Solving the BINPACKING Problem by Recognizer P Systems with Active Membranes

Mario J. PÉREZ-JIMÉNEZ, Francisco José 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
E-mail: {Mario.Perez, Francisco-Jose.Romero}@cs.us.es

Abstract. In this paper we present an effective solution to the BINPACKING problem using a family of recognizer P systems with active membranes, input membrane and external output. The analysis of the solution presented here will be done form the point of view of complexity classes.

1 Introduction

P systems are an emergent branch in the field of Natural Computing. This unconventional model of computation is presented as a kind of distributed parallel computing model and it is based upon the observation that the processes which take place in the complex structure of a living cell can be considered as computations.

Since Gh. Păun introduced it in [2] several variants have been considered from different approaches. A fairly complete compendium about P systems can be found in [3]. Many of the proposed variants have been proved to be computational complete,their computational power is that of Turing machines; besides some variants of P systems have been proved to be *computational efficient*, they have been shown to be able to solve **NP**-complete problems in polynomial time (see [3] Chapter 7).

The solution presented here has been designed through a family of recognizer P systems with active membranes, input membrane and external output. In particular, P systems with active membranes are studied in [3], section 7.2. We have followed the ideas and schemes used to solve others numerical **NP**-problems as the Subset–Sum in [9] and the Knapsack problem in [10]. Due to the strong similarities of the design of these solutions the idea of a *cellular programming language* seems possible as it is suggested in [12].

The analysis of the presented solution will be done from the point of view of the complexity classes. A *complexity 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. We will study the complexity of the proposed solution within the framework of the *complexity classes in P systems* studied in [7] and [8].

The paper is organized as follows: Section 2 recalls recognizer P systems with active membranes, input membrane and external output. In section 3 the complexity classes for P systems are briefly introduced. Sections 4, 5 and 6 show a cellular solution to the

BINPACKING problem. In section 7 we use a CLIPS simulator for recognizer P systems with acive membranes to show a session for the BINPACKING problem. Conclusions are given in section 8.

2 Recognizer P systems with Active Membranes, Input Membrane and External Output

Definition 2.1 *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.*

Definition 2.2 *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 2.3 *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 (μ_0, M_0), where $\mu_0 = \mu$, $M_0(j) = \mathcal{M}_j$, for each $j \neq i_\Pi$, and $M_0(i_\Pi) = \mathcal{M}_{i_\Pi} \cup m$.*

The computations of a P system with input $m \in M(\Sigma)$, a multiset 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 $m \in M(\Sigma)$.

In the case of P systems with input and with external output, the concept of computation is introduced in a similar way but with a slight variant. 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 2.4 *Let $\mu = (V(\mu), E(\mu))$ be a membrane structure. The* membrane structure with external 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* external environment *of the structure μ.*

Note that we have only included 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 informs about the contents of the external environment.

Definition 2.5 *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 its computations halt.*

3. *If $\mathcal{C}$ is a computation of Π, then either some object YES or some object N0 (but not both) mush have been released into the environment, and only in the last step of the computation. We say that $\mathcal{C}$ is an accepting computation (respectively, rejecting computation) if the object YES (respectively, N0) appears in the external environment associated to the corresponding halting configuration of $\mathcal{C}$.*

This recognizer systems are specially suitable when trying to solve decision problems.

In this paper we will deal with recognizer P-Systems with Active Membranes, Input Membrane and External Output. Let's remember that a P system with Active Membranes is a tuple:

$$\Pi = (\Sigma, H, \mu, \omega_1, \ldots, \omega_m, R)$$

where:

1. $m \geq 1$, is the initial degree of the system;
2. Σ is the alphabet of symbol-objects;
3. H is a finite set of labels for membranes;
4. μ is a membrane structure, of m membranes, labelled (not necessarily in a one-to-one manner) with elements of H;
5. $\omega_1, \ldots, \omega_m$ are strings over Σ, describing the initial multisets of objects placed in the m regions of μ;
6. R is a finite set of evolution rules, of the following forms:
   (a) $[\, a \to \omega\,]_h^\alpha$ for $h \in H$,$\alpha \in \{+, -, 0\}$, $a \in \Sigma$, $\omega \in \Sigma^*$, *object evolution rules*: This is an object evolution rule, associated with a membrane labelled with 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 $h \in H$, $\alpha_1, \alpha_2 \in \{+, -, 0\}$, $a, b \in \Sigma$, *communication rules (send in rules)*: An object from the region immediately outside a membrane labelled with 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 $h \in H$, $\alpha_1, \alpha_2 \in \{+, -, 0\}$, $a, b \in \Sigma$, *communication rules (send out rules)*: An object is sent out from membrane labelled with 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 $h \in H$, $\alpha \in \{+, -, 0\}$, $a, b \in \Sigma$, *dissolving rules*: A membrane labelled with 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 $h \in H$, $\alpha_1, \alpha_2, \alpha_3 \in \{+, -, 0\}$, $a, b, c \in \Sigma$, *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 with h are used for all copies of this membrane. At one step, a membrane labelled with h can be the subject of *only one* rule of types (b)-(e).

