Hindawi
Discrete Dynamics in Nature and Society
Volume 2023, Article ID 4993855, 9 pages
https://doi.org/10.1155/2023/4993855

Research Article

Solving the SAT Problem by Cell-Like P Systems with Channel States and Symport Rules

Xiaoming Wan,[1] **Chuchuan Liu**,[2] **and Yueguo Luo**[3]

[1]*College of Electronics and IoT Engineering, Chongqing Industry Polytechnic College, Chongqing 401120, China*
[2]*School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China*
[3]*College of Big Data and Intelligent Engineering, Yangtze Normal University, Chongqing 408100, China*

Correspondence should be addressed to Yueguo Luo; ygluo@yznu.edu.cn

Received 30 March 2023; Revised 22 May 2023; Accepted 29 May 2023; Published 10 June 2023

Academic Editor: Ya Jia

Cell-like P systems with channel states, which are a variant of tissue P systems in membrane computing, can be viewed as highly parallel computing devices based on the nested structure of cells, where communication rules are classified as symport rules and antiport rules. In this work, we remove the antiport rules and construct a novel variant, namely, cell-like P systems with channel states and symport rules, where one rule is only allowed to be nondeterministically applied once per channel. To explore the computational efficiency of the variant, we solve the $\mathcal{SAT}$ problem and obtain a uniform solution in polynomial time with the maximal length of rules 1. The results of our work are reflected in the following two aspects: first, communication rules are restricted to only one type, namely, symport rules; second, the maximal length of rules is decreased from 2 to 1. Our work indicates that the constructed variant with fewer rule types can still solve the $\mathcal{SAT}$ problem and obtain better results in terms of computational complexity. Hence, in terms of computational efficiency, our work is a notable improvement.

1. Introduction

Membrane computing, a new type of bioinspired computing model, was first proposed by Păun [1]. Like quantum computing and DNA computing, it can be used to construct various models and implement computer algorithms. In [2], Professor Adleman successfully solved the Hamiltonian path problem with 7 vertices. Membrane computing is inspired by biological cells, tissues, and nervous systems and can be implemented as distributed and parallel computing devices. At present, many new variants have been proposed [3–9], and many variants have been proven Turing universal. In the theoretical research of membrane computing, various computationally hard problems were solved [10–14]. Recently, inspired by membrane computing, Roy et al. proposed a new type of neural computing system [15], which will promote the development of membrane computing. In the application field, membrane computing has achieved excellent results in optimization algorithms [16, 17], biology-based approaches [18, 19], mobile robots [20], fault diagnosis [21, 22], and other related applications [23, 24]. Recently, some excellent results have been achieved in the field of machine learning [25–27], and some researchers combine these two research areas and obtain excellent results [28]. Currently, many scholars are focusing on developments in this field. For additional details, some review books including papers (e.g., [29]) and the website https://ppage.psystems.eu/ can be viewed to obtain the latest information.

Currently, there are primarily three types of P systems in membrane computing: cell-like P systems [1], tissue-like P systems [30], and neural-like P systems [31]. This article focuses on cell-like P systems (for additional details, see [1]). In [32], symport/antiport rules were introduced to cell-like P systems, and substances can move between two membranes; moreover, multisets of objects in adjacent regions can exchange positions. Freund et al. proposed the concept of the channel state by combining tissue P systems [33]. When

such a P system is running, the states of channels can be changed; in addition, channel states can activate rule execution. In [34], the channel state concept was introduced to cell-like P systems with symport/antiport rules to construct a new variant (abbreviated as *CCSSA* P systems); furthermore, the computational power of this new variant was explored, but the study of computational efficiency was not involved. Recently, Jiang et al. solved the $\mathcal{SAT}$ problem with *CCSSA* P systems to study its computational efficiency [35]; however, in the study, although a uniform $\mathcal{SAT}$ solution was obtained, the maximum rule length in the *CCSSA* P systems was 2. In terms of computational complexity, in membrane computing, a better method can solve the same *NP*-hard problem with a maximum rule length shorter than 2. If a membrane system can be constructed to solve the same problem, even with a shorter maximum rule length, then the system provides excellent computational property. For instance, in [36], $\mathcal{SAT}$ was solved with a maximum length rule of 8; however, in [37], the maximum length of rules was reduced to 3 with the same model. Hence, the latter paper improved upon the previous research results. Obviously, in membrane computing, the length of rules is an important factor in terms of computational complexity.

In [1], the notion of maximal parallelism was proposed, and it is an attractive strategy: at a given moment, multiple rules on each channel can be selected to use, where one rule may be executed multiple times. Nevertheless, in flat maximal parallelism [38], such a rule can only be activated once. In our work, we adopt a new method combined sequential manner with flat maximal parallelism, that is, on an arbitrary channel, even if more than one rule is employed at a given moment, regardless of the direction of movement of the rules on the channel and the channel states considered in the rules, only one rule can be nondeterministically applied once.

With respect to *CCSSA* P systems, communication rules are classified into two categories, namely, symport rules and antiport rules. In this work, we consider constructing a variant with fewer rule types; if the new variant can still solve the same problem, it is shown that the constructed variant is powerful enough. Hence, we remove the antiport rules of *CCSSA* P systems, thereby constructing a novel variant, that is, cell-like P systems with channel states and symport rules (abbreviated as *CCSS* P systems).

The following are the contributions of our work:

(i) We remove the antiport rules of *CCSSA* P systems, thereby constructing a novel variant, that is, cell-like P systems with channel states and symport rules, where, with respect to communication rules, ony symport rules are used. In such a variant, at a given moment, one rule is only allowed to be nondeterministically applied once per channel.

(ii) Membrane division is introduced to the constructed variant, and its computational complexity is studied. Specifically, we solve $\mathcal{SAT}$ and obtain a uniform solution. With respect to computational complexity, in the process of obtaining a theoretical proof, the maximal length of rules is 1, which reflects excellent performance. Our approach improves upon the current research method.

