Hindawi
Complexity
Volume 2020, Article ID 6765097, 10 pages
https://doi.org/10.1155/2020/6765097

Research Article

From NP-Completeness to DP-Completeness: A Membrane Computing Perspective

Luis Valencia-Cabrera, David Orellana-Martín, Miguel Á. Martínez-del-Amor, Ignacio Pérez-Hurtado, 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

Correspondence should be addressed to David Orellana-Martín; dorellana@us.es

Received 23 January 2020; Revised 21 July 2020; Accepted 28 July 2020; Published 30 August 2020

Academic Editor: Jose C. Valverde

Presumably efficient computing models are characterized by their capability to provide polynomial-time solutions for **NP**-complete problems. Given a class $\mathscr{R}$ of recognizer membrane systems, $\mathscr{R}$ denotes the set of decision problems solvable by families from $\mathscr{R}$ in polynomial time and in a uniform way. $\mathbf{PMC}_{\mathscr{R}}$ is closed under complement and under polynomial-time reduction. Therefore, if $\mathscr{R}$ is a presumably efficient computing model of recognizer membrane systems, then $\mathbf{NP} \cup \mathbf{co\text{-}NP} \subseteq \mathbf{PMC}_{\mathscr{R}}$. In this paper, the lower bound $\mathbf{NP} \cup \mathbf{co\text{-}NP}$ for the time complexity class $\mathbf{PMC}_{\mathscr{R}}$ is improved for any presumably efficient computing model $\mathscr{R}$ of recognizer membrane systems verifying some simple requirements. Specifically, it is shown that $\mathbf{DP} \cup \mathbf{co\text{-}DP}$ is a lower bound for such $\mathbf{PMC}_{\mathscr{R}}$, where **DP** is the class of differences of any two languages in **NP**. Since $\mathbf{NP} \cup \mathbf{co\text{-}NP} \subseteq \mathbf{DP} \cap \mathbf{co\text{-}DP}$, this lower bound for $\mathbf{PMC}_{\mathscr{R}}$ delimits a thinner frontier than that with $\mathbf{NP} \cup \mathbf{co\text{-}NP}$.

1. Introduction

Membrane Computing is a computing paradigm inspired by some basic biological features. Specifically, it is inspired by the structure and functioning of the living cells, as well as from the cooperation of cells in tissues, organs, and organisms. This paradigm provides distributed, parallel nondeterministic computing models whose computational devices are generically called *membrane systems* or *P systems* and has processing units called *compartments*. In this framework, there basically exist two ways to consider computational devices: cell-like membrane systems and tissue-like membrane systems. The former uses the biological membranes arranged hierarchically as compartments, inspired by the structure of the cell. The latter takes as compartments the cells placed in the nodes of a directed graph, which is inspired by the cell intercommunication in tissues.

Recognizer membrane systems were introduced in [1], in order to provide a natural framework in Membrane Computing, aiming to solve decision problems by means of language recognition. A recognizer membrane system has the following additional syntactic and semantic peculiarities: (a) the working alphabet has two distinguished objects (yes and no); (b) there exist an input compartment, and there is also an input alphabet strictly contained in the working alphabet; (c) the initial content of each compartment is a multiset of objects from the working alphabet not belonging to the input alphabet; (d) all computations halt; and (e) for each computation and only at its last step, either object yes or object no (but not both) must have been released to the environment. Recognizer membrane systems can only have accepting or rejecting computations, depending on whether they return yes or no to the environment, respectively.

On this basis, given a recognizer membrane system Π, for each multiset m over the input alphabet, we denote by $\Pi + m$ the membrane system obtained from Π by adding the multiset m to the content of the input membrane at the initial configuration. Unlike a Turing machine, all the

elements that make up a recognizer membrane system have a finite description. Therefore, while a decision problem (with infinite instances) can be solved by a single Turing machine, an infinite family of recognizer membrane systems is necessary to solve it. We say that a family $\Pi = \{\Pi(n) \mid n \in \mathbb{N}\}$ of recognizer membrane systems solves a decision problem X in polynomial time in a uniform way if such family Π can be generated by a deterministic Turing machine working in polynomial time, and there exists a pair (cod, s) of polynomial-time computable functions over the set of instances of X verifying the following: (a) for each instance $u \in I_X$, $s(u)$ is a natural number and $\mathrm{cod}(u)$ is an input multiset of the system $\Pi(s(u))$; (b) for each $n \in \mathbb{N}$, the set $s^{-1}(\{n\})$ is a finite set; and (c) the family Π is *polynomially bounded*, *sound*, and *complete* with regard to (X, cod, s). The pair (cod, s) is called a *polynomial encoding* from the decision problem X to the family Π. For more details, see [2]. Given a computing model $\mathscr{R}$ of recognizer membrane systems, $\mathbf{PMC}_{\mathscr{R}}$ denotes the set of decision problems solvable by families from $\mathscr{R}$ in polynomial time in a uniform way. The class $\mathbf{PMC}_{\mathscr{R}}$ is closed under complement and under polynomial-time reduction [2].

A computing model is said to be *efficient* (respectively, *presumably efficient*) if it has the ability to provide polynomial-time solutions for intractable problems (resp., **NP**-complete problems) [3]. The term *presumably efficient* refers to the fact that, as generally believed, if $\mathbf{P} \neq \mathbf{NP}$ then each **NP**-complete problem is intractable and, consequently, any presumably efficient model would be efficient. The computing model $\mathscr{AM}$ of recognizer P systems with active membranes and electrical charges is presumably efficient [4], as well as the computing model $\mathscr{AM}^0(+d, +ne)$ of recognizer polarizationless P systems with active membranes which makes use of dissolution rules and division for elementary and nonelementary membranes [5]. However, the computing model $\mathscr{AM}^0(-d)$ of recognizer polarizationless P systems with active membranes not making use of dissolution rules is not efficient, even when allowing division rules for elementary and nonelementary membranes [5]. Thus, if $\mathscr{R}$ is a presumably efficient computing model of recognizer membrane systems, then $\mathbf{NP} \cup \mathbf{co\text{-}NP} \subseteq \mathbf{PMC}_{\mathscr{R}}$.

Several variants of P systems have been demonstrated to be presumably efficient. A solution to QSAT has been given in the framework of P systems with active membranes and division rules for elementary and nonelementary membranes but only evolution rules of the RHS limited to length 1 [6]. When creation rules are used instead of division rules as a method of generating an exponential workspace in polynomial time, it is possible to get rid of polarizations to obtain a solution to the QSAT problem [7]. A new research line taking tissue P systems with symport/antiport rules where objects can evolve during the application of the rules has been developed recently. Several results have been obtained, and new solutions to **NP**-complete problems have been given in this framework [8].

The main contribution of this paper is to provide a lower bound for $\mathbf{PMC}_{\mathscr{R}}$ delimiting a frontier with such class that is thinner to the one found with respect to $\mathbf{NP} \cup \mathbf{co\text{-}NP}$, with $\mathscr{R}$ being a presumably efficient computing model of recognizer membrane systems verifying some simple requirements (e.g., in the case of cell-like membrane systems, allowing object evolution rules, dissolution rules, and communication rules). It is also worth pointing out that the methodology used to obtain this result is constructive in the following sense: given an **NP**-complete problem X, a **DP**-complete problem Y_X is associated, in such a manner that if $X \in \mathbf{PMC}_{\mathscr{R}}$ then $Y_X \in \mathbf{PMC}_{\mathscr{R}}$, for such kind of presumably efficient computing model $\mathscr{R}$ of recognizer membrane systems.

The remainder of this paper is structured as follows. Section 2 will be devoted to introduce some concepts that can be useful to understand the rest of the work. The complexity class **DP** is introduced in Section 3, where the main theorem presented in this paper is established. In Section 4, the methodology will be applied with an example. Finally, in Section 5, the most relevant conclusions are drawn, and some possible further developments are outlined.

2. Preliminaries

In order for the paper to be self-contained, several basic concepts and results will be introduced to facilitate a better understanding of the text.

An *alphabet* is a nonempty set whose elements are called *symbols*. A *string* u over an alphabet is an ordered finite sequence of symbols. The *length* of a string u, denoted by $|u|$, is the number of occurrences of symbols that it contains. The empty string (with length 0) contains no symbols, and it is denoted by λ. The set of all strings over an alphabet Γ is denoted by Γ^*. A *language* over Γ is a subset of Γ^*. If L_1, L_2 are languages over Γ, the set-theoretic *difference* of L_1 and L_2 (in this order), denoted by $L_1 \setminus L_2$, is defined as follows: $L_1 \setminus L_2 = \{u \in \Gamma^* \mid u \in L_1 \wedge u \notin L_2\}$.