Let us denote by $\mathcal{AM}$ the class of language recognizer P systems with active membranes using 2-division (see [3], section 7.2).

3 The complexity class $\mathbf{PMC}_{\mathcal{F}}$

Roughly speaking, a computational complexity study of a solution for a problem is an estimation of the resources (time, space, ...) that are required through all the processes that take place in the way from the bare instance of the problem up to the final answer.

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.

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.

Instead of looking for a single system that solves a problem, we prefer designing a family of P systems such that each element decides all the instances of "equivalent size", in certain sense.

Let us now introduce some basic concepts before the definition of the complexity class itself.

Definition 3.1 *Let L be a language, $\mathcal{F}$ a class of P systems with input and $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}^+}$ a family of P systems of $\mathcal{F}$. A polynomial encoding of L in $\mathbf{\Pi}$ is a pair (g,h) of polynomial-time computable functions $g : L \to \bigcup_{n\in\mathbf{N}^+} I_{\Pi(n)}$ 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 $L_1 \subseteq \Sigma_1$ and $L_2 \subseteq \Sigma_2$ be languages. Let $\mathcal{F}$ be a class of P systems with input and $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}^+}$ a family of P systems of $\mathcal{F}$. If $r : \Sigma_1 \to \Sigma_2$ 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 $\mathbf{\Pi}$.*

Definition 3.2 *Let $\mathcal{F}$ be a class of recognizer P systems, $f : \mathbf{N}^+ \to \mathbf{N}^+$ a total recursive function, and $X = (I_X, \theta_X)$ a decision problem. We say that $X \in \mathbf{MC}_{\mathcal{F}}(f)$ if there exists a family, $\mathbf{\Pi} = (\Pi(n))_{n\in\mathbf{N}^+}$, of P systems such that:*

- *$\mathbf{\Pi}$ is $\mathcal{F}$–consistent: $\forall n \in \mathbf{N}^+$, $\Pi(n) \in \mathcal{F}$.*
- *$\mathbf{\Pi}$ is uniform: there exists a deterministic Turing machine that from $n \in \mathbf{N}^+$ constructs $\Pi(n)$ in polynomial time.*
- *There exists a polynomial encoding (g, h) from I_X to $\mathbf{\Pi}$ verifying:*
  - *$\mathbf{\Pi}$ is f–bounded, regarding to (g, h).*
    For each $u \in I_X$, all computations of $\Pi(h(u))$ with input $g(u)$ halt in, at most, $f(|u|)$ steps.
  - *$\mathbf{\Pi}$ is X–sounded, regarding to (g, h).*
    For each $u \in I_X$, if every computation of $\Pi(h(u))$ with input $g(u)$ is an accepting computation, then $\theta_X(u) = 1$.
  - *$\mathbf{\Pi}$ is X–complete, regarding to (g, h).*
    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 computation.

Remark 3.1 *In the above definition we have imposed every P system $\Pi(n)$ to be confluent, in the following sense: for every input m, either every computation of $\Pi(n)$ with input m is an accepting computation, or every computation of $\Pi(n)$ with input m is a rejecting computation.*

Definition 3.3 *The polynomial complexity class associated with a collection of recognizer P systems, $\mathcal{F}$, is defined as follows:*

$$\mathbf{PMC}_{\mathcal{F}} = \bigcup_{f \text{ polynomial}} \mathbf{MC}_{\mathcal{F}}(f)$$

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

4 The BINPACKING Problem

The BINPACKING problem can be stated as follows:

> *Given a set $A = \{s_1, \ldots, s_n\}$, a weight function $\omega : A \to \mathbb{N}$ and two constants $b \in \mathbb{N}$, $c \in \mathbb{N}$ decide whether or not there exists a partition of A into b subsets such that their weights do not exceed c.*

This problem can be seen as the situation when we have n items, b bins of capacity c and we have to introduce the items in the bins.

We will represent the instances of the problem using tuples of the kind $(n, (\omega_1, \ldots, \omega_n), b, c)$, where n is the number of items, $(\omega_1, \ldots, \omega_n)$ are the weights, b is the number of bins and c their capacity.

We will face the resolution of this problem via a brute force algorithm, in the framework of recognizer P systems with active membranes using 2-division, without cooperation nor priority among rules. Our strategy will consist in:

- For each bin:
  - *Generation stage:* Membrane division is used until a specific membrane for each subset S of the *remaining* items is obtained.
  - *Calculation stage:* In each membrane the weight of the associated subset is calculated.
  - *Checking stage :* The condition $\omega(S) \leq c$ is checked for every subset $S \subseteq A$.
  - *Transition stage:* If the associated subset satisfies $\omega(S) \leq c$ then we introduce these items in this bin and we repeat the process with the remaining items and bins; otherwise the membrane is dissolved.
- *Output stage:* The answer is released into the environment according to the results in the checking stage for each bin.

