Solving the Subset Sum Problem Using Distributed Tissue-like P Systems with Cell Division

Justin Granda and Samuel Jose and Kelvin Cui Buño
Department of Computer Science (Algortihms & Complexity)
University of the Philippines Diliman
Diliman 1101 Quezon City, Philippines
justingranda@gmail.com,gsamueljose@gmail.com,kcbuno@up.edu.ph

ABSTRACT

The Subset Sum Problem is a decision problem where given a multiset of integers, a decision must be made on whether a subset of said set can be found where the sum of its elements is equal to a target value, or not. This problem is NP-Complete. Membrane computing is one of the ways used to approach these problems, using a computing model commonly referred to as P systems. In this work, we solve the Subset Sum Problem using dP systems where the components are tissue P systems with cell division. The 2-component solution proposed can generate candidate solutions twice as fast, as compared to the non-distributed solution it was based on. However, computation time is increased with respect to the target sum. Communication costs are analyzed and measured.
KEYWORDS: Membrane computing; dP systems; dP schemes; Tissue P systems; Subset-sum problem

1 INTRODUCTION

Membrane computing, as initiated in [10], involves the use of computational models that are inspired by biological structures and processes. These models are commonly referred to as P systems. This type of system usually follows a membrane structure where each membrane can contain objects, rules, and even other membranes. P systems are able to carry out computations by moving objects around the membranes with the use of evolution rules, and when a halting configuration is reached, the output is read from a predetermined output region. There are several variants of P systems, each with their own advantages and disadvantages in approaching different problems. Some of these variants can be seen in [5, 9, 12].

In tissue P systems, several systems known as "cells" communicate with each other through the use of symport/antiport rules in order to carry out a computation. In addition, the cells in this system have the ability to perform cell division. In this process a cell is able to multiply, where each new cell can be seen as an evolution of the previous cell. This leads to the possibility to create an exponential number of cells in linear time, and this space can be used in order to obtain polynomial-time solutions to computationally hard problems. There have been several studies shown solving NP-complete problems with the use of these systems, as seen in [4].

As stated above, tissue P systems with cell division have been used in creating solutions to NP-complete problems. One example of an NP-complete problem is the Subset Sum Problem. The problem involves having a finite set of integers, and finding its subset whose elements sum up to a specific value. What makes the Subset Sum Problem computationally difficult to solve is that as the input set grows, the number of possible subsets grows exponentially. When provided a sufficiently large enough input, looking through all candidate solutions would take an exponential amount of time. Several solutions to this problem using other types of P systems already exist as presented in [6, 7], where the solution has been improved to run in polynomial time.

Described by [13], distributed P systems(also known as dP systems) approach problems in a distributed manner. dP systems uses P systems as components, wherein each P system receives a partition of the input and performs computations using the inputs given to each of them. In addition to this, the components often communicate with each other through the use of certain rules that allow the exchange of objects or variables. The communication of these components can be measured in various ways, such as the number of rules used in a halting computation. Solutions to other NP-complete problems using dP systems are presented in [1, 2].

In this paper we present our dP system solution to the Subset Sum Problem, where the component P systems are tissue P systems with cell division. Our proposed solution builds on the P system solution presented in [3]. Given an input instance of the Subset Sum Problem, each component system receives a copy of the target value and a partition of the multiset. An analysis of the computation time compared to non-distributed solutions, as well as communication costs, are provided.

The rest of this work is organized as follows. Section 2 defines the terms used in this work. In Section 3, we present our solution to the Subset Sum Problem. In Section 4, we measure and analyze computation and communication complexity of our solution, and compare it with a non-distributed solution. Section 5 provides the summary of the findings of our work and gives some insight on future works.

2 PRELIMINARIES

The reader is assumed to be familiar with the fundamentals of formal language theory. Let Σ be an alphabet, then Σ^* denotes the set of all finite length strings over Σ. The number of symbols in a string s is the length of the string, denoted by $|s|$. The empty string will be denoted as λ. A multiset M over Σ is a mapping from Σ to the set of nonnegative integers. Multisets are represented using strings over Σ.

Definition 1. *[13] Let P be a non-empty finite set. A collection of $\{P_1, \ldots, P_n\}$ is called a partition P if and only if for all $1 \leq i, j \leq n, i \neq j$, P_i and P_j are disjoint, and $\bigcup_{i=1}^{n} P_i = P$. A partition $\{P_1, P_2, \ldots, P_k\}$ is called a balanced partition if and only if for all i, P_i have the same size or at most have a difference of 1. Otherwise it is called an unbalanced partition.*

2.1 dP Scheme

In this section, the concept of a dP scheme and the communication complexity of P system are briefly presented, then (recognizer) tissue P systems with cell division, as well as distributed tissue P systems with cell division, are introduced.

Definition 2. *A dP scheme of a finite degree $n \geq 1$ is a construct of the form [13]:*

$$\Delta = (O, \Pi_1, \ldots, \Pi_n, R),$$

where:

1. *O is an alphabet of objects.*
2. *$\Pi_1, \ldots, \Pi_n$ are cell-like P systems with O as the alphabet of objects and the skin membranes labeled with $s_1, \ldots, s_n$, respectively.*
3. *R is a finite set of inter-component communication rules of the form $(s_i, u/v, s_j)$, where $1 \leq i, j \leq n, i \neq j$, and $i, j \in O^*$, with $uv \neq \lambda$; $|uv|$ is called the weight of the rule.*

According to [13], the systems $\Pi_1, \ldots, \Pi_n$ are called the components of Δ. Each component can take in some input and perform computations independently. The system accepts if all components end in a halting computation, that is no more rules can be applied in all components. Each component can also communicate symbols with other components as defined by the rules in R.

Rules are used as usual in the framework of membrane computing, that is, in a maximally parallel way. In one timestep, each object in a membrane can only be used for one rule (non-deterministically chosen when there are several possibilities), but any object which can participate in a rule of any form must do it, i.e, in each step a maximal set of rules must be applied.