A *multiset* m over an alphabet Γ is a mapping from Γ onto the set $\mathbb{N}$ of natural numbers. A multiset m over Γ is finite (respectively, empty) if the set $\{x \in \Gamma \mid m(x) > 0\}$ is a finite (respectively, empty) set. If m, m' are multisets over Γ, then the *union* of m and m', denoted by $m + m'$, is the multiset over Γ defined as follows: $(m + m')(x) = m(x) + m'(x)$, for each $x \in \Gamma$.

2.1. The Cantor Pairing Function. The Cantor pairing function is the mapping from $\mathbb{N} \times \mathbb{N}$ onto $\mathbb{N}$ defined as follows: the natural number $[(t_1 + t_2) \cdot (t_1 + t_2 + 1)/2] + t_1$ is associated with every ordered pair (t_1, t_2) of natural numbers. We denote it by $\langle t_1, t_2 \rangle$, that is, $\langle t_1, t_2 \rangle = [(t_1 + t_2) \cdot (t_1 + t_2 + 1)/2] + t_1$. The Cantor pairing function is a primitive recursive function and bijective from $\mathbb{N} \times \mathbb{N}$ onto $\mathbb{N}$. Consequently, for each natural number t, there exists a unique pair of natural numbers t_1 and t_2 such that $t = \langle t_1, t_2 \rangle$.

2.2. Decision Problems and Languages. A decision problem X is an ordered pair (I_X, θ_X), where I_X is a language over a finite alphabet Σ_X and θ_X is a total Boolean function over I_X. The elements of I_X are called *instances* of the problem X.

If $\theta_X(u) = 1$ (respectively, $\theta_X(u) = 0$) then we say that the answer of the problem for instance u is "yes" (respectively, "no"). Each decision problem X has associated a language L_X over Σ_X: the set of instances whose answer is "yes." Conversely, every language L over an alphabet Σ has associated a decision problem whose set of instances is Σ^*, and for each $u \in \Sigma^*$ the answer of the problem is "yes" if and only if $u \in L$.

The *complement problem* $\overline{X}$ of a decision problem $X = (I_X, \theta_X)$ is the decision problem with the same set of instances, characterized by the following property. For each instance $u \in I_X$, the answer of $\overline{X}$ for u is "yes" if and only if the answer of X for u is "no." Obviously, $\overline{\overline{X}} = X$. In accordance with this definition, a problem X is an **NP**-complete problem if and only if its complement problem $\overline{X}$ is a **co**-**NP**-complete problem.

2.3. The Satisfiability Problem. Let us recall that the set of propositional Boolean variables VP is a countably infinite set $\{x_1, x_2, \ldots\}$ such that each of their elements can take only two possible Boolean values (*truth values*) from $\{0, 1\}$. *Boolean expressions* are captured through the recursive definition of the minimal set BE verifying: (a) any propositional Boolean variable is in BE; (b) if φ is in BE then its negation φ is also in BE; and (c) if φ_1 and φ_2 are in BE then its disjunction $\varphi_1 \vee \varphi_2$ is also in BE. Every Boolean expression φ has a finite number of Boolean variables associated with it, being denoted by $\mathrm{Var}(\varphi)$. A truth assignment σ for a Boolean expression φ is a mapping from $\mathrm{Var}(\varphi)$ onto the set $\{0, 1\}$. According to the classical definition of truth tables for negation and disjunction, every truth assignment σ for a Boolean expression φ provides a unique Boolean value to it. If the associated truth value is 1, then it is said that σ makes the Boolean expression φ true.

A *literal* is a Boolean variable or the negation of a Boolean variable. A *clause* is a finite disjunction of literals. Moreover, a Boolean expression is in conjunctive normal form if it is a finite conjunction of clauses. We say that a Boolean expression is in a *simplified conjunctive normal form* if it is in conjunctive normal form, literals are not repeated in each clause, and a literal and its negation do not appear in the same clause. We say that a Boolean expression φ is *satisfiable* if and only if there exists a truth assignment for φ making such an expression true. Let us recall that the satisfiability problem, denoted by SAT, is the problem of deciding whether a given Boolean formula in a simplified conjunctive normal form is satisfiable. The complement problem $\overline{\mathrm{SAT}}$ is denoted by UNSAT: *determining whether or not a Boolean expression in a simplified conjunctive normal form is unsatisfiable*. That is, given a Boolean expression φ in a simplified conjunctive normal form, determine whether any truth assignment σ for φ makes it false. It is well known that the SAT problem is an **NP** complete problem [9]. Thus, the UNSAT problem is a **co**-**NP** complete problem.

2.4. The Polynomial-Time Hierarchy and the Boolean Hierarchy. Let us recall that if $\mathscr{C}$ is a time complexity class and A is an oracle (a set of queries) then $\mathscr{C}^A$ is the class of all languages decidable by deterministic or nondeterministic Turing machines working in time bound as in $\mathscr{C}$ and having A as oracle (see [10], for details).

The *polynomial-time hierarchy* was introduced by Stockmeyer [11], inspired by the Kleene arithmetical hierarchy [12], where recursive time is replaced by deterministic polynomial time. The polynomial-time hierarchy is a sequence of complexity classes defined as follows: $\Delta_0\mathbf{P} = \Sigma_0\mathbf{P} = \Pi_0\mathbf{P} = \mathbf{P}$ and $\Delta_{k+1}\mathbf{P} = \mathbf{P}^{\Sigma_k\mathbf{P}}$, $\Sigma_{k+1}\mathbf{P} = \mathbf{NP}^{\Sigma_k\mathbf{P}}$, and $\Pi_{k+1}\mathbf{P} = \mathbf{co} - \mathbf{NP}^{\Sigma_k\mathbf{P}}$, for each $k \in \mathbb{N}$. The first level of this hierarchy is $\Delta_1\mathbf{P} = \mathbf{P}$, $\Sigma_1\mathbf{P} = \mathbf{NP}$, and $\Pi_1\mathbf{P} = \mathbf{co}$-$\mathbf{NP}$ and the second level is $\Delta_2\mathbf{P} = \mathbf{P}^{\mathbf{NP}}$, $\Sigma_2\mathbf{P} = \mathbf{NP}^{\mathbf{NP}}$, and $\Pi_2\mathbf{P} = \mathbf{co}$-$\mathbf{NP}^{\mathbf{NP}}$. The *cumulative polynomial-time hierarchy* is the class $\mathbf{PH} = \cup_{k\in\mathbb{N}}\Sigma_k\mathbf{P}$.

The *Boolean hierarchy* is a sequence of complexity classes obtained from **NP** by means of Boolean operations (union, intersection, and complementation). It was defined in [13] as follows: $\mathbf{NP}_0 = \mathbf{P}$, and $\mathbf{NP}_{2k+1} = \{L \cup L' \mid L \in \mathbf{NP}_{2k} \wedge L' \in \mathbf{NP}\}$, $\mathbf{NP}_{2k+2} = \{L \cap \overline{L'} \mid L \in \mathbf{NP}_{2k+1} \wedge L' \in \mathbf{NP}\}$, for each $k \in \mathbb{N}$. Sometimes, $\mathbf{NP}_k$ sets are denoted by $\mathbf{BH}_k$. The first level of this hierarchy is **NP** and the second level is the set of languages which are intersections of a language in **NP** and another one in **co**-**NP**.

3. The Complexity Class DP

The complexity class **DP** was introduced by Papadimitriou and Yannakis in 1982 [14] as follows.

Definition 1. A language L is in the class **DP** if and only if there are two languages L_1 and L_2 such that $L_1, L_2 \in \mathbf{NP}$ and $L = L_1 \setminus L_2$.

The complexity class **DP** can be easily characterized by the following property: a language L is in **DP** if and only if there exist two languages L_1 and L_2 in **NP** such that $L = L_1 \cap \overline{L_2}$. Class **DP** is the second level in the Boolean hierarchy. Besides, classes in the Boolean hierarchy can be described by finite unions of **DP** sets (or by finite intersections of **co**-**DP** sets). For more details, see [13, 15].

The complexity class **DP** can also be expressed in terms of decision problems as follows: a decision problem X is in class **DP** if and only if the language L_X associated with it belongs to class **DP**, that is, if there are two decision problems $X_1 = (I_{X_1}, \theta_{X_1})$ and $X_2 = (I_{X_2}, \theta_{X_2})$ such that L_{X_1} and L_{X_2} are languages in **NP** and $L_X = L_{X_1} \cap \overline{L_{X_2}}$. It is easy to prove that $\mathbf{NP} \cup \mathbf{co}$-$\mathbf{NP} \subseteq \mathbf{DP} \cap \mathbf{co}$-$\mathbf{DP}$.

Next, we define the *product of two decision problems*, and some properties of the new concept are analyzed.

