Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 424108, 11 pages
http://dx.doi.org/10.1155/2013/424108

Research Article

A Linear Time Complexity of Breadth-First Search Using *P* System with Membrane Division

Einallah Salehi, Siti Mariyam Shamsuddin, and Kourosh Nemati

Soft Computing Research Group, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia

Correspondence should be addressed to Einallah Salehi; esalehi50@yahoo.com

Received 18 January 2013; Revised 6 April 2013; Accepted 8 April 2013

Academic Editor: Jian Guo Zhou

One of the known methods for solving the problems with exponential time complexity such as NP-complete problems is using the brute force algorithms. Recently, a new parallel computational framework called Membrane Computing is introduced which can be applied in brute force algorithms. The usual way to find a solution for the problems with exponential time complexity with Membrane Computing techniques is by *P* System with active membrane using division rule. It makes an exponential workspace and solves the problems with exponential complexity in a polynomial (even linear) time. On the other hand, searching is currently one of the most used methods for finding solution for problems in real life, that the blind search algorithms are accurate, but their time complexity is exponential such as breadth-first search (BFS) algorithm. In this paper, we proposed a new approach for implementation of BFS by using *P* system with division rule technique for first time. The theorem shows time complexity of BSF in this framework on randomly binary trees reduced from $O(2^d)$ to $O(d)$.

1. Introduction

Membrane Computing, introduced in 1998, is an area of study in computer science [1]. Membrane Computing could be considered as a framework in the distributed parallel computing, and it is inspired from the computing ideas, structures, models, the living cells activities, and the cells organized in a hierarchy. Using Membrane Computing, we obtain a computing device called *P* system. Several membranes are included in the basic model of a *P* system, embedded in a main membrane that is known as the skin. The Euclidean space is divided by membranes into some regions consisting of some objects (that are generally signified by alphabetical symbols) and the evolution rules. Applying these rules, the objects can be evolved and/or moved from a region to their adjacent regions. Typically, these rules are employed in a nondeterministic and maximally parallel manner. After an initial system configuration, the computation starts, and it terminates once no evolution rule could be applied. The computation result can be a multiset of objects that are embedded in an output membrane or discharged from the system skin.

So far, several types of *P* system have been introduced and investigated [1–3]. *P* system with active membrane is one of the most well-known *P* systems in which the number of membranes increases during a computation by using some division rules, creation rules, or separation rules [4–7] (we call them in short *P* system with membrane division or *P* system with division rule, *P* system with membrane creation or *P* system with creation rule, and *P* system with separation rule, respectively). They make models of the division or creation of the cells in the nature. In the initial variant of *P* system with division rule [8], the division rules were defined in such a way that, from the division of a membrane, we could always obtain two membranes (we call it a 2-bounded division or, in short, division rule). A generalization was defined in [9] where a membrane can be divided into several (but a finite number of) membranes (we call it a k-bounded division). Also in *P* system with creation rules, some objects are so productive that they create membranes [10]. Thus, new membranes can be made exponentially.

Most of *P* systems are proved to be universal [1, 11–14] and efficient [8, 15, 16]. Several studies have been performed on the *P* system with active membrane. In most of these

studies, the time complexity is reduced from exponential to polynomial or linear time in an exponential workspace or, at least, they have improved the time complexity as follows: using P system with division rules; it was shown that the NP-complete problem SAT can be solved in a linear time [8]. As investigated in [9], the HPP can be solved using P systems with k-bounded division rules in polynomial time, but in [17], it was shown that HPP can be solved using P systems with division (2-bounded) rules in which the time complexity for this problem was linear, and the knapsack problem has been solved using a family of deterministic P system with division rules presented in [18] in linear time. Using P systems, some other problems (mostly NP-complete) have been solved in linear or polynomial time complexity, for example, SAT problem in [7, 19, 20], graph problem [21], subset sum [22], 3-COL problem [23], vertex cover problem [24], HAM-CYCLE problem [25], and HPP problem in [10]. P systems are also used to solve PSPACE problems [26–29].

In all researches conducted on this area of study, the time units are considered as the steps of computation in a P system, which are performed in parallel, in evolution rules, all membranes of the system, or the membranes division [8]. It is clear that execution of the P system framework programming on a sequential hardware does not give us a relevant result, unless it is applied on a parallel architecture. Recently, some studies were conducted on the implementation of Membrane Computing on the parallel architecture [30–34]. In all of the studies, the speed execution has been extremely increased.

On the other hand, classical search algorithms explore systematically the space of states through the saving of one or more paths in the memory and, then, recording the alternatives existing in each choice point. Once a final state is explored, the path considered as the transitions sequence is known as the solution to the problem. Many of the methods presented are designed specifically for solving the classical search problems. There are different heuristics and approximation algorithms that have found the solution in a reasonable time, but, at the same time, they may be less accurate or even they may have a good accuracy but with an exponential complexity in time [35]. One of these algorithms that have been proposed and designed already is breadth-first search (BFS). Therefore, like the parallel algorithms, few attempts have been made for improving the time complexity of the accurate searching methods.

So far, many researches (especially in real life problems) on classical searching have been conducted to reduce time implement and improve speed execution. Li et al. in [36] proposed a new algorithm that can be applied in searching for the multiple longest common subsequences (MLCS). In terms of the speed of the execution, it outperforms the previous leading parallel MLCS algorithm. In [37], the scheduling problem has been studied, in which a modified weight-combination search algorithm and a variable neighborhood search were employed to yield the optimal or near-optimal schedule. The results showed that the obtained solutions were near-optimal solutions in a reasonable computational time. Liu et al. [38], by combination of BFS and depth-first search (DFS) algorithms, obtained two new algorithms; these algorithms are known as memory function maximum degree algorithm (MD) and memory function preference degree algorithm (PD). In these cases, simulation results show that the two algorithms' performances are excellent at the same time, and the performances are improved at least 10 times. In another study, Shih et al. [39] proposed a new method for development of a splitting-based schema called Merged Search Tree to improve the schema of binary search tree defined in the radio frequency identification (RFID) system. Merged Search Tree method greatly improved the time and power performance simultaneously.

In this paper, we propose a new parallel method by P system with division rule through the exploration of BFS in a binary tree which includes positive integer weight and with finite depth. With this new method, the time complexity will be decreased from exponential to linear time. The paper is organized as follows: Section 2 describes the related works. In Section 3, we presented some basic definitions on P system with division rule. Section 4 introduces the BFS. The proposed method for BSF based on P system with membrane division is presented in Section 5, in which we provided some guidelines of the implementation of P system with membrane division for BFS. In Section 6, using two examples, experiments carried out in this study are presented. Finally, in Section 7, we presented the conclusion, and some ideas for future works were discussed.