Now we construct a family of recognizer P systems with active membranes using 2-division solving the BINPACKING problem.

Let us consider a polynomial bijection, $\langle\ \rangle$, between $\mathbf{N}^3$ and $\mathbf{N}$ (e.g. $\langle x, y, z\rangle = \langle\langle x, y\rangle, z\rangle$, induced by the pair function $\langle x, y\rangle = (x + y) \cdot (x + y + 1)/2 + x$).

The family presented here is

$$\mathbf{\Pi} = \{\ (\Pi(\langle n, b, c\rangle), \Sigma(n, b, c), i(n, b, c))\ :\ (n, b, c) \in \mathbb{N}^3\ \}$$

For each element of the family, the input alphabet is $\Sigma(n, b, c) = \{s_1, \ldots, s_n, z_1, \ldots, z_n\}$, the input membrane is $i(n, b, c) = 2$, and the P system $\Pi(\langle n, b, c\rangle) = (\Gamma(n, b, c), \{1, 2\}, \mu, \mathcal{M}_1, \mathcal{M}_2, R)$ is defined as follows:

- Working alphabet:

$$\Gamma(n, b, c) = \{z_{ijk}, s_{ijk}, Z_{ijk}, S_{ijk}, z_0, z, w, W, g_l, T, t_m, D, \overline{D}, \hat{D}, G, G_1, f_r, neg, d_e,$$

$$YES, NO\ :\ 1 \leq i \leq b, -1 \leq j \leq n, 1 \leq k \leq n, 0 \leq l \leq 2n + 1, 0 \leq m \leq 2n + 1\}$$

- Membrane structure: $\mu = [_1\ [_2\ \ ]_2\ ]_1$
- Initial Multisets: $\mathcal{M}_1 = \{f_1\},\ \ \mathcal{M}_2 = \{f_1, g_0, D^c\}$
- The set of evolution rules, R, consists of the following rules:
  1. $[\ s_k \rightarrow s_{1,\,k,\,k}\ ]_2^0;\ \ [\ z_k \rightarrow z_{1,\,k,\,k}\ ]_2^0, \quad 1 \leq k \leq n$
     These rules initialize the algorithm. The objects of type s and z will have three subindixes. The first one, $1 \leq i \leq b$, will represent the number of the bin where the item represented by this object can be added. The second one, $-1 \leq j \leq n$

will denote its position in the stack to be added in the current bin; if second subindixe is -1 then this item has not been chosen to be added in the bin. The third one, $1 \leq k \leq n$ will be use to show for which item this object is used to represent its weight.

2. $[\ z_{i,\,1,\,k}\ ]_2^0 \rightarrow [\ z\ ]_2^+\ \ [\ z_{i,\,-1,\,k}\ ]_2^0\ , \quad 1 \leq k \leq n\ ,\ 1 \leq i \leq b-1$
   The goal of these rules is to generate one membrane for each subset of the remaining items that can be added in the current bin. When the object $z_{i,\,1,\,k}$ is present in a neutrally charged membrane with label 2 it means that the system has to decide whether or not the item number k is chosen for the subset to be added in the bin number i.
   So the membrane is divided into two membranes: one positively charged which will represent the subset where the item number k is chosen to be introduced in the bin, the object z appears in this membrane; and the other one will be negatively charged and will represent the subset where the item number k is not added in the bin, so we set the second subindixe to -1, $z_{i,\,-1,\,k}$.
3. $[\ s_{i,\,0,\,k}\ \rightarrow\ w\ ]_2^+\ , \quad 1 \leq k \leq n,\ 1 \leq i \leq b-1$
   The presence of the objects $s_{i,\,0,\,k}$ in a positively charged membrane with label 2 means that the item number k is added to the subset associated to the membrane. The multiplicity of the object $s_{\cdot,\,\cdot,\,k}$ encodes the weight of the item k and the multiplicity of the object w encodes the weight of the subset associated to the membrane.So when an item is added to the subset associated to the membrane the objects $s_{i,\,0,\,k}$ evolve to w.
4. $[\ z\ ]_2^+\ \rightarrow\ \sharp\,[\ \ ]_2^0$
   The element z is used to change the polarization of the membranes with label 2 from positive to neutral.
5. $[\ s_{i,\,j,\,k}\ \rightarrow\ s_{i,\,j-1,\,k}]_2^0; [\ z_{i,\,j,\,k}\ \rightarrow\ z_{i,\,j-1,\,k}]_2^0;\ 0 \leq j \leq n,\ 1 \leq k \leq n,\ 1 \leq i \leq b-1$
   Once the item analyze has been or not introduced to the bin these rules update the stack of items by rotating the second subindixes of the objects of type s and z.
6. $[\ g_i\ \rightarrow\ g_{i+1}\ ]_2^0;\ \ [\ g_i\ \rightarrow\ g_{i+1}\ ]_2^+; \quad 0 \leq i \leq 2n-1$
   The objects g_i are counters used in the generation stage.
7. $[\ g_{2n}\ \rightarrow\ g_{2n+1}\ ,\ t_0\ ]_2^0;\ \ [\ g_{2n}\ \rightarrow\ g_{2n+1}\ ,\ t_0\ ]_2^+;$
   The generation stage takes $2n$ steps. The object g_{2n} will produce the objects g_{2n+1} and t_0 which will begin the preparation for the *checking stage*.
8. $[\ g_{2n+1}\ ]_2^0\ \rightarrow\ \sharp\,[\ \ ]_2^-;$
   The item g_{2n+1} will change the polarization of the membranes with label 2 from neutral to negative.
9. $[\ w\ \rightarrow\ W\ ]_2^-$
   In the preparation for the *checking stage* the objects w are renamed to W in order to avoid conflicts with the previous stage.
10. $[\ s_{i,\,-1,\,k}\ \rightarrow\ S_{i,\,k,\,k}\ ]_2^-; [\ z_{i,\,-1,\,k}\ \rightarrow\ Z_{i,\,k,\,k}\ ]_2^-; 1 \leq i \leq b-1,\ 1 \leq k \leq n$
    The objects $s_{i,\,-1,\,k}$ and $z_{i,\,-1,\,k}$ are renamed to $S_{i,\,-1,\,k}$ and $Z_{i,\,-1,\,k}$ before the *checking stage* in order to avoid conflicts with the previous stage.

