A Logarithmic Bound for Solving Subset Sum with P Systems

Daniel Díaz-Pernil, Miguel A. Gutiérrez-Naranjo,
Mario J. Pérez-Jiménez, and Agustín Riscos-Núñez

Research Group on Natural Computing
University of Sevilla, Spain
{sbdani,magutier,marper,ariscosn}@us.es

Abstract. The aim of our paper is twofold. On one hand we prove the ability of polarizationless P systems with dissolution and with division rules for non-elementary membranes to solve **NP**-complete problems in a polynomial number of steps, and we do this by presenting a solution to the Subset Sum problem. On the other hand, we improve some similar results obtained for different models of P systems by reducing the number of steps and the necessary resources to be of a logarithmic order with respect to k (recall that n and k are the two parameters used to indicate the size of an instance of the Subset Sum problem).

As the model we work with does not allow cooperative rules and does not consider the membranes to have an associated polarization, the strategy that we will follow consists on using objects to represent the weights of the subsets through their multiplicities, and comparing the number of objects against a fixed number of membranes. More precisely, we will generate k membranes in $\log k$ steps.

1 Introduction

This paper is the continuation of a series of results on Complexity Classes in Membrane Computing that are trying to establish the relevance, in terms of computing power, of each one of the possible features of a P system (see [3]).

The Subset Sum problem is a well-known **NP**-complete problem which can be formulated as follows: *Given a finite set A, a weight function, $w : A \to \mathbb{N}$, and a constant $k \in \mathbb{N}$, determine whether or not there exists a subset $B \subseteq A$ such that $w(B) = k$.* It has been a matter of study in Membrane Computing several times, being mainly used to prove the ability of different P system models in order to solve problems from the **NP** class in a polynomial time.

This speed-up is achieved by trading space for time, in the sense that the considered models allow that an exponential amount of membranes can be produced by a P system in a polynomial number of steps. For example, solutions to the Subset Sum problem working in a number of steps which is linear with respect to the parameters n and k have been designed using P systems with active membranes [9], using tissue P systems with cell division [2], and using P systems with membrane creation [4].

In this paper we work with P systems using division of non-elementary membranes and dissolution rules. Our aim goes beyond adding this P system model to the above mentioned list; we improve previous complexity results by solving the Subset Sum problem in a linear number of steps with respect to n and $\log k$. We also improve the pre-computation process, as the initial resources are also bounded by $\log k$.

The paper is structured as follows: in the next section we present the formal framework, i.e., we recall the definition of recognizing P systems, the P system model used along the paper is settled and the class $\mathbf{PMC}_{\mathcal{AM}^0(+d,+ne)}$ is presented. In Section 3, our design of the solution of the Subset Sum problem is presented and some conclusions are given in the last section.

2 Formal Framework

In this paper we are using cellular systems for attacking the resolution of decision problems. This means that for each instance of a problem that we try to solve, we are only interested in obtaining a Boolean answer (*Yes* or *No*). Therefore, the P system can behave as a *black box* to which the user supplies an input and from which an affirmative or negative answer is received. This is indeed the motivation for defining the concept of *recognizing P systems* (introduced in [13]).

2.1 Recognizing P Systems

Let us recall that a decision problem, X, is a pair (I_X, θ_X) where I_X is a language over an alphabet whose elements are called *instances* and θ_X is a total Boolean function over I_X. If u is an instance of the problem X such that $\theta_X(u) = 1$ (respectively, $\theta_X(u) = 0$), then we say that the answer to the problem for the instance considered is *Yes* (respectively, *No*).

Keeping this in mind, recognizing P systems are defined as a special class of membrane systems that will be used to solve decision problems, in the framework of the complexity classes theory. Note that this definition is stated informally, and it can be adapted for any kind of membrane system paradigm.

A recognizing P system is a P system with input and with external output having two distinguished objects `yes` and `no` in its working alphabet such that:

- All computations halt.
- If $\mathcal{C}$ is a computation of Π, then either the object `yes` or the object `no` (but not both) must have been released into the environment, and only in the last step of the computation.

2.2 The P System Model

The power of membrane division as a tool for efficiently solving **NP** problems in Membrane Computing has been widely proved. Many examples of designs of P systems solving **NP**-complete problems have been proposed in the framework of P systems with active membranes with two polarizations and three polarizations

and in the framework of P systems with non-elementary membrane division. The key of such solutions is the creation of an exponential amount of workspace (membranes) in a polynomial time.

In the literature, one can find two quite different rules for performing membrane division. On the one hand, in [7], P systems with active membranes were presented. In this model new membranes were obtained through the process of *mitosis* (membrane division). In these devices membranes have polarizations, one of the "electrical charges" $0, -, +$, and several times the problem was formulated whether or not these polarizations are necessary in order to obtain polynomial solutions to **NP**–complete problems. The last result is that from [1], where one proves that two polarizations suffice.