Definition 2. Let $X_1 = (I_{X_1}, \theta_{X_1})$ and $X_2 = (I_{X_2}, \theta_{X_2})$ be two decision problems. The *product problem* of X_1 and X_2, denoted by $X_1 \otimes X_2$, is the decision problem $(I_{X_1\otimes X_2}, \theta_{X_1\otimes X_2})$ defined as follows: $I_{X_1\otimes X_2} = I_{X_1} \times I_{X_2}$, and for each $(u_1, u_2) \in I_{X_1\otimes X_2}$, we have $\theta_{X_1\otimes X_2}(u_1, u_2) = 1$ if and only if $\theta_{X_1}(u_1) = 1$ and $\theta_{X_2}(u_2) = 1$.

A product problem Y_X can be naturally associated to a problem X. This problem Y_X is the *product problem*, defined as follows: $Y_X = (I_{Y_X}, \theta_{Y_X})$ such that $I_{Y_X} = \{(u_X,$

$u_{\overline{X}}) \mid u_X \in I_X, u_{\overline{X}} \in I_{\overline{X}}\}$, being $\overline{X}$ the complementary problem of X; $\theta_{Y_X}((u_1,u_2)) = 1$ if and only if $\theta_X(u_1) = \theta_{\overline{X}}(u_2) = 1$.

According to these definitions, the following results concerning to the product of two decision problems are obtained.

Proposition 1. *Let X_1 and X_2 be decision problems:*

*(i) If $X_1 \in$ **NP** and $X_2 \in$ **co-NP**, then $X_1 \otimes X_2 \in$ **DP***

*(ii) If X_1 is an **NP** complete problem and X_2 is a **co-NP** complete problem, then $X_1 \otimes X_2$ is a **DP** complete problem*

*(iii) If X_1 is an **NP** complete problem, then the product problem $X_1 \otimes \overline{X_1}$ is a **DP** complete problem*

*Since the SAT problem is an **NP** complete problem, we deduce that the product problem $SAT \otimes \overline{SAT}$ is a **DP** complete problem. This problem is referred as the SAT-UNSAT problem [10] and can be expressed as follows: given two Boolean expressions φ_1 and φ_2 in simplified conjunctive normal forms, determine whether or not φ_1 is satisfiable and φ_2 is unsatisfiable.*

4. The Methodology

The main result of this paper refers to the robustness of the complexity class $\mathbf{PMC}_{\mathscr{R}}$ associated with some computing models of recognizer membrane systems, with respect to the operation product of two decision problems.

Main Theorem 1. *Let $\mathscr{R}$ be a computing model of recognizer noncooperative cell-like P systems allowing dissolution rules, object evolution rules, and communication rules (send-in and send-out). If X_1 and X_2 are decision problems belonging to the time complexity class $\mathbf{PMC}_{\mathscr{R}}$, then $X_1 \otimes X_2 \in \mathbf{PMC}_{\mathscr{R}}$.*

Proof. First, without loss of generality, we can assume that the semantics associated with the application of the rules of systems from $\mathscr{R}$ is similar to P systems with active membranes, that is, at one transition step, a membrane can be a subject of only one rule of type dissolution, communication, or division, but object evolution rules are applied simultaneously with one of these rules and, moreover, in a maximal way. By using a different semantics, only few parameters that will appear in the proof should be modified.

For $i = 1, 2$, let $\Pi^{(i)} = \{\Pi^{(i)}(t) \mid t \in \mathbb{N}\}$ be a family of membrane systems from $\mathscr{R}$ solving X_i in polynomial-time in a uniform way. Let (cod_i, s_i) be a polynomial encoding from X_i into $\Pi^{(i)}$. Let $p_i(n)$ be a polynomial function such that, for each instance u_i from X_i, any computation of $\Pi^{(i)}(s_i(u_i)) + cod_i(u_i)$ performs at most $p_i(|u_i|)$ steps. In this situation, a family $\Pi = \{\Pi(t) \mid t \in \mathbb{N}\}$ of membrane systems from $\mathscr{R}$ will be defined from $\Pi^{(1)}$ and $\Pi^{(2)}$, in such a manner that Π provides a uniform and polynomial-time solution to the product problem $X_1 \otimes X_2$. The family Π will be called the *product family* of families $\Pi^{(1)}$ and $\Pi^{(2)}$, and it will be denoted by $\Pi^{(1)} \otimes \Pi^{(2)}$.

First, a pair of polynomial-time functions (cod, s) over the set of instances of $X_1 \otimes X_2$ is considered as follows: (a) $\mathrm{cod}(u_1,u_2) = \mathrm{cod}_1(u_1) + \mathrm{cod}_2(u_2)$ and (b) $s(u_1,u_2) = \langle s_1(u_1), s_2(u_2)\rangle$, for each $(u_1,u_2) \in I_{X_1 \otimes X_2}$. Hence, we want the instance (u_1,u_2) to be processed by the system $\Pi(s(u_1,u_2))$ with input multiset $\mathrm{cod}(u_1,u_2)$. In the following, the membrane system $\Pi(\langle s_1(u_1), s_2(u_2)\rangle)$ will be explicitly defined from $\Pi^{(1)}(s_1(u_1))$ and $\Pi^{(2)}(s_2(u_2))$, for each $(u_1,u_2) \in I_{X_1 \otimes X_2}$, so that (a) if there exist accepting computations of the systems $\Pi^{(1)}(s_1(u_1)) + \mathrm{cod}_1(u_1)$ and $\Pi^{(2)}(s_2(u_2)) + \mathrm{cod}_2(u_2)$, then $\theta_{X_1 \otimes X_2} = 1$ and (b) if $\theta_{X_1 \otimes X_2} = 1$, then any computation of $\Pi^{(1)}(s_1(u_1)) + \mathrm{cod}_1(u_1)$ and any computation of $\Pi^{(2)}(s_2(u_2)) + \mathrm{cod}_2(u_2)$ are accepting ones. According to this, (cod, s) will be a polynomial encoding from the product problem $X_1 \otimes X_2$ into the family Π. It is worth pointing out that, in fact, only membrane systems $\Pi(t)$ such that $t \in \mathrm{range}(s)$ are defined, and this is enough to achieve our goal.

In what follows, we will denote $t = \langle s_1(u_1), s_2(u_2)\rangle$, $t_1 = s_1(u_1)$, and $t_2 = s_2(u_2)$. Let us recall that given $t \in \mathbb{N}$, there exists a unique pair of natural numbers t_1 and t_2 such that $t = \langle t_1, t_2\rangle$. The underlying idea of the construction of $\Pi(t)$ is the following: when $\mathrm{cod}(u_1,u_2)$ is considered as the input multiset of this system; then, computations of $\Pi^{(1)}(t_1) + \mathrm{cod}_1(u_1)$ and $\Pi^{(2)}(t_2) + \mathrm{cod}_2(u_2)$ are simulated. The answers of these systems are placed in a membrane where the final decision is taken; accepting computations of $\Pi(t) + \mathrm{cod}(u_1,u_2)$ only come from accepting computations of $\Pi^{(1)}(t_1) + \mathrm{cod}_1(u_1)$ and $\Pi^{(2)}(t_2) + \mathrm{cod}_2(u_2)$.

Without loss of generality we can assume that $\Gamma^{(i)}(t_i) \backslash \{\mathrm{yes}, \mathrm{no}\}$, for $i = 1, 2$, are mutually disjoint, being $\Gamma^{(i)}(t_i)$ the working alphabet of $\Pi^{(i)}(t_i)$. Likewise, without loss of generality, we can assume the same with the corresponding sets of labels $H^{(1)}(t_1)$ and $H^{(2)}(t_2)$. For $1 \le i \le 2$, let q_i be the maximum value of $|\mathrm{cod}_i(u_i)|$ such that $s_i(u_i) = t_i$, and let d_i be the depth of the input membrane $\mathrm{in}_{\Pi^{(i)}(t_i)}$ of $\Pi^{(i)}(t_i)$ in the corresponding membrane structure. Let us denote $m_i = q_i + d_i + 2$, for $i = 1, 2$, and $n = 2 + \max\{d_i + q_i + p_i(|u_i|) : 1 \le i \le 2\}$. Systems $\Pi^{(i)}(t_i) + \mathrm{cod}_i(u_i)$ will be "prepared" after $m_i + 1$ steps, and computations of such systems will start at the next step. It will last $p_i(|u_i|)$ transition steps. Besides, it will take n steps to have both responses prepared to let $\Pi(t) + \mathrm{cod}(u_1,u_2)$ give the answer.

The syntactical ingredients of $\Pi(t)$ are the following:

(i) *Input alphabet.* The input alphabet is $\Sigma(t) = \Sigma^{(1*)}(t_1) \cup \Sigma^{(2*)}(t_2)$, being $\Sigma^{(i*)}(t_i)$ the input alphabet of $\Pi^{(i)}(t_i)$, by replacing objects yes and no by objects yes_i and no_i, for $i = 1, 2$, respectively.