The structure of this paper is as follows: Section 2 mainly gives the definition of *CCSS* P systems. In Section 3, based on the $\mathcal{SAT}$ problem, the computational efficiency of the system with respect to solving an *NP*-complete problem is studied; next, we explore a case study to verify the membrane system introduced in Section 4. Finally, some conclusions are presented, and future work is considered.

2. Cell-Like P Systems with Channel States and Symport Rules

In this section, we remove the antiport rules of *CCSSA* P systems, thereby constructing a novel variant, namely, *CCSS* P systems. For the formal language and the automaton theory, one can see [39, 40].

2.1. The Model of CCSS P Systems

Definition 1. A *CCSS* P system (degree $m \geq 1$, $1 \leq i \leq m$) is a tuple:

$$\Pi = (O, K, E, \mu, w_i, s_i, \mathcal{R}_i, i_{\text{out}}), \tag{1}$$

where

(i) O is an alphabet of objects

(ii) K represents the set of channel states (not necessarily exist in the set of O)

(iii) E represents an infinite number of objects in the environment

(iv) μ means the membrane structure denoted by a tree, and all nodes $1, \ldots, m$ in the tree correspond to the labels of membranes. Relative to each membrane i, $p(i)$ denotes the outside region of membrane i;

(v) $w_i \subseteq O (1 \leq i \leq m)$ denote the multisets of objects initially located in membrane i

(vi) $s_i (1 \leq i \leq m)$ represent channel states initially located on the membrane with label i, which is the channel between membrane i and $p(i)$

(vii) i_{out} is the output region ($i_{\text{out}} \in \{0, 1, \ldots, m\}$)

(viii) $\mathcal{R}_i, i \in \{1, \ldots, m\}$ represent a series of rules, including symport rules and division rules. In what follows, O^+ denotes the set of strings composed by the symbol in O but without λ; and given a string x, the length of x, which is the quantity of symbols in this string, is denoted by $|x|$.

2.1.1. Symport Rules

$$\left(S, (x, \text{in}), S'\right) \text{ or } \left(S, (x, \text{out}), S'\right), \tag{2}$$

where $S, S' \in K$, $x \in O^+$, $|x| > 0$. The length of a symport rule is $|x|$. Formally, the maximum rule length of a *CCSS* P system can be denoted by the maximum rule length of symport rules in the system.

At a given moment, when the channel state of a membrane is S and multiset x exists inside (resp., outside) membrane i, $(S, (x, \text{out}), S')$ (resp., $(S, (x, \text{in}), S')$) can be applied. By changing the channel state to S', multiset x is transferred to region $p(i)$ (resp., membrane i), where $p(i)$ represents the outer membrane of membrane i; if membrane i is the skin membrane (it does not have the upper neighbor membrane), $p(i)$ represents the environment.

2.1.2. Division Rules

$$[a]_i \longrightarrow [b]_i [c]_i, \tag{3}$$

where $i \in \{1, 2, \ldots, m\}, a, b, c \in O, i \neq i_{\text{out}}$.

At a given moment, relative to object a in membrane i (except for the skin membrane), a division rule is executed, and two membranes with the same label will appear. Simultaneously, object b including c appear in new membranes to replace the previous object a; other objects in the initial membrane can be copied to the two newly generated membranes. It is important to emphasize that objects a, b, and c can represent the same or different symbols. Moreover, the priority of division rules is higher than that of other rules, that is, if such a rule can be applied, symport rules cannot be applied at that moment.

2.2. System Operation. Even if more than one rule can be employed on a channel at a given moment, only a rule can be nondeterministically selected. In addition, if a rule can be executed more than one time at a given moment, this rule can only be activated once. When the application of a rule on a channel is completed, another rule can be applied. However, relative to all channels, at any step, rules can be executed in parallel on different channels; moreover, rules that can be executed must be executed on the corresponding channel. For example, multiset of objects u^2v^2 exists in membrane 1, and channel state S appears on the membrane; in addition, there are two rules associated with the membrane:

$$\begin{aligned} R_1&: \left(S, (u, \text{out}), S'\right), \\ R_2&: \left(S, (v, \text{out}), S'\right). \end{aligned} \tag{4}$$

First, we consider the strategy of maximal parallelism to run a system. Initially, rules R_1 and R_2 are activated simultaneously, and each rule can be executed multiple times. As a result, all copies of objects u and v will be transferred to the environment. The computing process is shown in Figure 1.

Next, we consider the method combined sequential manner with flat maximal parallelism. When the system starts running, one rule (R_1 or R_2) is only allowed to be nondeterministically applied on the channel; moreover, relative to the rule that is selected to be applied on the channel, it is only allowed to be applied once. Hence, the following two cases will occur: (i) Rule R_1 is only allowed to be applied. Eventually, one copy of object u is transferred to the environment; (ii) Rule R_2 is only allowed to be applied. Eventually, one copy of object v is transferred to the environment. The computing process is shown in Figure 2.

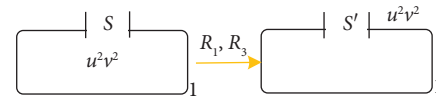

Figure 1: Applying maximal parallelism.

There is at most one channel on each membrane (between two adjacent regions). On each channel, we use the strategy of sequential manner combined with flat maximal parallelism to apply rules. The configuration of *CCSS* P systems is influenced by the following factors: the membrane structure, the multiset of objects in each region, and the channel states between adjacent regions. Before such a system starts running, we use the initial configuration to characterize it. As the computation continues, a series of configurations can be generated with the execution of rules at each step, and we call it a transition between two arbitrary configurations. A series of transitions is called a computation. Notably, the application of rules is based on sequential manner combined with flat maximal parallelism and follows the principle of nondeterminism [1]. Finally, when system operation stops, the halting configuration can be obtained; at that moment, the system halts, and no rules are applied, and the computing result is sent to the output region.

2.3. Recognizer CCSS P Systems. In membrane computing, the recognizer P systems are used to solve decision problems.

Definition 2. A recognizer *CCSS* P system (degree $m \geq 1$, $1 \leq i \leq m$) is a tuple:

$$\Pi = \left(O, K, \Sigma, E, \mu, w_i, s_i, \mathscr{R}_i, i_{in}, i_{\text{out}}\right), \tag{5}$$

where

(i) Σ represents an input alphabet $(\Sigma \subseteq O)$;

(ii) $\mathscr{YES}, \mathscr{NO} \in O$;

(iii) i_{in} (resp., i_{out}) represents the input (resp., output) region.