P systems with active membranes have been successfully used to design (uniform) solutions to well-known **NP**–complete problems, such as SAT [13], *Subset Sum* [9], *Knapsack* [10], *Bin Packing* [11], *Partition* [5], and the *Common Algorithmic Problem* [12].

The syntactic representation of membrane division rule is

$$[\,a\,]_h^{e_1} \rightarrow [\,b\,]_h^{e_2}\ [\,c\,]_h^{e_3} \tag{1}$$

where h is a label, e_1,e_2 and e_3 are electrical charges and a,b and c are objects. The interpretation is well-known: An elementary membrane can be divided into two membranes with the same label, possibly transforming some objects and changing the electrical charge. All objects present in the membrane except the object triggering the rule are copied into both new membranes.

In [6], a variant of this rule was used in which the polarization was dropped:

$$[\,a\,]_h \rightarrow [\,b\,]_h\ [\,c\,]_h. \tag{2}$$

In both cases (with and without polarizations) the key point is that the membranes are always elementary membranes. In the literature, there also exist rules for the division of non-elementary polarizationless membranes, as

$$[\,[\,]_{h_1}\,[\,]_{h_2}\,]_{h_0} \rightarrow [\,[\,]_{h_1}\,]_{h_0}\,[\,[\,]_{h_2}\,]_{h_0} \tag{3}$$

where h_0, h_1 and h_2 are labels. There exists an important difference with respect to elementary membrane division: in the case of (3), the rule is not triggered by the occurrence of an object inside a membrane, but by the membrane structure instead. This point has a crucial importance in the design of solutions, since a membrane can be divided by the corresponding rule even if there are no objects inside it.

According to the representation (3), the membrane h_0 divides into two new membranes also with label h_0 and all the information (objects and membranes) different from membranes h_1 and h_2 inside is duplicated.

In this paper we use a type of membrane division which is syntactically equivalent to (2)

$$[\,a\,]_h \rightarrow [\,b\,]_h\ [\,c\,]_h, \tag{4}$$

but we will consider a semantic difference; the dividing membrane can be elementary or non-elementary and after the division, all the objects and membranes

inside the dividing membrane are duplicated, except the object a that triggers the rule, which appears in the new membranes possibly modified (represented as objects b and c).

In this paper we work with a variant of P systems with active membranes which we call with weak division, and that does not use polarizations.

Definition 1. *A P system with active membranes with weak division is a P system with Γ as working alphabet, with H as the finite set of labels for membranes, and where the rules are of the following forms:*

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

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

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

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

(e) $[\, a \,]_h \to [\, b \,]_h\, [\, c \,]_h$ *for* $h \in H$, $a, b, c \in \Gamma$. *A membrane can be divided into two membranes with the same label, possibly transforming some objects. The content of the membrane is duplicated. The membrane can be elementary or not.*

These rules are applied according to the following principles:

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

Let us note that in this framework we work without cooperation, without priorities, with weak division, and without changing the labels of membranes.

In this paper we work within the model of *polarizationless P systems using weak division of non-elementary membranes and dissolution.* Let $\mathcal{AM}^0(+d, +ne)$ be the class of such systems.

2.3 The Class $\mathbf{PMC}_{\mathcal{AM}^0(+d,+ne)}$

Definition 2. *We say that a decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\mathbf{\Pi} = \{\Pi(n) : n \in \mathbb{N}\}$ of recognizing P systems from $\mathcal{AM}^0(+d, +ne)$ if the following holds:*

- *The family $\mathbf{\Pi}$ is* polynomially uniform *by Turing machines, that is, there exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(n)$ from $n \in \mathbb{N}$.*
- *There exists a pair (cod, s) of polynomial-time computable functions over I_X such that:*
  - *for each instance $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi(s(u))$;*
  - *the family $\mathbf{\Pi}$ is* polynomially bounded *with regard to (X, cod, s), that is, there exists a polynomial function p, such that for each $u \in I_X$ every computation of $\Pi(s(u))$ with input $cod(u)$ is halting and, moreover, it performs at most $p(|u|)$ steps;*
  - *the family $\mathbf{\Pi}$ is* sound *with regard to (X, cod, s), that is, for each $u \in I_X$, if there exists an accepting computation of $\Pi(s(u))$ with input $cod(u)$, then $\theta_X(u) = 1$;*
  - *the family $\mathbf{\Pi}$ is* complete *with regard to (X, cod, s), that is, for each $u \in I_X$, if $\theta_X(u) = 1$, then every computation of $\Pi(s(u))$ with input $cod(u)$ is an accepting one.*

In the above definition we have imposed to every P system $\Pi(n)$ a *confluent* condition, in the following sense: every computation of a system with the *same* input multiset must always give the *same* answer. The pair of functions (cod, s) is called a *polynomial encoding* of the problem in the family of P systems.