2. Related Works

The searching problem has been recently studied on the framework of Membrane Computing. A first study on DFS, was presented by Gutiérrez-Naranjo and Pérez-Jiménez [40, 41]; also the first study on local search was presented by Gutiérrez-Naranjo and Pérez-Jiménez [40, 41]. In both cases, the search is done using transition P systems; also N-Queen problem has been considered in these researches as the case study. The results have been compared to N-Queen problem in [42] and to one other. The solution presented for 4 queens in [42] was obtained after 20,583 seconds (more than 5 hours). The average time obtained for 20 queens in [40, 41] approaches are 15,944 and 0.133275 seconds, respectively.

Also DFS in [43] and BFS in [44] have been studied on disjoint paths using P systems. In the second one, the execution time was improved compared to the first study. In [45], BFS has also been proposed for solving disjoint paths problem using P systems. In this study, the execution time has been improved in comparison with the previous study. In all three cases, the searches were done without division rules.

3. P System with Membrane Division

One of the models that have been mostly studied in the field of Membrane Computing is the P system with membrane division, which is well recognized in the P system community. This model is one of the first models introduced in 2001 [8].

Definition 1. A P system with division rule for elementary membranes and with output is a construct of the form, where

(1) $m \geq 1$ signifies the initial *degree* of the P system;

(2) V denotes the alphabet of *symbol-objects*;

(3) $T \subseteq V$ is the output alphabet;

(4) H indicates a finite set of *labels* for membranes;

(5) μ stands for a membrane *structure*, consisting of m membranes that initially have neutral polarizations labeled with elements of H;

(6) $w_1, \ldots, w_m$ indicate strings over V, which describe the initial *multisets* of the objects placed in the m regions of μ;

(7) i_0 determines the output region;

(8) R stands for a finite set of *rules* of the following forms:

(a) $[x \to u]_h^p$, for $h \in H, p \in \{+, -, 0\}, x \in V, u \in V^*$ (V^* is set of strings over V).

This is an object *evolution rule* that is accompanied with a membrane which is labeled with h, and it depends on the membrane polarity. The empty string is represented by $\lambda \in V^*$.

(b) $x[]_h^{p_1} \to [y]_h^{p_2}$, for $h \in H, p_1, p_2 \in \{+, -, 0\}, x, y \in V$.

An object is sent in from the immediate outside region of the membrane with the label h; at the same time, this object would be changed into another object, and, simultaneously, the membrane polarity can also be changed.

(c) $[x]_h^{p_1} \to y[\,]_h^{p_2}$, for $h \in H, p_1, p_2 \in \{+, -, 0\}, x, y \in V$.

Here, an object is sent out from the membrane with the label h to the immediate outside region. This object, at the same time, changes into another object, and, simultaneously, the membrane polarity can be changed.

(d) $[x]_h^p \to y$, for $h \in H, \ p \in \{+, -, 0\}, \ x, y \in V$.

A membrane labeled with h is dissolved in reaction with an object. The skin is never dissolved.

(e) $[x]_h^{p_1} \to [y]_h^{p_2}[z]_h^{p_3}$, for $h \in H, p_1, p_2, p_3 \in \{+, -, 0\}, x, y, z \in V$.

A membrane by an object could be divided into two membranes with the same label, each of them containing one object (objects may be the same or different). And the polarities of these membranes can be altered.

These rules are employed based on the following principles.

(1) All rules are used in a *parallel* (we can determine the priority for some rules) and in a *maximal* manner. At each step, one object of a membrane could be utilized by only one rule that is chosen in a nondeterministic way, but any object that is able to evolve by one rule of any form should do its evolution.

(2) With dissolution of a membrane, its contents (multiset and internal membranes) are left free in the surrounding region.

(3) All objects and membranes that have not been specified in a rule and those that have not evolved remain unchanged to the next step.

(4) Those rules that are associated with the membranes that are labeled with h are utilized for the all copies of this membrane. At one step, a membrane that is labeled with h could only be subjected to one rule of types (b)–(e), and rule (a) could also be applied with other rules. In this case, our assumption is that the evolution rules of type (a) are used first, and after that, the other types of rules are applied. This process takes only one step to be performed.

(5) The skin membrane could not be divided. Like other membranes, it can have "electrically charge".

Definition 2. One says that rule R_i has priority (in the strong sense) over rule R_j; if R_i can be applied, then rule R_j cannot be applied. In this case one shows $R_i > R_j$.

Definition 3. Let i_0 be an output membrane (region) of P system Π; one says that i_0 is a dynamic output if the P system rules can apply on i_0.

It should be noticed that in the case of P systems with division rule, there is an evolution in the membrane structure during the time of computation. This evolution could be obtained not only by declining the number of membranes that may happen due to dissolution operations (rules of type (d)) but also by an increase in the number of membranes by division. It could be an exponential increase in a linear number of steps: by applying a division rule after n steps, and because of the maximal parallelism, we get 2^n copies from the membranes of the same labels. It is an approach that has been mostly investigated to obtain an exponential workspace to reduce the time through increasing the space for solving the computationally hard problems (such as NP-complete problems) in a polynomial or even linear time.

4. Breadth-First Search

Searching is based on numerous processes in the artificial intelligence. The most important point is that lots of real-life problems could be settled as a space of states. A state refers to description of the world in a certain instant (expressed in some languages), and two states are linked by a transition if the second state can be reached from the previous one by applying one elementary operation [35].

As mentioned above, using this abstraction, the searching methods have been widely studied regardless of the problems arisen in the real world. The studies have been conducted on aspects such as the completeness (where the searching method can find a solution, if any), time and space complexity, and optimality (whether the solution is optimal in some sense). Considering the searching tree, wherein the nodes indicate the states and the arcs represent the transitions, the classical search focuses on the order in which the nodes of the tree should be explored. In this search, there are two possible approaches; in the first instance, the blind search in which the search is guided only by the topology of the

tree without any information from the states. In the second instance, the informed search in which some information about the nodes' characteristics are employed for defining heuristics to determine the next node to be explored.

In each time unit of the sequential algorithms, a single node is considered. In this computation framework, the searching strategies are determined through the different orders in which the new nodes are explored. Within a typical framework, several nodes could be possibly explored, and, for continuing the search, we should select one of them. In the best state, there is a heuristic that can be helpful while choosing the best options from the existing candidates. In a certain sense, this heuristic indicates the distance of the considered node from a solution node, and this property provides us with information about the problem nature. In lots of other situations, we cannot have access to any information about this distance, and a blind strategy is required to be applied.

Due to the lack of information about the problem nature, the blind strategies are considered exclusively within the topology of the graph (tree) and also in the order where new nodes are reached. The BFS is one of the basic types of the blind search strategies, which expands the nodes in an order regarding to their distance from the root (Figure 1(a)). The time complexity for this search strategy is $O(b^d)$, where b represents the maximum branching factor and d signifies the lowest depth in which the solution could be found (for more information, see [35]).

5. Proposed Method

In this section, we present a first approach to BFS by P systems with membrane division. The goal of this approach is to show that Membrane Computing provides all the ingredients that we need to find a solution for any problem represented as a space of states and, hence, to be a useful tool to solve many real-life problems. In this approach, our aim is looking for a solution in search space (binary tree) by using a P system with membrane division according to BFS.