In such a system, the other parameters are defined as in Definition 1. A recognizer *CCSS* P system can start from the initial configuration with an input multiset; eventually, $\mathscr{YES}$ occurs in the region i_{out}.

Definition 3. $X = (I_X, \theta_X)$ denotes a decision problem, where I_X is the instance, and θ_X represents a predicate of the instances. The problem can be solved in polynomial time, if the following holds:

(i) Π is polynomially uniform by Turing machines

(ii) Relative to I_X, there is a pair (cod, s) of polynomial-time computable functions such that

(a) suppose u corresponds to an instance, $u \in I_X$, and $s(u)$ is a natural number; additionally,

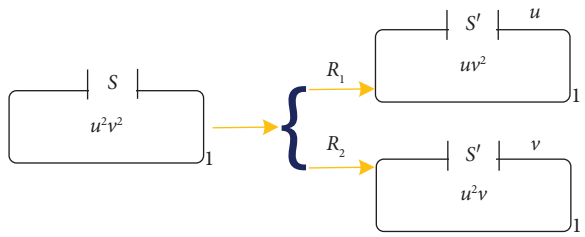

FIGURE 2: Applying sequential manner combined with flat maximal parallelism.

$\mathrm{cod}(u)$ represents an input multiset of *CCSS* P systems;

(b) relative to (X, cod, s), such a system is sound, that is, with regard to $u \in I_X$, if *CCSS* P systems have an accepting computation, then $\theta_X(u) = 1$;

(c) relative to (X, cod, s), such a system is complete, that is, suppose $u \in I_X$, relative to a problem, if $\theta_X(u) = 1$, computations of $\Pi(s(u))$ with $\mathrm{cod}(u)$ is an accepting one;

(d) such a system is polynomially bounded, that is, *CCSS* P systems stop computation and reach the halting configuration after $p(|u|)$ steps (p is a polynomial function).

Definition 4. The maximum rule length of a *CCSS* P system is equivalent to that of symport rules in the system. $\mathrm{PMC}_{\mathrm{CPS-CSSR}(k)}$ represents that the class of decision problems can be solved by a family of recognizer *CCSS* P systems in a uniform manner in polynomial time, where the maximal length of rules is k.

3. A Uniform Solution to the $\mathcal{SAT}$ Problem Based on *CCSS* P Systems

3.1. Constructing CCSS P Systems to Solve the $\mathcal{SAT}$

Theorem 5. $\mathcal{SAT} \in \mathrm{PMC}_{\mathrm{CPS-CSSR}(1)}$.

Proof. A $\mathcal{SAT}$ formula consists of n Boolean variables and m clauses:

$$C_j = y_{1,j} \vee \cdots \vee y_{p_j,j}, \tag{6}$$

where $y_{i,j} \in \{x_l, x_l | 1 \le l \le n\}$, $1 \le i \le p_j$, and $1 \le j \le m$; x_l is the negation of a propositional variable x_l.

A $\mathcal{SAT}$ formula γ is encoded by $\mathrm{cod}(\gamma)$:

$$\mathrm{cod}(\gamma) = \alpha_{1,1} \cdots \alpha_{n,1} \alpha_{1,2} \cdots \alpha_{n,2} \cdots \alpha_{1,m} \cdots \alpha_{n,m}. \tag{7}$$

We codify γ with the multiset ($1 \le j \le m$ and $1 \le i \le n$).

$$\alpha_{i,j} = \begin{cases} d_{i,j}, \text{which denotes } x_i \text{ is in } C_j; \\ \overline{d}_{i,j}, \text{which denotes } x_i \text{ is in } C_j; \\ d'_{i,j}, \text{which denotes that neither } x_i \text{ nor } x_i \text{ is in } C_j. \end{cases} \tag{8}$$

To solve the $\mathcal{SAT}$, a recognizer P system $\Pi_{\mathcal{SAT}(m,n)}$ is constructed:

$$\Pi = (O, K, \Sigma, E, \mu, w_1, \ldots, w_4, P_1, S_1, K, Q_1, \mathcal{R}_1, \ldots, \mathcal{R}_4, i_{\mathrm{in}}, i_{\mathrm{out}}), \tag{9}$$

where

(i) O is the set of objects:

$$O = \Sigma \cup \{g_i | 1 \le i \le n+1\} \cup \{b_i, t_i, f_i | 1 \le i \le n\} \cup \{e_j | 1 \le j \le m\} \cup \{a, b_1, \mathcal{YES}, \mathcal{NO}\}. \tag{10}$$

(ii) K is the set of channel states:

$$K = \{T_i, T'_i, F_i, F'_i, S'_i | 1 \le i \le n\} \cup \{T_{i,j}, F_{i,j} | 1 \le i \le n, 1 \le j \le m+1\} \cup \{P_i | 1 \le i \le 2mn + m + 9n + 3\} \cup \{T'_{i,j}, F'_{i,j}, \overline{T}_{i,j}, \overline{F}_{i,j} | 1 \le i \le n, 1 \le j \le m\} \cup \{S_i | 1 \le i \le n+1\} \cup \{Z_i | 1 \le i \le m+2\} \cup \{K, K'\} \cup \{Q_j | 1 \le i \le m+3\}. \tag{11}$$

(iii) $\Sigma = \{d_{i,j}, \overline{d}_{i,j}, d'_{i,j} | 1 \le i \le n, 1 \le j \le m\}$ denotes input objects

(iv) $E = \phi$

(v) $\mu = [_{234}]_1$

(vi) $w_1 = \{a, \mathcal{NO}\}, w_2 = \mathcal{YES}, w_4 = g_1 b_1 e_1, \ldots, e_m$.

(vii) $i_{\mathrm{in}} = 2$ and $i_{\mathrm{out}} = 0$.