A dP scheme computes in such a way that all its component P systems are aware of the problem they need to solve. Each component is assigned a partition of the input, and performs computations using the input given to them. The components may or may not differ from one another in terms of membrane structure, rules, and how they process the input. The concept of component "sameness" or "homogeneity" is based from homogeneous spiking neural P systems[14]. Homogeneity is applied on a component-level, rather than at the membrane/neuron-level, similar to the work done in [8].

Definition 3. *[8, 14] A dP scheme is said to have homogeneous components if and only if all the component P systems have the same initial membrane structure and same set of rules.*

Since individual components can work with a partitioned input encoded as objects with indices, the rules of the individual components may just vary with which range of indices they are working on. But the general process of each individual component is still the same.

The complexity measure we focus in studying dP schemes is the communication cost. The following defines the communication cost for a given computation step of the system:

Definition 4. *[13] Let Δ be a dP scheme, and $\delta : \delta_0 \Rightarrow \delta_1 \Rightarrow \ldots \Rightarrow \delta_h$ be a halting computation in Δ where δ_0 is the initial configuration and δ_h is a halting configuration, and R the set of inter-component communication rules, with each rule of the form $(s_i, u/v, s_j)$. Then for each $i = 0, 1, \ldots, h-1$, we have the following complexity measures:*

- *$ComN(\delta_i \Rightarrow \delta_{i+1}) = 1$, if at least one inter-component communication rule was used in this transition; 0 otherwise;*
- *$ComR(\delta_i \Rightarrow \delta_{i+1})$ is the number of inter-component communication rules used in this transition;*
- *$ComW(\delta_i \Rightarrow \delta_{i+1})$ is the sum of the weights of all inter-component communication rules used in this transition.*

Definition 5. *[13] Let the set of strings accepted by Δ be denoted as L(Δ). For ComX $\in$ {ComN,ComR,ComW}, we define:*

- *$ComX(\delta) = \sum_{i=0}^{h-1} ComX(\delta_i \Rightarrow \delta_i + 1)$, for δ which is a halting computation.*
- *$ComX(w, \Delta) = min\ \{ComX(\delta) \mid \delta$ is a computation of Δ that accepts the string $w\}$.*
- *$ComX(\Delta) = max\{ComX(w, \Delta) \mid w \in L(\Delta)\}$.*

The idea of parallelizability is introduced in [13], with respect to the communication measures described above.

DEFINITION 6. *[13] A language $L \subseteq V^*$ is said to be (n,m)-weakly ComX parallelizable, for some $n \geq 2, m \geq 1$, and $X \in \{N, R, W\}$, if there is a dP system with n components and there is a finite subset F_Δ of L such that each string $x \in L - F_\Delta$ can be written as $x = x_1 x_2 \ldots x_n$, each component Π_i of Δ takes as input the string x_i, $1 \leq i \leq n$, and the string x is accepted by Δ by a halting computation δ such that $ComX(\delta) \leq m$.*

A language L is said to be weakly ComX parallelizable if it is (n,m)-weakly ComX parallelizable for some $n \geq 2, m \geq 1$. A stronger version of parallelizability is also introduced.

2.2 Recognizing Tissue P Systems with Cell Division

In this subsection, we briefly introduce the notions of a (recognizer) tissue P system with cell division and how it performs computations.

DEFINITION 7. *Formally, a tissue P system with cell division of initial degree $q \geq 1$ is a tuple of the form [3]:*

$$\Pi = (\Gamma, w_1, \ldots, w_q, E, R, i_0),$$

where:

(1) *Γ is a finite alphabet, whose symbols will be called objects.*
(2) *$w_1, \ldots, w_q$ are strings over Γ, that describe the multisets of objects placed initially in the q cells of the system.*
(3) *$E \subseteq \Gamma$ is the set of objects placed in the environment, each one of them in an arbitrarily large amount of copies.*
(4) *R is a finite set of rules of the following form:*
    (a) *Communication rules: $(i, u/v, j)$, for $i, j \in \{0, 1, \ldots, q\}, i \neq j, u, v \in \Gamma^*$.*
    (b) *Division rules: $[a]_i \to [b]_i[c]_i$, where $i \in \{1, 2, \ldots, q\}$ and $a, b, c \in \Gamma$.*
(5) *$i_0 \in \{1, 2, \ldots, q\}$ is the output region.*

The main features of tissue P systems with cell division, from the computational point of view, are that cells obtained by division have the same labels as the original cell, and if a cell is divided, then its interaction with other cells or with the environment is blocked during the mitosis process. In some sense, this means that while a cell is dividing it closes the communication channels with other cells and with the environment. This features imply that the underlying graph is dynamic, as nodes can be added during the computation by division and the edges can be deleted/re-established for dividing cells.

The communication rule $(i, u/v, j)$ can be applied over two cells i and j such that u is contained in cell i and v is contained in cell j. The application of this rule means that the objects of the multisets represented by u and v are interchanged between the two cells.

The division rule $[a]_i \to [b]_i[c]_i$ can be applied over a cell i containing object a. The application of this rule divides this cell into two new cells with the same label. All the objects in the original cell are replicated and copied in each of the new cells, with the exception of the object a, which is replaced by the object b in the first new cell and by c in the second one.

Rules must be used in a maximally parallel way, but for this type of P system there is one other restriction: when a cell is divided, the division rule is the only one which is applied for that cell in that step; the objects inside that cell do not move in that step.

In this work, we define the representation of a configuration and computation of the component tissue P systems of our proposed dP scheme as the following:

DEFINITION 8. *A configuration δ_i of Π at time step i is a string over $\Sigma = \{[,]_h \mid h \in 1, \ldots, q\} \cup \Gamma$. The appearances of the character $[$ must be properly paired with $]_h$ in δ_i.*

A substring of δ_i of the form $x[u]_h y$, where $u, x, y \in \Sigma^*$, indicates that a membrane with label h contains u (possibly with other membranes as well). The initial configuration of Π is defined as δ_0. A halting of configuration of Π, denoted as δ_h is a configuration where no more rules can be applied.