11. $[\ D\ \rightarrow\ \overline{D}\,,\ \hat{D}\ ]_2^-$
    The multiplicity of the objects D represents the capacity of the bins. In the *checking stage* we have to check if the weight of the subset introduced in the current bin exceeds or not its capacity. At the beginning of this stage the objects D produce the objects $\overline{D}$ and $\hat{D}$. The objects $\hat{D}$ will be used in the *checking stage* of the current bin and the objects $\overline{D}$ will keep the capacity of the bins so this information can be used later in the computation.
12. $[\ \hat{D}\ ]_2^-\ \rightarrow\ \sharp\,[\ \ ]_2^0; [\ W\ ]_2^0 \rightarrow\ \sharp\,[\ \ ]_2^-$
    With these rules the system checks whether or not the weight of the subset introduced in the bin exceeds its capacity.
13. $[\ t_i\ \rightarrow\ t_{i+1}\ ]_2^-; [\ t_j\ \rightarrow\ t_{j+1}\ ]_2^0;\quad 0 \leq i \leq 2c-1,\ 1 \leq j \leq 2c-1$
    The objects t_i are counters used in the *checking stage*.
14. $[\ t_{2c}\ \rightarrow\ t_{2c+1}, G, z_0\ ]_2^-; [\ t_{2c}; \rightarrow\ t_{2c+1}, G, z_0\ ]_2^0;$
    The *checking stage* takes $2c$ steps. The objects t_{2c} will produce the objects t_{2c+1}, G and z_0 which will begin the transition to the next bin.
15. $[\ t_{2c+1}\ ]_2^-\ \rightarrow\ \sharp\,[\ \ ]_2^+; [\ t_{2c+1}\ ]_2^-\ \rightarrow\ \sharp\,[\ \ ]_2^+;$
    The object t_{2c+1} changes the polarization of membranes with label 2 from negative to positive and the *transition stage* begins.
16. $[\ W\ ]_2^+\ \rightarrow\ \sharp$
    If there are still objects W when the *checking stage* has finished it means that the multiplicity of objects W exceeded the multiplicity of objects $\hat{D}$. So the weight of the subset introduced in the bin exceeded its capacity, that is this is not a possible solution to the problem and the corresponding membrane is dissolved.
17. $[\ \hat{D}\ \rightarrow\ \sharp\ ]_2^+$
    The remaining objects $\hat{D}$ are "erased" in the *transition stage*.
18. $[\ \overline{D}\ \rightarrow\ D]_2^+$
    The objects $\overline{D}$ are renamed to D so they can be used in the computation for the next bin.
19. $[\ S_{i,\,k,\,k}\ \rightarrow\ s_{i+1,\,k,\,k}\ ]_2^+; [\ Z_{i,\,k,\,k}\ \rightarrow\ z_{i+1,\,k,\,k}\ ]_2^+;\quad 1 \leq i \leq b-2; 1 \leq k \leq n$
    The objects $S_{i,\,k,\,k}$ and $Z_{i,\,k,\,k}$ are renamed to $s_{i+1,\,k,\,k}$ and $z_{i+1,\,k,\,k}$ so they can be used in the computation for the next bin.
20. $[\ G\ \rightarrow\ G_1]_2^+; [\ G_1 \rightarrow g_1]_2^0$
    These rules produce the object g_1 that will be used as counter in the *generation stage* of the next bin.
21. $[\ z_0\ \rightarrow\ z\ ]_2^+$
    The object z is produced to finish the *transition stage*.
22. $[\ S_{b-1,\,k,\,k}\ \rightarrow\ w\ ]_2^+; [\ Z_{b-1,\,k,\,k}\ \rightarrow\ \sharp\ ]_2^+;\quad 1 \leq k \leq n$
    These rules introduce all the remaining items in the last bin.
23. $[\ f_i\ \rightarrow\ f_{i+1}\ ]_2^0\,;\ [\ f_i\ \rightarrow\ f_{i+1}\ ]_2^+\,;\ [\ f_i\ \rightarrow\ f_{i+1}\ ]_2^-\,; 0 \leq i \leq 2nb + 2cb + 5b - 2n - 2c - 5$
    The objects f_i are counters that will show when the *checking stage* for the last bin must begin.