5.1. An Overview of Breadth-First Search by P System with Membrane Division. In this section, we provide the design overview of a P system family with membrane division of searching according to the BFS binary trees with positive integer weight and with finite depth. The substance of the research is as follows: in initial configuration, the P system explores the root (start state); then, it checks to find the goal (final state), if any. Next, the computation will be halted and the remaining nodes show the path from start to goal. If the goal does not exist, then the P system divides the membranes (elementary membrane that located in the skin), explores the next level (depth) of nodes (Figure 2), and checks again to find the goal (see the proposed method in Figure 3). The computation will be continued until the goal is found. Considering the fact that each level of tree is explored in one stage (Figure 1(b)), and if the goal exists at the depth d, the goal will be discovered after $d + 1$ stages (stage 0 for root, stage 1 for depth 1, and stage d for depth d).

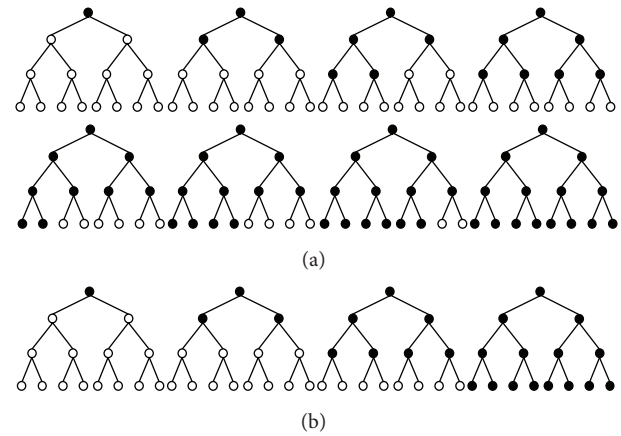

Figure 1: BFS on binary tree according to classical (a) and proposed (b) methods.

In Section 5.2, it will be demonstrated that each stage takes 4 steps (except stage one), and the time complexity of this method is $O(4d + 5) = O(d)$. So the time complexity for BFS at this method will decrease from $O(2^d)$ to $O(d)$. On the other hand, we know that one of the challenges that have arisen on the search threads is complexity in time, hence using a P system with membrane division can be a very convenient way for searching with low time complexity. Also, because of the parallel operation of the rules, the time execution will reduce in the P system.

5.2. Breadth-First Search Problem Formulation and P System Structure. In this section, we consider a P system with membrane division for BFS on binary trees and explain their features according to Section 5.1. We illustrate the rules and their implementation, step by step. It is noticeable that each binary tree's node has four fields: a key node, parent pointer, left child pointer, and right child pointer. The key node is the label or weight of a node, and pointers are the interface between the parent, the left and right children nodes. With this brief introduction, we explain the case study tree that considers the random binary tree with the following features.

(i) Depth of tree is integer number n, where $n \geq 0$;

(ii) the node j is shown by a_j;

(iii) the left and right child pointers for a_j are $2j$ and $2j+1$, respectively;

(iv) the key node for a_j is $x_j \in N$, we *show key* $(a_j) = x_j$;

(v) the key node for final state (goal) is G, we *show key* $(goal) = G$;

(vi) the relationship between a_j and its parent is considered by

$$y_j,\ y_j = \begin{cases} 1, & \text{if there exist } a_j \\ 0, & \text{if there exist not } a_j; \end{cases} \quad (1)$$

(vii) the numerical inputs of system are n, G, x_j, and y_j.

Note that in all of above relations $1 \leq j \leq 2^{n+1} - 1$.

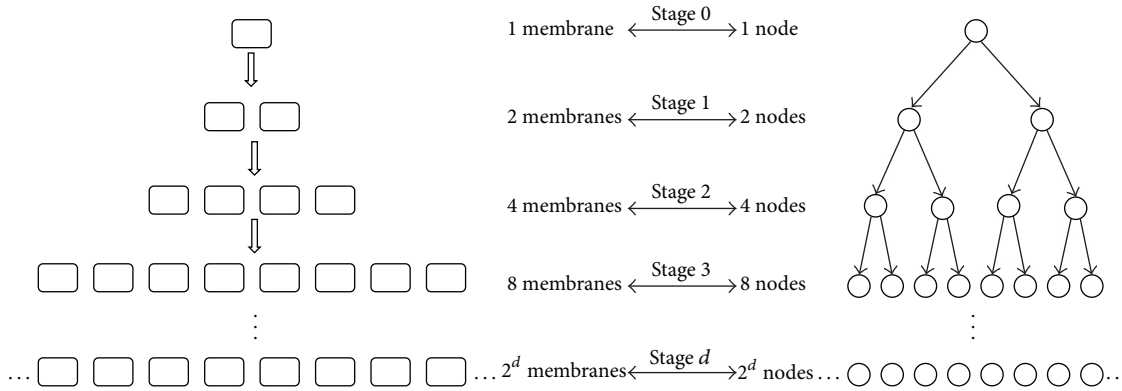

Figure 2: General view of the system.

