The Factorization Problem: A New Approach Through Membrane Systems

David Orellana-Martín, Luis Valencia-Cabrera, and Mario J. Pérez-Jiménez

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
{dorellana, lvalencia, marper}@us.es

Abstract. The *factorization problem* (given a natural number which is the product of two prime numbers, find its decomposition) is conjectured to be intractable and for that it has been used as the key to have secure current cryptosystems. Due to its relevance, this problem has been studied in various computational paradigms, in particular in *membrane computing*. In this framework, *recognizer* P systems were introduced to deal with *decision problems*, that is, problems whose solution/answer is either "`yes`" or "`no`". The factorization problem is a *search problem* (also called *function problem*), where the question is to identify/find *one* solution to the set of possible solutions associated with each instance. In this work, membrane systems *computing partial functions* are shown to (efficiently) solve the factorization problem, improving the previous solutions given in the framework of membrane computing. Specifically, a family of computing polarizationless P systems with active membranes using minimal cooperation and minimal production in object evolution rules, is provided to give a polynomial-time solution to the factorization problem.

1 Introduction

Membrane computing is a computing paradigm inspired by the structure and functioning of living cells. It was introduced in [10] by Gheorghe Păun, describing the basic behavior of these kinds of systems, called *membrane systems* or *P systems*. As a fields within *Natural Computing*, we take inspiration from nature in order to define the semantics of the computational model. Membrane computing takes the minimal functions required to have a living being, that is, replication of DNA, synthesis of proteins, the use of energy to perform metabolic processes and methods to regulate itself (e.g., *apoptosis*). In this way, we can abstract these chemical reactions by means of mathematical tools, for instance, rewriting rules, in order to perform computations. There are basically three approaches to consider computational devices: cell–like membrane systems [10, 11], using the biological membranes arranged hierarchically, inspired by the structure of the cell; tissue–like membrane systems [7], using the biological membranes placed in the nodes of a graph, inspired by the cell inter-communication in tissues; and

neuron-like P systems [5], inspired by the neurophysiological behavior of neurons sending electrical impulses (spikes) along axons from presynaptic neurons to post–synaptic neurons in a distributed and parallel manner. In these variants, polynomial-time solutions of some computationally hard decision problems has been provided: SAT [13], HAM-CYCLE [12], 3-COL [2], KNAPSACK, SUBSET-SUM and PARTITION [15], among others.

Cryptography is a discipline concerning the security of the information in the presence of possible intruders. For this purpose, several *cryptosystems* have been developed, that is, several protocols of security that make harder to a third party to discover the information that two different parts want to interchange. It is important to take into account that the protocol itself does not have to be secret (for instance, for public-key cryptosystems). In fact, the keys of the most important cryptosystems are well-known. The difficulty that resides in them is intrinsic to the problem behind the systems themselves. That is because there is not known any efficient classical algorithm to solve them. For instance, the key to have secure cryptosystems such as *RSA*, introduced by R. Rivest, A. Shamir and L. Adleman in [16], is the following version of the *factorization problem*: *given a natural number n which is the product of two large primes, find its decomposition.* Such a number n is used as the *modulus* for both public and private keys. In order to attack it, we need to factorize n into its prime factors. Many systems, such as banks, medical databases and critical systems with confidential information keep their data secure thanks to this method. The factorization problem is conjectured to be computationally intractable, as some of the problems used in cryptography, such as KNAPSACK [1] in Merkle-Hellman cryptosystem [8] and DISCRETE LOGARITHM used in Diffie-Hellman key exchange [3], among others.

In the framework of membrane computing, there were attempts to give a solution to the factorization problem in [6, 9, 22] with membrane systems. In [6], a solution is given by using a family of P system with active membranes and electrical charges, using 4-division rules, that is, by applying this kind of rules a membrane produces four new membranes, and object evolution rules whose left-hand side and right-hand sides can have three objects. In [9], a solution is given by means of a family of asynchronous P systems with active membranes and electrical charges which use cooperation in object evolution rules and division rules for non-elementary membranes. In [22], a solution is given by using a family of tissue P systems with cell division which use symport/antiport rules of length at most 4.

In this work, a solution of the factorization problem is given by means of a version of polarizationless P system with active membranes using 2-division rules only for elementary membranes (that is, by applying this kind of rule an elementary membrane produces only two new membranes) without dissolution rules. Specifically, these systems use *minimal cooperation* (the left-hand sides have at most two objects) and *minimal production* (the right-hand sides have only one object) in objects evolution rules. In order to develop simulators running on real computers, this kind of membrane system is interesting, for instance, from

[1] That is, in fact, an **NP**-complete problem.

the GPU computing point of view, just because the algorithm proposed seems easily translatable to this parallel computing paradigm.

The solution provided in this paper improves the previous ones given in [6, 9, 22]. In this regard, it should be recalled that families of polarizationless P system with active membranes using division rules and without dissolution rules only can (efficiently) solve problems in class **P** [4].

The paper is organized as follows. The next section briefly describes some basic aspects in order to make the work self-contained. In Section 3 we define the syntax and semantics of polarizationless P systems with active membranes by using membrane division rules and minimal cooperation in evolution rules. Next, *computing* membrane systems are introduced in Section 4. Section 5 is devoted to define the family of P systems that return the factorization of a given number, followed by an overview of the computation to know what is happening in each step. The paper ends with some open problems and concluding remarks.

2 Preliminaries

In order to have a precise definition of all the terms that are going to be used later, we are going to introduce them here.

2.1 Partial Functions

A function f is a set whose elements are ordered pairs verifying the following: $\forall x \forall y \forall z\, [(x, y) \in f \wedge (x, z) \in f \rightarrow y = z]$. The set $\{x \mid \exists y\, (x, y) \in f\}$ is called the *domain* of f and it is denoted by $dom(f)$. The set $\{x \mid \exists y\, (y, x) \in f\}$ is called the *range* of f and it is denoted by $rang(f)$.

Given two sets A, B, a *partial function* f from A onto B is a set verifying that $f \subseteq A \times B$, $dom(f) \subseteq A$ and $rang(f) \subseteq B$. In the case $dom(f) = A$ the function is called a *total function* from A onto B.

2.2 Alphabets and Multisets

An *alphabet* Γ is a non-empty set. A *multiset* over an alphabet Γ is an ordered pair (Γ, f) where f is a total function from Γ onto the set of natural numbers $\mathbb{N}$. The support of a multiset $\mathcal{M} = (\Gamma, f)$ is defined as $supp(\mathcal{M}) = \{x \in \Gamma \mid f(x) > 0\}$. A multiset is finite (respectively, empty) if its support is a finite (resp., empty) set. If Γ is a finite set then each multiset $\mathcal{M} = (\Gamma, f)$ is finite and it will be represented by $\{a^{f(a)} \mid a \in supp(\mathcal{M})\}$.

2.3 Graphs and Trees

A *free tree* (*tree*, for short) is a connected, acyclic, undirected graph. A *rooted tree* is a tree in which one of the vertices (called the *root* of the tree) is distinguished from the others. In a rooted tree the concepts of ascendants and descendants are defined in a usual way. Given a node x (different from the root), if the last edge

on the (unique) path from the root of the tree to the node x is $\{x, y\}$ (in this case, $x \neq y$), then y is **the** *parent* of node x and x is **a** *child* of node y. The root is the only node in the tree with no parent. A node with no children is called a *leaf* (see [1] for details).