We denote by $\mathbf{PMC}_{\mathcal{AM}^0(+d,+ne)}$ the set of all decision problems which can be solved by means of recognizing polarizationless P systems using division of non-elementary membranes and dissolution in polynomial time.

3 Designing the Solution to Subset Sum

In this section we address the resolution of the problem following a brute force algorithm, implemented in the framework of recognizing P systems from the $\mathcal{AM}^0(+d, +ne)$ class. 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 labeled by e is generated via membrane division.
- *Calculation stage*: in each membrane the weight of the associated subset is calculated (using the auxiliary membranes $e_0, \ldots, e_n$).
- *Checking stage*: in each membrane it is checked whether the weight of its associated subset is exactly k (using the auxiliary membranes ch).
- *Output stage*: the system sends out the answer to the environment, according to the result of the checking stage.

Let us now present a family of recognizing P systems from the $\mathcal{AM}^0(+d,+ne)$ class that solves Subset Sum, according to Definition 2.

We shall use a tuple $(n,(w_1,\ldots,w_n),k)$ to represent an instance of the Subset Sum 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. Let $g:\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ be a function defined by

$$g(n,k)=\frac{(n+k)(n+k+1)}{2}+n$$

This function is primitive recursive and bijective between $\mathbb{N}\times\mathbb{N}$ and $\mathbb{N}$ and computable in polynomial time. We define the polynomially computable function $s(u)=g(n,k)$.

We shall provide a family of P systems where each P system solves all the instances of the Subset Sum problem with the same size. Let us consider the binary decomposition of k, $\Sigma_{i\in I}2^i=k$, where the indices $i\in I$ indicate the positions of the binary expression of k where a 1 occurs. Let $I'=\{1,\ldots,\lfloor\log k\rfloor\}-I$ be the complementary set, that is, the positions where a 0 occurs. This binary encoding of k, together with the weight function w of the concrete instance, will be provided via an input multiset determined by the function cod as follows:

$$cod(u)=cod_1(u)\cup cod_2(u),$$

$$\begin{aligned}\text{where } cod_1(u)&=\{\{b_i^{w_i}\,:\,1\le i\le n\}\}\text{ and}\\ cod_2(u)&=\{\{c_j\,:\,j\in I\}\}\cup\{\{c'_j\,:\,j\in I'\}\}\end{aligned}$$

Next, we shall provide a family $\mathbf{\Pi}=\{\Pi(g(n,k))\,:\,n,k\in\mathbb{N}\}$ of recognizing P systems which solve the Subset Sum problem in a number of steps being of $O(n+\log k)$ order. We shall indicate for each system of the family its initial configuration and its set of rules. We shall present the list of rules divided by groups, and we shall provide for each of them some comments about the way their rules work.

Let us consider an arbitrary pair $(n,k)\in\mathbb{N}\times\mathbb{N}$. The system $\Pi(g(n,k))$ is determined by the tuple $(\Gamma,\Sigma,\mu,M,\mathcal{R},i_{in},i_0)$, that is described next:

- Alphabet:

$$\begin{aligned}\Gamma=\Sigma\cup&\{b_i^+,b_i^-,b_i^=,d_i,d_i^+,d_i^-,p_i,q_i\,:\,i=1,\ldots,n\}\\ \cup&\{g_0,\ldots,g_{2\lfloor\log k\rfloor+2},h_0,\ldots,h_{2\lfloor\log k\rfloor+2n+8},l_0,\ldots,l_{2\lfloor\log k\rfloor+2n+10}\}\\ \cup&\{v_0,\ldots,v_{2\lfloor\log k\rfloor+2n+12}\}\\ \cup&\{w_0,\ldots,w_{2\lfloor\log k\rfloor+2n+18}\}\\ \cup&\{x_0,\ldots,x_{2\lfloor\log k\rfloor+2n+15},z_0,\ldots,z_{2\lfloor\log k\rfloor+2n+7}\}\\ \cup&\{s,\texttt{yes},\texttt{no},Trash\}\end{aligned}$$

- Input alphabet: $\Sigma(n,k)=\{b_1,\ldots,b_n,c_0,\ldots,c_{\lfloor\log k\rfloor},c'_0,\ldots,c'_{\lfloor\log k\rfloor}\}$.