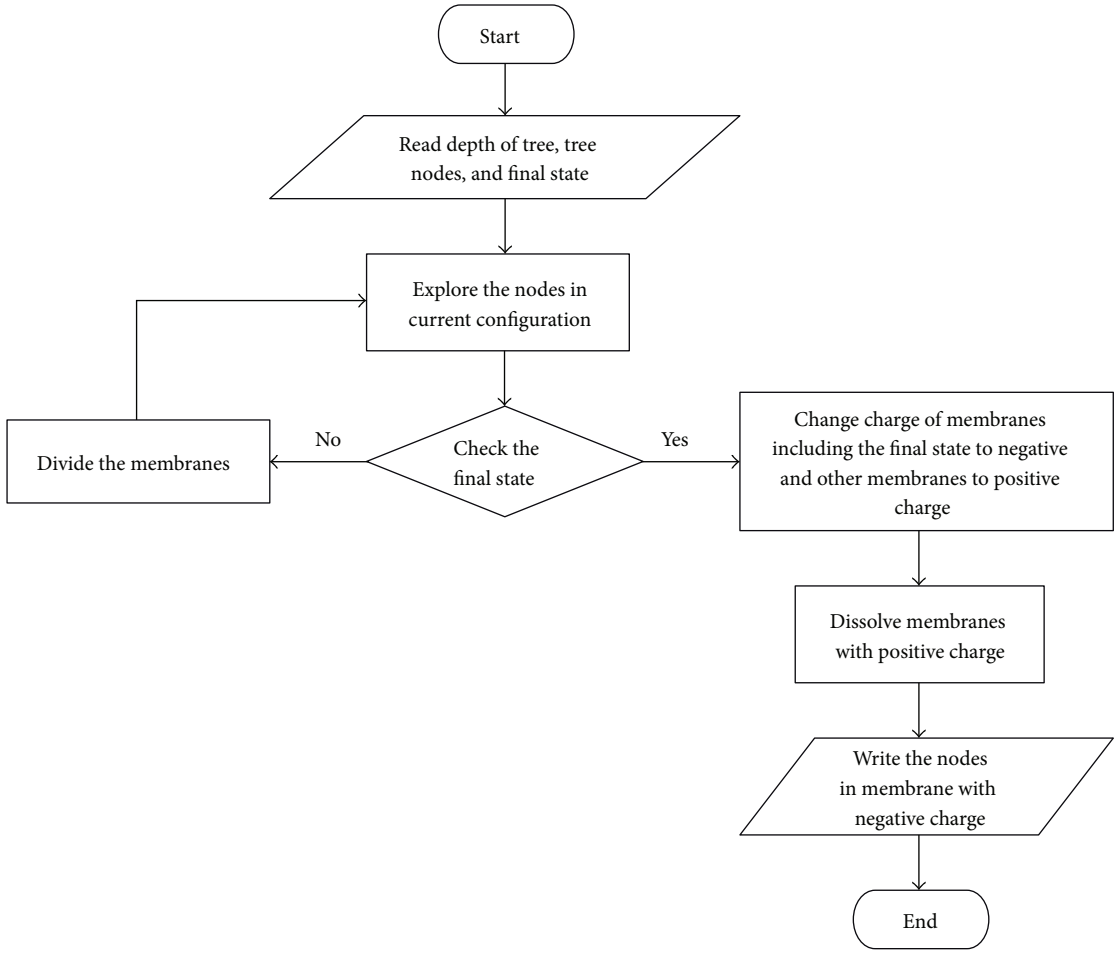

Figure 3: Logical scheme.

We also define the P system structure as follows.

We will consider P system $\Pi = (V, H, \Gamma, \mu, w_0, w_1, R_1, R_{2,1}, R_{2,2}, R_{3,1}, R_{3,2}, R_{4,1}, R_{4,2}, R_5, R_6, 1)$ with dynamic output $i_0 = 1$ and with the strong priorities $R_{3.1} > R_{4.1} > R_5$, where

(i) the working alphabet $V = \{k, t, y, z\} \cup \{a_j, b_j, c_j \mid 1 \leq j \leq 2^{n+1} - 1\}$;

(ii) the output alphabet $\Gamma = \{a_j \mid 1 \leq j \leq 2^{n+1} - 1\}$;

(iii) the set of labels $H = \{0, 1\}$;

(iv) the initial membrane structure $\mu = [[\,]_1^0]_0^0$;

(v) the initial multisets $w_0 =$ and $w_1 = \{c_1\}$.

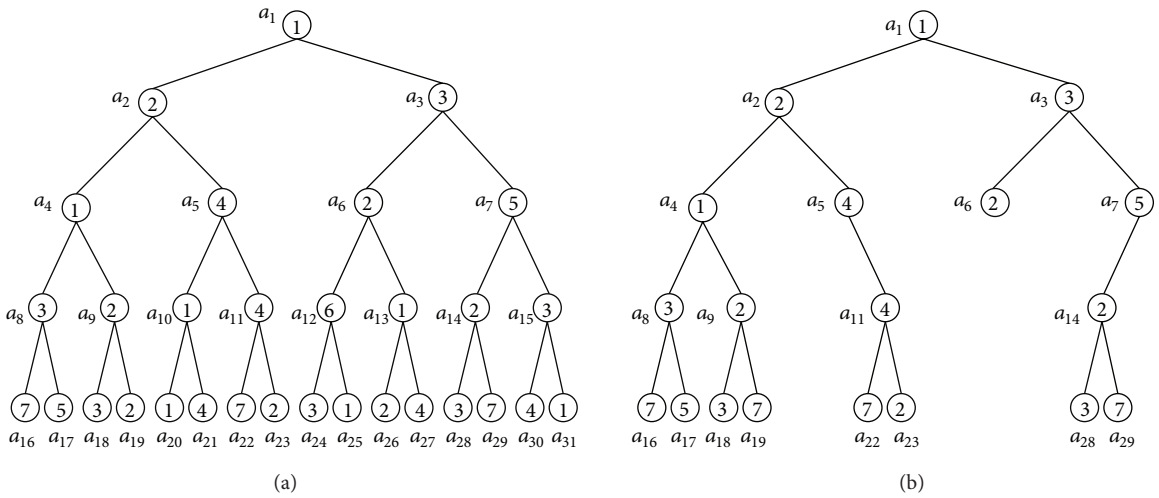

Figure 4: Binary random trees with positive integer weights.

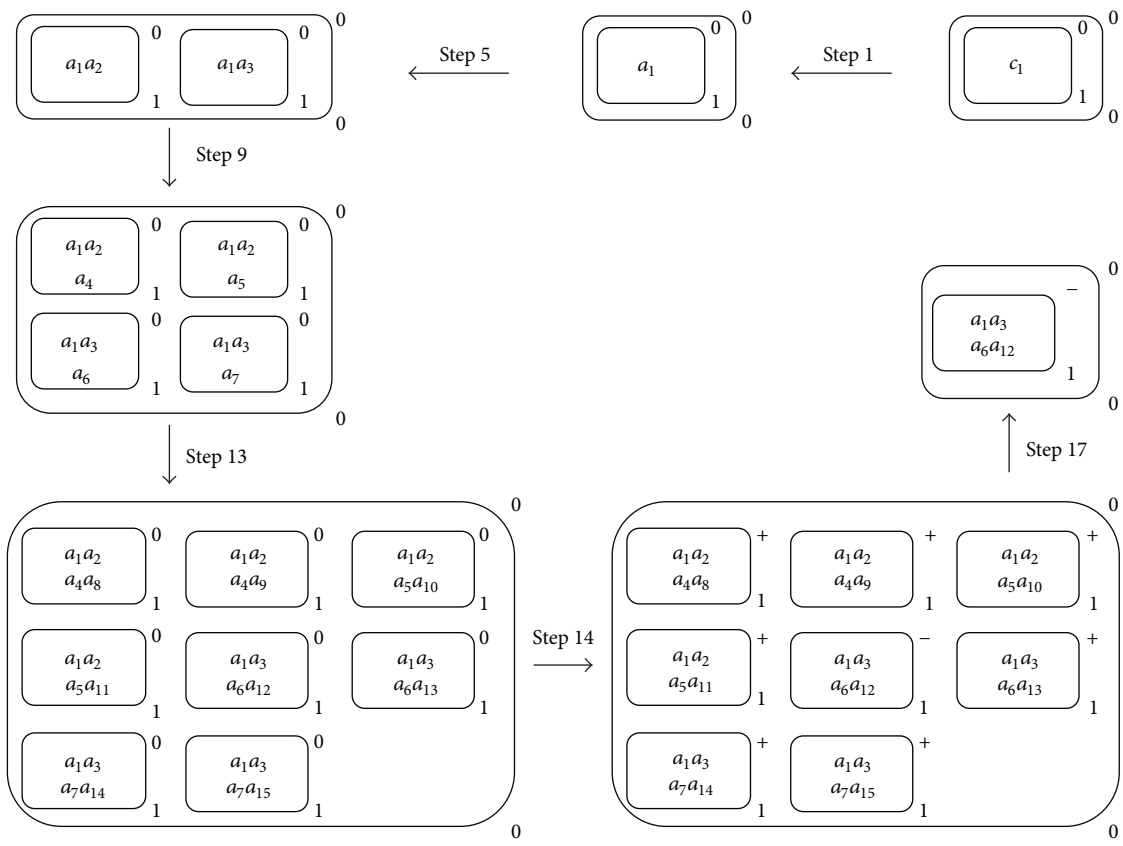

Figure 5: An overview of proposed P system process for searching number 6.