DEFINITION 9. *A configuration δ_i yields a configuration δ_{i+1}, denoted as $\delta_i \Rightarrow \delta_{i+1}$, if and only if δ_{i+1} is obtained from δ_i by applying division and communication rules in a maximally parallel manner.*

DEFINITION 10. *A computation in Π is the transition of configurations represented by a sequence $\delta : \delta_0 \Rightarrow \delta_1 \Rightarrow \ldots \Rightarrow \delta_h$, where δ_0 is the initial configuration and δ_h is the final or halting configuration.*

To study computational efficiency, a class of tissue P system with cell division is introduced in [11]. It is formally defined as follows:

DEFINITION 11. *A recognising tissue P system with cell division of degree $q \geq 1$ is a tuple $\Pi = (\Gamma, \Sigma, w_1, \ldots, w_q, E, R, i_{in}, i_0)$ where [3]:*

- *Γ has two distinguished objects yes and no;*
- *$\Sigma \subseteq \Gamma$ is the input alphabet;*
- *$i_{in} \in \{1, \ldots, q\}$ is the input cell;*
- *i_0, the output cell, is the environment;*
- *all computations halt;*
- *either an object yes or no(but not both) must be released into the environment, and only in the last step of any computation.*

For an input $w \in \Sigma^*$, w is added to the initial multiset of $w_{i_{in}}$. The input w is said to be recognized by Π if and only if the object **yes** is sent to the environment in the last step of its associated halting computations. A halting computation is said to be accepting if the **yes** object is released to the environment, while a halting computation is said to be rejecting if the **no** object is released to the environment.

In this work, we propose a dP system with recognizer tissue P systems with cell division as components to solve the Subset Sum Problem, where the input multiset will be partitioned with respect to the component systems. We define a so-called distributed tissue P system will cell division as follows:

DEFINITION 12. *A k-Distributed Tissue P System with Cell Division, or k-DTP for short, is defined as follows:*

$$k\text{-}\Delta_{DTP} = (\Gamma, \Pi_1, \Pi_2, \ldots, \Pi_k, E, R_\Delta)$$

where:

- Γ *is the set of all objects in the system* Δ*;*
- $\Pi_1, \Pi_2, \ldots, \Pi_k$ *are recognizing tissue P systems with cell division. Each* Π_i *has the alphabet of objects in* Γ*. Each cell of* Π_i *will be labelled* (i, j)*, where* $j = 1, 2, \ldots, d_i$*, and* d_i *denotes the number of cells in* Π_i*;*
- E *is the shared environment of all* $\Pi_i, i = 1, 2, \ldots, k$
- R_Δ *is a finite set of inter-component communication rules of the form* $((i, x), u/v, (j, y))$*, for* $i, j \in \{1, 2, \ldots, k\}, i \neq j$*,* $x \in \{1, \ldots, d_i\}$*,* $y \in \{1, \ldots, d_j\}$*, and* $u, v \in \Gamma^*$*.*

The inter-component communication rule functions as follows. Given two component systems Π_i and Π_j, if a multiset of objects u is found in cell x in Π_i, and a multiset of objects v is found in cell y in Π_j, then the inter-component communication rule can be invoked, and the multiset u is transferred to cell y in Π_j while the multiset v is transferred to cell x in Π_i.

3 SOLVING SUBSET SUM USING 2-Δ_{DTP}

3.1 Subset Sum Problem

The Subset Sum Problem (or *SSP* for short), is defined as follows:

DEFINITION 13. *Given a finite set* $S \subseteq \mathbb{N}$*, a weight function* $w : X \mapsto \mathbb{N}$ *and a constant* $T \in \mathbb{N}$*, determine whether or not there exists a set* $B \subset X$ *such that* $w(B) = T$*.*

In this work, an instance of the problem will be represented as a tuple $(n, (w_1, \ldots, w_n), T)$, where n stands for the cardinality of set $X = \{x_1, \ldots, x_n\}$, $w_i = w(x_i)$ and T is the target value.

3.2 Solution

In this section, a 2-Δ_{dTP} is presented that solves any instance ϕ of SSP as defined earlier. An overview of the computation is also provided. The proposed dP system will use the encoding used in [3] and will solve in a similar manner modified in such a way to solve *SSP* in a distributed manner.

THEOREM 1. *Let* ϕ *be any instance of the SSP with* n *elements and* T *as the target. Then there exists a solution using* 2-Δ_{dTP} *under a balanced partition of* A*.*

Let $X = x_1, \ldots, x_n$ be a finite set, $w : X \rightarrow \mathbb{N}$ a weight function with $n = |X|$ and $T \in \mathbb{N}$. X is partitioned into X_1 and X_2 with $\{X_1, X_2\}$ a balanced partition. Let $n_1 = |X_1|$ and $n_2 = |X_2|$. Component Π_k will receive the input instance $u_k = (n_k, W_k, (w_{k,1}, \ldots, w_{k,n_k}), T)$ where $w_{k,i} = w(a_{k,i}), 1 \leq k \leq 2, 1 \leq i \leq n_k$ where $x_{k,i} \in X_k$, and W is the maximum between the sum of weights of A_1 and A_2.

For $k = 1, 2$, given the input instance $u_k = (n_k, W, (w_{k,1}, ..., w_{k,n_k}), T)$, the input alphabet $\Sigma_k = \{q\} \cup \{v_i \mid 1 \leq i \leq n_k\}$. For $X_k = \{x_1, ..., x_{n_k}\}$, the number of copies of v_i represents the weight value of the element x_i, for $1 \leq i \leq n_k$. The number of copies of q represents the target sum T. The input multiset that is added to the input region of each component Π_k is $\{\{v_i^j \mid j = w_{k,i}, 1 \leq i \leq n_k\}\} \cup \{\{q^T\}\}$.