(ii) *Working alphabet.* The working alphabet is

$$\Gamma^{(1*)}(t) \cup \Gamma^{(2*)}(t) \cup \Gamma^{(3)}(t) \cup \Gamma^{(4)}(t) \cup \Gamma^{(5)}(t) \cup \Gamma^{(6)}(t), \tag{1}$$

where

$$\begin{aligned}
\Gamma^{(1*)}(t) &= \left(\Gamma^{(1)}(t_1) \cup \{\text{yes}, \text{no}\}\right) \cup \{\text{yes}_1, \text{no}_1\},\\
\Gamma^{(2*)}(t) &= \left(\Gamma^{(2)}(t_2) \cup \{\text{yes}, \text{no}\}\right) \cup \{\text{yes}_2, \text{no}_2\},\\
\Gamma^{(3)}(t) &= \{\#, \text{YES}, \text{NO}\} \cup \left\{a' \middle| a \in \Sigma(t)\right\},\\
\Gamma^{(4)}(t) &= \left\{a_j \middle| 0 \le j \le m_1 \wedge a \in \Gamma^{(1*)}(t)\right\}\\
&\quad \cup \left\{a_j \middle| 0 \le j \le m_2 \wedge a \in \Gamma^{(2*)}(t)\right\},\\
\Gamma^{(5)}(t) &= \{\alpha_0, \alpha_1\} \cup \left\{\beta_j \middle| 0 \le j \le n\right\},\\
\Gamma^{(6)}(t) &= \left\{\gamma_j \middle| 0 \le j \le \max\{m_1, m_2\}\right\},
\end{aligned} \tag{2}$$

that is, objects $a' \in \Gamma^{(3)}(t)$ and $a_j \in \Gamma^{(4)}(t)$ are "copies" of objects $a \in \Gamma^{(1)}(t_1) \cup \Gamma^{(2)}(t_2)$, and objects in $\Gamma^{(5)}(t)$ and $\Gamma^{(6)}(t)$ are counters for implementing a synchronization process.

(iii) *Set of labels.* The set of labels $H(t)$ is $H^{(1)}(t_1) \cup H^{(2)}(t_2) \cup \{r_t, r_t', r_t'', h^{(1)}(t_1), h^{(2)}(t_2)\}$, where r_t, r_t', r_t'', $h^{(1)}(t_1)$, $h^{(2)}(t_2)$ are different from each other, and $r_t, r_t', r_t'', h^{(1)}(t_1), h^{(2)}(t_2) \notin H^{(1)}(t_1) \cup H^{(2)}(t_2)$.

(iv) *Compartment structure.* The membrane structure of $\Pi(t)$ is obtained from the membrane structures $\mu_{\Pi^{(i)}(t_i)}$, for $i = 1, 2$, by adding four new nodes $r_t, r_t', h^{(1)}(t_1), h^{(2)}(t_2)$ such that (a) r_t is the skin membrane of $\Pi(t)$; (b) r_t is the father of r_t'; (c) r_t' is the father of r_t''; (d) r_t'' is the father of the skin membranes $r_{\Pi^{(i)}(t_i)}$ of $\Pi^{(i)}(t_i)$, for $i = 1, 2$; and (e) $h^{(i)}(t_i)$ is a new son of the input membrane $\text{in}_{\Pi^{(i)}(t_i)}$ of $\Pi^{(i)}(t_i)$, for $i = 1, 2$. More specifically, the membrane structure of $\Pi(t)$ is $\{(r_t, r_t'), (r_t', r_t''), (r_t'', r_{\Pi^{(1)}(t_1)}), (r_t'', r_{\Pi^{(2)}(t_2)})\} \cup \{(\text{in}_{\Pi^{(1)}(t_1)}, h^{(1)}(t_1)), (\text{in}_{\Pi^{(2)}(t_2)}, h^{(2)}(t_2))\} \cup \mu_{\Pi^{(1)}(t_1)} \cup \mu_{\Pi^{(2)}(t_2)}$. It is graphically depicted in Figure 1.

(v) *Initial multisets.* The initial multisets of $\Pi(t)$ are the following: if h is the label of a membrane from $\Pi^{(i)}(t_i)$ whose initial multiset $\mathcal{M}_h^{(i)}(t_i)$ associated with h is nonempty, then the initial multiset $\mathcal{M}_h(t)$ of $\Pi(t)$ is a "copy" of $\mathcal{M}_h^{(i)}(t_i)$ by replacing each symbol a by a_0. Moreover, $\mathcal{M}_{r_t}(t) = \mathcal{M}_{r_t'}(t) = \varnothing$, $\mathcal{M}_{r_t''}(t) = \{\alpha_0, \beta_0\}$ and $\mathcal{M}_{h^{(i)}(t_i)}(t) = \{\gamma_0\}$, for $i = 1, 2$.

(vi) *Set of rules.* The set of rules is $\mathcal{R}_{\Pi^{(1)}(t_1)} \cup \mathcal{R}_{\Pi^{(2)}(t_2)} \cup \mathcal{R}_t^*$, where $\mathcal{R}_{\Pi^{(i)}(t_i)}$ is the set of rules of $\Pi^{(i)}(t_i)$, for $i = 1, 2$, obtained through the replacement of objects yes and no by objects yes_i and no_i, respectively, in each rule. $\mathcal{R}_t^*$ is the following set of rules.

(1) *Rules for simultaneously transporting* $\text{cod}_i(u_i)$ *from* r_t'' *to* $h^{(i)}(t_i)$*, for* $i = 1, 2$ *and transforming it into* $\text{cod}_i(u_i')$*:*

(1.1) $a_{r_{\Pi^{(i)}(t_i)}} \longrightarrow [a']_{r_{\Pi^{(i)}(t_i)}}$, for $i = 1, 2$ and for each $a \in \Sigma^{(i*)}(t_i)$.

(1.2) $a'_h \longrightarrow [a']_h$, for each $a \in \Sigma(t)$ and each membrane h being either $h = h^{(i)}(t_i)$ or being h an ancestor of it.

(2) *Rules for simultaneously transporting* $\text{cod}_i(u_i')$ *from* $h^{(i)}(t_i)$ *to the input membrane* $\text{in}_{\Pi^{(i)}(t_i)}$*, by transforming it into* $\text{cod}_i(u_i)$*, for* $i = 1, 2$*:*

(2.1) $[\gamma_j \longrightarrow \gamma_{j+1}]_{h^{(i)}(t_i)}$, for $0 \le j \le m_i - 1$.

(2.2) $[\gamma_{m_i}]_{h^{(i)}(t_i)} \longrightarrow \#$.

(2.3) $[a' \longrightarrow a]_{h^{(i)}(t_i)}$, for each $a \in \Sigma(t)$.

(3) *Rules for synchronizing the simulation of* $\Pi^{(i)}(t_i) + \text{cod}_i(u_i)$*in order to start at the* $(m_i + 2)$*th transition step, for* $i = 1, 2$*:*

(3.1) $[a_j \longrightarrow a_{j+1}]_h$, for each $a \in \Gamma(t), 0 \le j \le m_i - 1, a_0 \in \mathcal{M}_h$.

(3.2) $[a_{m_i} \longrightarrow a]_h$, for each $a \in \Gamma(t), a_0 \in \mathcal{M}_h$.

(4) *Rules for synchronizing the instant* $n = 2 + \max\{d_i + q_i + p_i(|u_i|): 1 \le i \le 2\}$*in which the answers of* $\Pi^{(1)}(t_1) + \text{cod}_1(u_1)$ *and* $\Pi^{(2)}(t_2) + \text{cod}_2(u_2)$ *reach the membrane* r_t''*:*

(4.1) $[\beta_j \longrightarrow \beta_{j+1}]_{r_t''}$, for $0 \le j \le n - 1$.

(5) *Rules for the output of the system* $\Pi(t) + \text{cod}(u_1, u_2)$*:*

(5.1) $[\beta_n]_{r_t''} \longrightarrow \#$.

(5.2) $[\text{no}_i]_{r_t'} \longrightarrow \text{NO}$, for $i = 1, 2$.

(5.3) $[\text{NO}]_{r_t} \longrightarrow \text{no}_{r_t}$.

(5.4) $[\alpha_0 \longrightarrow \alpha_1]_{r_t'}$.

(5.5) $[\alpha_1]_{r_t'} \longrightarrow \text{YES}_{r_t'}$.

(5.6) $[\text{YES}]_{r_t} \longrightarrow \text{yes}_{r_t}$. □

4.1. Input Membrane. The input membrane is the membrane labelled by r_t''.

An overview of the computations of $\Pi(t) + \text{cod}(u_1, u_2)$ is as follows.