We also consider the set of rules R as follows:

$$R_1 \equiv \left[y \longrightarrow z^{2^n} \right]_0^0. \tag{2}$$

This rule produces 2^n copies of z by y in membrane 0 (skin);

$$R_2 \equiv \begin{cases} (1) & z[\,]_1^0 \longrightarrow [z]_1^+ \\ (2) & [z]_1^+ \longrightarrow \lambda. \end{cases} \tag{3}$$

Using these rules, the membrane with label 1 is recharged and dissolved; these rules are applied when the answer is obtained, and computation halts

$$R_3 \equiv \begin{cases} (1) & [t]_1^0 \longrightarrow t[\,]_1^+ \\ (2) & [t \longrightarrow \lambda]_1^+. \end{cases} \tag{4}$$

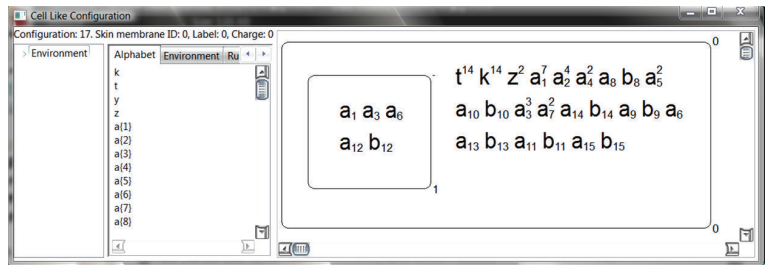

Figure 6: The final configuration of the search tree by P-Lingua simulator of the proposed method for finding number 6 and its path from start (root) until goal (number 6) located in membrane with label 1 and neutral charge.

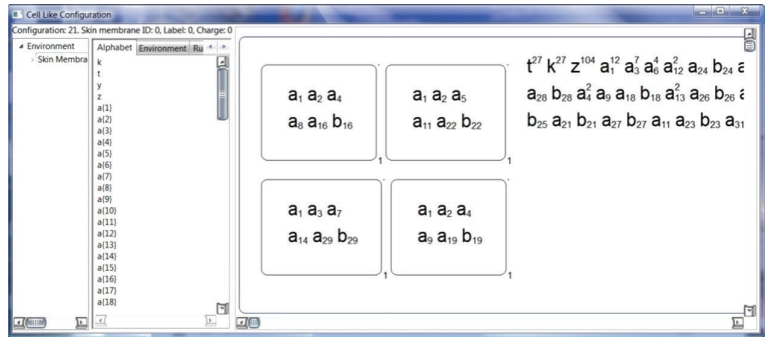

Figure 7: The final configuration of the search tree by P-Lingua simulator of the proposed method for finding number 7 and its paths from start (root) until goals (number 7) located in membranes with label 1 and neutral charge.

After applying these rules, a copy of t will be sent to outside and the charge of membrane is changed; after that, all copies of t disappeared, in membrane 1 with positive charge

$$R_4 \equiv \begin{cases} (1) & [k]_1^0 \longrightarrow y[\,]_1^- \\ (2) & [k]_1^+ \longrightarrow [\,]_1^0. \end{cases} \quad (5)$$

In first case, object k changes the charge of membrane 1 from neutral to negative and sends object y out of membrane; at the same time, k disappeared. This rule will be applied when there is no any object of t in membrane 1 with neutral charge (see R_3), and it means that the answer is here (see R_6). In the second case, k changed the charge of membrane 1 from positive to neutral, and the process is ready to continue

$$R_5 \equiv \left[b_j\right]_1^0 \longrightarrow \left[c_{2j}\right]_1^0\left[c_{2j+1}\right]_1^0, \quad 1 \le j \le 2^n - 1. \quad (6)$$

Object b_j divides membrane 1 and sends c_{2j} and c_{2j+1} to core of each new membrane, that they produce the children of current parent's node in the next step. If node j is nil, then $y_j = 0$, and, in the next step, node a_j will not be produced. If node a_j is not nil, then $y_j = 1$, and, in next step, node a_j will be produced:

$$R_6 \equiv \left[c_j \longrightarrow t^{(G-x_j)^2} a_j \left(b_j\right)^{y_j} k\right]_1^0, \quad 1 \le j \le 2^{n+1} - 1. \quad (7)$$

From c_j, rules R_6 makes a_j, $(b_j)^{y_j}$, k, and some copies of t, among them, a_j is a current node. If $y_j = 1$, then we have one copy b_j, and it makes the properties of a_j's children in the next step (see R_5). If $y_j = 0$, then the computation in this branch will be stopped. The appearance of t means that the goal (G) is not yet reached; therefore, rule R_3 is applied and the process will continue. If there is any copy of t, it means that the goal is reached; therefore, rule $R_{4.1}$ is applied and the process will be stopped after applying rules R_1 and R_2.

Remark 4. We denote the P system Π configuration in step i by C_i, and C_0 is initial configuration.

Remark 5. We consider a path from root until node a_{i_n} in a binary tree with depth, by $a_{i_0} a_{i_1} \ldots a_{i_m} a_{i_{(m+1)}} \ldots a_{i_n} = A_{i_n}$, where $2^m \le i_m \le 2^{m+1} - 1$, $0 \le m \le n$, $i_{(m+1)} = 2(i_m)$ or $i_{(m+1)} = 2(i_m) + 1$.

Lemma 6. *Let be a nonnegative integer number, and be natural numbers, and let $y_j = 1$ and $G \neq x_j$, where $2^i \le j \le 2^{i+1} - 1$ and $0 \le i \le n$. If Π is the P system mentioned above, then there exist 2^i membranes with label 1 and neutral charge including a multiset in the form $t^{(G-x_j)^2} A_j b_j k$ in configuration C_{4i+1}.*

Proof. We will prove the lemma by induction on i.

Case 1 ($i = 0$). In initial configuration (C_0), there exists only object c_1 in P system that is located inside membrane 1 with neutral charge. Therefore, in step one, the rules of type R_6 will be applied, object c_1 is changed into $t^{(G-x_1)^2} a_1 b_1 k = t^{(G-x_1)^2} A_1 b_1 k$, and thus the proposition is true for $i = 0$.