2.4 Binary Representation of Natural Numbers

For each natural number $n \in \mathbb{N}$ we denote $[0, 2^{n+1}) = \{x \in \mathbb{N} \mid 0 \leq x < 2^{n+1}\}$. Next, we define the binary representation of natural numbers. For each natural number $x \in \mathbb{N}$, $x \geq 1$, there exist a unique tuple $(x_0, \ldots, x_n) \in \{0,1\}^{n+1}$, with $n \in \mathbb{N}$ and $x_n = 1$, such that $x = x_0 \cdot 2^0 + x_1 \cdot 2^1 + \ldots + x_n \cdot 2^n$, that is, $x \in [0, 2^{n+1})$. We say that the finite sequence $x_n \cdots x_1 x_0$ or the tuple $(x_0, \ldots, x_n)$, is the *binary representation* of natural number x. We denote $k_x = n$, that is, $1 + k_x$ is the number of digits of natural number $x \geq 1$ in its binary representation.

Binary representation and, in general, representation of numbers different to unary one, is useful because the *size* of a natural number x (the number of bits used) in unary representation is x but in binary representation its size is $1 + \lfloor \log_2(x) \rfloor$. Consequently, the size of a natural number expressed in unary form is exponential in the size of that number expressed in binary form. It is worth pointing out that within the framework of Membrane Computing work is being carried out on multisets, and it is usual represents instances of abstract problems in unary representation (natural numbers are encoded by the multiplicities of some objects in a multiset).

Given a partial function f from $\mathbb{N}^q$ into $\mathbb{N}^r$, with $q \geq 1, r \geq 1$, for each $\boldsymbol{x} = (x_1, \ldots, x_q) \in \mathbb{N}^q$ and $\boldsymbol{y} = (y_1, \ldots, y_r) \in \mathbb{N}^r$ such that $f(\boldsymbol{x}) = \boldsymbol{y}$ there exists a unique natural number $k_{(\boldsymbol{x},\boldsymbol{y})}$ defined as follows:

$$k_{(\boldsymbol{x},\boldsymbol{y})} = \min\{k \in \mathbb{N} \mid [k \geq 1] \wedge [x_1, \ldots, x_q, y_1, \ldots, y_r \in [0, 2^k)]\}$$

That is, $k_{(\boldsymbol{x},\boldsymbol{y})}$ is the smallest natural number where natural numbers $x_1, \ldots, x_q$, $y_1, \ldots, y_r$ can be represented in binary form with, at most, $k_{(\boldsymbol{x},\boldsymbol{y})}$ digits.

3 Polarizationless P Systems with Active Membranes

In [11], P systems with active membranes are introduced as a universal computing model, that is, it has the same computational power than a Turing machine. There, a linear time solution to `SAT` is given, using P systems with active membranes with polarizations and membrane division. As polarizations seem to be a very powerful tool from the computational complexity point of view, a new framework not using them is created, the so-called *polarizationless* P systems with active membranes. In [4], a frontier of efficiency is given by means of dissolution rules. Passing from forbidding them to allowing them is the same as passing from non-efficiency to (strong) efficiency. In fact, not only **NP**-complete problems can be solved in an efficient way, but a solution to the problem `QSAT`, a

well-known **PSPACE**-complete problem [14], by means of recognizer polarizationless P systems with membrane division for elementary and non-elementary membranes and dissolution rules in linear time is given.

In P systems with active membranes, the rules are non-cooperative, that is, the left-hand side of the rules have only one object. In [18] and [19], a cooperative version of object evolution rules was introduced. In following investigations, more restrictions were added to these rules, by considering *minimal cooperation* (the left-hand side of the rules have exactly two objects) and *minimal production* (the right-hand side of the rules have only one object) in objects evolution rules. Even restricting these rules to this, a linear-time solution to `SAT` was provided in [20] when division rules only for elementary membranes are considered and dissolution rules are forbidden.

3.1 Syntax

Definition 1. *A polarizationless P system with active membranes of degree $p \geq 1$ that makes use of minimal cooperation and minimal production in object evolution rules is a tuple $\Pi = (\Gamma, H, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_p, \mathcal{R}, i_{out})$, where:*

- *Γ is a finite alphabet;*
- *H is a finite alphabet such that $H \cap \Gamma = \emptyset$;*
- *μ is a labelled rooted tree with p nodes;*
- *$\mathcal{M}_1, \ldots, \mathcal{M}_p$ are multisets over Γ;*
- *$\mathcal{R}$ is a finite set of rules, of the following forms:*
  - *(a) $[\, a \to c\,]_h$ or $[\, a\, b \to c\,]_h$, for $h \in H$, $a, b, c \in \Gamma$ (object evolution rules).*
  - *(b) $a\,[\quad]_h \to [\, b\,]_h$, for $h \in H$, $a, b \in \Gamma$ (send-in communication rules).*
  - *(c) $[\, a\,]_h \to b\,[\quad]_h$, for $h \in H$, $a, b \in \Gamma$ (send-out communication rules).*
  - *(d) $[\, a\,]_h \to b$, for $h \in H$, $a, b \in \Gamma$ (dissolution rules).*
  - *(e) $[\, a\,]_h \to [\, b\,]_h [\, c\,]_h$, for $h \in H$, $a, b, c \in \Gamma$ (division rules for elementary membranes).*
  - *(f) $[\,[\quad]_{h_0}\,[\quad]_{h_1}\,]_h \to [\,[\quad]_{h_0}\,]_h\,[\,[\quad]_{h_1}\,]_h$, for $h, h_0, h_1 \in H$ (division rules for non-elementary membranes).*

A polarizationless P system with active membranes of degree p can be viewed as a set of p membranes, labelled by elements of H, arranged in a hierarchical structure μ given by a rooted tree (called membrane structure) whose root is called the *skin membrane*, such that: (a) $\mathcal{M}_1, \ldots, \mathcal{M}_p$ represent the finite multisets of *objects* initially placed in the p membranes of the system; (b) $\mathcal{R}$ is a finite set of rules over Γ associated with the labels; and (c) $i_{out} \in H \cup \{env\}$ indicates the output zone. We use the term *zone i* to refer to membrane i in the case $i \in H$ and to refer to the "environment" of the system in the case $i = env$. The leaves of μ are called *elementary membranes*. In these kind of P systems where are mechanisms, implemented by division rules, able to generate an exponential workspace (in terms of number of membranes and objects) in polynomial time. This allows us to describe brute force algorithms in these systems in an "efficient" way.

3.2 Semantics

A *configuration* $\mathcal{C}_t$ at an instant t of a polarizationless P system with active membranes is described by the following elements: (a) the membrane structure at instant t, and (b) all multisets of objects over Γ associated with all the membranes present in the system at that moment.

An object evolution rule $[\, a \to c\,]_h$ (resp., $[\, a\, b \to c\,]_h$) is *applicable* to a configuration $\mathcal{C}_t$ at an instant t, if there exists a membrane labelled by h in $\mathcal{C}_t$ which contains object a (resp., objects a and b). When applying such a rule, object a (resp., objects a and b) is consumed and object c is produced in that membrane.

A send-in communication rule $a\,[\quad]_h \to [\, b\,]_h$ is *applicable* to a configuration $\mathcal{C}_t$ at an instant t, if there exists a membrane labelled by h in $\mathcal{C}_t$ such that h is not the label of the root of μ and its parent membrane contains object a. When applying such a rule, object a is consumed from the parent membrane and object b is produced in the corresponding membrane labelled by h.

A send-out communication rule $[\, a\,]_h \to b\,[\quad]_h$ is *applicable* to a configuration $\mathcal{C}_t$ at an instant t, if there exists a membrane labelled by h in $\mathcal{C}_t$ such that it contains object a. When applying such a rule, object a is consumed from such membrane h and object b is produced in the parent of such membrane (in the case that such membrane is the skin then object b is produced in the environment).

A dissolution rule $[\, a\,]_h \to b$ is *applicable* to a configuration $\mathcal{C}_t$ at an instant t, if there exists a membrane labelled by h in $\mathcal{C}_t$, different from the skin membrane and the output zone, such that it contains object a. When applying such a rule, object a is consumed, membrane h is dissolved and its objects beside an object b are sent to the parent (or the first ancestor that has not been dissolved).