The proposed solution can be structured in the following stages.

(i) Transport stage

It consists of several phases: for $i = 1, 2$, in a simultaneous way, (a) the input multiset $\text{cod}_i(u_i)$ is transported from the input membrane r_t'' of $\Pi(t)$ to the skin membrane $r_{\Pi^{(i)}(t_i)}$ of $\Pi^{(i)}(t_i)$, being transformed into the multiset $\text{cod}_i(u_i')$ obtained from $\text{cod}_i(u_i)$ by replacing each symbol a in $\text{cod}_i(u_i)$ by a'; (b) then, $\text{cod}_i(u_i')$ is transported from the skin membrane $r_{\Pi^{(i)}(t_i)}$ to membrane $h_{t_i}^{(i)}$; finally, (c) $\text{cod}_i(u_i')$ moves from $h_{t_i}^{(i)}$ to the input membrane of $\Pi^{(i)}(t_i)$, being transformed into $\text{cod}_i(u_i)$.

(ii) Simulation stage

Computations of the system $\Pi^{(i)}(t_i)$ with input multiset $\text{cod}_i(u_i)$ are simulated, for $i = 1, 2$.

(iii) Output stage

The system $\Pi(t)$ with input multiset $\text{cod}(u_1, u_2)$ sends the right answer to the environment according to the results obtained in the previous stage.

4.1.1. Transport Stage. Once the input multiset $\mathrm{cod}(u_1, u_2) = \mathrm{cod}_1(u_1) + \mathrm{cod}_2(u_2)$ is supplied to the system $\Pi(t)$, by applying the rules from 1.1, multiset $cod_i(u_i)$, for $i = 1, 2$, enters into membrane $r_{\Pi^{(i)}(t_i)}$ transformed in $\mathrm{cod}_i(u_i')$. Next, by applying the rules from 1.2, multiset u_i' is transported to membrane $h_{t_i}^{(i)}$. Then, multiset $\mathrm{cod}_i(u_i')$ is transported from membrane $h_{t_i}^{(i)}$ to the input membrane of $\Pi^{(i)}(t_i)$ being transformed in $\mathrm{cod}_i(u_i)$, by dissolving the membrane $h_{t_i}^{(i)}$ (rule 2.2) and evolving objects a' to objects a by applying the rules from 2.3. This process can be observed in a graphical way in Figure 1.

It is worth pointing out that, in this process, we must prevent the rules of $\Pi^{(i)}(t_i)$ from being applied. In order to do that rules from 3.1 handle copies a_j of objects initially placed in membranes during m_i transition steps, and rules from 3.2 transform those objects in objects a in order to be used in the simulation of $\Pi^{(i)}(t_i) + \mathrm{cod}_i(u_i)$. Consequently, at the configuration $\mathscr{C}_{m_i+1}$, the simulation of $\Pi^{(i)}(t_i) + \mathrm{cod}_i(u_i)$, for $i = 1, 2$, will start.

4.1.2. Simulation Stage. The simulation of $\Pi^{(i)}(t_i) + \mathrm{cod}_i(u_i)$ starts at the $(m_i + 2)$th transition step. Then, after at most $p_i(|u_i|)$ transition steps, an answer will be sent to the membrane r_t'. Consequently, after at most $n = 2 + \max\{d_i + q_i + p_i(|u_i|): 1 \le i \le 2\}$ transition steps, systems $\Pi^{(1)}(t_1) + \mathrm{cod}_1(u_1)$ and $\Pi^{(2)}(t_2) + \mathrm{cod}_2(u_2)$ will send their responses to membrane r_t'', where counter β_j has been evolving from β_0 to β_n by applying rules from 4.1.

4.1.3. Output Stage. At the $(n + 1)$th transition step, the membrane r_t'' is dissolved by applying the rule 5.1. Then, membrane r_t' will contain objects $\alpha_0, \#$ along with the two answers (yes_i and/or no_i) provided by the simulations of $\Pi^{(i)}(t_i) + \mathrm{cod}_i(u_i)$, for $i = 1, 2$.

Case 1. Membrane r_t' from configuration $\mathscr{C}_{n+1}$ contains at least an object no_i. In this case, the answer must be negative. In fact, at the $(n + 2)$th step, by applying rule 5.2 one object no_i dissolves the membrane r_t' and evolves to object NO. Simultaneously, by applying the rule 5.4, object α_0 evolves to object α_1. Then, at configuration $\mathscr{C}_{n+2}$ the skin membrane r_t contains objects $\alpha_1, \#$, NO and one object yes_i or no_i. Finally, at the next step, by applying the rule 5.3, object no is sent out of system. A graphical interpretation of this case is present in Figure 2.

Case 2. Membrane r_t' from configuration $\mathscr{C}_{n+1}$ does not contain any object no_i. In this case, the answer must be affirmative. In fact, at the $(n + 2)$th step, by applying rule 5.4, object α_0 evolves to α_1 inside membrane r_t'. Then, membrane r_t' from configuration $\mathscr{C}_{n+2}$ will contain objects $\alpha_1, \#, \mathrm{yes}_1$, and yes_2. Next, by applying rule 5.5, object YES reaches membrane r_t. Finally, an object yes is sent out to the environment by applying rule 5.6. A graphical interpretation of this case is present in Figure 3.

Remark 1. Concerning to the proof of Theorem 1, we would like to draw attention to two aspects: (a) recognizer membrane systems from class $\mathscr{R}$ are required to allow the use of dissolution rules, object evolution rules, and communication rules (in both directions: send-in and send-out) and (b) an explicit solution of the product problem product is provided from two solutions of the problems involved in that operation.

Corollary 1. *Let $\mathscr{R}$ be a presumably efficient computing model of recognizer cell-like P systems allowing dissolution rules, object evolution rules, and communication rules (send-in and send-out). Then, $\boldsymbol{DP} \cup \boldsymbol{co}\text{-}\boldsymbol{DP} \subseteq \mathbf{PMC}_{\mathscr{R}}$.*

Proof. It suffices to note that if X is an **NP** complete problem such that $X \in \mathbf{PMC}_{\mathscr{R}}$, then the product problem $X \otimes \overline{X}$ is a **DP** complete problem such that $X \otimes \overline{X} \in \mathbf{PMC}_{\mathscr{R}}$. Since the complexity class $\mathbf{PMC}_{\mathscr{R}}$ is closed under complement and under polynomial-time reduction, $\mathbf{DP} \cup \mathbf{co}\text{-}\mathbf{DP} \subseteq \mathbf{PMC}_{\mathscr{R}}$ follows. □

5. Applying the Methodology to MAJORITY

In this section, we detail the application of the methodology through a simple example. For the sake of simplicity, a simple problem X will be selected so the focus can be on the application of the problem. The application of the methodology to an **NP**-complete problem will be applied in [16].

Let $M_f(A)$ be the set of all finite multisets over A, and let $f_u(x)$ be the number of appearances of symbol x in the multiset u. Let $\mathrm{size}(u)$ be the number of elements of u.

Let $X = (I_X, \theta_X)$ be the MAJORITY problem. Let $I_X = \{x \mid x \in M_f(\{0, 1\})\}$ and $\theta_X(u) = 1$ if and only if $f_u(1) > f_u(0)$, that is, if $f_u(1) \ge 1 + \lfloor n/2 \rfloor$.

A uniform family of polarizationless P systems with dissolution rules can be designed for this purpose (the MAJORITY problem falls under the class **P**; therefore, it could be enough to use the cod function to encode the input as yes in the affirmative case and no in the negative case, but for the application of the methodology, we will make that $\mathrm{cod}(u)$ will encode a 1 as an object t and a 0 as an object f).

Let $n > 0$ be the number of elements of the input instance. We define the recognizer polarizationless P system with dissolution rules:

$$\Pi(n) = \left(\Gamma, \Sigma, \mu, \mathscr{M}_1, \ldots, \mathscr{M}_{3+\lfloor n/2 \rfloor}, \mathscr{R}, i_{\mathrm{in}}, i_{\mathrm{out}}\right), \tag{3}$$

where

(1) $\Gamma = \Sigma \cup \{\mathrm{yes}, \mathrm{no}, y, \#\} \cup \{d_k \mid 0 \le k \le 3 + \lfloor n/2 \rfloor\}$.

(2) $\Sigma = \{f, t\}$.

(3) $\mu = [\,[\ldots_{3+\lfloor n/2 \rfloor} \ldots]_{32}]_1$.

(4) $\mathscr{M}_{3+\lfloor n/2 \rfloor} = \{y\}, \mathscr{M}_2 = \{d_0\}, \mathscr{M}_i = \varnothing, i \notin \{2, 3 + \lfloor n/2 \rfloor\}$.

