ (the cardinality of set $\mathsf{A} = \{\mathsf{a}_1, \ldots, \mathsf{a}_n\}$) a P system with active membranes, input and external output is defined as follows: $(\Pi_n, \Sigma_n, i_{\Pi_n})$ where $\Pi_n = (\Gamma_n, \mathsf{H}, \mu_\Pi, \mathcal{M}_\mathsf{s}, \mathcal{M}_{\mathsf{e},n}, \mathcal{M}_\mathsf{r}, \mathsf{R}_n)$, P system of degree.

- Working alphabet: $\Gamma_n = \{\mathsf{x}_i \mid 0 \leq i \leq n\} \cup \{\#, \mathsf{yes}, \mathsf{no}, \overline{\mathsf{no}}, \mathsf{q}, \mathsf{q}_0, \mathsf{q}_1, \mathsf{q}_2, \mathsf{q}_3, \mathsf{c}, \mathsf{g}, \overline{\mathsf{g}}, \mathsf{d}, \mathsf{f}_0, \mathsf{f}, \mathsf{f}_+, \overline{\mathsf{b}}, \mathsf{b}, \overline{\mathsf{x}}_0, \overline{\mathsf{b}}_0, \mathsf{b}_0, \mathsf{z}, \mathsf{z}_0, \mathsf{z}_+, \overline{\mathsf{z}}, \mathsf{h}_0, \overline{\mathsf{h}}_1, \mathsf{h}_1, \mathsf{p}, \mathsf{t}\}$.
- Set of labels: $\mathsf{H} = \{\mathsf{s}, \mathsf{e}, \mathsf{r}\}$.
- Membrane structure: $\mu_\Pi = [_\mathsf{s} [_\mathsf{e}]_\mathsf{e} [_\mathsf{r}]_\mathsf{r}]_\mathsf{s}$.
- Initial multisets: $\mathcal{M}_\mathsf{s} = \overline{\mathsf{no}}$, $\mathcal{M}_{\mathsf{e},n} = \overline{\mathsf{g}} \, \mathsf{f}_0 \, \mathsf{d}^n \, \mathsf{z}_0$ and $\mathcal{M}_\mathsf{r} = \overline{\mathsf{h}}_1$.
- Set of rules: R_n that consists of the following rules:

(a) $[_\mathsf{e} \mathsf{f}_0]_\mathsf{e}^0 \to [_\mathsf{e} \mathsf{q}]_\mathsf{e}^- [_\mathsf{e} \mathsf{f}]_\mathsf{e}^+$ $[_\mathsf{e} \mathsf{z}_0 \to \mathsf{z}]_\mathsf{e}^0$ $[_\mathsf{e} \mathsf{z}_+ \to \mathsf{z}_0]_\mathsf{e}^0$

$[_\mathsf{e} \mathsf{f}_+]_\mathsf{e}^0 \to [_\mathsf{e} \mathsf{f}_0]_\mathsf{e}^0 [_\mathsf{e} \mathsf{f}]_\mathsf{e}^+$ $[_\mathsf{e} \mathsf{z}_0 \to]_\mathsf{e}^+$ $[_\mathsf{e} \mathsf{z}_+]_\mathsf{e}^+ \to \mathsf{z} [_\mathsf{e}]_\mathsf{e}^+$

$[_\mathsf{e} \mathsf{f} \to \mathsf{f}_+]_\mathsf{e}^+$ $[_\mathsf{e} \mathsf{d}]_\mathsf{e}^+ \to \# [_\mathsf{e}]_\mathsf{e}^0$ $[_\mathsf{e} \mathsf{z} \to \mathsf{z}_+]_\mathsf{e}^+$

The goal of these rules is the generation of one membrane for each subset of A. When an object f_0 is present in a neutrally charged membrane we pick a new element from A for its associated subset (summing its weight to the previous ones) and then divide the membrane. In the membranes where q appears no further objects will be added, and the charge of the membrane changes in order to activate the checking stage. The multiplicity of object d controls the number of divisions that must take place. The object z evolves in order to remain only in the last generated membrane, collaborating to control the beginning of the output stage.