A division rule $[\, a\,]_h \to [\, b\,]_h [\, c\,]_h$ for elementary membrane is *applicable* to a configuration $\mathcal{C}_t$ at an instant t, if there exists an elementary membrane labelled by h in $\mathcal{C}_t$, different from the skin membrane and the output zone, such that it contains object a. When applying such a rule, the membrane with label h is divided into two membranes with the same label; in the first copy, object a is replaced by object b, in the second one, object a is replaced by object c; all the other objects are replicated and copies of them are placed in the two new membranes.

A division rule $[\,[\quad]_{h_0}\,[\quad]_{h_1}\,]_h \to [\,[\quad]_{h_0}\,]_h\,[\,[\quad]_{h_1}\,]_h$ for non-elementary membrane is *applicable* to a configuration $\mathcal{C}_t$ at an instant t, if there exists a membrane labelled by h in $\mathcal{C}_t$, different from the skin membrane and the output zone, which contains a membrane labelled by h_0 and another membrane labelled by h_1. When applying such a rule, the membrane with label h is divided into two membranes with the same label; the first copy inherits membrane h_0 with its contents, and the second copy inherits membrane h_1 with its contents. Besides, if the membrane labelled by h contains more membranes other that those with the labels h_0, h_1, then such membranes are duplicated so that they become part of the contents of both new copies of the membrane h.

In polarizationless P systems with active membranes, the rules are applied according to the following principles:

- At one transition step, one object and one membrane can be used by only one rule, selected in a non-deterministic way.
- At one transition step, a *membrane* can be the subject of *only one* rule of types $(b) - (f)$, and then it is applied at most once.
- Object evolution rules can be simultaneously applied to a membrane with one rule of types $(b) - (f)$. In any case, object evolution rules are applied in a maximally parallel manner.
- If at the same time a membrane labelled with h is divided by a rule of type (e) or (f) 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, changing the objects, and then the division is produced. Of course, this process takes only one transition step.
- The skin membrane and the output membrane, if any, can never get divided nor dissolved.

Let us notice that in these kind of P systems the environment plays a passive role in the following sense: along any computation, the environment only can receive objects from the system but it cannot send objects into the system.

4 Computing Membrane Systems

Let us recall that *counting membrane systems*, was introduced as a framework where counting problems (a special case of *search problems*) can be solved in a natural way [21]. These systems are inspired from *counting Turing machines* introduced by L. Valiant [23] and from recognizer membrane systems where the Boolean answer of these systems is replaced by an answer encoded by a natural number expressed in a binary representation (placed in the environment associated with the halting configuration). On the other hand, the concept of *computing P system* was introduced in [17] providing devices in Membrane Computing to compute *partial functions* from $\mathbb{N}^q$ to $\mathbb{N}^r$, with $q \geq 1, r \geq 1$.

Inspired by the previous concepts, (*binary*) *computing membrane systems* is defined in order to compute partial function from $\mathbb{N}^q$ to $\mathbb{N}^r$ $(q \geq 1, r \geq 1)$ when the natural numbers are considered by means of the corresponding binary representation.

Definition 2. *A (binary) computing membrane system Π of degree (p, q, r), $p \geq 1, q \geq 1, r \geq 1$, and order $n \geq 1$ is a tuple $\Pi = (\Pi', \Sigma, \Phi, i_{in})$, where*

- *$\Pi' = (\Gamma', \mu', \mathcal{M}'_1, \ldots, \mathcal{M}'_p, \mathcal{R}')$ is a membrane system with external output of degree p.*
- *$\Sigma = \{a_{1,0}, \ldots, a_{1,n-1}, \ldots, a_{q,0}, \ldots, a_{q,n-1}\}$ is an ordered set (the input alphabet) strictly contained in Γ'.*
- *$\Phi = \{b_{1,0}, \ldots, b_{1,n-1}, \ldots, b_{r,0}, \ldots, b_{r,n-1}\}$ is an ordered set (the final alphabet) strictly contained in Γ' and $\Phi \cap \Sigma = \emptyset$.*
- *i_{in} is the label of a distinguished membrane of Π' (the input membrane).*

Given a (binary) computing membrane system $\Pi = (\Pi', \Sigma, \Phi, i_{in})$ of degree (p, q, r) and order n, for each tuple $\boldsymbol{x} = (x_1, \ldots, x_q) \in \mathbb{N}^q$ such that $x_i \in [0, 2^n)$, for $1 \leq i \leq q$, there are uniques $x_{i,j} \in \{0, 1\}$, for $1 \leq i \leq q, 0 \leq j \leq n-1$, verifying $x_i = \sum_{j=0}^{n-1} x_{i,j} \cdot 2^j$, we use the following notations:

- $cod(\boldsymbol{x})$ is the set $\{a_{1,0}^{x_{1,0}}, \ldots, a_{1,n-1}^{x_{1,n-1}}, \ldots, a_{q,0}^{x_{q,0}}, \ldots, a_{q,n-1}^{x_{q,n-1}}\}$.
- $\Pi + cod(\boldsymbol{x})$ is the membrane system Π whose the initial configuration is the tuple $(\mu', \mathcal{M}'_1, \ldots, \mathcal{M}'_{i_{in}} + cod(\boldsymbol{x}), \ldots \mathcal{M}'_p)$.

In a (binary) computing membrane system Π of degree (p, q, r) and order n, the following semantics conditions are required: for each natural number $\boldsymbol{x} = (x_1, \ldots, x_q) \in \mathbb{N}^q$ such that $x_i \in [0, 2^n)$, for $1 \leq i \leq q$,

- Either any computation of $\Pi + cod(\boldsymbol{x})$ is a non-halting computation, or all computations of $\Pi + cod(\boldsymbol{x})$ halt.
- If all computations of $\Pi + cod(\boldsymbol{x})$ halt, then there exists a tuple

$$(y_{1,0}, \ldots, y_{1,n-1}, \ldots, y_{r,0}, \ldots, y_{r,n-1}) \in \{0, 1\}^{r \cdot n}$$

  such that for any computation of $\Pi + cod(\boldsymbol{x})$, the subset of the final alphabet Φ contained in the environment associated with the corresponding halting configuration is $\{b_{1,0}^{y_{1,0}}, \ldots, b_{1,n-1}^{y_{1,n-1}}, \ldots, b_{r,0}^{y_{r,0}}, \ldots, b_{r,n-1}^{y_{r,n-1}}\}$.

According with this, the output of the membrane system $\Pi + cod(\boldsymbol{x})$, in the case that all their computations halt, denoted by $Output(\Pi + cod(\boldsymbol{x}))$, is the tuple

$$(y_{1,0}, \ldots, y_{1,n-1}, \ldots, y_{r,0}, \ldots, y_{r,n-1}) \in \{0, 1\}^{r \cdot n}$$

That is, the output of the membrane system $\Pi + cod(\boldsymbol{x})$ encodes a tuple $(y_1, \ldots, y_r) \in \mathbb{N}^r$ such that $y_l \in [0, 2^n)$, for $1 \leq l \leq r$, and $(y_{l,0}, \ldots, y_{l,n-1})$ is the binary representation of y_l.