(5) The set $\mathscr{R}$ of rules is formed by the following rules:

(a) These rules will carry the object y to the membrane labelled by 1 if and only if there are at least $1 + \lfloor n/2 \rfloor$ objects 1:

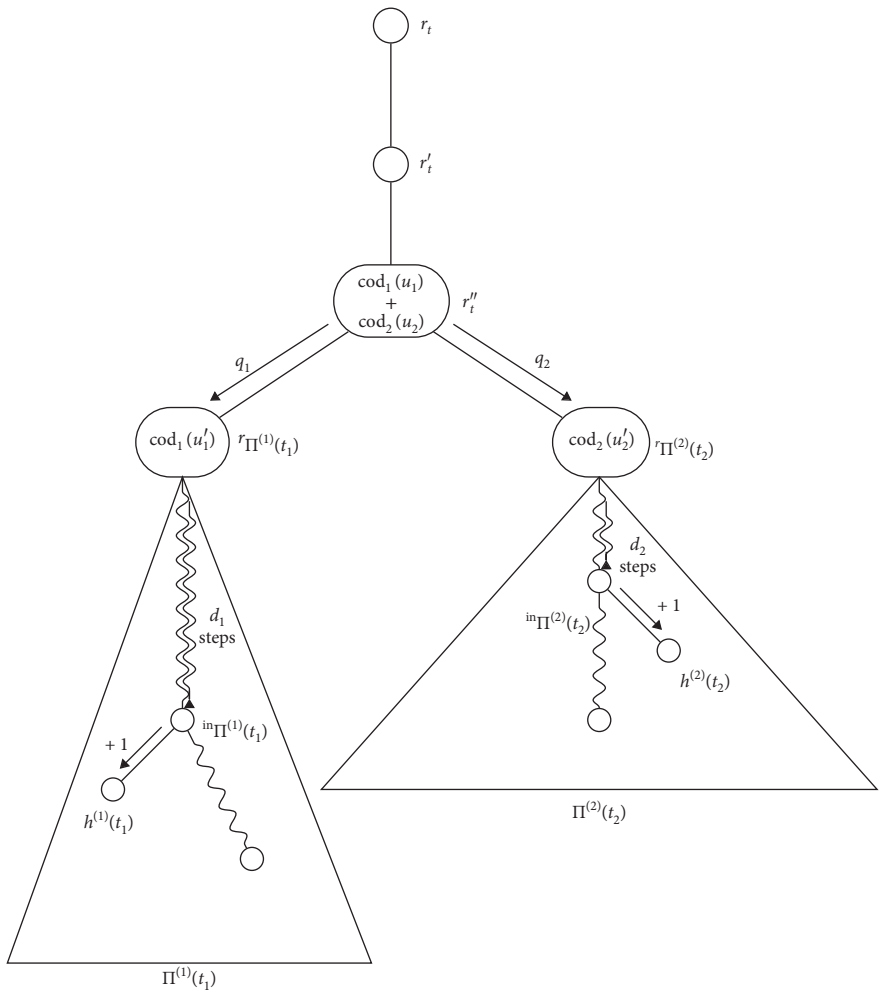

Figure 1: Transport of objects from r''_t to the corresponding $\mathrm{in}_{\Pi^{(i)}(t_i)}$. For this purpose, rules from 1 and 2 are applied. Besides these rules, rules from 3 are applied to make the objects from $\mathscr{M}_h^{(i)}$ be available in their corresponding membranes only when objects from $\Sigma(t)$ are in their corresponding input membranes.

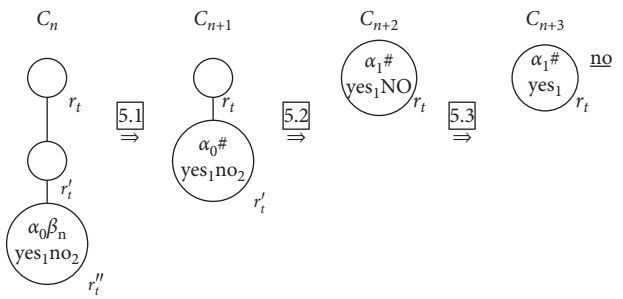

Figure 2: Negative answer. With rules from 4 and 5, an object no is sent to the environment in the last step of the computation.

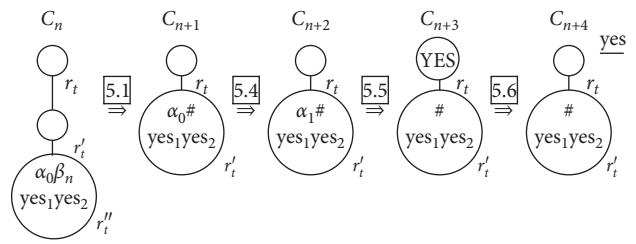

Figure 3: Affirmative answer. With rules from 4 and 5, an object yes is sent to the environment in the last step of the computation.

$$\{\,[t]_i \longrightarrow \#, \quad \text{for } 3 \le i \le 3 + \left\lfloor \frac{n}{2} \right\rfloor. \tag{4}$$

(b) These rules will make the object n_0 evolve to $n_{\lfloor n/2 \rfloor + 2}$:

$$\{\,[d_k \longrightarrow d_{k+1}]_2, \quad \text{for } 0 \le k \le 2 + \left\lfloor \frac{n}{2} \right\rfloor. \tag{5}$$

(c) Rules for returning yes if object y has arrived the membrane labelled by 1 in the configuration $\lfloor n/2 \rfloor + 1$:

$$\begin{aligned} y_2 &\longrightarrow [y]_2, \\ [y]_2 &\longrightarrow \mathrm{yes}, \\ [\mathrm{yes}]_1 &\longrightarrow \mathrm{yes}_1. \end{aligned} \tag{6}$$

(d) Rules for returning an object no in case the object y is not in membrane 1 in configuration $\lfloor n/2 \rfloor$:

$$\begin{aligned} \left[d_{\lfloor n/2 \rfloor + 2}\right]_2 &\longrightarrow \mathrm{no}_2, \\ [\mathrm{no}]_1 &\longrightarrow \mathrm{no}_1. \end{aligned} \tag{7}$$

(6) $i_{\mathrm{in}} = 2 + \lceil n/2 \rceil$ and $i_{\mathrm{out}} = \mathrm{env}$.

The system works are as follows: the instance is introduced in the input membrane as a multiset of objects t and f ($\mathrm{cod}(u)$ will encode a 1 as an object t and a 0 as an object f). Objects t will start dissolving membranes starting from the membrane labelled by $\lceil n/2 \rceil$ to the membrane labelled by 3. If there are enough objects t (that is, if there is a majority of 1 in the instance), then object y will be in the membrane labelled by 1 in configuration $\mathscr{C}_{\lfloor (n/2)+1 \rfloor}$. Therefore, object y will go into membrane 2 and will dissolve it, avoiding object $d_{\lceil n/2 \rceil + 3}$ evolving into an object no. While dissolving, object y will be transformed into an object yes, that will go to the environment in the last step of the computation. If object y is not present in the membrane 1, then object $d_{3+\lceil n/2 \rceil}$ will be transported to membrane 1 as an object no in the membrane 1 and will be sent to the environment in the last step of the computation. The system will take $5 + \lceil n/2 \rceil$ steps in the affirmative case, and $6 + \lceil n/2 \rceil$ steps in the negative case. With this solution to the MAJORITY problem, we can proceed to apply the methodology.

Let $\overline{\mathrm{MAJORITY}}$ be the complementary problem of MAJORITY, that is, $I_{\overline{\mathrm{MAJORITY}}} = I_{\mathrm{MAJORITY}}$ and $\theta_{\overline{\mathrm{MAJORITY}}}(u) = 1 - \theta_{\mathrm{MAJORITY}}(u)$. A solution to this problem could be given simply changing the role of objects yes and no.

Let

$$\Pi^{(1)}(n) = \left(\Gamma^{(1)}, \Sigma^{(1)}, \mu^{(1)}, \mathscr{M}_1^{(1)}, \ldots, \mathscr{M}_{3+\lfloor n/2 \rfloor}^{(1)}, \mathscr{R}^{(1)}, i_{\mathrm{in}}^{(1)}, i_{\mathrm{out}}^{(1)}\right), \tag{8}$$

be recognizer polarizationless P systems with dissolution rules that solves the instances of MAJORITY of size n as described above. We can define a recognizer polarizationless P system with dissolution rules:

$$\begin{aligned} \Pi^{(2)}(n) = \Big(&\Gamma^{(2)}, \Sigma^{(2)}, \mu^{(2)}, \mathscr{M}_{4+\lfloor n_1/2 \rfloor}^{(2)}, \ldots, \\ &\mathscr{M}_{6+\lfloor n_1/2 \rfloor + \lfloor n_2/2 \rfloor}^{(2)}, \mathscr{R}^{(2)}, i_{\mathrm{in}}^{(2)}, i_{\mathrm{out}}^{(2)}\Big), \end{aligned} \tag{9}$$

which solves the instances of $\overline{\mathrm{MAJORITY}}$ of size n as follows. For doing $\Gamma^{(i)}(n_i)$ disjoint, $\Gamma^{(2)}(t_2)$ will consist on the symbols from $\Gamma^{(1)}(t_1)$ in capital letters. All the rules are changed according to this. To make $H^{(1)}$ and $H^{(2)}$ disjoint, given $n_1, n_2 > 0$, we change the label i of $H^{(2)}$ to $3 + \lfloor n_1/2 \rfloor + i$. In $\Pi^{(i)}$, we change objects yes and no by yes_i and no_i, respectively.