24. $[\ f_{2nb+2cb+5b-2n-2c-4}\ \rightarrow\ neg, T, d_0\ ]_2^+$;
This rule will force the system to skip the *generating stage* for the last bin and will force the *checking stage* begin.

25. $[\ T\ \rightarrow\ t_0\ ]_2^0 ; [\ neg\ ]_2^0\ \rightarrow\ \sharp\ [\ \ ]_2^-$
These rules initialize the *checking stage* for the last bin.

26. $[\ d_i\ \rightarrow\ d_{i+1}]_2^0 , [\ d_i\ \rightarrow\ d_{i+1}]_2^- , [\ d_{2c+3}\ ]_2^0 \rightarrow\ YES;\ 0 \leq i \leq 2c+2$
The objects d_i are counters in the membranes with label 2 that eventually will produce the answer YES.

27. $[\ d_i\ \rightarrow\ d_{i+1}\ ]_1^0 ; [\ d_{2nb+2cb+5b-2n+3}\ \rightarrow\ NO\ ]_1^0$
The objects d_i are counters in the skin that will eventually produce the answer NO.

28. $[\ YES\ ]_1^0\ \rightarrow\ YES[\ \ ]_1^+ ;\ [\ NO\ ]_1^0\ \rightarrow\ YES[\ \ ]_1^-$
These rules released the answer into the environment. Note that if the answer of the system must be YES this object will appear in the skin one step before the object NO so no conflict occurs.

5 An Overview of the Computation

First of all we must define a polynomial encoding of the Binpacking problem in the family $\mathbf{\Pi}$ in order to study the complexity of the problem with respect to it. Given an instance $u = (n, (\omega_1, \ldots, \omega_n), b, c)$ of the Binpacking problem, we define $h(u) = \langle n, b, c\rangle$ (recall the bijection mentioned in the previous section) and $g(u) = \{z_1, \ldots z_n, s_1^{\omega_1}, \ldots, s_n^{\omega_n}\}$. Now we will informally describe how the system $\mathbf{\Pi}(h(u))$ with input $g(u)$ works.

In the first step of the computation, the rules $[\ s_k \rightarrow s_{1,\,k,\,k}\ ]_2^0 ;\ \ [\ z_k \rightarrow z_{1,\,k,\,k}\ ]_2^0$ are applied to initialize the computation. The first subindixe of these objects represents the bin we are dealing with.

For each bin i, for $1 \leq i \leq b-1$, the generation and calculation stages take place in parallel, following the instructions from the rules in 1 - 7. This two stages end when the object g_{2n+1} set the polarization of the membrane to negative. We generate every subset of the remaining items, associating each subset to a single working membrane.

Let us introduce the concept of subset *associated* with an internal membrane through the following recursive definition:

- The subset associated with the initial membrane is the empty one.
- When an object $z_{i,\,\cdot,\,k}$ does not appears in a inner membrane it means that the k-th item of A has been introduced in the bin number i. In the other hand when an object $z_{i,\,-1,\,k}$ appears in an inner membrane, it means that the k-th item of A has been left out of the bin number i and so it can be introduced in the following bins.
- When a division rule is applied, the two newborn membranes inherit the associated subset form the original membrane.

As we have mentioned above, the two first stages are carried out in parallel. Indeed, there is only a gap of one step of computation between the moment when an item is added to the associated subset and the moment when the new weight of the subset is updated. For example, for the item number 1 which is represented by the objects s_1 and

z_1: after two steps of the computation we can see that there are two inner membranes in the configuration. In one of them no object $z_{1,\cdot,1}$ appears, thus the item number 1 has been introduced in the bin number 1 and it can be proved that according to the rule 3 in the next configuration ω_1 copies of w will appear.

Meanwhile in the other inner membrane appears the object $z_{1,-1,1}$, thus the item number 1 has been left out of the bin number 1.