We formally define a family of dP systems for solving the *SSP* as follows:

DEFINITION 14. *For every* $(n, T) \in \mathbb{N}^2$, 2-$\Delta_{dTP}(n, T) = (\Gamma, \Pi_1, \Pi_2, 0, R_\Delta)$*, is a construct where:*

- $\Gamma = \Gamma_1 \cup \Gamma_2$,
- *0 is the shared environment of* Π_1 *and* Π_2,
- Π_k*, for* $k = 1, 2$ *are recognizer tissue P systems with cell division defined as:*

$$\Pi_k = (\Gamma_k, \Sigma_k, w_{k,1}, w_{k,2}, E_k, R_k, i_{k_{in}} = (k, 2), i_{k_{out}} = 0),$$

where:

  - $\Sigma_k = \{q\} \cup \{v_i : 1 \leq i \leq n\}$
  - $w_{k,1} = S_k a_1 \bar{a}_1 b c_1 yes no$
  - $w_{k,2} = D A_1 \ldots A_{n_k}$
  - $\Gamma_k = \Sigma_k \cup \{A_i, B_i, : 1 \leq i \leq n_k\}$ $\cup \{a_i : 1 \leq i \leq n_k + \lceil \log_2 n_k \rceil + \lceil \log_2(T+1) \rceil + T + 11\}$ $\cup \{\bar{a}_i : 1 \leq i \leq n_k + \lceil \log_2 n_k \rceil + \lceil \log_2(T+1) \rceil + T + 11\}$ $\cup \{\beta_i : 0 \leq i \leq 2\}$ $\cup \{c_i : 1 \leq i \leq n_k + 1\}$ $\cup \{d_i : 1 \leq i \leq \lceil \log_2 n_k \rceil + \lceil \log_2(T+1) \rceil + 4\}$ $\cup \{e_i : 1 \leq i \leq \lceil \log_2 n_k \rceil + 1\}$ $\cup \{B_{i,j} : 1 \leq i \leq n_k \wedge 1 \leq j \leq \lceil \log_2(T+1) \rceil + 1\}$ $\cup \{s_i : 0 \leq i \leq T\}$ $\cup \{b, D, p, r, g_1, g_2, f_1, Y, S, N, \delta, \alpha, S_1, S_2, yes, no\}$
  - $E_k = \Gamma - \{yes, no\}$
  - R_k *is the set of rules of each* $k = 1, 2$ *component.*
    (1) *Division Rules:*
    $r_{1,i} \equiv [A_i]_2 \rightarrow [B_i]_2 [\lambda]_2$ *for* $i = 1, \ldots, n_k$

(2) *Communication Rules*
$r_{2a,i} \equiv (1, a_i/a_{i+1}^2, 0)$ *for* $i = 1, \ldots, n_k$
$r_{2\bar{a},i} \equiv (1, \bar{a}_i/\bar{a}_{i+1}, 0)$ *for* $i = 1, \ldots, n_k$
$r_{2b,i} \equiv (1, a_i/a_{i+1}, 0)$ *for* $i = n_k + 1, \ldots, \lceil \log_2 n_k \rceil + \lceil \log_2(T+1) \rceil + 10 + T$
$r_{2\bar{b},i} \equiv (1, \bar{a}_i/\bar{a}_{i+1}, 0)$ *for* $i = n_k + 1, \ldots, \lceil \log_2 n_k \rceil + \lceil \log_2(T+1) \rceil + 10 + T$
$r_{3,i} \equiv (1, bc_i/b^2c_{i+1}^2, 0)$ *for* $i = 1, \ldots, n_k$
$r_4 \equiv (1, c_{n+1}/D, 2)$
$r_5 \equiv (2, c_{n+1}/d_1e_1, 0)$
$r_{6,i} \equiv (2, e_i/e_{i+1}^2, 0)$ *for* $i = 1, \ldots, \lceil \log_2 n_k \rceil$
$r_{7,i} \equiv (2, d_i/d_{i+1}, 0)$ *for* $i = 1, \ldots, \lceil \log_2 n_k \rceil + \lceil \log_2(T+1) \rceil + 3$
$r_{8,i} \equiv (2, e_{\lceil \log_2 n_k \rceil + 1}B_i/B_{i1}, 0)$ *for* $i = 1, \ldots, n_k$
$r_{9,i,j} \equiv (2, B_{ij}/B_{ij+1}^2, 0)$ *for* $i = 1, \ldots, n_k$, $j = 1, \ldots, \lceil \log_2(T+1) \rceil$
$r_{10,i} \equiv (2, B_{i\lceil \log_2(T+1) \rceil + 1}v_i/pr, 0)$, *for* $i = 1, \ldots, n_k$
$r_{11} \equiv (2, pq/\lambda, 0)$
$r_{12} \equiv (2, d_{\lceil \log_2 n \rceil + \lceil \log_2(T+1) \rceil + 4}/g_1f_1, 0)$
$r_{13} \equiv (2, f_1p/\lambda, 0)$
$r_{14} \equiv (2, f_1q/\lambda, 0)$
$r_{15} \equiv (2, g_1/g_2s_0, 0)$
$r_{16} \equiv (2, g_2f_1/Y, 0)$
$r_{17} \equiv (2, Y/\lambda, 1)$
$r_{18} \equiv (1, bY/S, 0)$
$r_{19} \equiv (1, S_kSyes/\lambda, 0)$
$r_{20} \equiv (1, a_{n+\lceil \log_2 n \rceil + \lceil \log_2(T+1) \rceil + W + 11}b/N, 0)$
$r_{20a} \equiv (1, \bar{a}_{n+\lceil \log_2 n \rceil + \lceil \log_2(T+1) \rceil + W + 11}/\alpha, 0)$
$r_{21,i} \equiv (2, s_i/N, 1)$ *for* $i = 0, \ldots, W_k$
$r_{22,i} \equiv (2, rs_i/s_{i+1}, 0)$ *for* $i = 0, \ldots, W_k - 1$
$r_{23} \equiv (1, \alpha/\beta_0\delta, 0)$
$r_{24} \equiv (1, u\delta yes/\lambda, 0)$ *where* $u = S_2$ *if* $k = 1$, *else* $u = S_1$
$r_{25,i} \equiv (1, \beta_i/\beta_{i+1}, 0)$ *for* $i = 0, 1$
$r_{26} \equiv (1, \beta_2\delta no/\lambda, 0)$

- *R_Δ is the set of inter-component communication rules:* $r_c \equiv ((1,1), S_1s_x/S_2s_y, (2,1))$ *for* $x + y = T$

3.3 Overview of the Computation

First of all, we recall the solution to the SSP discussed in [3]. We will describe briefly how each stage works. For $k = 1, 2$, let $X_k = \{x_1, ..., x_{n_k}\}$ be the input set to Π_k, encoded as the input instance $u_k = (n_k, (w_{k,1}, ..., w_{k,n_k}), T)$.