(viii) The rules in $\mathcal{R}_1$ are as follows:

$R_{1,i} \equiv (P_i, (a, \mathrm{out}), P_{i+1}), 1 \le i \le 2mn + m + 9n + 3$.

$R_{2,i} \equiv (P_i, (a, \text{in}), P_{i+1}), 1 \le i \le 2mn + m + 9n + 3.$

$R_3 \equiv (P_{2mn+2m+11n}, (\mathcal{NO}, \text{out}), P_{2mn+2m+11n+1}).$

$R_4 \equiv (P_{2mn+2m+11n+1}, (\mathcal{YES}, \text{out}), P_{2mn+2m+11n+2}).$

(ix) The rules in $\mathcal{R}_2$ are as follows:

Division rules:

$R_{5,i} \equiv [b_i]_2 \longrightarrow [t_i]_2[f_i]_2, 1 \le i \le n.$

Symport rules:

$R_{6,i} \equiv (S_i, (b_i, \text{in}), S_i'), 1 \le i \le n.$

$R_{7,i} \equiv (S_i', (f_i, \text{out}), F_i), 1 \le i \le n.$

$R_{8,i} \equiv (F_i, (g_{n+1}, \text{in}), F_i'), 1 \le i \le n.$

$R_{9,i} \equiv (S_i', (f_i, \text{in}), T_i), 1 \le i \le n.$

$R_{10,i} \equiv (T_i, (f_i, \text{out}), T_i'), 1 \le i \le n.$

$R_{11,i} \equiv (T_i', (b_1, \text{in}), T_{i,1}), 1 \le i \le n.$

$R_{12,i} \equiv (F_i', (f_i, \text{in}), F_{i,1}), 1 \le i \le n.$

$R_{13,i,j} \equiv (T_{i,j}, (d_{i,j}, \text{out}), T_{i,j}'), 1 \le i \le n, 1 \le j \le m.$

$R_{14,i,j} \equiv (T_{i,j}, (\overline{d}_{i,j}, \text{out}), \overline{T}_{i,j}), 1 \le i \le n, 1 \le j \le m.$

$R_{15,i,j} \equiv (T_{i,j}, (d_{i,j}', \text{out}), \overline{T}_{i,j}), 1 \le i \le n, 1 \le j \le m.$

$R_{16,i,j} \equiv (F_{i,j}, (\overline{d}_{i,j}, \text{out}), F_{i,j}'), 1 \le i \le n, 1 \le j \le m.$

$R_{17,i,j} \equiv (F_{i,j}, (d_{i,j}, \text{out}), \overline{F}_{i,j}), 1 \le i \le n, 1 \le j \le m.$

$R_{18,i,j} \equiv (F_{i,j}, (d_{i,j}', \text{out}), \overline{F}_{i,j}), 1 \le i \le n, 1 \le j \le m.$

$R_{19,i,j} \equiv (T_{i,j}', (e_j, \text{in}), T_{i,j+1}), 1 \le i \le n, 1 \le j \le m.$

$R_{20,i,j} \equiv (F_{i,j}', (e_j, \text{in}), F_{i,j+1}), 1 \le i \le n, 1 \le j \le m.$

$R_{21,i,j} \equiv (\overline{T}_{i,j}, (\overline{d}_{i,j}, \text{in}), T_{i,j+1}), 1 \le i \le n, 1 \le j \le m.$

$R_{22,i,j} \equiv (\overline{T}_{i,j}, (d_{i,j}', \text{in}), T_{i,j+1}), 1 \le i \le n, 1 \le j \le m.$

$R_{23,i,j} \equiv (\overline{F}_{i,j}, (d_{i,j}, \text{in}), F_{i,j+1}), 1 \le i \le n, 1 \le j \le m.$

$R_{24,i,j} \equiv (\overline{F}_{i,j}, (d_{i,j}', \text{in}), F_{i,j+1}), 1 \le i \le n, 1 \le j \le m.$

$R_{25,i} \equiv (T_{i,m+1}, (t_i, \text{out}), S_{i+1}), 1 \le i \le n.$

$R_{26,i} \equiv (F_{i,m+1}, (f_i, \text{out}), S_{i+1}), 1 \le i \le n.$

$R_{27} \equiv (S_{n+1}, (e_1, \text{out}), Z_2).$

$R_{28,j} \equiv (Z_j, (e_j, \text{out}), Z_{j+1}), 2 \le j \le m.$

$R_{29} \equiv (Z_{m+1}, (\mathcal{NO}, \text{in}), Z_{m+2}).$

$R_{30} \equiv (Z_{m+2}, (\mathcal{YES}, \text{out}), Z_{m+2}).$

(x) The rules in $\mathcal{R}_3$ are as follows:

Division rules:

$R_{31,i} \equiv [t_i]_3 \longrightarrow [b_{i+1}]_3[b_{i+1}]_3, 1 \le i \le n - 1.$

Symport rules:

$R_{32,i} \equiv (K, (t_i, \text{in}), K'), 1 \le i \le n - 1.$

$R_{33,i} \equiv (K', (b_i, \text{out}), K), 2 \le i \le n.$

(xi) The rules in $\mathcal{R}_4$ are as follows:

Division rules:

$R_{34,i} \equiv [g_i]_4 \longrightarrow [g_{i+1}]_4[g_{i+1}]_4, 1 \le i \le n.$

Symport rules:

$R_{35,j} \equiv (Q_j, (e_j, \text{out}), Q_{j+1}), 1 \le j \le m.$

$R_{36} \equiv (Q_{m+1}, (g_{n+1}, \text{out}), Q_{m+2}).$

$R_{37} \equiv (Q_{m+2}, (b_1, \text{out}), Q_{m+3}).$

The whole computing process is primarily classified into two main phases: the generation phase and the checking/output phase. There are many rules involved in the generation phase, which is relatively complicated; hence, we will describe the generation phase in detail.