Definition 3. *We say that a (binary) computing membrane system Π of degree (p, q, r) and order n, computes a partial function f from $[0, 2^n) \times \overset{(q \cdot n)}{\cdots} \times [0, 2^n)$ into $[0, 2^n) \times \overset{(r \cdot n)}{\cdots} \times [0, 2^n)$, if for each $\boldsymbol{x} = (x_1 \ldots x_n) \in [0, 2^n) \times \overset{(q \cdot n)}{\cdots} \times [0, 2^n)$, the following holds:*

- *$f(\boldsymbol{x})$ is defined, that is, $\boldsymbol{x} \in dom(f)$, if and only if all computations of system $\Pi + cod(\boldsymbol{x})$ halt.*
- *$f(\boldsymbol{x}) = \boldsymbol{y}$, with $\boldsymbol{y} = \sum_{j=0}^{n-1} y_{i,j} \cdot 2^j$, for $1 \leq i \leq r$, if and only if*

$$Output(\Pi + cod(\boldsymbol{x})) = (y_{1,0}, \ldots, y_{1,n-1}, \ldots, y_{r,0}, \ldots, y_{r,n-1})$$

Definition 4. *We say that a family $\{\Pi(k) \mid k \in \mathbb{N}\}$ of (binary) computing membrane systems computes a partial function f from $\mathbb{N}^q$ into $\mathbb{N}^r$ if the following holds:*

- *For each $k \in \mathbb{N}$, $\Pi(k)$ is a (binary) computing membrane system of order $k + 1$.*
- *For each $\boldsymbol{x} \in \mathbb{N}^q$, $f(\boldsymbol{x})$ is defined and $f(\boldsymbol{x}) = \boldsymbol{y}$, with $\boldsymbol{y} = \sum_{j=0}^{k_{(\boldsymbol{x}, \boldsymbol{y})}} y_{i,j} \cdot 2^j$, for $1 \leq i \leq r$, if and only if the output of the system $\Pi(k_{(\boldsymbol{x}, \boldsymbol{y})}) + cod(\boldsymbol{x})$ is the tuple $(y_{1,0}, \ldots, y_{1,k_{(\boldsymbol{x}, \boldsymbol{y})}}, \ldots, y_{r,0}, \ldots, y_{r,k_{(\boldsymbol{x}, \boldsymbol{y})}})$.*

5 Solving the FACTORIZATION problem by Computing Membrane Systems

Let us recall that the FACTORIZATION problem is the following: *given a natural number which is the product of two prime numbers, find its decomposition.* This problem can be characterized by a *partial function* FACT from $\mathbb{N}$ to $\mathbb{N}^2$ defined as follows: for each natural number x which is the product of two prime numbers y, z, with $y \geq z$, we have $\texttt{FACT}(x) = (y, z)$. In other words, to solve the FACTORIZATION problem is equivalent to compute the partial function FACT.

In this paper, a solution to the FACTORIZATION problem is presented by providing a family $\{\Pi(n) \mid n \in \mathbb{N}\}$ of (binary) computing polarizationless P systems with active membranes which make use of minimal cooperation and minimal production (without dissolution rules and without division rules for non-elementary membranes), that computes the partial function FACT from $\mathbb{N}$ to $\mathbb{N}^2$, previously defined. Specifically, an instance x of the FACTORIZATION problem will be processed by the membrane system $\Pi(k_x)$ with input multiset $cod(x)$, where $cod(x)$ encodes the binary representation of instance x through the input alphabet of $\Pi(k_x)$. Bearing in mind that $2 \leq y, z < x$ we have $k_x = k_{(x,y,z)}$, and so $x, y, z \in [0, 2)^{1+k_x}$. Besides, $1+k_x$ is the maximum number of digits of natural numbers $x, y, z \geq 1$ in its binary representation and the system $\Pi(k_x)$ has order $k_x + 1$. For each natural number $n \in \mathbb{N}$, we consider the computing polarizationless P system with active membranes which makes use of minimal cooperation and minimal production but without dissolution rules and without division for non-elementary membranes, $\Pi(n) = (\Gamma, \Sigma, \Phi, H, \mu, \mathcal{M}_1, \mathcal{M}_2, \mathcal{R}, i_{in}, i_{out})$ of degree $(2, 1, 2)$ and order $n + 1$, defined as follows:

(1) Working alphabet:

$$
\begin{aligned}
\Gamma = \Sigma \cup \Phi \cup \{\#\} \cup \{\omega_{j,k} \mid 0 \leq j \leq n, 0 \leq k \leq 17n + 26\} \cup \\
\{\alpha_{1,j,s} \mid 0 \leq j \leq n, 0 \leq s < j\} \cup \\
\{\alpha_{2,j,s} \mid 0 \leq j \leq n, 0 \leq s < n + 1 + j\} \cup \\
\{T_{h,j}, \overline{T}_{h,j} \mid 1 \leq h \leq 2, 0 \leq j \leq n\} \cup \\
\{p_{j,k} \mid 0 \leq j \leq n, 0 \leq k \leq 5n + 10\} \cup \\
\{\beta_{h,j,s} \mid 1 \leq h \leq 2, 0 \leq j \leq n, 0 \leq s \leq 3n + 6\} \cup \\
\{\overline{P}_{j,k} \mid 0 \leq j \leq n, 0 \leq k \leq 5n + 8\} \cup \{X_j, \overline{X}_j \mid 0 \leq j \leq n\} \cup \\
\{t_{1,j,s}, \bar{t}_{1,j,s} \mid 0 \leq j \leq n, j \leq s \leq 3n + 5\} \cup \\
\{t_{2,j,s}, \bar{t}_{2,j,s} \mid 0 \leq j \leq n, n + 1 + j \leq s \leq 3n + 5\} \cup \\
\{T^*_{h,j}, \overline{T}^*_{h,j} \mid 1 \leq h \leq 2, 0 \leq j \leq n\} \cup \\
\{P_j, \overline{P}_j, P^*_j, \overline{P}^*_j \mid 0 \leq j \leq n\} \cup \{e_j, e^*_j \mid 1 \leq j \leq n\} \cup \\
\{d_j \mid 1 \leq j \leq 4n + 3\} \cup \{d^*_j \mid 2n + 2 \leq j \leq 4n + 2\} \cup \\
\{G_k \mid 1 \leq k \leq 8n + 6\} \cup \{T'_{1,j}, \overline{T}'_{1,j} \mid 0 \leq j \leq n\} \cup \\
\{C_{h,j,i} \mid 0 \leq h \leq 2, 0 \leq j \leq n, 0 \leq i \leq n - j\} \cup \\
\{C_{h,j} \mid 0 \leq h \leq 2, 0 \leq j \leq n\} \cup \\
\{T_{1,j,k}, \overline{T}_{1,j,k} \mid 0 \leq j \leq n, 0 \leq k \leq j\} \cup \\
\{T_{2,j,k}, \overline{T}_{2,j,k} \mid 0 \leq j \leq n, 0 \leq k \leq n + 1 + j\} \cup \\
\{y_j, \overline{y}_j, z_j, \overline{z}_j, y^*_j, z^*_j \mid 0 \leq j \leq n\}
\end{aligned}
$$

where the input alphabet is $\Sigma = \{a_i \mid 0 \leq i \leq n\}$ and the final alphabet is $\Phi = \{b_{1,j}, b_{2,j} \mid 0 \leq j \leq n\}$;

(2) $H = \{1, 2\}$

(3) Membrane structure: $\mu = [\,[\quad]_2\,]_1$, that is, $\mu = (V, E)$ where $V = \{1, 2\}$ and $E = \{(1, 2)\}$

(4) Initial multisets: $\mathcal{M}_1 = \{\omega^2_{j,0} \mid 0 \leq j \leq n\}$, $\mathcal{M}_2 = \{\overline{X}_j, \alpha_{1,j,0}, \alpha_{2,j,0}, T^{n+4}_{1,j}, \overline{T}^{n+4}_{1,j}, T^{n+4}_{2,j}, \overline{T}^{n+4}_{2,j}, p_{j,0}, \omega^2_{j,0}, \tau_{j,0}, \overline{z}_j, \beta^{n+1}_{1,j,0}, \beta^{n+1}_{2,j,0} \mid 0 \leq j \leq n\} \cup \{\overline{P}_{j,0} \mid 0 \leq i \leq 2n+1\}$