After a division rule $[\ z_{i,1,k}\ ]_2^0 \rightarrow [\ z\ ]_2^+\ \ [\ z_{i,-1,k}\ ]_2^0$ is applied, the two new membranes will behave in a different way. The positively charged membrane will update the weight of its associated subset because a new item has been introduced; meanwhile the neutrally charged membrane can be divided if there still are more items to be added in the bin.

The *generation and calculation stages* continue this way till the object g_{2n+1} set the polarization of the membranes to negative following the instructions of the rule 8 and then the *checking stage* begins.

Before the *checking stage* the system renames and duplicate some objects following the rules 9, 10, and 11 in order to avoid conflicts with the previous stages.

The purpose of this stage is to compare the multiplicities of objects W and $\hat{D}$. This task is carried out by the rules in 12. The objects $t_1, \cdots, t_{2c+1}$ are used as counter in this stage. When the object t_{2c+1} appears in the membrane it means that the stage is over and it sets the polarization to positive using the rule in 15. In the next step if there are any objects W it would mean that the multiplicity of the objects W was greater than the multiplicity of objects $\hat{D}$, that is the weight of the associated subset exceeded the capacity of the bin and so the membrane is dissolved by the rule in 16. Otherwise the weight of the associated subset did not exceeded its capacity and the systems have to move to the next bin. Using the rules in 17, 18, 19, 20 and 21 the computation for the bin number i, $1 \leq i \leq b-1$ ends and the computation for the $i+1$ bin begins.

The computation for the bin number b, the last one, is quite different from the rest. The system skip the *generation stage* for this bin using the rules in 22, 24 and 25 because all the remaining items have to be introduced in this bin. The counters of rule 23 are used to show when the computation for the last bin begins.

The rules in 26 are counters that can eventually produce the object YES. If the object d_{2c+3} appears in a working membrane it would mean that the subsets associated to this membrane during the computation is a solution to the problem.

The counters in 27 will eventually release the object NO in the skin. Observe that the object NO will be produced a step later than the objects YES so following the rules 28 the correct answer in released into the environment.

6 Necessary Resources

The presented family of recognizer P systems that solves the BINPACKING problem is polynomially uniform by Turing machines. It can be observed that the definition of the family is done in a recursive manner from a given instance, in particular from the constants n, b and c. Futhermore the necessary resources to buil an element of the family are:

- Size of the alphabet: $4n^2b - 4n^2 + 4nb + 4bc + 14b - 4n - 2c - 1 \in O(\max\{n, b, c\}^3)$
- Number of membranes: $2 \in \Theta(1)$
- $|\mathcal{M}_1| + |\mathcal{M}_2| = c + 3 \in O(c)$

- Sum of the rules' lengths: $10n^2b + 10n^2 + 110nb + 60cb + 40n + 150b + 20c + 120 \in O(\max\{n, b, c\}^3)$

Note that the instance $u = (n, (\omega_1, \ldots, \omega_n), b, c)$ is introduced in the initial configuration through an input multiset; that is, encoded in an unary representation and, thus, we have that $|u| \in O(\omega_1 + \cdots + \omega_n + c)$.

The number of steps in each stage are the following:

1. Initialition: 1 step
2. For each bin from 1 to $b - 1$:
   - Generation and calculation stages: $2n$ steps
   - Transition to the checking stage: 2 steps
   - Checking stage: $2c$ steps
   - Transition to the next bin: 3 steps
3. For the bin number b: $2c + 6$ steps
4. In the output stage: 2 steps

So the overall number of steps is:

$$1 + (b-1)(2n + 2c + 5) + 2c + 12 = 2nb + 2bc + 5b - 2n + 4 \in O(\max\{nb, nc, bc\}^2)$$

From these discussion we deduce the following results:

Theorem 6.1 $BINPACKING \in \mathbf{PMC}_{\mathcal{AM}}$

Although the next result is a corollary of Theorem 1, we formulate it as another theorem, in order to stress its relevance.

Theorem 6.2 $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$

Proof. It suffices to make the following observations: the BINPACKING problem is $\mathbf{NP}-$complete, $BINPACKING \in \mathbf{PMC}_{\mathcal{AM}}$ and the class $\mathbf{PMC}_{\mathcal{AM}}$ is closed under polybomial-time reduction. □

This theorem can be extended, if we notice that the class $\mathbf{PMC}_{\mathcal{AM}}$ is closed under complement.