Case 2 ($i < n \rightarrow i + 1$). Let i be such that $0 \le i < n$, by inductive hypothesis, there exist 2^i membranes with

label 1 and neutral charge including a multiset in the form $t^{(G-x_j)^2} A_j b_j k$ in configuration C_{4i+1}. In this configuration, if $(G - x_j)^2 = 1$, then rule $R_{3.1}$ will be applied (rules $R_{3.1}, R_{4.1}$, and R_5 can be applied on objects t, k, and b_j, respectively, but with this order $R_{3.1} > R_{4.1} > R_5$) in step $4i + 2$, and this rule changes the polarization of membranes 1 from neutral to positive. After that, in step $4i + 3$, rule $R_{4.2}$ will be applied, and it changes the polarization of membranes 1 from positive to neutral. If $(G - x_j)^2 > 1$, then rule $R_{3.1}$ will be applied in step $4i + 2$; after that, rules $R_{3.2}$ and $R_{4.2}$ will be applied simultaneously (see Section 3, principal 4) and C_{4i+3} will be obtained. In both conditions, k and t will disappear and we obtain 2^i membranes with label 1 and neutral charge, and each membrane includes a multiset in the form $A_j b_j$. Now rules R_5 is applied in step $4i+4$ and each membrane will be duplicated; half of them include $A_j c_{2j}(1)$, and the remaining include $A_j c_{2j+1}(2)$. Therefore, we have 2^{i+1} membranes with label 1 and neutral charge; each of them includes one multiset such as (1) or (2). In step $4i+5$, rules R_6 will be applied and multisets $A_j c_{2j}$ and $A_j c_{2j+1}$ will be changed to $t^{(G-x_{2j})} A_j a_{2j} b_{2j} k$ and $t^{(G-x_{2j+1})} A_j a_{2j+1} b_{2j+1} k$, respectively. On the other hand, we know that $A_j a_{2j} = A_{2j}$ and $A_j a_{2j+1} = A_{2j+1}$; then, in step $4i + 5 = 4(i + 1) + 1$, we have 2^{i+1} membranes with label 1 and neutral charge, and each membrane includes one multiset in the form $t^{(G-x_J)} A_J b_J k$, where $2^{i+1} \leq J \leq 2^{i+2} - 1$. Thus, the proposition is true for all numbers i, where $0 \leq i \leq n$, and, in configuration C_{4i+1}, there exist 2^i membranes with label 1 and neutral charge, and each membrane includes a multiset in the form $t^{(G-x_j)^2} A_j b_j k$. □

Note. According to Lemma 6, if there is not any goal, computations will continue to depth n.

Lemma 7. *If the assumptions to Lemma 6 are satisfied, then for each j, there exists a membrane with label 1 and neutral charge, including A_j in configuration C_{4i+1}.*

Proof. It is evident according to the definition of A_j, there exist 2^i copies of A_j ($2^i \leq j \leq 2^{i+1} - 1, 1 \leq i \leq n$); also according to Lemma 6, in configuration C_{4i+1} there exist 2^i membranes with label 1 and neutral charge, including a multiset in the form $t^{(G-x_j)^2} A_j b_j k$, and thus the lemma holds for each j. □

Theorem 8. *The BFS can be implemented on a binary tree by a P system with division rule in a linear time.*

Proof. Let the goal be located in the nearest depth d; it means there exists at least a j, $2^d \leq j \leq 2^{d+1} - 1$ such that $G = x_j$ and $G \neq x_j$, for all j ($2^i \leq j \leq 2^{i+1} - 1, i < d$). We consider a P system and a complete binary tree ($y_i = 1, 1 \leq i \leq 2^{n+1} - 1$) according to assumptions presented in Lemma 6 and A_j according to Remark 5 (A_j is the path from root until node a_j). By Lemmas 6 and 7 the whole of paths from root until depth d are explored in configuration C_{4d+1}. Hence, in step $4d + 1$, one of membranes (perhaps more than one) with neutral charge includes just $A_j b_j k$ (there is no t; it means that the goal is here and A_j shows the path from root until goal). We consider this membrane with label $1'$. Next, in step $4d+2$, for membrane $1'$, rule $R_{4.1}$ and for membranes 1, rule $R_{3.1}$ will be applied simultaneously. In this case, object k changes the polarization of membrane $1'$, and it sends an object y out of the membranes and disappears, simultaneously. In this moment, membrane $1'$ has the negative polarization, so any rules cannot be applied on it. Also in step $4d + 2$, object t changes the polarization of membranes 1 from neutral to positive charge. In step $4d + 3$, each object y evolves to z^{2^n} in membrane 0 by rule R_1, and, at the same time, object t will be dissolved in membranes 1 by rule $R_{3.2}$, and object k will be sent out of membranes 1 and changes the polarization of membrane from positive to neutral charge by rule $R_{4.2}$. The rules apply maximally parallel manner. In step $4d + 4$, one copy of object z that have been produced in step $4d + 3$ will be sent to each membrane with label 1, and the polarization will be changed from neutral to positive by applying rule $R_{2.1}$. In step $4d + 5$ (the last step), object z dissolves membranes 1, and the computation will be halted. Finally, membrane (perhaps some membrane) with label 1 ($1'$) will be remained with negative charge that shows the place of the goal by the path (multiset $A_j b_j$) that is located in each membrane. Therefore, the presented P system explores a complete binary tree according to BFS method in a linear time. It is clear that even if the tree is not complete, the theorem is true. □

6. Experimental Simulation

We will give some hints on the computation by following the computation of the examples that were shown in Figures 4(a) and 4(b).

Example 9. Let us find number 6 in the tree according to Figure 4(a). Therefore $n = 4$, $G = 6$, and you can see the weights on the tree. In configuration C_0, there is just one object c_1 that occurs in membrane 1; therefore, for the first time, rules R_6 is applied and we achieve $t^{25} a_1 b_1 k$ in membrane 1 for configuration C_1. Next, rule $R_{3.1}$ and after that $R_{3.2}$ and $R_{4.2}$ are applied ($R_{3.2}$ and $R_{4.2}$ are applied simultaneously), objects t and k will be dissolved (obtaining C_3), and the system will be ready to continue; then we can apply rules R_5 (obtaining C_4) in the next step. This is the first time that we divide membrane 1, so there are two membranes with label 1 in C_4 in which one of them contains $a_1 c_2$ and another one contains $a_1 c_3$. In step (5), rules R_6 can be applied and multisets are changed to $t^{16} a_1 a_2 b_2 k$ and $t^9 a_1 a_3 b_3 k$.