(5) The set of rules $\mathcal{R}$ consists of the following rules:

5.1 Counters

$$[\,a_j\,\overline{X}_j \to X_j\,]\ \text{, for } 0 \leq j \leq n$$
$$[\,\alpha_{1,j,s} \to \alpha_{1,j,s+1}\,]_2\ \text{, for } 0 \leq j \leq n \text{ and } 0 \leq s < j$$
$$[\,\alpha_{2,j,s} \to \alpha_{2,j,s+1}\,]_2\ \text{, for } 0 \leq j \leq n \text{ and } 0 \leq s < n+1+j$$
$$\left.\begin{array}{l}[\,\beta_{1,j,k} \to \beta_{1,j,k+1}\,]_2\\ [\,\beta_{2,j,k} \to \beta_{2,j,k+1}\,]_2\end{array}\right\} \text{for } 0 \leq j \leq n, 0 \leq k \leq 3n+5$$
$$[\,\overline{P}_{j,k} \to \overline{P}_{j,k+1}\,]_2\ \text{, for } 0 \leq j \leq 2n+1, 0 \leq k \leq 5n+7$$
$$[\,p_{i,j} \to p_{i,j+1}\,]_2\ \text{, for } 0 \leq i \leq n, 0 \leq j \leq 5n+9$$
$$[\,\tau_{j,k} \to \tau_{j,k+1}\,]_2\ \text{, for } 0 \leq j \leq n, 0 \leq k \leq 14n+11$$
$$[\,\omega_{j,k} \to \omega_{j,k+1}\,]_2\ \text{, for } 0 \leq j \leq n, 0 \leq k \leq 15n+21$$
$$[\,\omega_{i,j} \to \omega_{i,j+1}\,]_1\ \text{, for } 0 \leq i \leq n, 0 \leq j \leq 17n+25$$

5.2 Generation Stage

$$\left.\begin{array}{l}[\,\alpha_{1,j,j}\,]_2 \to [\,t_{1,j,j}\,]_2\,[\,\bar{t}_{1,j,j}\,]_2\\ [\,\alpha_{2,j,n+1+j}\,]_2 \to [\,t_{2,j,n+1+j}\,]_2\,[\,\bar{t}_{2,j,n+1+j}\,]_2\end{array}\right\} \text{for } 0 \leq j \leq n$$
$$\left.\begin{array}{l}[\,t_{1,j,v} \to t_{1,j,v+1}\,]_2\\ [\,\bar{t}_{1,j,v} \to \bar{t}_{1,j,v+1}\,]_2\end{array}\right\} \text{for } 0 \leq j \leq n \text{ and } j \leq v \leq 2n$$
$$\left.\begin{array}{l}[\,t_{2,j,v} \to t_{2,j,v+1}\,]_2\\ [\,\bar{t}_{2,j,v} \to \bar{t}_{2,j,v+1}\,]_2\end{array}\right\} \text{for } 0 \leq j \leq n-1 \text{ and } n+1+j \leq v \leq 2n$$
$$\left.\begin{array}{l}[\,t_{h,j,2n+s}\,\overline{T}_{h,j} \to t_{h,j,2n+s+1}\,]_2\\ [\,\bar{t}_{h,j,2n+s}\,T_{h,j} \to \bar{t}_{h,j,2n+s+1}\,]_2\end{array}\right\} \text{for } \begin{array}{c}1 \leq h \leq 2\\ 0 \leq j \leq n\\ 1 \leq s \leq 3n+4\end{array}$$
$$\left.\begin{array}{l}[\,t_{h,j,3n+5}\,\overline{T}_{h,j} \to \#\,]_2\\ [\,\bar{t}_{h,j,3n+5}\,T_{h,j} \to \#\,]_2\\ [\,\beta_{1,j,3n+6}\,T_{1,j} \to T^*_{1,j}\,]_2\\ [\,\beta_{1,j,3n+6}\,\overline{T}_{1,j} \to \overline{T}^*_{1,j}\,]_2\\ [\,\beta_{2,j,3n+6}\,T_{2,j} \to T^*_{2,j}\,]_2\\ [\,\beta_{2,j,3n+6}\,\overline{T}_{2,j} \to \overline{T}^*_{2,j}\,]_2\end{array}\right\} \text{for } 0 \leq j \leq n, 0 \leq k \leq 3n+5$$

5.3 Multiplication Stage

$$\left.\begin{array}{l}[\,T^*_{1,j}\,T^*_{2,j'} \to P_{j+j'}\,]_2\\ [\,T^*_{1,j}\,\overline{T}^*_{2,j'} \to P_j\,]_2\\ [\,\overline{T}^*_{1,j}\,T^*_{2,j'} \to P_{j'}\,]_2\\ [\,\overline{T}^*_{1,j}\,\overline{T}^*_{2,j'} \to \#\,]_2\end{array}\right\} \text{for } 0 \leq j, j' \leq n$$

$$\left.\begin{array}{l}[\,\overline{P}_j\,P_j \to P_{j+1}\,]_2\\ [\,\overline{P}_{j,5n+8} \to \overline{P}_j\,]_2\\ [\,P_j\overline{P}_j \to P_j\,]_2\\ [\,p_{j,5n+10}\,P_j \to P_j^*\,]_2\\ [\,p_{j,5n+10}\,\overline{P}_j \to \overline{P}_j^*\,]_2\end{array}\right\} \text{for } 0 \le j \le 2n+1$$

5.4 Equality Checking Stage

$$\left.\begin{array}{l}[\,P_j^*\,X_j \to e_j\,]_2\\ [\,\overline{P}_j^*\,\overline{X}_j \to e_j\,]_2\\ [\,\overline{P}_j^*\,X_j \to e_j^*\,]_2\\ [\,P_j^*\,\overline{X}_j \to e_j^*\,]_2\end{array}\right\} \text{for } 0 \le j \le n$$

$$[\,e_0\,e_1 \to d_1]_2$$

$$[\,d_j\,e_{j+1} \to d_{j+1}\,]_2 \text{ , for } 0 \le j \le n-1$$

$$[\,e_0^*\,e_1 \to G_1\,]_2$$

$$[\,e_0\,e_1^* \to G_1\,]_2$$

$$[\,e_0^*\,e_1^* \to G_1\,]_2$$

$$\left.\begin{array}{l}[\,d_j\,e_{j+1}^* \to G_{j+1}\,]_2\\ [\,G_j\,e_{j+1} \to G_{j+1}\,]_2\\ [\,G_j\,e_{j+1}^* \to G_{j+1}\,]_2\end{array}\right\} \text{for } 0 \le j \le n$$

$$\left.\begin{array}{l}[\,d_j\,\overline{P}_{j+1} \to d_{j+1}]_2\\ [\,d_j\,P_{j+1} \to G_{j+1}\,]_2\\ [\,G_j\,P_{j+1} \to G_{j+1}\,]_2\\ [\,G_j\,\overline{P}_{j+1} \to G_{j+1}\,]_2\end{array}\right\} \text{for } n \le j \le 2n$$

$$\left.\begin{array}{l}[\,G_{2n+1+j}\,T_{1,j} \to G_{2n+2+j}\,]_2\\ [\,G_{2n+1+j}\,\overline{T}_{1,j} \to G_{2n+2+j}\,]_2\\ [\,G_{3n+2+j}\,T_{2,j} \to G_{3n+3+j}\,]_2\\ [\,G_{3n+2+j}\,\overline{T}_{2,j} \to G_{3n+3+j}\,]_2\end{array}\right\} \text{for } 0 \le j \le n$$