The initial configuration consists of $n+\lfloor\log k\rfloor+9$ membranes, arranged as shown in Figure 1. Formally, the membrane structure μ is

$$[[[[[[\overset{n}{\cdots}\,[[[[[]_{ch}\cdots[]_{ch}]_{a_1}]_{a_2}]_{e_0}]_{e_1}\ \overset{n}{\cdots}]_{e_n}]_{a_3}\,[]_c]_{a_4}]_e]_f]_{skin}$$

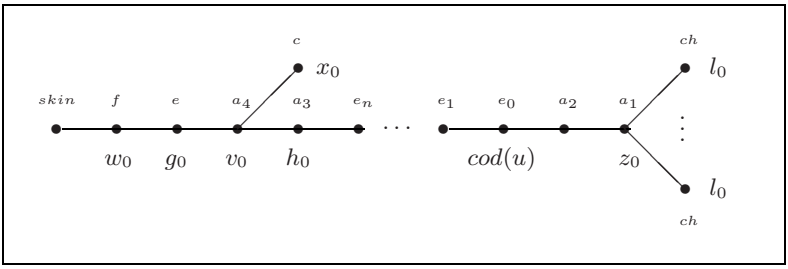

Fig. 1. Initial Configuration

where there are exactly $\lfloor \log k \rfloor + 1$ copies of membrane $[\,]_{ch}$.

Roughly speaking (more precise explanations will be given for the rules), we can classify the membranes according to their role as follows:

- $n+2$ membranes that take care of the generation stage, namely those labeled by $e_0, e_1, \ldots e_n$ and e.
- $\lfloor \log k \rfloor + 3$ membranes that take care of preparing and implementing the checking stage, namely those labeled by ch, a_1 and a_2.
- 4 membranes that take care of the answer stage, handling and synchronizing the results of the checking, namely those labeled by a_3, a_4, c and f.

• The initial multisets are:

$$M(f) = \{\{w_0\}\};\ M(e) = \{\{g_0\}\};\ M(a_4) = \{\{v_0\}\};\ M(a_3) = \{\{h_0\}\};$$

$$M(c) = \{\{x_0\}\};\ M(a_1) = \{\{z_0\}\};\ M(ch) = \{\{l_0\}\}$$

$$M(skin) = M(a_2) = M(e_0) = \cdots = M(e_n) = \emptyset$$

• The input membrane is $i_{in} = e_0$, and the output region is the environment ($i_0 = env$).

First task: generate k membranes ch. At the beginning of the computation, k membranes ch will be generated inside the innermost region of the structure.

The strategy works as follows:

1. Initially, there are $\lfloor \log k \rfloor$ membranes ch in the region a_1, and the input multiset is located in region e_0 (recall that $cod_2(u)$ consists of $\lfloor \log k \rfloor$ objects c_i or c'_i representing the binary encoding of k).
2. In the first $\lfloor \log k \rfloor$ steps, the objects from $cod_2(u)$ get into membrane a_2 (the objects enter one by one membrane a_2). Simultaneously, the counter z_i is evolving inside membrane a_1 and dissolves it at the $\lfloor \log k \rfloor$ step.
3. Thus, in the next step each element from $cod_2(u)$ will go inside a membrane ch (all objects go in parallel into different membranes in a one-to-one manner).

4. Objects c'_i will dissolve the membranes where they enter, while each object c_i will generate by division 2^i membranes ch.
5. After at most $\lfloor \log k \rfloor$ further steps all divisions have been completed, and the number of membranes ch is exactly k.

Membrane a_2 will not be divided until the generation and weight calculation stages have been completed, acting as a separator between objects from $cod_1(u)$ and membranes ch.

Set (A1).
$$\left.\begin{array}{l} c_i[\,]_{a_2} \to [c_i]_{a_2} \\ c'_i[\,]_{a_2} \to [c'_i]_{a_2} \\ \\ c_i[\,]_{ch} \to [c_i]_{ch} \\ c'_i[\,]_{ch} \to [c'_i]_{ch} \\ [c'_i\,]_{ch} \to Trash \end{array}\right\} \text{for } i \in \{0, \ldots, \lfloor \log k \rfloor\}.$$

Set (A2).
$$\begin{array}{ll} [c_0 \to Trash]_{ch} & \\ [c_i]_{ch} \to [c_{i-1}]_{ch}\,[c_{i-1}]_{ch} & \text{for } i = 1, \ldots, \lfloor \log k \rfloor \\ [z_i \to z_{i+1}]_{a_1} & \text{for } i = 0, \ldots, \lfloor \log k \rfloor - 1 \\ [z_{\lfloor \log k \rfloor}]_{a_1} \to z_{\lfloor \log k \rfloor + 1} & \\ [g_i \to g_{i+1}]_e & \text{for } i = 0, \ldots, 2\lfloor \log k \rfloor + 1 \\ [g_{2\lfloor \log k \rfloor + 2} \to d_1 s]_e & \end{array}$$

In the last step of this stage, the counter g_i produces the objects d_1 and s which will trigger the beginning of the next stage.