The generation phase generates all possible assignments of all variables and detects satisfiable clauses with these assignments. The computing process of this phase is shown in Figure 3. Initially, the computation begins by using the membranes with label 1 and label 4. Relative to these two membranes, a parallel computing process is used. The function of membrane 1 is to count the steps during system operation so that the computing result can be output at a given moment (we will explain this computing process later). Next, we only consider the computations involving membranes with label 4. At the initial configuration, the multiset of objects $g_1b_1e_1, \ldots, e_m$ exists in membrane 4. Because the system includes object g_1 present in membrane 4, the system executes the division rule $R_{34,i}$, which can generate two cells with the same label, and object g_2 will appear in the two membranes. Since division rules have higher priority, they can be used in membrane 4 because of g_2. Finally, after n steps, 2^n membranes will appear, and object g_{n+1} will be generated in each membrane. Subsequently, relative to channel state Q_1 on cell 4, rule $R_{35,i}$ is activated and can be executed m times; the channel state will increase by 1 at each step, and finally, object Q_{m+1} will appear. At that moment, rule R_{36} can be executed, sending object g_{n+1} to the membrane with label 1. Based on the channel state on membrane 4, the system will execute R_{37} at the next step, and object b_1 in membrane 4 will be sent to membrane 1; these objects will be used in the subsequent computing process.

After the above computation involving membrane 4 ends, the rules involving membrane 2 begin to be applied. The following computing process assigns values (e.g., "true" or "false") to the variables x_1 of all clauses in a $\mathcal{SAT}$ formula, and the rules from $R_{6,1}$ to $R_{33,2}$ are applied. First, $R_{6,i}$ is activated, and then, division rule $R_{5,i}$ is executed. Objects t_1 and f_1 appear in the two newly generated membranes. After a division rule is used, f_i is sent to the region outside membrane 2 by applying rule $R_{7,i}$, and the channel state of this membrane will be F_i. Next, rules $R_{8,i}$ and $R_{9,i}$ can be activated; objects f_i and g_{n+1} generated from membrane 4 will enter membrane 2. Then, only rule $R_{10,i}$ is used, and the channel state of this membrane changes to T_i'. Next, $R_{11,i}$ and $R_{12,i}$ are simultaneously executed, b_1 (resp., f_1) enters membrane 2 based on channel state T_i' (resp., F_i'), and $T_{i,1}$ (resp., $F_{i,1}$) will become the new channel state.

The subsequent computing process is primarily used to compare the assignment of the current variables with the corresponding variables in all clauses, and it runs in parallel related to two different membranes simultaneously. In label 2, one rule in set $\{R_{13,i,j}, R_{14,i,j}, \text{and } R_{15,i,j}\}$ is executed first and then one rule in set $\{R_{19,i,j}, R_{21,i,j}, \text{and } R_{22,i,j}\}$ will be used; this process checks the current variable assignment in each clause labelled "true" in the sequential manner

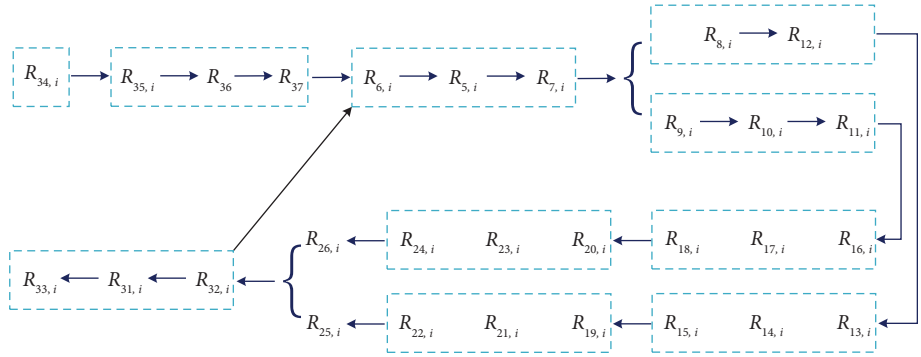

Figure 3: The computing process during the generation phase.

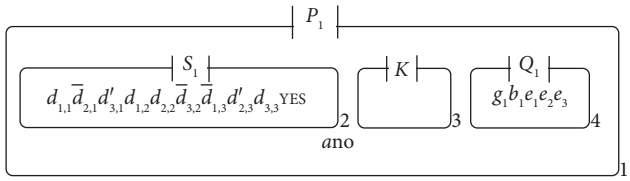

Figure 4: The initial configuration.

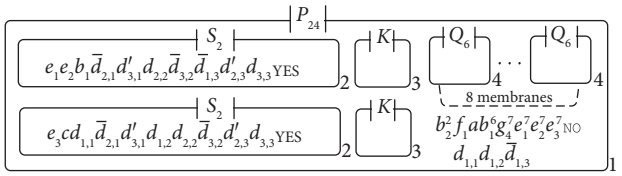

Figure 5: When the process corresponding to x_1 completes.

combined with flat maximal parallelism. A similar computation is executed in another membrane with label 2: one rule in set $\{R_{16,i,j}, R_{17,i,j}$, and $R_{18,i,j}\}$ is applied first and then one rule in set $\{R_{20,i,j}, R_{23,i,j}$, and $R_{24,i,j}\}$ will be used; this process checks the current variable assignment in each clause labelled "false." In the computing process above, e_j is an object that has been generated in membrane 4. When rules are executed, the value of the second subscript of the channel state corresponding to membrane 2 increases by 1; when the value increases to $m + 1$, which means that each clause associated with the current variable has been checked, the system executes rule $R_{25,i}$ (resp., $R_{26,i}$); notably, t_i (resp, f_i) would exist in membrane 1, and the channel state of the corresponding membrane is changed to S_2. The newly generated object t_i is moved to membrane 3 by employing rule $R_{32,i}$, and then, division rule $R_{31,i}$ is activated based on the influence of t_i; simultaneously, object b_2 appears in the newly generated membrane. Next, rule $R_{33,i}$ is employed, object b_2 is sent to membrane 1, and the next iteration proceeds in the computation.