5.5 Trivial Solution Check Stage

$$[\,d_{2n+1}\,T_{1,0} \to d_{2n+2}\,]_2$$

$$[\,d_{2n+1}\,\overline{T}_{1,0} \to d_{2n+2}^*\,]_2$$

$$\left.\begin{array}{l}[\,d_{2n+2+j}\,\overline{T}_{1,j+1} \to d_{2n+3+j}\,]_2\\ [\,d_{2n+2+j}\,T_{1,j+1} \to d_{2n+3+j}^*\,]_2\\ [\,d_{2n+2+j}^*\,T_{1,j+1} \to d_{2n+3+j}^*\,]_2\\ [\,d_{2n+2+j}^*\,\overline{T}_{1,j+1} \to d_{2n+3+j}^*\,]_2\end{array}\right\} \text{for } 0 \le j \le n-2$$

$$[\,d_{3n+1}\,\overline{T}_{1,n} \to T_{3n-1}\,]_2$$

$$[\,d_{3n+1}\,T_{1,n} \to d_{3n-1}\,]_2$$

$$[\,d_{3n+1}^*\,T_{1,n} \to d_{3n+2}\,]_2$$

$$[\,d_{3n+1}^*\,\overline{T}_{1,n} \to d_{3n+2}\,]_2$$

$$[\,d_{3n+2}\,T_{2,0} \to d_{3n}\,]_2$$

$$[\,d_{3n+2}\,\overline{T}_{2,0} \to d_{3n}^*\,]_2$$

$$\left.\begin{array}{l}[\,d_{3n+3+j}\,\overline{T}_{2,j+1} \to d_{3n+4+j}\,]_2\\ [\,d_{3n+3+j}\,T_{2,j+1} \to d_{3n+4+j}^*\,]_2\\ [\,d_{3n+3+j}^*\,T_{2,j+1} \to d_{3n+4+j}^*\,]_2\\ [\,d_{3n+3+j}^*\,\overline{T}_{2,j+1} \to d_{3n+4+j}^*\,]_2\end{array}\right\} \text{for } 0 \le j \le n-2$$

$$[\, d_{4n+2}\, \overline{T}_{2,n} \to T_{4n+3}\,]_2$$

$$[\, d_{4n+2}\, T_{2,n} \to d_{4n+3}\,]_2$$

$$[\, d^*_{4n+2}\, T_{2,n} \to d_{4n+3}\,]_2$$
$$[\, d^*_{4n+2}\, \overline{T}_{2,n} \to d_{4n+3}\,]_2$$

5.6 First Delete Stage

$$\left.\begin{array}{l} [\, G_{4n+3+j}\, T_{1,j} \to G_{4n+4+j}\,]_2 \\ [\, G_{4n+3+j}\, \overline{T}_{1,j} \to G_{4n+4+j}\,]_2 \\ [\, G_{5n+4+j}\, T_{2,j} \to G_{5n+5+j}\,]_2 \\ [\, G_{5n+4+j}\, \overline{T}_{2,j} \to G_{5n+5+j}\,]_2 \\ [\, G_{6n+5+j}\, T_{1,j} \to G_{6n+6+j}\,]_2 \\ [\, G_{6n+5+j}\, \overline{T}_{1,j} \to G_{6n+6+j}\,]_2 \end{array}\right\} \text{for } 0 \leq j \leq n$$

$$\left.\begin{array}{l} [\, G_{7n+6+j}\, T_{2,j} \to G_{7n+7+j}\,]_2 \\ [\, G_{7n+6+j}\, \overline{T}_{2,j} \to G_{7n+7+j}\,]_2 \end{array}\right\} \text{for } 0 \leq j \leq n-1$$

$$[\, G_{8n+6}\, T_{2,n} \to \#\,]_2$$
$$[\, G_{8n+6}\, \overline{T}_{2,n} \to \#\,]_2$$

5.7 Second Delete Stage

$$\left.\begin{array}{l} [\, \tau_{j,14n+12}\, T_{1,j} \to T'_{1,j}\,]_2 \\ [\, \tau_{j,14n+12}\, \overline{T}_{1,j} \to \overline{T}'_{1,j}\,]_2 \end{array}\right\} \text{for } 0 \leq j \leq n$$

$$\left.\begin{array}{l} [\, \overline{T}'_{1,j}\, T_{2,j} \to C_{2,j,n-j}\,]_2 \\ [\, \overline{T}'_{1,i}\, \overline{T}_{2,j} \to C_{1,j,n-j}\,]_2 \\ [\, T'_{1,j}\, T_{2,j} \to C_{1,j,n-j}\,]_2 \\ [\, T'_{1,j}\, \overline{T}_{2,j} \to C_{0,j,n-j}\,]_2 \end{array}\right\} \text{for } 0 \leq j \leq n$$

$$\left.\begin{array}{l} [\, C_{1,j,0}\, C_{2,j-1,1} \to C_{2,j-1,0}\,]_2 \\ [\, C_{0,j,0}\, C_{2,j-1,1} \to C_{0,j-1,0}\,]_2 \\ [\, C_{1,j,0}\, C_{1,j-1,1} \to C_{1,j-1,0}\,]_2 \end{array}\right\} \text{for } 2 \leq j \leq n$$

$$\left.\begin{array}{l} [\, C_{2,j,0}\, C_{i,j-1,1} \to C_{2,j-1,0}\,]_2 \\ [\, C_{0,j,0}\, C_{i,j-1,1} \to C_{0,j-1,0}\,]_2 \end{array}\right\} \text{for } 0 \leq i \leq 2, 2 \leq j \leq n$$

$$[\, C_{1,1,0}\, C_{2,0,1} \to C_{2,0}\,]_2$$

$$[\, C_{1,1,0}\, C_{0,0,1} \to C_{0,0}\,]_2$$

$$[\, C_{1,1,0}\, C_{1,0,1} \to C_{1,0}\,]_2$$

$$\left.\begin{array}{l} [\, C_{2,1,0}\, C_{j,0,1} \to C_{2,0}\,]_2 \\ [\, C_{0,1,0}\, C_{j,0,1} \to C_{0,0}\,]_2 \end{array}\right\} \text{for } 0 \leq j \leq 2$$

$$[\, C_{i,j,k} \to C_{i,j,k-1}\,]_2 \text{ , for } 0 \leq i \leq 2, 2 \leq j \leq n, 0 \leq k \leq n$$

5.8 Output 1 Phase

$$\left.\begin{array}{l}
[\, C_{2,j}\, T_{1,j} \to C_{2,j+1}\,]_2 \\
[\, C_{2,j}\, \overline{T}_{1,j} \to C_{2,j+1}\,]_2 \\
[\, C_{2,n+1+j}\, T_{2,j} \to C_{2,n+2+j}\,]_2 \\
[\, C_{2,n+1+j}\, \overline{T}_{2,j} \to C_{2,n+2+j}\,]_2 \\
[\, \omega_{j,15n+22}\, T_{1,j} \to T_{1,j,j}\,]_2 \\
[\, \omega_{j,15n+22}\, \overline{T}_{1,j} \to \overline{T}_{1,j,j}\,]_2 \\
[\, \omega_{j,15n+22}\, T_{2,j} \to T_{2,j,n+1+j}\,]_2 \\
[\, \omega_{j,15n+22}\, \overline{T}_{2,j} \to \overline{T}_{2,j,n+1+j}\,]_2 \\
[\, T_{1,j,0}\,]_2 \to y_j\, [\quad]_2 \\
[\, \overline{T}_{1,j,0}\,]_2 \to \overline{y}_j\, [\quad]_2 \\
[\, T_{2,j,0}\,]_2 \to z_j\, [\quad]_2 \\
[\, \overline{T}_{2,j,0}\,]_2 \to \overline{z}_j\, [\quad]_2
\end{array}\right\} \text{for } 0 \leq j \leq n$$

$$\left.\begin{array}{l}
[\, T_{1,j,k} \to T_{1,j,k-1}\,]_2 \\
[\, \overline{T}_{1,j,k} \to \overline{T}_{1,j,k-1}\,]_2
\end{array}\right\} \text{for } 0 \leq j \leq n, 1 \leq k \leq n$$

$$\left.\begin{array}{l}
[\, T_{2,j,k} \to T_{2,j,k-1}\,]_2 \\
[\, \overline{T}_{2,j,k} \to \overline{T}_{2,j,k-1}\,]_2
\end{array}\right\} \text{for } 0 \leq j \leq n, 1 \leq k \leq 2n+1$$