In the *Generation Stage*, The input cell $(k, 2)$ is divided using the division rules $r_{1,i}$ which will produce 2^{n_k} cells, which represents all possible subsets of the input set. Each cell will contain some B_i objects, for $1 \leq i \leq n_k$. If a cell contains B_i, then it represents a subset that contains $x_i \in X_k$. For example, a cell containing B_1, B_3, B_5, represents the subset of $\{x_1, x_3, x_5\}$. Likewise this cell would also contain $v_1^{w(x_1)}$, $v_3^{w(x_3)}$, $v_5^{w(x_5)}$, representing the weight values of the elements x_1, x_3, x_5 respectively. This stage runs for n_k steps. For each time step i, $1 \leq i \leq n_k$ of the generation stage, using rule $r_{1,i}$, for $i = 1, ..., n$, A_i is used as a trigger to perform a division of cell $(k, 2)$, with one cell containing B_i, and the other cell does not.

In the *Pre-Checking Stage*, p and r objects are produced in each cell. The rule r_4 interchanges D and c_{n+1} between the cell $(k, 2)$ and $(k, 1)$. Rule r_5 then transforms c_{n+1} in each cells into d_1e_1. After the transition, $r_{6,i}$ and $r_{7,i}$ evolves d_1 and $2^{\lceil \log_2 n \rceil}$ copies of $e_{\lceil \log_2 n \rceil + 1}$ are produced. Using $r_{8,i}$, the $e_{\lceil \log_2 n \rceil + 1}$ and B_i pairs creates $B_{i,1}$. Rule $r_{9,i,j}$ creates $2^{\lceil \log_2(T+1) \rceil}$ copies of $B_{i,1+\lceil \log_2(T+1) \rceil}$ in $\lceil \log_2(T+1) \rceil$ steps. These $B_{i,1+\lceil \log_2(T+1) \rceil}$ objects, together with v_i, are exchanged for copies of p and r objects using rule $r_{10,i}$. The multiplicity of the p and r objects represents the weight of the subset represented by each cell. The p objects will be used to check if the weight of the subset is equal to the target sum, represented by the number of copies of q during the checking stage. In case the number of copies of p and q are not equal, the r objects will be used later to introduce the sum objects s_i, for $0 \leq i \leq T$ for the communication stage.

In the *Checking Stage*, using rule r_{11}, each cell $(k, 2)$ will remove pairs of p and q objects (sent to the environment) to check if the weight of the subset is equal to the target sum. The d_i objects, for $i = 1, ..., \lceil \log_2 n_k \rceil + \lceil \log_2(T+1) \rceil + 4$, acts as a counter to later on introduce the objects f_1 and g_1 using rule r_{12}. The object g_1 is exchanged for g_2 and s_0. For f_1, the following can occur:

- If the number of copies of p and q are equal, f_1 together with g_2, will be exchanged for a copy of Y. This means that component Π_k has a subset whose weight is equal to the target sum, hence a solution exists. Rules r_{17} through r_{19} are then used to send a **yes** object to the environment.
- Otherwise, if the number of copies of p and q are not equal, f_1 will be removed from the cell, together with any one copy of p or q. The system will then enter the communication stage.

In the *Communication Stage*, using rule $r_{22,i}$ each component will first produce in cell $(k, 1)$ sum objects, denoted by s_i, for $i = 0, ..., W$, which represents the possible weights from the subsets of each cell $(k, 2)$. For each cell $(k, 2)$, if there are i copies of r, then this cell would produce s_i, for some $i = 0, ..., W$. This would take at most W steps. Cell $(k, 1)$ will then produce 2^{n_k} copies of N objects, and an α object using rules

r_{20} and r_{20a} respectively. The N objects are used to transfer the sum objects to cell $(k, 1)$ from cell $(k, 2)$ using rule $r_{21,i}$. Once all s objects are present in cell $(k, 1)$, Π_k is now ready for inter-component communication. Π_1 will check with Π_2 if cell $(1, 1)$ and cell $(2, 1)$ contains sum objects, s_x, and s_y respectively, such that $x + y = T$. Now,

- If there exists s_x and s_y such that $x + y = T$, these sum objects, along with the token objects S_1 and S_2 are exchanged between the two components. Component Π_1 will then use rule r_{24} to send a **yes** to the environment, along with S_2 and object δ, ending the accepting computation. Likewise, Π_2 does the same, but using S_1. This is done using rule r_{24}.
- Otherwise, no communication occurs. The α object would have been used to introduce β_0 and δ using rule r_{23}. The β_i, for $i = 0, 1, 2$, acts a counter. When β_2 is produced and δ is still present, each component will then send a **no** to the environment, indicating a rejecting computation.

An example is given to demonstrate how the proposed distributed P system computes SSP.