The subsequent computation is similar to the previous process, that is, values are assigned to other variables (from x_2 to x_n) in the $\mathcal{SAT}$ formula, and the corresponding process used above for variable x_1 is implemented. Finally, all variables corresponding to clauses in the $\mathcal{SAT}$ formula are compared with assignments, and the result "true" for the corresponding clauses is obtained. The system uses the strategy of sequential manner combined with flat maximal parallelism; notably, relative to some rules (e.g., $R_{6,i}$ and $R_{8,i}$), even if more than one copy of a object can be used to by a rule simultaneously, such a rule cannot be executed multiple times and only used once at the step. Finally, when the n-th variable is executed, the application of rules $R_{25,n}$ and $R_{26,n}$ ends, and the above computing process ceases. Although t_n and f_n appear in membrane 1 at that moment, the rules in membrane 3 (e.g., $R_{31,i}$, $R_{32,i}$, and $R_{33,i}$) are not applied.

When S_{n+1} exists on the channel state of membrane 2, the system begins the computation for the checking/output phase. At this point, rule R_{27} is activated, and object e_1 is transferred out of membrane 2; additionally, Z_2 would be the new channel state. Next, rule $R_{28,j}$ is applied as long as objects $e_2, e_3, \ldots, e_m$ exist in a membrane with label 2, and these objects can be removed from membrane 2; simultaneously, the value of the corresponding channel state will be continuously increased. If the channel state of membrane 2 reaches Z_{m+1}, the $\mathcal{SAT}$ formula has a satisfiable solution; at that moment, rules R_{29} and R_{30} can be applied to send $\mathcal{NO}$ to membrane 2 and $\mathcal{YES}$ to the membrane with label 1. If the channel state of membrane 2 does not reach Z_{m+1}, the conclusion is no satisfiable solution; in this case, rules R_{29} and R_{30} are not executed, and $\mathcal{NO}$ remains in membrane 1. At step $2mn + 2m + 11n$, if $\mathcal{NO}$ is in membrane 1, rule R_3 is activated, and $\mathcal{NO}$ appears in the environment as the computing result. Therefore, if an $\mathcal{SAT}$ solution is unsatisfactory, the entire computation requires $2mn + 2m + 11n + 1$ steps; otherwise, rule R_3 will not be executed. Additionally, when the system remains at step $2mn + 2m + 11n + 1$, rule R_4 is activated, and $\mathcal{YES}$ appears in the environment as the computing result. Therefore, if an $\mathcal{SAT}$ solution is satisfied, the whole computation requires $2mn + 2m + 11n + 2$ steps. □

3.2. Some Formal Details

(i) Size of the set O: $3mn + m + 4n + 5$;

(ii) Size of the set K: $8mn + 3m + 17n + 11$;

(iii) Initial number of membranes: 4;

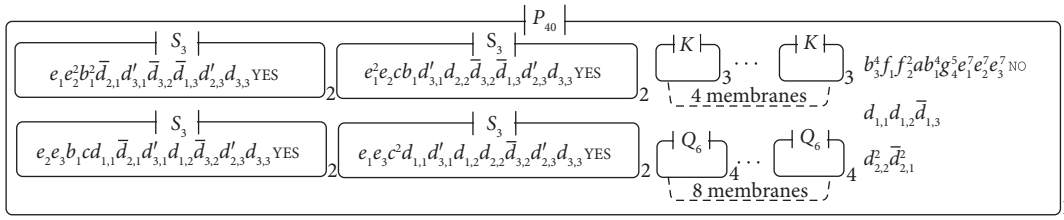

Figure 6: When the process corresponding to x_2 completes.

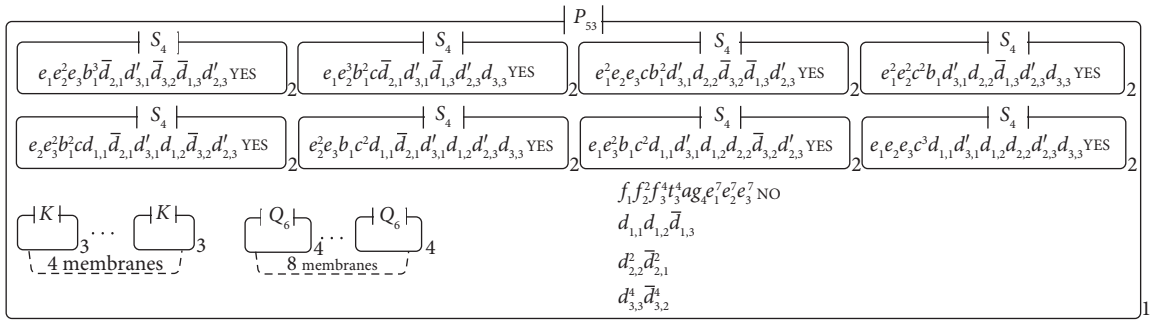

Figure 7: When the process corresponding to x_3 completes.

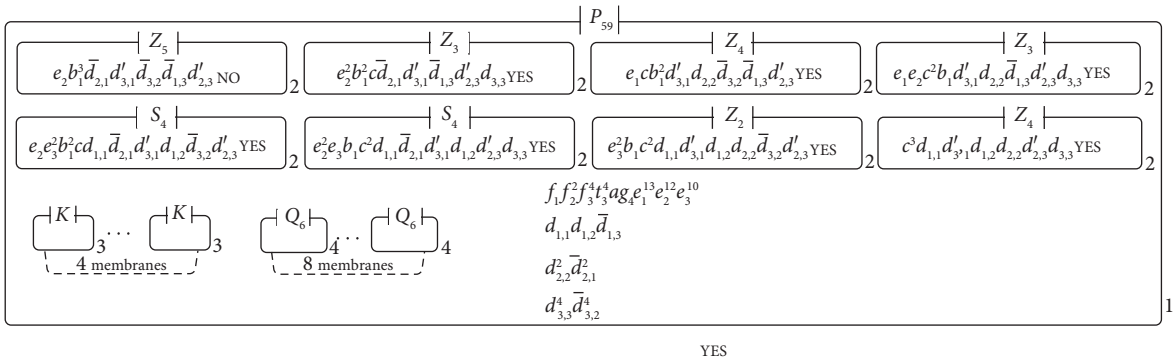

Figure 8: The halting configuration.