5.10 Output 2 Phase

$$\left.\begin{array}{l}
[\, y_j\, y_j \to y_j\,]_1 \\
[\, \overline{y}_j\, \overline{y}_j \to \overline{y}_j\,]_1 \\
[\, z_j\, z_j \to z_j\,]_1 \\
[\, \overline{z}_j\, \overline{z}_j \to \overline{z}_j\,]_1 \\
[\, \omega_{i,17n+26}\, y_j \to y_j^*\,]_1 \\
[\, \omega_{i,17n+26}\, z_j \to z_j^*\,]_1 \\
[\, y_j^*\,]_1 \to b_{1,j}\, [\quad]_1 \\
[\, z_j^*\,]_1 \to b_{2,j}\, [\quad]_1
\end{array}\right\} \text{for } 0 \leq j \leq n$$

(6) The input membrane is the membrane labelled by 1 ($i_{in} = 2$) and the output region is the environment ($i_{out} = env$).

6 An Overview of the Computations

Let $x \in \mathbb{N}$ an instance of the `FACTORIZATION` problem, that is, x is a natural number whose binary representation is given by $(x_0, \ldots, x_n)$, and such that $x = y \cdot z$ being y and z prime numbers with $y \geq z$. Then, x will be processed by the membrane system $\Pi(k_x) + cod(x)$, where $cod(x) = \{a_0^{x_0}, \ldots, a_n^{x_n}\}$.

The family $\{\Pi(n) \mid n \in \mathbb{N}\}$ designed to solve the `FACTORIZATION` problem captures the behaviour of a brute force algorithm: (a) all possible pairs of natural numbers y, z, with $y, z \leq x$ are produced; (b) the product $y \cdot z$ is computed; and (c) the output is the pair $\{y, z\}$ if and only if $x = y \cdot z$. Next, we briefly describe the stages in which the computations of membrane system $\Pi(n)$ are structured, where $n = k_x$, being x an instance of the `FACTORIZATION` problem.

6.1 Generation Stage

At this stage, 2^{2n+2} membranes labelled by 2 are generated in such manner that each of them contains $n + 4$ copies of possible candidate pairs of natural

numbers y, z, whose binary representation have at most $n+1$ digits, which will be represented by symbols $T^*_{h,j}$ and $\overline{T}^*_{h,j}$. For that, first of all, the code $cod(x)$ of the instance $x = (x_0, \ldots, x_n)$ changes to $\{\rho_i \mid 0 \leq i \leq n\}$, where $\rho_j = X_j$ if $x_j = 1$, $\rho_j = \overline{X}_j$ if $x_j = 0$. From the beginning, division rules to objects $\alpha_{1,j,j}$ and $\alpha_{2,j,n+1+j}$ are applied. Second, objects $t_{h,j,v}$ and $\bar{t}_{h,j,v}$ are used in order to remove undesired objects $T_{h,j}$ and $\overline{T}_{h,j}$. Finally, objects $\beta_{h,j,3n+6}$ will produce objects $T^*_{h,j}$ and $\overline{T}^*_{h,j}$ encoding all possible different candidates y, z of natural numbers in each membrane labelled by 2. This stage takes $3n+7$ steps.

6.2 Multiplication Stage

At this stage, the pair of natural numbers encoded in each membrane labelled by 2, is multiplied. For that, first all bits represented by objects $T^*_{1,j}$ or $\overline{T}^*_{1,j}$ are multiplied with all bits represented by objects $T^*_{2,j'}$ or $\overline{T}^*_{2,j'}$, and objects $P_{j+j'}$ are produced. Second, objects $P_{j+j'}$ are handled in order to be sure that there is, at most, one bit for each position. In order to have a complete binary representation of these numbers, that is, the representation of each bit of the product, we use object P_j to represent that the bit j equals 1, and object $\overline{P}_j$ if bit j equals 0. This stage takes $2n+4$ steps.

6.3 Equality Checking Stage

Here, in each membrane labelled by 2, the instance x encoded by objects X_j and $\overline{X}_j$ is compared to the product $y \cdot z$, represented by objects P_j and $\overline{P}_j$. If they are equal, that is, $y \cdot z = x$, then objects encoding y, z remain in that membrane, and they are removed otherwise. First, objects X_j and $\overline{X}_j$ are compared with objects P_j and $\overline{P}_j$. Next, these partial comparisons represented by objects e_j and e^*_j are used to compare the entire number. If some object $T_{h,j}$, with $j \geq n+1$, appears, that is, the binary representation of the product has more "useful" bits than the original number, then all the objects are erased. This stage takes $4n+4$ steps.

6.4 Trivial Solution Check Stage

Next, solutions y, z with either $y = 1$ or $z = 1$ (*trivial* solutions) are removed. For that, bits are checked to be sure that the two numbers are different from 1, and remove them otherwise. This stage takes $2n+2$ steps.

6.5 First Delete Stage

In order to remove remaining objects from a membrane, a garbage recollection strategy is used, so if an object G_{4n+3} appears in a membrane, then all objects in such a membrane are removed. This stage takes $4n+4$ steps.

6.6 Second Delete Stage

If $y \cdot z = x$ and $y \neq z$ then we have two membranes labelled by 2 such that objects $T_{1,j}$ $\overline{T}_{1,j}$ encode y and objects $T_{2,j}$ $\overline{T}_{2,j}$ encode z, but one of them represents that $y > z$ and the other one represents that $y < z$. In this situation, membrane containing objects encoding $y > z$ is distinguished and the corresponding objects of the other membrane are removed. In the case $y = z$, objects encoding these natural numbers will be kept in both membranes. For that, objects $C_{r,j,k}$ will be produced. If the j-th bit of number y will be smaller than the j-th bit of z, then $r = 2$, on the contrary $r = 0$. If j-th bit of both y and z are the same one then $r = 1$. Later, these objects are used to compare the entire numbers. This stage takes $n + 2$ steps.

6.7 Output 1 Stage

In this stage, objects representing numbers y and z are going to be sent out to membrane 1. To make this stage deterministic, first objects $T_{1,j}$ and $\overline{T}_{1,j}$ and second objects $T_{2,j}$ and $\overline{T}_{2,j}$ are released to membrane labelled by 1. At the end of the stage, objects y_j and $\overline{y}_j$ represent $T_{1,j}$ and $\overline{T}_{1,j}$ in membrane 1. Similarly, objects z_j and $\overline{z}_j$ represent $T_{2,j}$ and $\overline{T}_{2,j}$ in membrane 1. This stage takes $4n+5$ steps.

6.8 Output 2 Stage

Finally, binary representations of the numbers y and z are going to be sent out to the environment by using objects of the final alphabet. First, the perfect square case (two copies of objects y_j or $\overline{y}_j$ and two copies of objects z_j or $\overline{z}_j$ appear) has to be taken into account. For that, two objects y_j (or $\overline{y}_j$) will produce only one object y_j (or $\overline{y}_j$), and similarly for objects z_j and $\overline{z}_j$. Next, each object y_j (resp., z_j) will produce an object y_j^* (resp., z_j^*) cooperating with object ω_{17n+26}. Finally, each object y_j^* produce an object $b_{1,j}$ at the environment, and each object z_j^* produce an object $b_{2,j}$ at the environment. This stages takes at most $2n + 3$ steps.

At Table 1, the steps used by each stage, besides the initial and final configuration of each one are indicated.