(b) $[_e x_i \to x_{i-1}]_e^+ \qquad 1 \le i \le n$

$[_e x_0 \to \bar{x}_0]_e^0 \qquad\qquad [_e \bar{x}_0 \to \bar{b}_0]_e^0 \qquad\qquad [_e \bar{x}_0 \to]_e^+$

In the beginning, objects x_i, $1 \le i \le n$, are introduced encoding the weights of the corresponding elements of A. When the generation stage ends, the multiplicity of object $\bar{b}_0$ will encode the weight of the subset associated with the membrane.

(c) $[_e q \to q_0]_e^- \qquad\qquad [_e \bar{b}_0 \to b_0]_e^- \qquad\qquad [_e \bar{b} \to b]_e^-$

These rules mark the beginning of the checking stage in a membrane. Now, the multiplicity of object b_0 encode the weight of the corresponding subset of A and the multiplicity of object b encode the value of the constant k.

$[_e \bar{g}]_e^- \to \bar{g} [_e]_e^-$

Object $\bar{g}$ will be used to mark the beginning of the output stage.

(d) $[_e b_0]_e^- \to \# [_e]_e^+ \qquad\qquad [_e b]_e^+ \to \# [_e]_e^-$

We compare the number of occurrences of objects b_0 and b sending them out alternatively.

$[_e q_0 \to q_1]_e^- \qquad\qquad [_e q_1 \to q_0]_e^+ \qquad\qquad [_e q_1 \to q_2 c]_e^-$

$[_e c]_e^- \to \# [_e]_e^+ \qquad\qquad [_e q_2 \to q_3]_e^+$

Objects q_i and c control if both objects have been actually sent out or not (if there is an excess or lack of any of them).

$[_e q_3]_e^+ \to yes [_e]_e^0 \qquad [_e q_3]_e^- \to \# [_e]_e^0 \qquad [_e q_0]_e^+ \to \# [_e]_e^0$

These rules deal with the different checking results.

(e) $[_s z \to \bar{z} z]_s^0 \qquad\qquad \bar{z} [_r]_r^0 \to [_r \bar{z}]_r^0 \qquad\qquad [_r \bar{z} \to p]_r^0$

Object z controls the beginning of a process in membrane r that will trigger the output stage. When z appears in membrane s 2^n objects $\bar{g}$ are present in it.

$[_s \bar{z}]_s^0 \to \# [_s]_s^+ \qquad\qquad [_s \bar{g} \to g]_s^+ \qquad\qquad g [_e]_e^0 \to [_e g]_e^+$

When a membrane ends its checking stage it admits one object g.

(f) $g [_r]_r^+ \to [_r g]_r^- \qquad\qquad [_r h_1 \to h_0]_r^+ \qquad\qquad [_r h_0 \to h_1]_r^-$

$[_r p]_r^- \to p [_r]_r^0 \qquad\qquad [_r g]_r^0 \to g [_r]_r^- \qquad\qquad p [_r]_r^- \to [_r p]_r^+$

$[_r h_0]_r^+ \to t [_r]_r^+ \qquad\qquad [_r \bar{h}_1 \to h_1]_r^+$

We will use membrane r to detect when all objects g have been admitted in a membrane e. That will mean that the checking stage has finished in all membranes and then, the output stage is triggered.

(g) $[_s t]_s^+ \to \# [_s]_s^- \qquad\qquad [_s yes]_s^- \to yes [_s]_s^0$

$[_s \overline{no} \to no]_s^- \qquad\qquad [_s no]_s^- \to no [_s]_s^0$

The presence of object t in membrane s activates the answering process. If there is any object yes then it must be sent out. Otherwise, an object no goes out.

(h) Also, some cleaning can be done during the process.

$$[_e x_i \rightarrow \ ]_e^- \qquad 1 \leq i \leq n \qquad [_e z \rightarrow \ ]_e^- \qquad\qquad [_e d \rightarrow \ ]_e^-$$
$$[_e b \rightarrow \ ]_e^0 \qquad\qquad\qquad\qquad [_e b_0 \rightarrow \ ]_e^0$$

- Input alphabet: $\Sigma_n = \{\bar{b}\} \cup \{x_i \mid 1 \leq i \leq n\}$.
- Input membrane: $i_\Pi = e$.