(iv) Initial number of objects: $m + 5$;

(v) The total number of rules: $14mn + 3m + 23n + 11$;

(vi) The maximal length of rules: 1.

Corollary 6. $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP} \subseteq \mathrm{PMC}_{\mathrm{CPS-CSSR}(1)}$.

In [35], the $\mathcal{SAT}$ problem was solved with the maximum rule length 2, where communication rules include symport rules and antiport rules.

In this work, however, on the one hand, communication rules are restricted to only one type, namely, symport rules; on the other hand, the maximal length of rules is decreased from 2 to 1. The proof indicates that our work has improved upon the current research method.

4. Case Study

In what follows, a case study is explored to demonstrate the previous proof. A formula of the $\mathcal{SAT}$ is denoted by $\gamma = (x_1 \vee x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3)$, which has 3 clauses and 3 variables, and the input multiset is denoted by $\Sigma = d_{1,1}\overline{d}_{2,1}d'_{3,1}d_{1,2}d_{2,2}\overline{d}_{3,2}\overline{d}_{1,3}d'_{2,3}d_{3,3}$.

Figure 4 denotes the initial configuration. When the system begins operation, rules involving membrane 1 and membrane 4 are applied in parallel. In membrane 4, many objects $e_1, e_2, \ldots, e_m$ are generated because of the division rule, which is useful later. There are three variables in the formula, which correspond to the iterations in the computing process. During the first iteration, the division rule is also used in membrane 2, and two assignments ("true" and

"false") for variable x_1 occur in each membrane; then, the system checks the two assignments of variable x_1 in each clause, and finally, the configuration is obtained (Figure 5).

The execution process for variable x_2 is similar to that for variable x_1, as shown in Figure 6. When the final iteration is completed, Figure 7 denotes the configuration of the system. The details of the above computing process for variables x_1, x_2, and x_3 can be found in Figure 3 of Section 3.1.

Finally, we determine whether multiset $e_1e_2e_3$ exists in each membrane with label 2. In this instance, it is obvious that the formula has satisfiable solutions. Hence, when the system stops, $\mathcal{YES}$ is output to the environment as the computing result (Figure 8).

5. Conclusions

In this work, we have constructed a novel variant, namely, *CCSS* P systems; with respect to communication rules, only symport rules are employed. The computational efficiency of this variant has been explored. The proof indicates that the $\mathcal{SAT}$ problem is solved by applying symport rules and membrane division. With regard to the system we constructed, the maximal length of rules is 1; moreover, the rule types of communication rules decreased from 2 to 1, that is, only symport rules are applied. Thus, in terms of computational complexity, our method improves upon the current research method.

Membrane separation and cell separation have obtained some satisfactory results in the existing literature (e.g., [37]). In this work, we adopt membrane division in the proposed variant; nevertheless, readers can perform membrane separation to potentially construct a new variant and explore its computational efficiency.

In this work, during system operation, we use sequential manner combined with flat maximal parallelism as the main strategy. Inspired by some actual biological phenomena, other methods have been introduced in membrane computing, such as time-freeness [41], local synchronization [42], rule synchronization [43], asynchronism [44], and minimal parallel [45] approaches. Especially for time-freeness, the execution time of each rule may be different. Therefore, based on the variant of this article, readers can introduce time-freeness to *CCSS* P systems and construct a more robust computing system, which is worthy of further study.

In our work, multiple membranes can work in parallel to perform high-efficiency computations. However, the parallelization of certain computing processes is not particularly excellent. Readers can attempt to improve parallelism among different membranes to construct a system with enhanced performance.

Data Availability

No datasets were analyzed or generated during the course of the current study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Science and Technology Research Program of the Chongqing Municipal Education Commission (Grant no. KJZD-K202003201).

References

[1] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[2] L. Adleman, "Molecular computation of solutions to combinatorial problems," *Science*, vol. 266, no. 5187, pp. 1021–1024, 1994.

[3] Z. Gazdag and G. Kolonits, "A new method to simulate restricted variants of polarizationless P systems with active membranes," *Journal of Membrane Computing*, vol. 1, no. 4, pp. 251–261, 2019.

[4] H. Peng, T. Bao, X. Luo et al., "Dendrite P systems," *Neural Networks*, vol. 127, pp. 110–120, 2020.

[5] H. Peng and J. Wang, "Coupled neural P systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 6, pp. 1672–1682, 2019.

[6] B. Song, X. Zeng, M. Jiang, and M. J. Pérez-Jiménez, "Monodirectional tissue P systems with promoters," *IEEE Transactions on Cybernetics*, vol. 51, no. 1, pp. 438–450, 2021.

[7] B. Song, X. Zeng, and A. Rodríguez-Patón, "Monodirectional tissue P systems with channel states," *Information Sciences*, vol. 546, pp. 206–219, 2021.

[8] T. Wu, Z. Zhang, and L. Pan, "On languages generated by cell-like spiking neural P systems," *IEEE Transactions on Nanobioscience*, vol. 15, no. 5, pp. 455–467, 2016.

[9] Y. Zhao, X. Liu, M. Sun, J. Qu, and Y. Zheng, "Time-free cell-like P systems with multiple promoters/inhibitors," *Theoretical Computer Science*, vol. 843, pp. 73–83, 2020.

[10] J. Cooper and R. Nicolescu, "Alternative representations of P systems solutions to the graph colouring problem," *Journal of Membrane Computing*, vol. 1, no. 2, pp. 112–126, 2019.

[11] P. Guo, J. Zhu, H. Chen, and R. Yang, "A linear-time solution for all-SAT problem based on P system," *Chinese Journal of Electronics*, vol. 27, no. 2, pp. 367–373, 2018.

[12] Y. Luo, H. Tan, Y. Zhang, and Y. Jiang, "The computational power of timed P systems with active membranes using promoters," *Mathematical Structures in Computer Science*, vol. 29, no. 5, pp. 663–680, 2019.

[13] Y. Luo, Y. Zhao, and C. Chen, "Homeostasis tissue-like P systems," *IEEE Transactions on NanoBioscience*, vol. 20, no. 1, pp. 126–136, 2021.