Set(B).
$$\left.\begin{array}{l} [w_i \to w_{i+1}]_f \\ [v_i \to v_{i+1}]_{a_4} \\ [h_i \to h_{i+1}]_{a_3} \\ [x_i \to x_{i+1}]_c \\ [l_i \to l_{i+1}]_{ch} \end{array}\right\} \text{for } i \in \{0, \ldots, 2\lfloor \log k \rfloor + 2\}.$$
$$[z_i \to z_{i+1}]_{a_2} \quad \text{for } i \in \{\lfloor \log k \rfloor + 1, \ldots, 2\lfloor \log k \rfloor + 2\}.$$

The rest of the counters simply increase their indices in this stage. (See Fig. 2.)

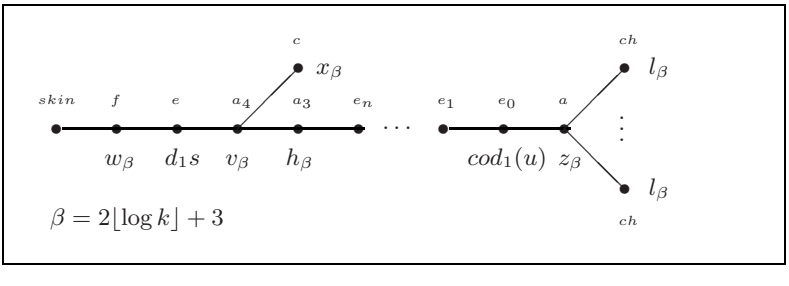

Fig. 2. TIME $2\lfloor \log k \rfloor + 3$

Second task: generate 2^n membranes e. Objects d_i residing inside membrane(s) e will produce n consecutive divisions, thus yielding 2^n copies of membrane e. To each one of them, a subset of A is associated in the following way: after each division, the membranes where object p_i occurs correspond to subsets of A containing a_i, and conversely, membranes where q_i occurs will be associated with subsets not containing a_i.

Set (C).

$$
\begin{array}{ll}
[d_i]_e \to [d_i^+]_e[d_i^-]_e & \text{for } i = 1, \dots n \\
[d_i^+ \to p_i d_{i+1}]_e & \text{for } i = 1, \dots n-1 \\
[d_i^- \to q_i d_{i+1}]_e & \text{for } i = 1, \dots n-1 \\
[d_n^+ \to p_n]_e & \\
[d_n^- \to q_n]_e &
\end{array}
$$

Membrane divisions take place every two steps, so in the $(2\lfloor \log k \rfloor + 2n + 2)$-th step there will be 2^n membranes e.

Set (D).

$$
\begin{array}{ll}
s\,[\,]_{a_i} \to [s]_{a_i} & \text{for } i = 3, 4 \\
s\,[\,]_{e_i} \to [s]_{e_i} & \text{for } i = 0, \dots, n \\
[s]_{e_0} \to Trash & \\
p_j\,[\,]_{a_i} \to [p_j]_{a_i} & \text{for } i = 3, 4 \quad j = 1, \dots, n \\
p_j\,[\,]_{e_i} \to [p_j]_{e_i} & \text{for } j = 1, \dots, n \quad i = j, \dots, n \\
[p_i \to q_i]_{e_i} & \text{for } i = 1, \dots, n \\
q_j\,[\,]_{a_i} \to [q_j]_{a_i} & \text{for } i = 3, 4 \quad j = 1, \dots, n \\
q_j\,[\,]_{e_i} \to [q_j]_{e_i} & \text{for } j = 1, \dots, n \quad i = j, \dots, n \\
[q_i]_{e_i} \to Trash & \text{for } i = 1, \dots, n
\end{array}
$$

While the divisions are being carried out, objects s, p_j and q_j, for $j = 1, \dots, n$, travel into inner membranes (recall that whenever membrane e gets divided, the internal nested structure of membranes e_i is duplicated). In the $(2\lfloor \log k \rfloor + n + 2)$-th step, an object s arrives to every membrane e_0. This object dissolves the membrane in the next step, and therefore in the $(2\lfloor \log k \rfloor + n + 3)$-th step we find inside every membrane e_1 the multiset $cod_1(u)$, and in this moment the weight calculation stage begins (see rules in **Set (E)**).

As we said before, objects p_j and q_j are traveling into inner membranes, until they reach e_j. This is done in such a way that in the $(2\lfloor \log k \rfloor + n + 3)$-th step there is in each membrane e_1 either an object p_1 or an object q_1, in addition to the multiset $cod_1(u)$.

Before going on, let us state two points. First, recall that in the input multiset, introduced in e_0 at the beginning of the computation, there are $w(a_i)$ copies of b_i, for $i = 1, \dots, n$. Second, let us note that objects q_i dissolve membrane e_i immediately after arriving to it, while objects p_i take two steps to dissolve membrane e_i (first they are transformed into q_i and in the next step the dissolution takes place).