So we have defined a family of P systems $\{\Pi_n\}_{n\in\mathbb{N}}$. Each of the members of the family, Π_n, solves all the instances of the Subset-Sum problem for a finite set A with cardinality n. Each instance will be determined by the values of the weight function, ω_i for $i = 1, \ldots, n$, and the value of the constant k. The set of possible input multisets is $I_{\Pi_n} = \{\bar{b}^k \ x_1^{\omega_1} \ldots x_n^{\omega_n} \mid k, \omega_1, \ldots, \omega_n \in \mathbb{N}\}$. As we can see, all the members of the family can be constructed by a Turing machine in polynomial time from n.

Let us consider $I_X = \{(n, (\omega_1, \ldots, \omega_n), k) \mid n, \omega_1, \ldots, \omega_n, k \in \mathbb{N}\}$ (all the instances of the Subset-Sum problem). The pair of functions $(\mathsf{cod}, \mathsf{s})$ defined by $\mathsf{cod}(n, (\omega_1, \ldots, \omega_n), k) = \bar{b}^k \ x_1^{\omega_1} \ldots x_n^{\omega_n}$ and $\mathsf{s}(n, (\omega_1, \ldots, \omega_n), k) = n$ is a polynomial encoding of I_X into $\{\Pi_n\}_{n\in\mathbb{N}}$.

The following data gives us an idea of Π_n complexity:

- Size of the working alphabet: $n + 31 \in O(n)$.
- Number of membranes: $3 \in O(1)$.
- $|\mathcal{M}_s| + |\mathcal{M}_{e,n}| + |\mathcal{M}_r| = n + 5 \in O(n)$.
- Input size: $k + \omega(A)$
- Number of rules: $2n + 48$
- Number of computation steps needed in the worst case: $3n + 2\min(k, \omega(A)) + 19$

In what follows we will prove that the systems of the family are recognizer P systems that solve the Subset-Sum problem in linear time; that is, that the family is sound, complete, and polynomially bounded.

3   Formal Verification

Proposition 1. *Consider $k, n \in \mathbb{N}$ and a weight function $\omega : A \rightarrow \mathbb{N}$ such that $\omega(a_i) = \omega_i$ for $i = 1, \ldots, n$. For any $l \in \mathbb{N}$ and i, $1 \leq i \leq n$, if l is the weight of a subset $D \subseteq \{a_1, \ldots, a_{i-1}\}$, then from a membrane of the following form $[_e \bar{b}^k$ $\bar{g} \ f_+ \ d^{n-i} \ \bar{b}_0^l \ x_0^{\omega_i} \cdots x_{n-i}^{\omega_n} \ ]_e^0$ we obtain the set of membranes*
$$\{[_e b^k \ q_0 \ b_0^{l'} \ ]_e^- \mid \text{where } l' \text{ is the weight of } D \cup D' \text{ for } \emptyset \neq D' \subseteq \{a_i, \ldots, a_n\}\}$$
They will be called relevant membranes.

The last membrane of this set will be generated after $3(n-i+1)$ steps. During the process $2^{n-i+1} - 1$ objects $\bar{g}$ will appear in membrane s.

Moreover, we also obtain the following set of membranes:
$$\{[_e \bar{b}^k \ \bar{g} \ f_+ \ \bar{b}_0^{l'} \ ]_e^+ \mid l' = \omega(D \cup D') \text{ for } D' \subseteq \{a_i, \ldots, a_n\}\}$$
These membranes will be called irrelevant and the last one will be generated after $3(n - i + 1)$ steps.

If an abject z_+ is present in the considered membrane, then it will only remain, as an object z, in the last generated irrelevant membrane.

Proof: By decreasing induction on i, starting from $i = n$.