Theorem 6.3 $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{AM}}$

7 A CLIPS Session for $n = b = c = 2$

In this section we show a session with the CLIPS simulator presented in [11] for a instance of the BINPACKING problem: $u = (2, (1, 2), 2, 2)$.

```
 CLIPS (V6.10 07/01/98)

CLIPS> (load "SIMULATOR4.clp")

CLIPS> (reset)
CLIPS> (run)

Write the path and the file where the P-system is written:
/home/Fran/binpacking10.clp

                    ******************************
                    * P-SYSTEM SUCESSFULLY LOADED *
                    ******************************

Write the value of the parameter n : 2
Write the value of the parameter c : 2
Write the value of the parameter b : 2

Write the input multiset following the instructions given above:
z 1 , s 1 , z 2 , s 2 , s 2


Configuration number: 0

[environment [multiset ]]

[skin     [children 2]
          [label 1]
          [polarity 0]
          [multiset , f 1 ,]]

[membrane
    [number 2]
    [children ]
    [father 1]
    [label 2]
    [polarity 0]
    [multiset , f 1 , g 0 , C , C , z 1 , s 1 , z 2 , s 2 , s 2 ,]]


Configuration number: 1

[environment [multiset ]]

[skin     [children 2]
          [label 1]
          [polarity 0]
          [multiset , f 2 ,]]

[membrane
    [number 2]
    [children ]
    [father 1]
```

```
    [label 2]
    [polarity 0]
    [multiset , f 2 , g 1 , C , C , z 1 1 1 , s 1 1 1 , z 1 2 2 , s 1 2 2 ,
               s 1 2 2 ,]]
```

Above we have the initial configuration and the first step that initialize the computation.

Now the *generation stage* takes place. In the first configuration of this stage can be seen how it works. We get two membranes, one with the item number 1 in the first bin and the other one with the item number 1 left out of the first bin.

```
Configuration number: 2

[environment [multiset ]]
[skin     [children 3 4]
          [label 1]
          [polarity 0]
          [multiset , f 3 ,]]
[membrane
    [number 4]
    [children ]
    [father 1]
    [label 2]
    [polarity 0]
    [multiset , f 3 , g 2 , C , C , z 1 -1 1 , s 1 0 1 , z 1 1 2 , s 1 1 2 ,
               s 1 1 2 ,]]
[membrane
    [number 3]
    [children ]
    [father 1]
    [label 2]
    [polarity +]
    [multiset , f 3 , g 2 , C , C , z , s 1 0 1 , z 1 1 2 , s 1 1 2 , s 1 1 2 ,]]
```

At the end of the *generation stage* we have four working membranes representing the four possible subsets. Now the system moves to the *checking stage*.

```
Configuration number: 5

[environment [multiset ]]
[skin     [children 7 8 5 6]
          [label 1]
          [polarity 0]
          [multiset , f 6 , ]]
[membrane
    [number 8]
    [children ]
    [father 1]
    [label 2]
    [polarity 0]
    [multiset , f 6 , g 5 , t 0 , C , C , w , z 1 -1 2 , s 1 -1 2 , s 1 -1 2 ,]]
[membrane
    [number 5]
    [children ]
```

```
    [father 1]
    [label 2]
    [polarity 0]
    [multiset , f 6 , g 5 , t 0 , C , C , z 1 -1 1 , s 1 -1 1 , w , w ,]]
[membrane
    [number 6]
    [children ]
    [father 1]
    [label 2]
    [polarity 0]
    [multiset , f 6 , g 5 , t 0 , C , C , z 1 -1 1 , s 1 -1 1 , z 1 -1 2 ,
               s 1 -1 2 , s 1 -1 2 ,]]
[membrane
    [number 7]
    [children ]
    [father 1]
    [label 2]
    [polarity 0]
    [multiset , f 6 , g 5 , t 0 , C , C , w , w , w ,]]
```

At the end of the *checking stage* we can see that in the membrane number 7 there still are one object W so the weight of its associated subset exceeded the capacity of the bin. Thus this membrane will be dissolved in the next step becuase it did not represent a solution to the problem.

```
Configuration number: 12

[environment [multiset ]]
[skin     [children 7 8 5 6]
          [label 1]
          [polarity 0]
          [multiset , f 13 , ]]
[membrane
    [number 8]
    [children ]
    [father 1]
    [label 2]
    [polarity +]
    [multiset , f 13 , G , z 0 , C1 , C1 , Z 1 2 2 , S 1 2 2 , S 1 2 2 ,]]
[membrane
    [number 7]
    [children ]
    [father 1]
    [label 2]
    [polarity +]
    [multiset , f 13 , G , z 0 , C1 , C1 , W ,]]
[membrane
    [number 5]
    [children ]
    [father 1]
    [label 2]
    [polarity +]
    [multiset , f 13 , G , z 0 , C1 , C1 , Z 1 1 1 , S 1 1 1 ,]]
[membrane
```

```
    [number 6]
    [children ]
    [father 1]
    [label 2]
    [polarity +]
    [multiset , f 13 , G , z 0 , C1 , C2 , C1 , Z 1 1 1 , S 1 1 1 , Z 1 2 2 ,
               S 1 2 2 , S 1 2 2 ,]]
```