EXAMPLE 1. *Consider* 2-Δ_{DTP} *working on an instance of the problem where* $X = \{1, 2, 3, 4\}, n = 4, T = 8$. *Suppose the partition* $\{X_1, X_2\}$ *is a balanced partition of* X*, where* $X_1 = \{1, 2\}$, $X_2 = \{3, 4\}$. Π_1 *and* Π_2 *will solve for the instances,* $u_1 = (2, 7, (1, 2), 8)$ *and* $u_2 = (2, 7, (3, 4), 8)$ *respectively. For brevity, we only give the computation of* Π_1 *as shown in Table 1.*

EXAMPLE 2. *Consider* 2-Δ_{DTP} *working on an instance of the problem where* $X = \{1, 2\}, n = 2, T = 4$. *Suppose the partition* $\{X_1, X_2\}$ *is a balanced partition of* X*, where* $X_1 = \{1\}$, $X_2 = \{2\}$. Π_1 *and* Π_2 *will solve for the instances,* $u_1 = (1, 2, (1), 4)$ *and* $u_2 = (1, 2, (2), 4)$ *respectively. Again, for brevity, we only give the computation of* Π_1 *as shown in Table 2. For this example, we only highlight the important objects and time steps.*

4 ANALYSIS OF THE COMPUTATION COST

4.1 Running Time

Given an instance of the SSP, a set X, target sum T, and W being the maximum of the weights between the sets of the partition, if communication does not occur, then our solution takes $\left\lceil \frac{n}{2} \right\rceil + \left\lceil \log_2 \frac{n}{2} \right\rceil + \lceil \log_2(T + 1) \rceil + W + 9$ time steps to complete. If communication was required and a subset sum pair exists (i.e. case 4), then our solution takes $\left\lceil \frac{n}{2} \right\rceil + \left\lceil \log_2 \frac{n}{2} \right\rceil + \lceil \log_2(T + 1) \rceil + W + 13$ time steps to complete; if no pair exists (i.e case 5), the solution takes $\left\lceil \frac{n}{2} \right\rceil + \left\lceil \log_2 \frac{n}{2} \right\rceil + \lceil \log_2(T + 1) \rceil + W + 14$ time steps.

4.2 Communication Cost and Parallelizability

Communication only occurs during one time step in the entire computation, so $ComN = 1$. The inter-component communication rules are only used once since only one copy each of S_1 and S_2 exist in the system. Thus we have $ComR = 1$. Four objects in total are exchanged for each time the inter-component communication rule is used. Since the inter-component communication rule is used once, then we have $ComW = 4$. Thus in terms of parallelizability we can say that:

THEOREM 2. *The SSP is (2,r)-weakly ComX parallelizable, where* $(r, ComX) \in \{(1, ComN), (1, ComR), (4, ComW)\}$.

We denote the worst case scenario, that is the case where communication was required, as $TIME_\Delta(n, T)$. From Theorem 2, we know that our solution to the SSP is weakly parallelizable. We compare its running time with the solution presented in [3]. The worst case running time of the non-distributed solution presented is $n + \lceil \log_2 n \rceil + \lceil \log_2(T + 1) \rceil + 12$, which will be denoted as $TIME_\Pi(n, T)$. Taking the limits of the ratio of the two running times:

$$\lim_{n,T\to\infty} \frac{TIME_\Pi(n, T)}{TIME_\Delta(n, T)} =$$
$$\frac{n + \lceil \log_2 n \rceil + \lceil \log_2(T + 1) \rceil + 12}{\left\lceil \frac{n}{2} \right\rceil + \left\lceil \log_2 \frac{n}{2} \right\rceil + \lceil \log_2(T + 1) \rceil + W + 14} = \frac{2}{W}$$

This indicates that the proposed distributed P system solution actually runs slower than the solution presented in [3]. This slow down further increases when the sum of weights W increases. The proposed solution does not perform well on input instances with large weight values.

5 CONCLUSION

In this work, we presented a solution to the SSP using a distributed tissue P system with cell division. The solution is a distributed system where components solve independently in a similar manner as presented in [3]. In this way we can observe the effects of partitioning the input and the addition of inter-component communication rules. By partitioning the input multiset into 2 parts, the generation stage only takes half the time to complete compared to the generation stage in the non-distributed solution. But whatever time steps saved from the generation stage is largely offset by the need to produce the sum objects, which takes additional time steps equal to the total weight of the input set. Now since one component of the distributed system is equivalent to the tissue P system solution in [3], this would also mean that we used double the number of cells in the distributed solution without any