Figure 1 shows the evolution of a membrane $[_e \overline{b}^k \overline{g} f_+ \overline{b}_0^l x_0^{\omega_n}]_e^0$ where l is the weight of $D \subseteq \{a_1, \ldots, a_{n-1}\}$. The branching represents new generated membranes obtained by division.

$$[_e b^k \overline{g} f_+ b_0^l x_0^{\omega_n} \underline{z_+}]_e^0$$

$$[_e b^k \overline{g} f_0 b_0^l \vec{x}_0^{\omega_n} \underline{z_0}]_e^0 \qquad\qquad [_e b^k \overline{g} f b_0^l \vec{x}_0^{\omega_n} \underline{z_0}]_e^+$$

$$[_e b^k \overline{g} q b_0^{l+\omega_n} \underline{z}]_e^- \qquad [_e b^k \overline{g} f b_0^{l+\omega_n} \underline{z}]_e^+ \qquad [_e b^k \overline{g} f_+ b_0^l]_e^+ \ (3)$$

$$\overline{g} [_e b^k q_0 b_0^{l+\omega_n}]_e^- \ (1) \qquad [_e b^k \overline{g} f_+ b_0^{l+\omega_n} \underline{z}]_e^+ \ (2)$$
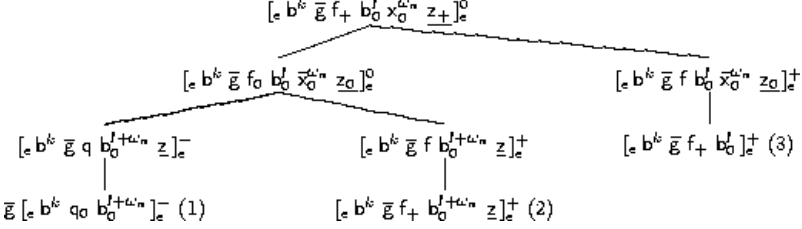
Fig. 1. Case $i = n$

In node (1) we can see that the relevant membrane $[_e b^k q_0 b_0^{l+\omega_n}]_e^-$ (where $l + \omega_n = \omega(D \cup \{a_n\})$) is obtained after $3 = 3(n - n + 1)$ steps. Besides, in the last step $1 = 2^{n-n+1} - 1$ object $\overline{g}$ has been sent to membrane s.

In (2) the irrelevant membrane $[_e \overline{b}^k \overline{g} f_+ \overline{b}_0^{l+\omega_n}]_e^+$ (where $l + \omega_n$ is the weight of $D \cup \{a_n\}$) is obtained after $3 = 3(n - n + 1)$ steps.

In (3) we have obtained the irrelevant membrane $[_e \overline{b}^k \overline{g} f_+ \overline{b}_0^l]_e^+$, after 2 steps ($l$ is the weight of $D = D \cup \emptyset$).

Moreover, Figure 1 shows (underlined) the evolution of an object z_+ when it is present in the initial membrane and we can see that it only remains, as an object z, in the last obtained irrelevant membrane.

Thus, the proposition holds for $i = n$.

Induction step: $i + 1 \to i$

The evolution of $[_e \overline{b}^k \overline{g} f_+ d^{n-i} \overline{b}_0^l x_0^{\omega_i} \cdots x_{n-i}^{\omega_n}]_e^0$ where l is the weight of a subset $D \subseteq \{a_1, \ldots, a_{i-1}\}$ is shown in Figure 2.

In (1) the relevant membrane $[_e b^k q_0 b_0^{l+\omega_i}]_e^-$ is obtained. In it $l + \omega_i$ is the weight of $D \cup \{a_i\}$. In the last step one object $\overline{g}$ appears in membrane s.

In (2), a membrane $[_e \overline{b}^k \overline{g} f_+ d^{n-(i+1)} \overline{b}_0^{l+\omega_i} x_0^{\omega_{i+1}} \cdots x_{n-(i+1)}^{\omega_n}]_e^0$, in which l is the weight of $D \cup \{a_i\} \subseteq \{a_1, \ldots, a_i\}$, is obtained.

By induction hypothesis, from this membrane we obtain the set of membranes $\{[_e b^k q_0 b_0^{l'}]_e^- \mid l' = \omega((D \cup \{a_i\}) \cup D')$ for $\emptyset \neq D' \subseteq \{a_{i+1}, \ldots, a_n\}\}$. The last member of this set will be generated after $3(n - i) + 3 = 3(n - i + 1)$ steps. During this process $2^{n-i} - 1$ objects $\overline{g}$ will appear in membrane s.

In addition, the set of irrelevant membranes $\{[_e \overline{b}^k \overline{g} f_+ \overline{b}_0^{l'}]_e^+ \mid l'$ is the weight of $(D \cup \{a_i\}) \cup D'$ for $D' \subseteq \{a_{i+1}, \ldots, a_n\}\}$ is obtained. The last member of this set will be generated after $3(n - i) + 3 = 3(n - i + 1)$ steps.