For the last bin the system skip the *generation stage*, it introduces all the remaining items in the bin and goes directly to the *checking stage*.

```
Configuration number: 16

[environment [multiset ]]
[skin     [children 8 5 6]
          [label 1]
          [polarity 0]
          [multiset , f 17 , ]]
[membrane
    [number 8]
    [children ]
    [father 1]
    [label 2]
    [polarity -]
    [multiset , t 1 , d 2 , C1 , C2 , C1 , C2 , W , W ,]]
[membrane
    [number 5]
    [children ]
    [father 1]
    [label 2]
    [polarity -]
    [multiset , t 1 , d 2 , C1 , C2 , C1 , C2 , W ,]]
[membrane
    [number 6]
    [children ]
    [father 1]
    [label 2]
    [polarity -]
    [multiset , t 1 , d 2 , C1 , C2 , C1 , C2 , W , W , W ,]]
```

When the *checking stage* ends the system detects that one membrane of the three remaining membranes must be dissolved because it did not encodes a solution of the problem. Nevertheless the other two membranes do enconde a solution and they released objects YES in the skin.

```
Configuration number: 24

[environment [multiset ]]
[skin     [children ]
          [label 1]
          [polarity 0]
          [multiset , f 25  , YES ,  YES , ]]
```

At the last step of the computation the system released the answer into the environment.

```
Configuration number: 25

[environment [multiset YES ,]]
[skin     [children ]
          [label 1]
          [polarity +]
          [multiset , NO , YES , ]]


The system has reached a halting configuration in the step number 25
and the element YES has been released into the environment.
```

8 Conclusions

In this paper we have presented an effective solution for the BINPACKING problem through a family of recognizer P systems with active membranes. This has been done in the framework of complexity classes in cellular computing with membranes.

The design presented here is very similar to the solutions to numerial **NP**-complete problems studied in [9], [10] and [12]. The strong similarities in their designs show that the idea of a *cellular programming language is posible*, indicating some "subroutines" that can be used in a variety of situations and therefore could be useful for attacking new problems in the future. As an example of the usefulness of the subroutines outlined in [12], let us see how the design of the solutions for the BINPACKING would look like:

BINPACKING
for i=1, . . . , b-1 do
$gen - subsets(n_i)$
$calc - weight(n_i)$
$rename$
$check - weight$
$marker - leq$
$counter(n)$
$clean - dissolve$
end for.
$calc - weight(n_b)$
$rename$
$check - weight$
$marker - leq$
$counter(n)$
$clean - dissolve$
$detector$
$answer$

The CLIPS simulator for P systems presented in [11] is a very useful tool that has helped to debug the design and to understand better how the P systems from the family **Π** work.

Acknowledgement. This work is supported by the Ministerio de Ciencia y Tecnología of Spain, by the *Plan Nacional de I+D+I (2000–2003)* (TIC2002-04220-C03-01),

cofinanced by FEDER funds, and by a FPI fellowship (of the second author) from the University of Seville.

References

[1] Cordón-Franco, A., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Sancho-Caparrini, F.: A Prolog simulator for deterministic P systems with active membranes, *New Generation Computing*, in press.

[2] Păun, Gh.: Computing with membranes, *Journal of Computer and Systems Sciences*, 61, 1 (2000), 108–143.

[3] Păun, Gh.: *Membrane Computing. An Introduction*, Springer-Verlag, 2002.

[4] Păun, Gh., Rozenberg, G.: A guide to membrane computing, *Theoretical Computer Sciences*, 287 (2002), 73–100.

[5] Păun, Gh., Rozenberg, G., Salomaa, A.: Membrane computing with external output, *Fundamenta Informaticae*, 41, 3 (2000), 313–340.

[6] Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: Solving VALIDITY problem by active membranes with input, *Proceedings of the Brainstorming Week on Membrane Computing*, M. Cavaliere, C. Martin-Vide, and Gh. Păun (eds), Report GRLMC 26/03, 279–290.

[7] Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: *Teoría de la Complejidad en modelos de computacion celular con membranas*, Editorial Kronos, 2002.

[8] Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: A polynomial complexity class in P systems using membrane division, *Proceedings of the 5th Workshop on Descriptional Complexity of Formal Systems*, E. Csuhaj-Varjú, C. Kintala, D. Wotschke, and Gy. Vaszyl (eds.), 2003, 284–294.

[9] Pérez-Jiménez, M.J., Riscos-Núñez, A.: Solving the Subset-Sum problem by active membranes, *New Generation Computing*, in press.

[10] Pérez-Jiménez, M.J., Riscos-Núñez, A.: A linear-time solution for the Knapsack problem using active membranes, *Lecture Notes in Computer Science*, 2933 (2004), 140–152.

[11] Pérez-Jiménez, M.J., Romero-Campero, F.J.: A CLIPS Simulator for Recognizer P Systems with Active Membranes, in this volume.

[12] Riscos-Núñez, A., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: Towards a programming language in cellular computing, in this volume.

[13] CLIPS Web Page: http:// www.ghg.net/clips/CLIPS.html

[14] The P Systems Web Page: http://psytems.disco.unimib.it/