Set (E).

$$
\begin{array}{lll}
[b_1 \to b_1^+]_{e_1} & & \\
[b_{i+1} \to b_{i+1}^-]_{e_i} & \text{for } i = 1, \ldots, n-1 & \\
[b_{i+2} \to b_{i+2}^=]_{e_i} & \text{for } i = 1, \ldots, n-2 & \\
[b_{i+3} \to b_{i+3}^=]_{e_i} & \text{for } i = 1, \ldots, n-3 & \\
[b_i^+ \to b_0]_{e_i} & \text{for } i = 1, \ldots, n & \\
[b_i^+ \to Trash]_{e_j} & \text{for } i = 1, \ldots, n-1, & j = i+1 \\
[b_i^- \to b_i^+]_{e_i} & \text{for } i = 2, \ldots, n & \\
[b_{i+1}^- \to b_{i+1}^+]_{e_i} & \text{for } i = 1, \ldots, n-1 & \\
[b_i^= \to b_i^+]_{e_i} & \text{for } i = 3, \ldots, n & \\
[b_{i+1}^= \to b_{i+1}^-]_{e_i} & \text{for } i = 2, \ldots, n-1 & \\
[b_{i+2}^= \to b_{i+2}^-]_{e_i} & \text{for } i = 1, \ldots, n-2 & \\
[b_n^+ \to Trash]_{a_3} & &
\end{array}
$$

The basic strategy consists on allowing objects b_i to get transformed into objects b_0 only if the element $a_i \in A$ belongs to the associated multiset.

Let us summarize informally the evolution of objects b_i for all possible cases. Recall that in the $(2\lfloor \log k \rfloor + 2)$-th step, the counter g_i produces an object s in membrane e:

- At step $t = 2\lfloor \log k \rfloor + 3$ object s enters in e_n and either d_1^+ or d_1^- appear in each one of the two existing copies of membrane e.
- At step $t = 2\lfloor \log k \rfloor + 4$ object s enters in e_{n-1} and either p_1 or q_1 appear in membranes e.
- At step $t = 2\lfloor \log k \rfloor + 5$, after the second division has been carried out, there are 4 membranes labeled by e. Object s enters in e_{n-2} (this happens in all 4 copies) and p_1 or q_1 get into e_n (there are two of each).
- ...
- At step $t = 2\lfloor \log k \rfloor + n + 3$ object s arrives into e_0, and p_1 or q_1 enter in e_2.
- At step $t = 2\lfloor \log k \rfloor + n + 4$ object s dissolves e_0 (and hence objects b_i are moved to e_1), and p_1 or q_1 arrive into e_1.
- At step $t = 2\lfloor \log k \rfloor + n + 5$ objects b_1, b_2 and b_3 have been transformed in b_1^+, b_2^- and $b_3^=$, respectively, and they will be located either in e_1 (if the membrane contained an object p_1) or in e_2 (if there was an object q_1 in e_1). Besides, in the same step p_2 or q_2 get into e_2.
- At step $t = 2\lfloor \log k \rfloor + n + 6$
  - Objects b_1^+ evolve to b_0 (if they were in e_1) or to $Trash$ (if they were in e_2).
  - Objects b_2^- evolve to b_2^+.
  - Objects $b_3^=$ have been transformed into b_3^- (both those that were in e_2 and those in e_1).
  - All the objects b_i^α ($i = 1, \ldots, n$ and $\alpha \in \{+, -, =\}$) will be located either in membrane e_2 (if the latter contained an object p_2) or in e_3 (if there was an object q_2 in e_2).
  - Besides, in this moment p_3 or q_3 get into e_3.

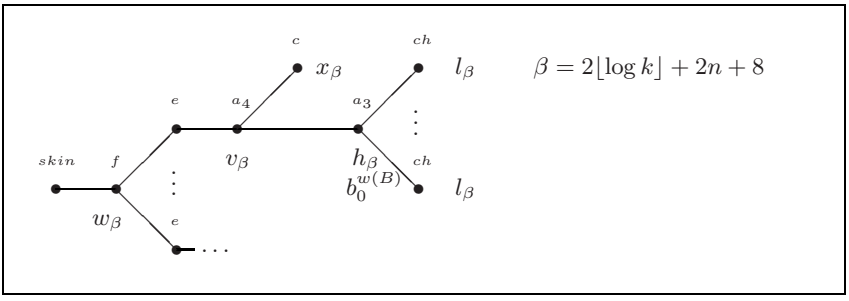

Fig. 3. TIME $2\lfloor \log k \rfloor + 2n + 8$

The design has been adjusted in such a way that in the moment when objects p_i and q_i arrive into membranes e_i it happens that the objects b_j^α ($j = i, \ldots, n$ and $\alpha \in \{+, -, =\}$) are located in e_i in half of the membranes or in e_{i+1} in the rest of membranes. In the next step there will be objects b_i^+ in e_i only for those cases where there was an object p_i, and hence the weight of element $a_i \in A$ should be added to the weight of the associated multiset (that is, $w(a_i)$ copies of b_0 will be produced in those membranes).