In (3), a membrane $[_e \overline{b}^k \overline{g} f_+ d^{n-(i+1)} \overline{b}_0^l x_0^{\omega_{i+1}} \cdots x_{n-(i+1)}^{\omega_n}]_e^0$ in which l is the weight of D is obtained.

By induction hypothesis, from this membrane the set of relevant membranes $\{[_e b^k q_0 b_0^{l'}]_e^- \mid l' = \omega(D \cup D')$ for $\emptyset \neq D' \subseteq \{a_{i+1}, \ldots, a_n\}\}$ is generated. The

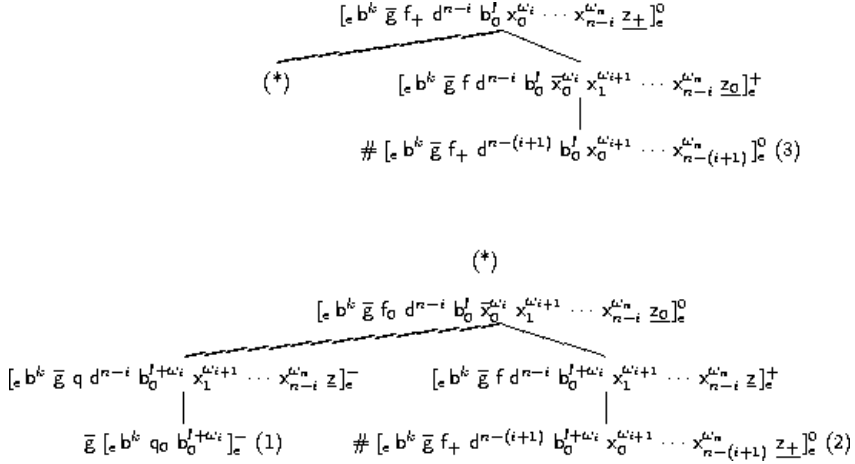$$[_e\, b^k\, \overline{g}\, f_+\, d^{n-i}\, b_0^l\, x_0^{\omega_i} \cdots x_{n-i}^{\omega_n}\, z_+]_e^0$$

(*)
$$[_e\, b^k\, \overline{g}\, f\, d^{n-i}\, b_0^l\, \overline{x}_0^{\omega_i}\, x_1^{\omega_i+1} \cdots x_{n-i}^{\omega_n}\, z_0]_e^+$$

$$\# [_e\, b^k\, \overline{g}\, f_+\, d^{n-(i+1)}\, b_0^l\, x_0^{\omega_i+1} \cdots x_{n-(i+1)}^{\omega_n}]_e^0 \quad (3)$$

(*)
$$[_e\, b^k\, \overline{g}\, f_0\, d^{n-i}\, b_0^l\, \overline{x}_0^{\omega_i}\, x_1^{\omega_i+1} \cdots x_{n-i}^{\omega_n}\, z_0]_e^0$$

$$[_e\, b^k\, \overline{g}\, q\, d^{n-i}\, b_0^{l+\omega_i}\, x_1^{\omega_i+1} \cdots x_{n-i}^{\omega_n}\, z]_e^- \qquad [_e\, b^k\, \overline{g}\, f\, d^{n-i}\, b_0^{l+\omega_i}\, x_1^{\omega_i+1} \cdots x_{n-i}^{\omega_n}\, z]_e^+$$

$$\overline{g}\, [_e\, b^k\, q_0\, b_0^{l+\omega_i}]_e^- \quad (1) \qquad\qquad \# [_e\, b^k\, \overline{g}\, f_+\, d^{n-(i+1)}\, b_0^{l+\omega_i}\, x_0^{\omega_i+1} \cdots x_{n-(i+1)}^{\omega_n}\, z_+]_e^0 \quad (2)$$

Fig. 2. Induction step $i + 1 \rightarrow i$

last member of this set will be generated after $3(n - i) + 2 = 3(n - i + 1) - 1$ steps and during the process $2^{n-i} - 1$ objects $\overline{g}$ will appear in membrane s. From (3), it is also generated the set of irrelevant membranes $\{[_e \overline{b}^k\, \overline{g}\, f_+\, \overline{b}_0^{l'}]_e^+ \mid l'$ is the weight of $D \cup D'$ for $D' \subseteq \{a_{i+1}, \ldots, a_n\}\}$. The last member of this set will be generated after $3(n - i) + 3 = 3(n - i + 1)$ steps.