From here, we are going to define P system that will solve the problem $\mathrm{MAJORITY} \otimes \overline{\mathrm{MAJORITY}}$, which is defined as follows. Let $u_1, u_2 \in M_f(\{0,1\})$.

$\theta_{\mathrm{MAJORITY} \otimes \overline{\mathrm{MAJORITY}}}(u_1, u_2) = 1$ if and only if $\theta_{\mathrm{MAJORITY}}(u_1) = 1$ and $\theta_{\overline{\mathrm{MAJORITY}}}(u_2) = 1$. Let $\mathrm{cod}(u_1, u_2) = \mathrm{cod}_1(u_1) + \mathrm{cod}_2(u_2)$. Let $t_i = \mathrm{size}(u_i)$, and $t = \langle t_1, t_2 \rangle$. Let $q_i = t_i$ and $d_i = 3 + \lfloor t_i/2 \rfloor$. Let $p_i(|u_i|) = 6 + \lfloor t_i/2 \rfloor$. Let $m_i = q_i + d_i + 2 = t_i + 4 + \lfloor t_i/2 \rfloor$ and $n = \max\{d_i + q_i + p_i(|u_i|)\} = \max\{t_1 + 8 + 2 \cdot \lfloor t_1/2 \rfloor, t_2 + 8 + 2 \cdot \lfloor t_2/2 \rfloor\}$. We define a recognizer polarizationless P system with dissolution rules:

$$\begin{aligned} \Pi(t) = \Big(&\Gamma, \Sigma, \mu, \mathscr{M}_1, \ldots, \mathscr{M}_{6+\lfloor t_1/2 \rfloor + \lfloor t_2/2 \rfloor}, \mathscr{M}_{r_t}, \mathscr{M}_{r_f(t)}, \\ &\mathscr{M}_{r''(t)}, \mathscr{M}_{h^{(1)}(t_1)}, \mathscr{M}_{h^{(2)}(t_2)}, \mathscr{R}, i_{\mathrm{in}}, i_{\mathrm{out}}\Big), \end{aligned} \tag{10}$$

where

(i) $\Gamma = \Gamma^{(1*)}(t) \cup \Gamma^{(2*)}(t) \cup \Gamma^{(3)}(t) \cup \Gamma^{(4)}(t) \cup \Gamma^{(5)}(t) \cup \Gamma^{(6)}(t)$, where

$$\begin{aligned} \Gamma^{(1*)}(t) &= \Sigma^{(1)}(t_1) \cup \{\mathrm{yes}_1, \mathrm{no}_1, y, \#\} \cup \left\{d_k \middle| 0 \le k \le 3 + \left\lfloor \frac{t_1}{2} \right\rfloor\right\}, \\ \Gamma^{(2*)}(t) &= \Sigma^{(2)}(t_2) \cup \{\mathrm{yes}_2, \mathrm{no}_2, Y, \#\} \cup \left\{D_k \middle| 0 \le k \le 3 + \left\lfloor \frac{t_2}{2} \right\rfloor\right\}, \\ \Gamma^{(3)}(t) &= \{\#, \mathrm{YES}, \mathrm{NO}\} \cup \{t', f', T', F'\}, \\ \Gamma^{(4)}(t) &= \left\{yes_{1_j}, no_{1_j}, y_j \middle| 0 \le j \le m_1\right\} \cup \left\{d_{k_j} \middle| 0 \le k \le 3 + \left\lfloor \frac{t_1}{2} \right\rfloor, \ 0 \le j \le m_1\right\} \cup \\ &\quad \left\{\mathrm{yes}_{2_j}, \mathrm{no}_{2_j}, Y_j \middle| 0 \le j \le m_2\right\} \cup \left\{D_{k_j} \middle| 0 \le k \le 3 + \left\lfloor \frac{t_1}{2} \right\rfloor, \ 0 \le j \le m_2\right\} \cup, \\ \Gamma^{(5)}(t) &= \{\alpha_0, \alpha_1\} \cup \left\{\beta_j \middle| 0 \le j \le n\right\}, \\ \Gamma^{(6)}(t) &= \left\{\gamma_j \middle| 0 \le j \le \max\{m_1, m_2\}\right\}. \end{aligned} \tag{11}$$

(ii) $\Sigma(t) = \Sigma^{(1)}(t_1) \cup \Sigma^{(2)}(t_2)$, being $\Sigma^{(1)}(t_1) = \{t, f\}$ and $\Sigma^{(2)}(t_2) = \{T, F\}$.

(iii) $H(t) = H^{(1)} \cup H^{(2)} \cup \{r_t, r_t', r_t'', h^{(1)}(t_1), h^{(2)}(t_2)\}$, where

$r_t, r_t', r_t'', h^{(1)}(t_1), h^{(2)}(t_2)$ are different from each other, and

$$r_t, r_t', r_t'', h^{(1)}(t_1), h^{(2)}(t_2) \notin H^{(1)} \cup H^{(2)}. \quad (12)$$

(iv) $\mu = \mu = [[[[\ldots[_{h^{(1)}(t_1)}]_{3+\lfloor t_1/2 \rfloor} \ldots]_1 [\ldots [_{h^{(2)}(t_2)}]_{6+\lceil t_1/2 \rceil + \lfloor t_2/2 \rfloor} \ldots]_{4+\lfloor t_1/2 \rfloor_t}]_{r_t''}]_{r_t}]_{r_t}$.

(v) $\mathcal{M}_{3+\lfloor t_1/2 \rfloor} = \{y_0\}, \mathcal{M}_{6+\lfloor t_1/2 \rfloor + \lfloor t_2/2 \rfloor} = \{Y_0\}, \mathcal{M}_2 = \{d_{0_0}\}, \mathcal{M}_{2+\lfloor t_1/2 \rfloor} = \{D_{0_0}\}, \mathcal{M}_{r_t''}(t) = \{\alpha_0, \beta_0\}$ and $\mathcal{M}_{h^{(i)}(t_i)}(t) = \{\gamma_0\}$, for $i = 1, 2$ and $\mathcal{M}_j = \varnothing$ for the rest of membranes.

(vi) *Set of rules.* The set of rules is $\mathcal{R}_{\Pi^{(1)}(t_1)} \cup \mathcal{R}_{\Pi^{(2)}(t_2)} \cup \mathcal{R}_t^*$, where $\mathcal{R}_{\Pi^{(i)}(t_i)}$ is the set of rules of $\Pi^{(i)}(t_i)$, for $i = 1, 2$, obtained through the replacement of objects yes and no by objects yes_i and no_i, respectively, in each rule. $\mathcal{R}_t^*$ is the following set of rules:

(1) Rules for simultaneously transporting $\text{cod}_i(u_i)$ from r_t'' to $h^{(i)}(t_i)$, for $i = 1, 2$ and transforming it into $\text{cod}_i(u_i')$:

$$\left.\begin{array}{l} a_1 \longrightarrow [a']_1 \\ A_{4+\lfloor t_1/2 \rfloor} \longrightarrow [A']_{4+\lfloor t_1/2 \rfloor} \end{array}\right\}, \quad \text{for } a \in \{t, f\},\ A \in \{T, F\},$$

$$\left.\begin{array}{l} a'_i \longrightarrow [a']_1 \\ a'_{h^{(1)}(t_1)} \longrightarrow [a']_{h^{(1)}(t_1)} \\ A'_j \longrightarrow [A']_j \\ A'_{h^{(2)}(t_2)} \longrightarrow [A']_{h^{(2)}(t_2)} \end{array}\right\}, \quad \text{for } \left\{\begin{array}{l} a \in \{t, f\},\ 3 \le i \le 3 + \left\lfloor \frac{n_1}{2} \right\rfloor, \\ A \in \{T, F\},\ 6 + \left\lfloor \frac{n_1}{2} \right\rfloor \le j \le + \left\lfloor \frac{n_2}{2} \right\rfloor, \end{array}\right. \quad (13)$$