Now the computation will be repeated as above. Therefore, in C_{13}, we have eight membranes with label 1 and neutral charge that each one contains one of the multisets $t^9 a_1 a_2 a_4 a_8 b_8 k$, $t^{16} a_1 a_2 a_4 a_9 b_9 k$, $t^{25} a_1 a_2 a_5 a_{10} b_{10} k$, $t^4 a_1 a_2 a_5 a_{11} b_{11} k$, $a_1 a_3 a_6 a_{12} b_{12} k$, $t^{25} a_1 a_3 a_6 a_{13} b_{13} k$, $t^{16} a_1 a_3 a_7 a_{14} b_{14} k$, or $t^9 a_1 a_3 a_7 a_{15} b_{15} k$. In the next step, two rules are applied at the same time, $R_{3.1}$ and $R_{4.1}$, and C_{14} is obtained. Rule $R_{4.1}$ is applied on membrane that contains multiset $a_1 a_3 a_6 a_{12} b_{12} k$ (goal is here); this rule makes k disappear, makes one copy of

y in region 0, and changes the polarization of the membranes from neutral to negative charge, but $R_{3.1}$ is applied on other membranes and changes their charge from neutral to positive charge. C_{15} will be obtained after applying rules R_1, $R_{3.2}$, and $R_{4.2}$, simultaneously. In this case, objects t are dissolved by $R_{3.2}$, and, by applying rule $R_{4.2}$ on the whole membranes, the charges will be changed from positive to neutral, except the membrane containing multiset $a_1a_3a_6a_{12}b_{12}$ and has negative charge; also there will be 2^4 copies of z by applying rule R_1. After applying rules $R_{2.1}$ and $R_{2.2}$, all of the membranes will be dissolved, except for the membrane containing $a_1a_3a_6a_{12}b_{12}$ with negative charge; we achieve C_{17}, and computation is halted. It can be seen that the path for searching from start (root) to the goal in the membrane remained $a_1a_3a_6a_{12}$. In addition, it can be seen also that the brief of computation in Figure 5 shows paths' multisets.

For verification, we run the program in an updated version of the P-Lingua [46]. The final configuration (configuration 17) is shown by P-Lingua simulator in Figure 6.

Example 10. Let us find number 7 in the tree shown in Figure 4(b). Therefore, $n = 4, G = 7$, and you can see the weights on the tree. This tree is not complete and there are 4 goals in depth 4, the proposed method can find all of the goals. The simulation is carried out using P-Lingua shown in Figure 7.

7. Conclusion and Future Works

Due to the high computational cost, many classical methods have exponential time complexity. Since breadth-first search is one of these cases, for searching solution to real-life hard problems, it is very important to decrease time complexity. The proposed method has been successful in decreasing the time complexity.

Using P system with membrane division, this study presents an advanced approach to find a solution to the problem of breadth-first search. An exponential workspace has been created to decrease the time complexity from an exponential to a linear time. For the future studies, it is recommended to improve the design from the P system aspect that may include new ingredients and polynomial or linear uniform. Another recommendation is to conduct new case studies that are more connected with real-life problems. A study can be carried out on the implementation of this design on one parallel architecture and compare the results with the obtained ones from a sequential architecture.

Conflict of Interests

The authors confirm that there is no conflict of interest between the authors and the P-Lingua company, which has been used for programming the algorithms in the paper. The reason that the name of the programming code has been mentioned is that the authors would like to help the readers to compare their future studies with this manuscript. The authors also confirm that there is no conflict of interests, either financially or intellectually, that may bias the revision of this manuscript.

Acknowledgments

This work is supported by The Ministry of Higher Education (MOHE) under Research University Grant (RUG 03H72). The authors would like to thank Research Management Centre (RMC), Universiti Teknologi Malaysia (UTM) for the support in R&D and Soft Computing Research Group (SCRG) for the inspiration in making this study a success. The authors would also like to thank the anonymous reviewers who have contributed enormously to this work.

References

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

[2] 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.

[3] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta Informaticae*, vol. 71, no. 2-3, pp. 279–308, 2006.

[4] 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.

[5] X. Zhang, S. Wang, Y. Niu, and L. Pan, "Tissue P systems with cell separation: attacking the partition problem," *Science China Information Sciences*, vol. 54, no. 2, pp. 293–304, 2011.

[6] L. Pan and A. Alhazov, "Solving HPP and SAT by P systems with active membranes and separation rules," *Acta Informatica*, vol. 43, no. 2, pp. 131–145, 2006.

[7] L. Pan and T. O. Ishdorj, "P systems with active membranes and separation rules," *Journal of Universal Computer Science*, vol. 10, no. 5, pp. 630–649, 2004.

[8] G. Păun, "P systems with active membranes: attacking NP-complete problems," *Journal of Automata, Languages and Combinatorics*, vol. 6, no. 1, pp. 75–90, 2001.

[9] S. N. Krishna and R. Rama, "A variant of P systems with active membranes: solving NP-complete problems," *Romanian Journal of Information Science and Technology*, vol. 2, pp. 357–367, 1999.

[10] M. Mutyam and K. Krithivasan, "P systems with membrane creation: universality and efficiency," in *Machines, Computations, and Universality (Chişinău, 2001)*, M. Margenstern and Y. Rogozhin, Eds., vol. 2055, pp. 276–287, Springer, Berlin, Germany, 2001.

[11] L. Pan and G. Păun, "Spiking neural P systems: an improved normal form," *Theoretical Computer Science*, vol. 411, no. 6, pp. 906–918, 2010.

[12] L. Pan, X. Zeng, and X. Zhang, "Time-free spiking neural P systems," *Neural Computation*, vol. 23, no. 5, pp. 1320–1342, 2011.

[13] L. Pan and X. Zeng, "Small universal spiking neural P systems working in exhaustive mode," *IEEE Transactions on NanoBioscience*, vol. 10, no. 2, pp. 99–105, 2011.

[14] X. Zhang, Y. Jiang, and L. Pan, "Small universal spiking neural P systems with exhaustive use of rules," in *Proceedings of the 3rd International Conference on Bio-Inspired Computing: Theories and Applications (BICTA '08)*, pp. 117–127, October 2008.

[15] Y. Niu, L. Pan, M. J. Pérez-Jiménez, and M. R. Font, "A tissue P systems based uniform solution to tripartite matching problem," *Fundamenta Informaticae*, vol. 109, no. 2, pp. 179–188, 2011.

[16] L. Pan, G. Păun, and M. J. Pérez-Jiménez, "Spiking neural P systems with neuron division and budding," *Science China Information Sciences*, vol. 54, no. 8, pp. 1596–1607, 2011.

[17] A. Păun, "On P systems with active membranes," in *Unconventional Models of Computation, UMC'2K*, I. Antoniou, C. S. Calude, and M. J. Dinneen, Eds., pp. 187–201, Springer, London, UK, 2001.

[18] M. Pérez-Jiménez and A. Riscos-Núñez, "A linear-time solution to the knapsack problem using P systems with active membranes," in *Membrane Computing*, C. Martín-Vide, G. Mauri, G. Păun, G. Rozenberg, and A. Salomaa, Eds., vol. 2933, pp. 250–268, Springer, Berlin, Germany, 2004.

[19] G. Păun, Y. Suzuki, H. Tanaka, and T. Yokomori, "On the power of membrane division in P systems," *Theoretical Computer Science*, vol. 324, no. 1, pp. 61–85, 2004.

[20] A. E. Porreca, A. Leporati, G. Mauri, and C. Zandron, "P systems with active membranes: trading time for space," *Natural Computing*, vol. 10, no. 1, pp. 167–182, 2011.