Thus, from a membrane $[_e \overline{b}^k\, \overline{g}\, f_+\, d^{n-i}\, \overline{b}_0^l\, x_0^{\omega_i} \cdots x_{n-i}^{\omega_n}]_e^0$ we will obtain the set of relevant membranes $\{[_e b^k\, q_0\, b_0^{l'}]_e \mid l'$ is the weight of $D \cup D'$ for a subset $\emptyset \neq D' \subseteq \{a_i, \ldots, a_n\}\}$ (the last member after $3(n - i + 1)$ steps). During the process $2(2^{n-i} - 1) + 1 = 2^{n-i+1} - 1$ objects $\overline{g}$ appear in membrane s.

Besides, we obtain the set of irrelevant membranes $\{[_e \overline{b}^k\, \overline{g}\, f_+\, \overline{b}_0^{l'}]_e^+ \mid l'$ is the weight of $D \cup D'$ for $D' \subseteq \{a_i, \ldots, a_n\}\}$ (the last member after $3(n - i + 1)$ steps).

Finally, if an object z_+ is present in the initial membrane (underlined in Figure 2) it only appears in (2), then by induction hypothesis it will only remain, as an object z, in the last generated irrelevant membrane. $\qquad\square$

Theorem 2. *Given $k, n \in \mathbb{N}$ and a weight function $\omega : A \rightarrow \mathbb{N}$, $\omega(a_i) = \omega_i$ for $i = 1, \ldots, n$ from a membrane of the form $[_e \overline{b}^k\, \overline{g}\, f_0\, d^n\, x_1^{\omega_1} \cdots x_n^{\omega_n}\, z_0]_e^0$ the set of relevant membranes $\{[_e b^k\, q_0\, b_0^l]_e \mid l = \omega(D)$ for $D \subseteq A\}$ is obtained (last one after $3n + 2$ steps). During the process 2^n objects $\overline{g}$ appear in membrane s.*

The set of irrelevant membranes $\{[_e \overline{b}^k\, \overline{g}\, f_+\, \overline{b}_0^l]_e^+ \mid l = \omega(D)$ for $D \subseteq A\}$ is also obtained (the last of them after $3n + 2$ steps). The object z_0 will evolve to an object z that will only remain in the last generated irrelevant membrane.

Proof:

Figure 3 shows the evolution of $[_e \overline{b}^k\, \overline{g}\, f_0\, d^n\, x_1^{\omega_1} \cdots x_n^{\omega_n}\, z_0]_e^0$.

In (1) the relevant membrane $[_e b^k\, q_0]_e^-$ is obtained and during the process an object $\overline{g}$ appears in membrane s.
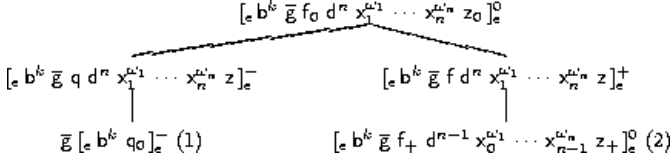
$$[_e b^k \bar{g} f_0 d^n x_1^{\omega_1} \cdots x_n^{\omega_n} z_0]_e^0$$

$$[_e b^k \bar{g} q d^n x_1^{\omega_1} \cdots x_n^{\omega_n} z]_e^- \qquad\qquad [_e b^k \bar{g} f d^n x_1^{\omega_1} \cdots x_n^{\omega_n} z]_e^+$$

$$\bar{g} [_e b^k q_0]_e^- \ (1) \qquad\qquad [_e b^k \bar{g} f_+ d^{n-1} x_0^{\omega_1} \cdots x_{n-1}^{\omega_n} z_+]_e^0 \ (2)$$

Fig. 3. Evolution scheme

In (2) we have $[_e \bar{b}^k \bar{g} f_+ d^{n-1} x_0^{\omega_1} \cdots x_{n-1}^{\omega_n}]_e^0$, case $i = 1$ and $l = 0$ of proposition 1, and from it we will obtain the set $\{[_e b^k q_0 b_0^l]_e^- \mid l = \omega(D)$ for $\emptyset \neq D \subseteq A\}$ of relevant membranes (the last one after $3(n - 1 + 1) + 2 = 3n + 2$ steps) and $2^{n-1+1} - 1 = 2^n - 1$ objects $\bar{g}$ in membrane s.