7 Conclusions and Future Work

The `FACTORIZATION` problem (*given a natural number n which is product of two large primes, find its decomposition*) can be characterized by a *partial function* `FACT` from $\mathbb{N}$ to $\mathbb{N}^2$ defined as follows: for each natural number x which is the product of two prime numbers y, z, with $y \geq z$, we have $\texttt{FACT}(x) = (y, z)$. This problem belongs to the class **FNP** and it is conjectured that it is an intractable problem, assuming that $\mathbf{P} \neq \mathbf{NP}$. Besides, it is the basis for some cryptographic systems as important as RSA, a *de facto* standard for digital signatures. In order

Stage	Steps	Initial configuration	Final configuration
Generation	$3n+7$	0	$3n+7$
Multiplication	$2n+4$	$3n+7$	$5n+11$
Equality checking stage	$4n+4$	$5n+11$	$9n+15$
Trivial solution check	$2n+2$	$7n+13$	$9n+15$
First delete	$4n+4$	$9n+15$	$13n+19$
Second delete	$n+2$	$12n+18$	$13n+20$
Output 1	$4n+5$	$13n+20$	$17n+25$
Output 2	$\leq 2n+3$	$17n+25$	$\leq 19n+28$

Table 1. Number of steps by each stage

to provide solutions in the framework of Membrane Computing, new membrane systems computing partial functions between natural numbers are introduced.

In this work, a linear time solution to the `FACTORIZATION` problem is presented by means of a family of polarizationless P system with active membranes without dissolution rules which use minimal cooperation and minimal production in object evolution rules. This solution improves the previous ones given in the membrane computing framework, in the sense that the use of syntactical ingredients is significantly lower.

P-Lingua [25] and MeCoSim [24] have been very useful as assistant tools for the process of verifying this design. An interesting future work is to use this model in a GPU-based simulator, since it can accelerate the processing of the computation. Several simulators of P systems have been implemented using the NVIDIA CUDA framework. In fact, in the PMCGPU project [26] we can see some simulators for different types of P systems. Some stages could be optimized in order to have faster communications between the multiple cores of the graphic card, like the encoding of objects into integers or the omission of some objects that only act to synchronize the system. Another way to speed up the algorithm would be to omit all the membranes containing an element $c_{1,i}$, because we know that these bits equal zero in our initial number x.

8 Acknowledgements

This work was supported by Project TIN2017-89842-P of the Ministerio de Economía y Competitividad of Spain and by Grant No 61320106005 of the National Natural Science Foundation of China.

References

1. T.H. Cormen, C.E. Leiserson, R.L. Rivest: An Introduction to Algorithms. *The MIT Press*, Cambridge, Massachussets, 1994.
2. D. Díaz-Pernil, H.A. Christinal, M.Á. Gutiérrez-Naranjo: Solving the 3-COL Problem by Using Tissue P Systems without Environment and Proteins on Cells. In: *Proceedings of the Fourteenth Brainstorming Week on Membrane Computing*, 1-5 February, 2016, Sevilla, Spain, Fénix Editora, 163–172.

3. W. Diffie, M. Hellman: New directions in cryptography. *IEEE Transactions on Information Theory* **22**, 6, 644–654.
4. M.A. Gutiérrez–Naranjo, M.J. Pérez–Jiménez, A. Riscos–Núñez, F.J. Romero–Campero: On the power of dissolution in P systems with active membranes. *Lecture Notes in Computer Science* **3850** (2006), 224–240.
5. M. Ionescu, Gh. Păun, T. Yokomori: Spiking Neural P Systems. *Fundamenta Informaticae* **71**, 2,3 (February 2006), 279–308.
6. A. Leporati, C. Zandron, G. Mauri: Solving the factorization problem with P systems. *Progress in Natural Science* **17**, 4 (2007), 471–478.
7. C. Martín-Vide, Gh. Păun, J. Pazos, A. Rodríguez-Patón: Tissue P systems. *Theoretical Computer Science* **296**, 2 (2003), 295–326.
8. R. Merkle, M. Hellman: Hiding information and signatures in trapdoor knapsacks. *IEEE Transactions on Information Theory* **24**, 5, 525–530.
9. T. Murakawa, A. Fujiwara: Operations and Factorization using Asynchronous P systems. *International Journal of Networking and Computing* **2**, 2 (2012), 217–233.
10. Gh. Păun. Computing with membranes. *Journal of Computer and Systems Science* **61**, 1 (2000), 108–143.
11. Gh. Păun: P systems with active membranes: Attacking **NP**–complete problems. *Journal of Automata, Languages and Combinatorics* **6** (2001), 75–90. A preliminary version appeared in *Centre for Discrete Mathematics and Theoretical Computer Science Research Reports Series*, CDMTCS-102, May 1999.
12. M.J. Pérez-Jiménez, A. Riscos-Núñez, M. Rius-Font, L. Valencia-Cabrera: The relevance of the environment on the efficiency of tissue P systems. *Lecture Notes in Computer Science* **8340** (2014), 308–321.
13. M.J. Pérez-Jiménez, Á. Romero-Jiménez, F. Sancho-Caparrini: Complexity classes in models of cellular computing with membranes. *Natural Computing* **2**, 3 (2003), 265–285.
14. M. Sipser: Introduction to the Theory of Computation. *International Thomson Publishing* (1996).
15. A. Riscos-Núñez: Programación celular: Resolución eficiente de problemas numéricos NP-completos. PhD. Thesis, University of Seville, Spain, 2003.
16. R.L. Rivest, A. Shamir, L. Adleman: A method for obtaining digital signatures and public-key cryptosystems. *CAMC* **21**, 2 (1978), 120–126.
17. Á. Romero-Jiménez, M.J. Pérez-Jiménez: Generation of Diophantine Sets by Computing P Systems with External Output. *Lecture Notes in Computer Science* **2509** (2002), 176–190.
18. L. Valencia-Cabrera, D. Orellana-Martín, A. Riscos-Núñez, M.J. Pérez-Jiménez: Minimal cooperation in polarizationless P systems with active membranes. In C. Graciani, Gh. Păun, D. Orellana-Martn, A. Riscos-Nez, L. Valencia-Cabrera (eds.): *Proceedings of the Fourteenth Brainstorming Week on Membrane Computing*, 1-5 February, 2016, Sevilla, Spain, Fénix Editora, 327–356.
19. L. Valencia-Cabrera, D. Orellana-Martín, M.Á. Martínez-del-Amor, A. Riscos-Núñez, M.J. Pérez-Jiménez: Polarizationless P systems with active membranes: Computational complexity aspects. *Journal of Automata, Languages and Combinatorics* **21**, 1-2 (2016), 101–117.
20. L. Valencia-Cabrera, D. Orellana, M.A. Martínez-del-Amor, A. Riscos, M.J. Pérez-Jiménez: Reaching efficiency through collaboration in membrane systems: Dissolution, polarization and cooperation. *Theoretical Computer Science* **701** (2017), 226–234.
21. L. Valencia-Cabrera, D. Orellana, A. Riscos, M.J. Pérez-Jiménez: Counting membrane systems. *Lecture Notes in Computer Science* **10725** (2017), 74–87.

22. X. Zang, Y. Niu, L. Pan, M.J. Pérez-Jiménez: Linear Time Solution to Prime Factorization by Tissue P Systems with Cell Division. *Proceedings of the Ninth Brainstorming Week on Membrane Computing*, 2011, 355–372.
23. L.G. Valiant: The complexity of computing the permanent. *Theoretical Computer Science* **8**, 2 (1979), 189–201.
24. MeCoSim Website: `http://www.p-lingua.org/mecosim/`
25. P-Lingua Website: `http://www.p-lingua.org/wiki/index.php/Main_Page`
26. PMCGPU Website: `https://sourceforge.net/projects/pmcgpu/`