Time	Rules Applied	Π_1 Configuration
0	-	$[DA_1A_2v_1v_2^2q^8]_{(1,2)}$ $[S_1a_1\bar{a}_1bc_1$ **yes no** $]_{(1,1)}$
$2(n=2)$	$r_{1,i}, r_{2a,i}, r_{2\bar{a},i}, r_{3,i}$	$[DB_1B_2v_1v_2^2q^8]_{(1,2)}$ $[DB_1v_1v_2^2q^8]_{(1,2)}$ $[DB_2v_1v_2^2q^8]_{(1,2)}$ $[Dv_1v_2^2q^8]_{(1,2)}$ $[S_1a_3^4\bar{a}_3b^4c_3^4$ **yes no** $]_{(1,1)}$
3	$r_4, r_{2b,i}, r_{2\bar{b},i}$	$[c_3B_1B_2v_1v_2^2q^8]_{(1,2)}$ $[c_3B_1v_1v_2^2q^8]_{(1,2)}$ $[c_3B_2v_1v_2^2q^8]_{(1,2)}$ $[c_3v_1v_2^2q^8]_{(1,2)}$ $[S_1a_4^4\bar{a}_4b^4D^4$ **yes no** $]_{(1,1)}$
4	$r_5, r_{2b,i}, r_{2\bar{b},i}$	$[d_1e_1B_1B_2v_1v_2^2q^8]_{(1,2)}$ $[d_1e_1B_1v_1v_2^2q^8]_{(1,2)}$ $[d_1e_1B_2v_1v_2^2q^8]_{(1,2)}$ $[d_1e_1v_1v_2^2q^8]_{(1,2)}$ $[S_1a_5^4\bar{a}_5b^4D^4$ **yes no** $]_{(1,1)}$
$5(\lceil\log_2 n\rceil = 1)$	$r_{6,i}, r_{7,i}, r_{2b,i}, r_{2\bar{b},i}$	$[d_2e_2^2B_1B_2v_1v_2^2q^8]_{(1,2)}$ $[d_2e_2^2B_1v_1v_2^2q^8]_{(1,2)}$ $[d_2e_2^2B_2v_1v_2^2q^8]_{(1,2)}$ $[d_2e_2^2v_1v_2^2q^8]_{(1,2)}$ $[S_1a_6^4\bar{a}_6b^4D^4$ **yes no** $]_{(1,1)}$
6	$r_{7,i}, r_{8,i}, r_{2b,i}, r_{2\bar{b},i}$	$[d_3B_{1,1}B_{2,1}v_1v_2^2q^8]_{(1,2)}$ $[d_3e_2B_{1,1}v_1v_2^2q^8]_{(1,2)}$ $[d_3e_2B_{2,1}v_1v_2^2q^8]_{(1,2)}$ $[d_3e_2^2v_1v_2^2q^8]_{(1,2)}$ $[S_1a_7^4\bar{a}_7b^4D^4$ **yes no** $]_{(1,1)}$
$10(\lceil\log_2(T+1)\rceil = 4)$	$r_{7,i}, r_{9,i,j}, r_{2b,i}, r_{2\bar{b},i}$	$[d_6B_{1,5}^{16}B_{2,5}^{16}v_1v_2^2q^8]_{(1,2)}$ $[d_6e_2B_{1,5}^{16}v_1v_2^2q^8]_{(1,2)}$ $[d_6e_2B_{2,5}^{16}v_1v_2^2q^8]_{(1,2)}$ $[d_6e_2^2v_1v_2^2q^8]_{(1,2)}$ $[S_1a_{11}^4\bar{a}_{11}b^4D^4$ **yes no** $]_{(1,1)}$
11	$r_{7,i}, r_{10,i}, r_{2b,i}, r_{2\bar{b},i}$	$[d_7B_{1,5}^{15}B_{2,5}^{14}p^3r^3q^8]_{(1,2)}$ $[d_7e_2B_{1,5}^{15}prv_2^2q^8]_{(1,2)}$ $[d_7e_2B_{2,5}^{14}v_1p^2r^2q^8]_{(1,2)}$ $[d_7e_2^2v_1v_2^2q^8]_{(1,2)}$ $[S_1a_{12}^4\bar{a}_{12}b^4D^4$ **yes no** $]_{(1,1)}$
12	$r_{7,i}, r_{11}, r_{2b,i}, r_{2\bar{b},i}$	$[d_8B_{1,5}^{15}B_{2,5}^{14}r^3q^5]_{(1,2)}$ $[d_8e_2B_{1,5}^{15}rv_2^2q^7]_{(1,2)}$ $[d_8e_2B_{2,5}^{14}v_1r^2q^6]_{(1,2)}$ $[d_8e_2^2v_1v_2^2q^8]_{(1,2)}$ $[S_1a_{13}^4\bar{a}_{13}b^4D^4$ **yes no** $]_{(1,1)}$
13	$r_{12}, r_{2b,i}, r_{2\bar{b},i}$	$[g_1f_1B_{1,5}^7B_{2,5}^{14}r^3q^5]_{(1,2)}$ $[g_1f_1e_2B_{1,5}^{15}rv_2^2q^7]_{(1,2)}$ $[g_1f_1e_2B_{2,5}^{14}v_1r^2q^6]_{(1,2)}$ $[g_1f_1e_2^2v_1v_2^2q^8]_{(1,2)}$ $[S_1a_{14}^4\bar{a}_{14}b^4D^4$ **yes no** $]_{(1,1)}$
14	$r_{14}, r_{15}, r_{2b,i}, r_{2\bar{b},i}$	$[g_2s_0B_{1,5}^7B_{2,5}^{14}r^3q^4]_{(1,2)}$ $[g_2s_0e_2B_{1,5}^{15}rv_2^2q^6]_{(1,2)}$ $[g_2s_0e_2B_{2,5}^{14}v_1r^2q^5]_{(1,2)}$ $[g_2s_0e_2^2v_1v_2^2q^7]_{(1,2)}$ $[S_1a_{15}^4\bar{a}_{15}b^4D^4$ **yes no** $]_{(1,1)}$
17	$r_{22,i}, r_{2b,i}, r_{2\bar{b},i}$	$[g_2s_3B_{1,5}^{15}B_{2,5}^{14}q^4]_{(1,2)}$ $[g_2s_1e_2B_{1,5}^{15}v_2^2q^6]_{(1,2)}$ $[g_2s_2e_2B_{2,5}^{14}v_1q^5]_{(1,2)}$ $[g_2s_0e_2^2v_1v_2^2q^7]_{(1,2)}$ $[S_1a_{18}^4\bar{a}_{18}b^4D^4$ **yes no** $]_{(1,1)}$
$24(W=7)$	$r_{2b,i}, r_{2\bar{b},i}$	$[g_2s_3B_{1,5}^{15}B_{2,5}^{14}q^4]_{(1,2)}$ $[g_2s_1e_2B_{1,5}^{15}v_2^2q^6]_{(1,2)}$ $[g_2s_2e_2B_{2,5}^{14}v_1q^5]_{(1,2)}$ $[g_2s_0e_2^2v_1v_2^2q^7]_{(1,2)}$ $[S_1a_{25}^4\bar{a}_{25}b^4D^4$ **yes no** $]_{(1,1)}$
25	r_{20}, r_{20a}	$[g_2s_3B_{1,5}^{15}B_{2,5}^{14}q^4]_{(1,2)}$ $[g_2s_1e_2B_{1,5}^{15}v_2^2q^6]_{(1,2)}$ $[g_2s_2e_2B_{2,5}^{14}v_1q^5]_{(1,2)}$ $[g_2s_0e_2^2v_1v_2^2q^7]_{(1,2)}$ $[S_1\alpha N^4D^4$ **yes no** $]_{(1,1)}$
26	$r_{21,i}, r_{23}$	$[g_2NB_{1,5}^{15}B_{2,5}^{14}q^4]_{(1,2)}$ $[g_2Ne_2B_{1,5}^{15}v_2^2q^6]_{(1,2)}$ $[g_2Ne_2B_{2,5}^{14}v_1q^5]_{(1,2)}$ $[g_2Ne_2^2v_1v_2^2q^7]_{(1,2)}$ $[S_1\beta_0\delta s_0s_1s_2s_3D^4$ **yes no** $]_{(1,1)}$
27	r_c	$[g_2NB_{1,5}^{15}B_{2,5}^{14}q^4]_{(1,2)}$ $[g_2Ne_2B_{1,5}78v_2^2q^6]_{(1,2)}$ $[g_2Ne_2B_{2,5}^{14}v_1q^5]_{(1,2)}$ $[g_2Ne_2^2v_1v_2^2q^7]_{(1,2)}$ $[S_2\beta_1\delta s_0s_7s_2s_3D^4$ **yes no** $]_{(1,1)}$
28	r_{24}	$[g_2NB_{1,5}^{15}B_{2,5}^{14}q^4]_{(1,2)}$ $[g_2Ne_2B_{1,5}^{15}v_2^2q^6]_{(1,2)}$ $[g_2Ne_2B_{2,5}^{14}v_1q^5]_{(1,2)}$ $[g_2Ne_2^2v_1v_2^2q^7]_{(1,2)}$ $[\beta_2s_0s_7s_2s_3D^4$ **no** $]_{(1,1)}$ $S_2\delta$**yes**