(2) *Rules for simultaneously transporting* $\text{cod}_i(u_i')$ *from* $h^{(i)}(t_i)$ *to the input membrane* $\text{in}_{\Pi^{(i)}(t_i)}$, *by transforming it into* $\text{cod}_i(u_i)$, *for* $i = 1, 2$:

$$\left\{ \left[\gamma_j \to \gamma_{j+1}\right]_{h^{(i)}(t_i)}, \quad \text{for } i \in \{1, 2\}, 0 \le j \le m_i - 1, \right.$$

$$\left\{ \left[\gamma_{m_i}\right]_{h^{(i)}(t_i)} \longrightarrow \#, \quad \text{for } i \in \{1, 2\}, \right.$$

$$\left.\begin{array}{l} [a' \longrightarrow a]_{h^{(1)}(t_1)} \\ [A' \longrightarrow A]_{h^{(2)}(t_2)} \end{array}\right\}, \quad \text{for } a \in \{t, f\}, A \in \{T, F\}. \quad (14)$$

(3) *Rules for synchronizing the simulation of* $\Pi^{(i)}(t_i) + \text{cod}_i(u_i)$ *in order to start at the* $(m_i + 2)$*th transition step, for* $i = 1, 2$,

$$\left.\begin{array}{l} \left[d_0 \to d_{0_{j+1}}\right]_2, \\ \left[y_i \to y_{j+1}\right]_{3+\lfloor t_1+2 \rfloor}, \end{array}\right\} \quad \text{for } 0 \le j \le m_1,$$

$$\left.\begin{array}{l} \left[D_{0_j} \to D_{0_{j+1}}\right]_{2\lfloor t_1+2 \rfloor}, \\ \left[Y_i \to Y_{j+1}\right]_{6+\lfloor t_1+2 \rfloor + \lfloor t_2+2 \rfloor}, \end{array}\right\} \quad \text{for } 0 \le h \le m_2, \quad (15)$$

$$\left[d_{0_{m1}} \to d_0\right]_2,$$

$$\left[D_{0_{m2}} \to D_0\right]_{2+\lfloor t_1+2 \rfloor},$$

$$[y_{m1} \to y]_{3+\lfloor t_1+2 \rfloor},$$

$$[Y_{m2} \to Y]_{6+\lfloor t_1+2 \rfloor + \lfloor t_2/2 \rfloor}.$$

(4) *Rules for synchronizing the instan* $n = 2 + \max\{d_i + q_i + p_i(|u_i|): 1 \le i \le 2\}$. *in which the answers of* $\Pi^{(1)}(t_1) + \text{cod}_1(u_1)$ *and* $\Pi^{(2)}(t_2) + \text{cod}_2(u_2)$ *reach the membraner*$_t''$: $\left\{ [\beta_j \longrightarrow \beta_{j+1}]_{r_t''}, \text{for } 0 \le j \le n - 1. \right.$

(5) *Rules for the output of the system* $\Pi(t) + \text{cod}(u_1, u_2)$:

$$\left.\begin{array}{l} [\beta_n]_{r_t''} \longrightarrow \# \\ [\text{no}_i]_{r_t'} \longrightarrow \text{NO} \\ [\alpha_0 \longrightarrow \alpha_1]_{r_t'} \\ [\alpha_1]_{r_t'} \longrightarrow \text{YES}_{r_t'} \\ [\text{YES}]_{r_t} \longrightarrow \text{yes}_{r_t} \end{array}\right\}, \quad \text{for } i \in \{1, 2\}, \quad (16)$$

$$i_{\text{in}} = r_t'',$$

$$i_{\text{out}} = \text{env}.$$

This system will return an object yes or an object no, but not both, only in the last step of the computation. The computation will take $n + 4$ computational steps in the affirmative case and $n + 3$ steps in the negative case.

6. Conclusions

Given a computing model $\mathcal{R}$ of recognizer membrane systems with the ability to provide polynomial-time solutions to **NP** complete problems (*presumably efficient* computing model), the time complexity class $\mathbf{PMC}_{\mathcal{R}}$ contains

the complexity classes **NP** and co-**NP**, since $\mathbf{PMC}_{\mathscr{R}}$ is closed under complement and under polynomial-time reductions [2]. In this paper, a lower bound thinner than $\mathbf{NP} \cup \text{co-}\mathbf{NP}$ is provided for the class $\mathbf{PMC}_{\mathscr{R}}$ associated with a presumably efficient computing model $\mathscr{R}$ of recognizer P systems allowing object evolution rules, dissolution rules, and communication rules. Specifically, for such kind of computing models, the following result has been obtained: $\mathbf{DP} \cup \text{co-}\mathbf{DP} \subseteq \mathbf{PMC}_{\mathscr{R}}$, where **DP** is the set of languages which can be expressed as the difference of two languages in **NP** or, equivalently, as the intersection of a language in **NP** and a language in **co**-**NP**.

In this paper, we have applied the methodology to the MAJORITY problem. We have chosen such a simple problem for the sake of simplicity. An efficient solution to the SAT-UNSAT problem, a **DP**-complete problem, is given in [16].

In this research line, it would be interesting to analyze the possibility of extending the main theorem to other presumably efficient computing models of recognizer membrane systems; for instance, cooperative cell-like membrane systems without dissolution rules, tissue-like P systems with symport/antiport rules or spiking neural P systems.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the research project TIN2017-89842-P, co-financed by Ministerio de Economía, Industria y Competitividad (MINECO) of Spain, through the Agencia Estatal de Investigación (AEI), and by Fondo Europeo de Desarrollo Regional (FEDER) of the European Union.

References

[1] M. J. Pérez-Jiménez, A. Romero-Jiménez, and F. Sancho-Caparrini, "Decision P systems and the $\mathbf{P} \neq \mathbf{NP}$ conjecture," *Lecture Notes in Computer Science*, vol. 2597, pp. 388–399, Springer, Berlin, Germany, 2003.

[2] M. J. Pérez-Jiménez, Á. Romero-Jiménez, and F. Sancho-Caparrini, "Complexity classes in models of cellular computing with membranes," *Natural Computing*, vol. 2, no. 3, pp. 265–285, 2003.

[3] L. F. Macías-Ramos, M. J. Pérez-Jiménez, A. Riscos-Núñez, and L. Valencia-Cabrera, "Membrane fission versus cell division: when membrane proliferation is not enough," *Theoretical Computer Science*, vol. 608, pp. 57–65, 2015.

[4] G. H. Păun, "Attacking NP-complete problems," *Unconventional Models of Computation*, pp. 94–115, Springer-Verlag, Berlin, Germany, 2000.

[5] M. Á. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, A. Riscos-Núñez, and F. J. Romero-Campero, "On the power of dissolution in P systems with active membranes," *Lecture Notes in Computer Science*, vol. 3850, pp. 224–240, Springer, Berlin, Germany, 2006.

[6] L. Pan, D. Orellana-Martín, B. Song, and M. J. Pérez-Jiménez, "Cell-like P systems with polarizations and minimal rules," *Theoretical Computer Science*, vol. 816, pp. 1–18, 2020.

[7] D. Orellana-Martín, L. Valencia-Cabrera, A. Riscos-Núñez, and M. J. Pérez-Jiménez, "Membrane creation in polarizationless P systems with active membranes," *Fundamenta Informaticae*, vol. 171, no. 1–4, pp. 297–311, 2019.

[8] B. Song, K. Li, D. Orellana-Martín, L. Valencia-Cabrera, and M. J. Pérez-Jiménez, "Cell-like P systems with evolutional symport/antiport rules and membrane creation," *Information and Computation*, Article ID 104542, 2020.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, NY, USA, 1979.

[10] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley Publishing Company, Boston, MA, USA, 1994.

[11] L. Stockmeyer, "The polynomial-time hierarchy," *Theoretical Computer Science*, vol. 3, no. 1, pp. 1–22, 1977.

[12] H. Rogers Jr., *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, NY, USA, 1967.

[13] G. Wechsung, "On the Boolean closure of NP," *Lecture Notes in Computer Science*, vol. 199, pp. 485–493, Springer, Berlin, Germany, 1985.

[14] C. H. Papadimitriou and M. Yannakis, "The complexity of facets (and some facets of complexity)," in *Proceedings of the 24th ACM Symposium on the Theory of Computing*, pp. 229–234, Victoria, Canada, 1982.

[15] J.-Y. Cai, T. Gundermann, J. Hartmanis et al., "The Boolean hierarchy I: structural properties," *SIAM Journal on Computing*, vol. 17, no. 6, pp. 1232–1252, 1988.

[16] D. Orellana-Martín, L. Valencia-Cabrera, and M. J. Pérez-Jiménez, "From SAT to SAT-UNSAT using P systems with dissolution rules," in *Proceedings of the 2020 Membrane Computing: 21th International Conference on Membrane Computing, ICMC 2020*, Vienna, Austria, September 2020.