Moreover, from (2) (by Proposition 1) we will also obtain the set of irrelevant membranes $\{[_e \bar{b}^k \bar{g} f_+ \bar{b}_0^l]_e^+ \mid l = \omega(D)$ for $D \subseteq A\}$ (the last one after $3(n - 1 + 1) + 2 = 3n + 2$ steps.

Finally, as an object z_+ appears in (2), from the evolution of z_0 (again by Proposition 1) it will only remain as an object z in the last generated irrelevant membrane. □

Let us see now, given $0 \leq m \leq \min(k, l)$, the evolution of a relevant membrane of the form: $[_e q_0 b^{k-m} b_0^{l-m}]_e^-$

(a) Case: $m = k, m = l$
$$[_e q_0]_e^- \Rightarrow [_e q_1]_e^- \Rightarrow [_e q_2 c]_e^- \Rightarrow \# [_e q_2]_e^+ \Rightarrow [_e q_3]_e^+ \Rightarrow \text{yes} [_e]_e^0$$
(b) Case: $m < k, m = l$
$$[_e q_0 b^{k-m}]_e^- \Rightarrow [_e q_1 b^{k-m}]_e^- \Rightarrow [_e q_2 c b^{k-m}]_e^- \Rightarrow \# [_e q_2 b^{k-m}]_e^+ \Rightarrow$$
$$[_e q_3 b^{k-(m+1)}]_e^- \Rightarrow \# [_e b^{k-(m+1)}]_e^0 \text{ (objects } b \text{ will be consumed in the following step).}$$
(c) Case: $m = k, m < l$
$$[_e q_0 b_0^{l-m}]_e^- \Rightarrow \# [_e q_1 b_0^{l-(m+1)}]_e^+ \Rightarrow [_e q_0 b_0^{l-(m+1)}]_e^+ \Rightarrow$$
$$\# [_e b_0^{l-(m+1)}]_e^0 \text{ (objects } b_0 \text{ will be consumed in the following step).}$$
(d) Case: $m < k, m < l$
$$[_e q_0 b^{k-m} b_0^{l-m}]_e^- \Rightarrow \# [_e q_1 b^{k-m} b_0^{l-(m+1)}]_e^+ \Rightarrow$$
$$\# [_e q_0 b^{k-(m+1)} b_0^{l-(m+1)}]_e^- \text{ (that will continue evolving as shown)}$$

Beginning with $m = 0$, pairs of objects b_0 and b are sent out (case (d)) until both of them are finished (case (a)) or a lack of any of them is detected (cases (b) and (c)).

At the end, the membrane will be neutrally charged and ready to admit one object g, after that it will remain inactive.

The only irrelevant membrane that continues evolving is the one with an object z_+ that will be sent out (so this object will appear in membrane s after the last object $\bar{g}$ has also appeared).

Object z in membrane s will activate the evolution of membrane r:

z $[_r \overline{h}_1]_r^0 \Rightarrow \overline{z} z [_r \overline{h}_1]_r^0 \Rightarrow [_r \overline{h}_1 z]_r^+$ (also an object $\overline{z}$ is sent out membrane s changing its charge to positive) $\Rightarrow [_r h_1 p]_r^+$ (in membrane s each object $\overline{g}$ evolves to an object g).

In membrane r begins now a process to detect if there is any object g in membrane s (remember that 2^n relevant membranes are at the checking stage and that when they finish they will admit one object g). Figure 4 shows a scheme of the process.

When this process finishes an object t has appeared in membrane s and two cases can take place: there is an object yes in membrane s or there is not.