[14] 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*, vol. 275, Article ID 104542, 2020.

[15] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[16] G. Singh and K. Deep, "Effectiveness of new multiple-PSO based membrane optimization algorithms on CEC 2014

benchmarks and Iris classification," *Natural Computing*, vol. 16, no. 3, pp. 473–496, 2017.

[17] G. Zhang, H. Rong, F. Neri, and M. J. Pérez-Jiménez, "An optimization spiking neural p system for approximately solving combinatorial optimization problems," *International Journal of Neural Systems*, vol. 24, no. 5, Article ID 1440006, 2014.

[18] P. Frisco, M. Gheorghe, and M. J. Pérez-Jiménez, *Applications of Membrane Computing in Systems and Synthetic Biology*, Springer-Verlag, Heidelberg, Germany, 2014.

[19] M. García-Quismondo, M. Levin, and D. Lobo, "Modeling regenerative processes with membrane computing," *Information Sciences*, vol. 381, pp. 229–249, 2017.

[20] C. Buiu and A. G. Florea, "Membrane computing models and robot controller design, current results and challenges," *Journal of Membrane Computing*, vol. 1, no. 4, pp. 262–269, 2019.

[21] J. Wang, H. Peng, W. Yu et al., "Interval-valued fuzzy spiking neural P systems for fault diagnosis of power transmission networks," *Engineering Applications of Artificial Intelligence*, vol. 82, pp. 102–109, 2019.

[22] G. Zhang, M. J. Pérez-Jiménez, and M. Gheorghe, *Real-life Applications with Membrane Computing*, Springer-Verlag, Heidelberg, Germany, 2017.

[23] R. Xian, R. Lugu, H. Peng, Q. Yang, X. Luo, and J. Wang, "Edge detection method based on nonlinear spiking neural systems," *International Journal of Neural Systems*, vol. 33, no. 1, Article ID 2250060, 2023.

[24] X. Chen, H. Peng, J. Wang, and F. Hao, "Supervisory control of discrete event systems under asynchronous spiking neuron P systems," *Information Sciences*, vol. 597, pp. 253–273, 2022.

[25] S. Xia, G. Wang, Z. Chen, Y. Duan, and Q. liu, "Complete random forest based class noise filtering learning for improving the generalizability of classifiers," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 11, pp. 2063–2078, 2019.

[26] S. Xia, Y. Liu, X. Ding, G. Wang, H. Yu, and Y. Luo, "Granular ball computing classifiers for efficient, scalable and robust learning," *Information Sciences*, vol. 483, pp. 136–152, 2019.

[27] S. Xia, D. Peng, D. Meng et al., "Ball k-means: fast adaptive clustering with No bounds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 87–99, 2022.

[28] J. Hu, H. Peng, J. Wang, and W. Yu, "kNN-P: a kNN classifier optimized by P systems," *Theoretical Computer Science*, vol. 817, pp. 55–65, 2020.

[29] B. Song, K. Li, D. Orellana-Martín, M. J. Pérez-Jiménez, and I. PéRez-Hurtado, "A survey of nature-inspired computing: membrane computing," *ACM Computing Surveys*, vol. 54, no. 1, pp. 1–31, 2021.

[30] C. Martín-Vide, G. Păun, J. Pazos, and A. Rodríguez-Patón, "Tissue P systems," *Theoretical Computer Science*, vol. 296, no. 2, pp. 295–326, 2003.

[31] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta Informaticae*, vol. 71, no. 2, pp. 279–308, 2006.

[32] A. Păun and G. Păun, "The power of communication: P systems with symport/antiport," *New Generation Computing*, vol. 20, no. 3, pp. 295–305, 2002.

[33] R. Freund, G. Păun, and M. J. Pérez-Jiménez, "Tissue P systems with channel states," *Theoretical Computer Science*, vol. 330, no. 1, pp. 101–116, 2005.

[34] B. Song, L. Pan, and M. J. Pérez-Jiménez, "Cell-like P systems with channel states and symport/antiportrules," *IEEE Transactions on NanoBioscience*, vol. 15, no. 6, pp. 555–566, 2016.

[35] S. Jiang, Y. Wang, and Y. Su, "A uniform solution to SAT problem by symport/antiport P systems with channel states and membrane division," *Soft Computing*, vol. 23, no. 12, pp. 3903–3911, 2019.

[36] L. Pan and M. J. Pérez-Jiménez, "Computational complexity of tissue-like P systems," *Journal of Complexity*, vol. 26, no. 3, pp. 296–315, 2010.

[37] M. J. Pérez-Jiménez and P. Sosík, "An optimal Frontier of the efficiency of tissue P systems with cell separation," *Fundamenta Informaticae*, vol. 138, no. 1–2, pp. 45–60, 2015.

[38] L. Pan, G. Păun, and B. Song, "Flat maximal parallelism in P systems with promoters," *Theoretical Computer Science*, vol. 623, pp. 83–91, 2016.

[39] M. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, Englewood Cliffs, New Jersey, 1967.

[40] G. Rozenberg and A. Salomaa, *Handbook of Formal Languages*, Springer, Berlin, Germany, 1997.

[41] M. Cavaliere and D. Sburlan, "Time-independent P systems," *Lecture Notes in Computer Science*, pp. 239–258, Springer-Verlag, Berlin, Germany, 2005.

[42] L. Pan, A. Alhazov, H. Su, and B. Song, "Local synchronization on asynchronous tissue P systems with symport/antiport rules," *IEEE Transactions on NanoBioscience*, vol. 19, no. 2, pp. 315–320, 2020.

[43] B. Song and L. Pan, "Rule synchronization for tissue P systems," *Information and Computation*, vol. 281, Article ID 104685, 2021.

[44] P. Frisco, G. Govan, and A. Leporati, "Asynchronous P systems with active membranes," *Theoretical Computer Science*, vol. 429, pp. 74–86, 2012.

[45] G. Ciobanu, L. Pan, G. Păun, and M. J. Pérez-Jiménez, "P systems with minimal parallelism," *Theoretical Computer Science*, vol. 378, no. 1, pp. 117–130, 2007.