[21] A. Alhazov, C. Martín-Vide, and L. Pan, "Solving graph problems by P systems with restricted elementary active membranes," in *Aspects of Molecular Computing*, N. Jonoska, G. Păun, and G. Rozenberg, Eds., vol. 2950 of *Lecture Notes in Computer Science*, pp. 1–22, Springer, Berlin, Germany, 2004.

[22] D. Díaz-Pernil, A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "Solving subset sum in linear time by using tissue P systems with cell division," in *Bio-Inspired Modeling of Cognitive Tasks*, J. Mira and J. Álvarez, Eds., vol. 4527, pp. 170–179, Springer, Berlin, Germany, 2007.

[23] D. Díaz-Pernil, M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "A uniform family of tissue P systems with cell division solving 3-COL in a linear time," *Theoretical Computer Science*, vol. 404, no. 1-2, pp. 76–87, 2008.

[24] D. Diaz-Pernil, M. J. Pérez-Jiménez, A. Riscos-Núñez, and A. Romero-Jiménez, "Computational efficiency of cellular division in tissue-like membrane systems," *Romanian Journal of Information Science and Technology*, vol. 11, pp. 229–241, 2008.

[25] A. E. Porreca, N. Murphy, and M. J. Pérez-Jiménez, "An optimal frontier of the efficiency of tissue P systems with cell division," in *Proceedings of the 10th Brainstorming Week on Membrane Computing*, pp. 141–166, Seville, Spain, 2012.

[26] P. Sosík, "The computational power of cell division in P systems: beating down parallel computers?" *Natural Computing*, vol. 2, no. 3, pp. 287–298, 2003.

[27] M. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and F. J. Romero-Campero, "A linear solution for QSAT with membrane creation," in *Membrane Computing*, R. Freund, G. Păun, G. Rozenberg, and A. Salomaa, Eds., vol. 3850, pp. 241–252, Springer, Berlin, Germany, 2006.

[28] A. Alhazov, L. Pan, and C. Martín-Vide, "Solving a PSPACE-complete problem by recognizing P systems with restricted active membranes," *Fundamenta Informaticae*, vol. 58, no. 2, pp. 67–77, 2003.

[29] T. O. Ishdorj, A. Leporati, L. Pan, X. Zeng, and X. Zhang, "Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources," *Theoretical Computer Science*, vol. 411, no. 25, pp. 2345–2358, 2010.

[30] F. G. C. Cabarle, H. Adorna, and M. A. Martínez-del-Amor, "A spiking neural P system simulator based on CUDA," in *Membrane Computing*, M. Gheorghe, G. Păun, G. Rozenberg, A. Salomaa, and S. Verlan, Eds., vol. 7184, pp. 87–103, Springer, Berlin, Germany, 2012.

[31] F. Cabarle, H. Adorna, M. A. Martínez-del-Amor, and M. J. Pérez-Jiménez, "Spiking neural P system simulations on a high performance GPU platform," in *Algorithms and Architectures for Parallel Processing*, Y. Xiang, A. Cuzzocrea, M. Hobbs, and W. Zhou, Eds., vol. 7017, pp. 99–108, Springer, Berlin, Germany, 2011.

[32] J. Cecilia, J. M. García, G. D. Guerrero, M. A. Martínez-del-Amor, M. J. Pérez-Jiménez, and M. Ujaldan, "The GPU on the simulation of cellular computing models," *Soft Computing*, vol. 16, no. 2, pp. 231–246, 2012.

[33] J. M. Cecilia, G. D. Guerrero, J. M. García, M. A. Martínez-del-Amor, I. Pérez-Hurtado, and M. J. Pérez-Jiménez, "Simulation of P systems with active membranes on CUDA," in *Proceedings of the International Workshop on High Performance Computational Systems Biology (HIBI '09)*, pp. 61–70, October 2009.

[34] G. D. Guerrero, J. M. Cecilia, J. M. García, M. A. Martínez-del-Amor, I. Pérez-Hurtado, and M. J. Pérez-Jiménez, "Analysis of P systems simulation on CUDA," in *XX Jornadas de Paralelismo*, pp. 289–294, A Coruña, Spain, 2009.

[35] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, NJ, USA, 2nd edition, 2002.

[36] Y. Li, Y. Wang, and L. Bao, "FACC: a novel finite automaton based on cloud computing for the multiple longest common subsequences search," *Mathematical Problems in Engineering*, vol. 2012, Article ID 310328, 17 pages, 2012.

[37] W. Cheng, P. Guo, Z. Zhang, M. Zeng, and J. Liang, "Variable neighborhood search for parallel machines scheduling problem with step deteriorating jobs," *Mathematical Problems in Engineering*, vol. 2012, Article ID 928312, 20 pages, 2012.

[38] G. Liu, H. Peng, L. Li, Y. Yang, and Q. Luo, "Improved degree search algorithms in unstructured P2P networks," *Mathematical Problems in Engineering*, vol. 2012, Article ID 923023, 18 pages, 2012.

[39] B.-Y. Shih, C.-W. Chen, C.-Y. Chen, and T.-W. Lo, "Merged search algorithms for radio frequency identification anticollision," *Mathematical Problems in Engineering*, vol. 2012, Article ID 609035, 20 pages, 2012.

[40] M. Gutiérrez-Naranjo and M. Pérez-Jiménez, "Depth-first search with P systems," in *Membrane Computing*, M. Gheorghe, T. Hinze, G. Păun, G. Rozenberg, and A. Salomaa, Eds., vol. 6501, pp. 257–264, Springer, Berlin, Germany, 2011.

[41] M. A. Gutiérrez-Naranjo and M. J. Pérez-Jiménez, "Implementing local search with membrane computing," in *Proceedings of the 9th Brainstorming Week on Membrane Computing*, pp. 159–168, Seville, Spain, 2011.

[42] M. A. Gutiérrez-Naranjo, M. A. Martinez del Amor, I. Pérez-Hurtado, and M. J. Pérez-Jiménez, "Solving the N-queens puzzle with P systems," in *Proceedings of the 7th Brainstorming Week on Membrane Computing*, pp. 199–210, Sevilla, Spain, 2009.

[43] M. J. Dinneen, Y.-B. Kim, and R. Nicolescu, "Edge- and node-disjoint paths in P systems," *Electronic Proceedings in Theoretical Computer Science*, vol. 40, pp. 121–141, 2010.

[44] R. Nicolescu and H. Wu, "BFS solutions for disjoint paths in P systems," in *Unconventional Computation*, C. S. Calude, J. Kari, I. Petre, and G. Rozenberg, Eds., vol. 6714 of *Lecture Notes in Computer Science*, pp. 164–176, Springer, Berlin, Germany, 2011.

[45] R. Nicolescu and H. Wu, "New solutions for disjoint paths in P systems," *Natural Computing*, vol. 11, pp. 637–651, 2012.

[46] D. Díaz-Pernil, I. Pérez-Hurtado, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "A P-lingua programming environment for membrane computing," in *Membrane Computing*, D. Corne, P. Frisco, G. Păun, G. Rozenberg, and A. Salomaa, Eds., vol. 5391, pp. 187–203, Springer, Berlin, Germany, 2009.