Set (F).
$$\left.\begin{array}{l} [w_i \to w_{i+1}]_f \\ [v_i \to v_{i+1}]_{a_4} \\ [h_i \to h_{i+1}]_{a_3} \\ [x_i \to x_{i+1}]_c \\ [z_i \to z_{i+1}]_{a_2} \\ [l_i \to l_{i+1}]_{ch} \end{array}\right\} \text{for } i \in \{2\lfloor \log k \rfloor + 3, \ldots, 2\lfloor \log k \rfloor + 2n + 6\}.$$
$$[z_{2\lfloor \log k \rfloor + 2n + 7}]_{a_2} \to Trash$$

The rest of the counters simply increase their indices during this stage. At the end of the stage, in the $(2\lfloor \log k \rfloor + 2n + 7)$-th step, z_i will dissolve all membranes a_2. Therefore, in the next step we have 2^n membranes labeled by e, and inside them (more precisely, inside membranes a_3) we have multisets of objects b_0 encoding the weights of all possible subsets $B \subseteq A$ (each membrane encodes a different subset) and also exactly k copies of membrane ch, see Fig. 3.

Third task: compare k to the weight of each subset. We shall focus next on the checking stage. That is, the system has to check in all membranes a_3 if the number of objects b_0 (encoding the weight of the associated subset) matches or not the parameter k (represented as the number of membranes ch). This task is performed by the following set of rules (for the sake of simplicity, we denote $\beta = 2\lfloor \log k \rfloor + 2n + 8$):

Set (G).
$$b_0\,[\,]_{ch} \to [c^*]_{ch}$$
$$[b_0 \to u_1]_{a_4}$$
$$[c^*]_{ch} \to Trash$$
$$[h_\beta]_{a_3} \to Trash$$

At the step $t = \beta$, objects b_0 get into membranes ch, and simultaneously membrane a_3 is dissolved. There are three possible situations:

1. There are exactly k objects b_0. In this case at step $t = \beta + 1$ there will not be any object b_0 remaining, and all membranes ch have been dissolved.
2. The number of objects b_0 is lower than k. In this case at step $t = \beta + 1$ there will not be any object b_0 remaining, but there will be some membranes ch that have not been dissolved (because no object b_0 entered them).
3. The number of objects b_0 is greater than k. In this case there are some objects b_0 that could not get inside a membrane ch (recall that the rules are applied in a maximal parallel way, but for each membrane only one object can cross it at a time).

In the second case, inside each membrane ch that has not been dissolved the rules $[l_{\beta+1} \to l_{\beta+2}]_{ch}$ and $[l_{\beta+2}]_{ch} \to u_2$ are applied in the steps $t = \beta + 1$ and $t = \beta + 2$, respectively. Hence at step $t = \beta + 3$ there will be an object u_2 in a_4.

In the third case, the exceeding objects b_0 may, nondeterministically, either get into a membrane ch (avoiding that the dissolution rule is applied to that membrane) or evolve into object u_1. Irrespectively of the nondeterministic choice, we know that there will be no more objects b_0 in a_4 at step $t = \beta + 2$.

Of course, during this stage the rest of the counters continue evolving:

Set (H). $[l_{\beta+i-1} \to l_{\beta+i}]_{ch}$ for $i = 0, 1, 2$
$[v_{\beta+i-1} \to v_{\beta+i}]_{a_4}$ for $i = 0, \ldots, 4$
$[x_{\beta+i-1} \to x_{\beta+i}]_c$ for $i = 0, \ldots, 7$
$[w_{\beta+i-1} \to w_{\beta+i}]_f$ for $i = 0, \ldots, 10$

The next set of rules guarantees that in every membrane where the weight of the associated subset was different from k (and only in such membranes) there will be some objects u_3.

Set (I1). $[u_i \to u_{i+1}]_{a_4}$ for $i = 1, 2$
$[l_{\beta+2}]_{ch} \to u_2$
$[l_{\beta+2} \to u_3]_{a_4}$
$[c^* \to u_3]_{a_4}$

These objects u_3, being in membrane a_4, will go into membranes c and dissolve them. We have here a similar situation as before, as there may be several objects u_3 willing to go into a membrane c. The counter v_i takes care of dissolving membrane a_4 so that any exceeding object u_3 will be moved to membrane e and subsequently transformed into $Trash$.

Set (I2). $u_3\,[\,]_c \to [u_4]_c$
$[v_{\beta+4}]_{a_4} \to Trash$
$[u_3 \to Trash]_e$
$[u_4 \to u_5]_c$
$[u_5]_c \to Trash$

Final task: answer stage. Therefore, only in the branches where the number of objects b_0 were equal to k we have a membrane c inside membrane e at step $\beta + 7$. Besides, we also have a counter w_i evolving in membrane f:

- If the instance of the Subset Sum problem has an affirmative answer, i.e., if there exists a subset of A whose weight is k, then in the step $\beta + 7$ there will be a membrane e with a membrane c inside and an object $x_{\beta+7}$ in it. This object will produce an object `yes` which will dissolve his way out to the environment.