(a) $[_s t \text{ yes } \overline{no}]_s^+ \Rightarrow [_s \text{ yes } \overline{no}]_s^- \Rightarrow \text{ yes } [_s \text{ no }]_s^0$
(b) $[_s t \overline{no}]_s^+ \Rightarrow [_s \overline{no}]_s^- \Rightarrow [_s \text{ no }]_s^- \Rightarrow \text{ no } [_s]_s^0$
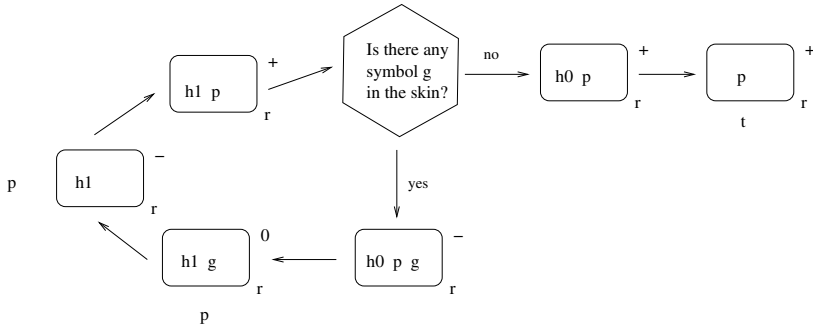

Fig. 4. Checking the existence of objects g in the skin

4 Conclusions

In this paper we have presented a family of recognizer P systems solving the Subset-Sum problem. Given a "size", n, member Π_n of the family solves all the instances of the SubsetSum problem for a finite set A with cardinal n. Each instance is determined by the value of k and the values of the weight function, ω_i for $i = 1, \ldots, n$.

This solution is more uniform than the one presented in [4], as the construction of the family only depends on n, the cardinality of A, and not on the parameter k.

On the other hand, we remark that the rules that control the generation, checking, and output stages depend neither on n nor on k. Only the number of rules that handle objects x_i (corresponding to the different elements of A) is determined by n.

Another interesting point is that the number of computation steps, in the cases where $\omega(A) < k$, is smaller than in the solution given in [4].

This solution has also been adapted to other numerical NP-complete problems as the Knapsack or the Partition problems. Moreover, different approaches

and skeleton designs to solve those problems have been studied. For example, in order to obtain 2^n membranes (each of them representing a subset of A) there is another solution that only generates those membranes (in the solution given in this paper, another 2^n irrelevant membranes are also generated).

Those results allow us to be optimistic about describing a "language" for the design of P systems to solve relevant numerical problems and, why not, other kinds of problems.

Another issue related to the present paper is the computer simulation of P systems. An implementation *in silico* (in CLIPS) for P systems with active membranes has been developed by the Research Group on Natural Computing from the University of Seville [5]. This simulation has helped us 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 detect possible bugs in the implementation.

The CLIPS code of the simulator, some instructions of use and some examples (including the problem presented in this paper) are available on the Web at http://www.gcn.us.es.

References

1. M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and A. Ricos-Núñez. Towards a programming language in cellular computing. *Electronic Notes in Theoretical Computer Science*, 123(1):93–110, 2005.
2. Gh. Păun. *Membrane Computing: An Introduction.* Springer-Verlag, Berlin, 2002.
3. Gh. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000. and Turku Center for Computer Science-TUCS Report No 208.
4. M. J. Pérez-Jiménez and A. Riscos-Núñez. Solving the subset-sum problem by p systems with active membranes. *New Generation Computing, Springer–Verlag, Tokyo*, to appear.
5. M. J. Pérez-Jiménez and F. J. Romero-Campero. A CLIPS simulator for recognizer p systems with active membranes. In Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, and F. Sancho-Caparrini, editors, *Proceedings of the Second Brainstorming Week on Membrane Computing*, RGNC Report (01/04), pages 387–413, 2004.
6. M. J. Pérez-Jiménez, Á. Romero-Jiménez, and F. Sancho-Caparrini. Complexity classes in models of cellular computing with membranes. *Natural Computing*, 2(3):265–285, September 2003.
7. A. Riscos-Núñez. *Programación celular: resolución eficiente de problemas numéricos NP-completos.* PhD thesis, University of Seville, 2004.
8. C. Zandron. *A Model for Molecular Computing: Membrane Systems.* PhD thesis, Universit degli Studi di Milano, 2001.