  On the contrary, if the instance has a negative answer, then there will not exist any membrane c in the system in the step $\beta + 7$ and the object `yes` will not be produced. Hence, the membrane f will not be dissolved by `yes` and when the counter w_i reaches $w_{\beta+10}$, an object `no` will appear and will be sent to the environment.

The set of rules is the following one:

Set (J).
$$[x_{\beta+7}]_c \rightarrow yes$$
$$[yes]_e \rightarrow yes$$
$$[yes]_f \rightarrow yes$$
$$[yes]_{skin} \rightarrow yes\,[\,]_{skin}$$
$$[w_{\beta+10}]_f \rightarrow no$$
$$[no]_{skin} \rightarrow no\,[\,]_{skin}$$

Consequently, if the answer is affirmative the P system halts after $\beta+11$ steps and otherwise after $\beta + 12$ steps.

4 Conclusions

In this paper we have combined different techniques for designing P systems in order to get a uniform family of P systems that solves the Subset Sum problem in the framework of P systems with weak division, with dissolution and without polarization. The main contribution of this paper is related to the Complexity Theory of P systems. The best solution of the **NP**-complete problem Subset Sum in any P system model up to now was linear in both input parameters n and k. In this paper we show that the dependency on k can be significantly reduced, since we show a solution where the resources and the number of steps are of a logarithmic order with respect to k.

Acknowledgement

The authors acknowledge the support of the project TIN2006-13425 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and the support of the project of excellence TIC-581 of the Junta de Andalucía.

References

1. Alhazov, A., Freund, R., Păun, G.: P Systems with Active Membranes and Two Polarizations. In: Păun, G., Riscos-Núñez, A., Romero-Jiménez, A., Sancho-Caparrini, F. (eds.) Proc. Second Brainstorming Week on Membrane Computing, Report RGNC 01/04, pp. 20–35 (2004)
2. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: Solving Subset Sum in Linear Time by Using Tissue P Systems with Cell Division. LNCS, vol. 4527, pp. 170–179 (2007)
3. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Campero, F.J.: Computational Efficiency of Dissolution Rules in Membrane Systems. International Journal of Computer Mathematics 83(7), 593–611 (2006)
4. Gutiérrez Naranjo, M.A., Pérez Jiménez, M.J., Romero-Campero, F.J.: A Linear Solution of Subset Sum Problem by Using Membrane Creation. In: Mira, J.M., Álvarez, J.R. (eds.) IWINAC 2005. LNCS, vol. 3561, pp. 258–267. Springer, Heidelberg (2005)
5. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A Fast P System for Finding a Balanced 2-Partition. Soft Computing 9(9), 673–678 (2005)
6. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Campero, F.: On the Power of Dissolution in P Systems with Active Membranes. In: Freund, R., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2005. LNCS, vol. 3850, pp. 373–394. Springer, Heidelberg (2006)
7. Păun, G.: Computing with Membranes: Attacking **NP**–complete Problems. In: Antoniou, I., Calude, C., Dinneen, M.J. (eds.) UMC 2000. Unconventional Models of Computation, pp. 94–115. Springer, Berlin (2000)
8. Păun, G.: Membrane Computing. An Introduction. Springer, Berlin (2002)
9. Pérez-Jiménez, M.J., Riscos-Núñez, A.: Solving the Subset-Sum problem by P Systems with Active Membranes. New Generation Computing 23(4), 367–384 (2005)
10. Pérez-Jiménez, M.J., Riscos-Núñez, A.: A Linear–time Solution to the Knapsack Problem Using P Systems with Active Membranes. In: Martín-Vide, C., Mauri, G., Păun, G., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing. LNCS, vol. 2933, pp. 250–268. Springer, Heidelberg (2004)
11. Pérez-Jiménez, M.J., Romero-Campero, F.J.: Solving the Bin Packing Problem by Recognizer P Systems with Active Membranes. In: Păun, G., Riscos-Núñez, A., Romero-Jiménez, A., Sancho-Caparrini, F. (eds.) Proc. Second Brainstorming Week on Membrane Computing, Report RGNC 01/04, University of Seville, pp. 414–430 (2004)
12. Pérez-Jiménez, M.J., Romero–Campero, F.J.: Attacking the Common Algorithmic Problem by Recognizer P Systems. In: Margenstern, M. (ed.) MCU 2004. LNCS, vol. 3354, pp. 304–315. Springer, Heidelberg (2005)
13. Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: A Polynomial Complexity Class in P Systems Using Membrane Division. In: DCFS 2003. Proc. 5th Workshop on Descriptional Complexity of Formal Systems, pp. 284–294 (2003)