Table 1: Sample computation for Π_1 in Example 1. To summarize the computation table: For the first n steps ($n = 2$), the system generates all possible subsets, e.g., the cell with label $(1, 2)$ containing B_1B_2 represents the subset that contains the first and second elements. In this case, the values 1 and 2, respectively. By step 11, the system is now prepared to do the checking stage. Note here that the number of copies of the p and r objects represent the sum of the subset, e.g., the subset containing the values 1 and 2 has 3 copies of p and r (p^3, r^3). The number of copies of p will then be checked against the number of copies of q (which represents the target sum). Since there is no cell with an equal number of p and q objects, by step 26, the system will produce all the possible sums from the given input. With input $\{1, 2\}$, the possible sums are $\{0, 1, 2, 3\}$, represented by the sum objects $\{s_0, s_1, s_2, s_3\}$, respectively. At step 27, Π_1 will check with Π_2 if there is a sum object from Π_1, denoted as s_x, and a sum object from Π_2, denoted as s_y, such that $x + y = T$. In this case, Π_1 has s_1, and Π_2 has s_7 (since Π_2 has $\{3, 4\}$). Hence, a inter-component communication occurs. Π_1 receives S_2s_7 from Π_2, while Π_2 receives S_1s_1 from Π_1. Both components will then conclude that there is a solution to the input instance $X = \{1, 2, 3, 4\}$, $T = 8$.

gain (and actually performing worse). One could consider a method on how we can generate the correct sum object from the r objects in a single step, hence not requiring W steps.

For future works on the topic of distributed tissue P system solving SSP, we could analyze how the system would perform given an unbalanced partition (e.g. one component receives two-thirds of the input set, while the other receives only one-third). One of the issues to consider would be the timing of the counters. The component with the larger input would run longer than the component with the smaller input. Hence, the running time of the generation stage would approach that of the non-distributed solution in [3]. This is still not taking into account the running time introduced by the additional communication stage. What would not change however would be the communication cost, as it would still require one communication step, rule, and the same number of objects communicated. What would affect the communication cost would be increasing the number of components of the distributed tissue P system solving SSP. We would need additional communication steps to coordinate the candidate sum values from all components. We could lessen the communication cost if only one component is assigned the role of deciding the input instance as accept or reject.

Now in terms of communication cost, we have shown that SSP is weakly ComX parallelizable. It takes only one communication step and rule, and as few as four objects needed for communication. In contrast to another distributed P system solution for an NP-hard problem[1, 2], the communication costs are mostly the same. But what is good about this proposed distributed tissue P system solution to SSP is that it only needs a linear number of inter-component communication rules to solve SSP, as opposed to the distributed P system solution to the satisfiability problem (SAT) which requires an exponential number of inter-component communication rules. This indicates that there are some NP-hard problems in which a distributed P system solution does not require an exponential amount of resource. This would mean that it could be possible to design a more efficient inter-component communication protocol for problems such as SAT, or there could be easier or harder NP-hard problems for distributed P systems.

Time	Π_1 Configuration
0	$[A_1 v_1 q^4]_{(1,2)}\ [S_1 a_1 \bar{a}_1 b c_1$ **yes no** $]_{(1,1)}$
1	$[B_1 v_1 q^4]_{(1,2)}\ [v_1 q^4]_{(1,2)}\ [S_1 a_1 \bar{a}_1$ **yes no** $]_{(1,1)}$
8	$[B_{14}^7 p r q^4]_{(1,2)}\ [v_1 q^4]_{(1,2)}\ [S_1 a_9^2 \bar{a}_9$ **yes no** $]_{(1,1)}$
9	$[B_{14}^7 r q^3]_{(1,2)}\ [v_1 q^4]_{(1,2)}\ [S_1 a_{10}^2 \bar{a}_{10}$ **yes no** $]_{(1,1)}$
17	$[s_1 B_{14}^7 q^2]_{(1,2)}\ [s_0 v_1 q^3]_{(1,2)}\ [S_1 a_{18}^2 \bar{a}_{18}$ **yes no** $]_{(1,1)}$
21	$[B_{14}^7 q^2]_{(1,2)}\ [v_1 q^3]_{(1,2)}\ [S_1 \beta_2 \delta s_1 s_0$ **yes no** $]_{(1,1)}$
22	$[B_{14}^7 q^2]_{(1,2)}\ [v_1 q^3]_{(1,2)}\ [S_1 s_1 s_0$ **yes** $]_{(1,1)}\ \beta_2 \delta$**no**

Table 2: Sample computation for Π_1 in Example 2. This instance has no solution hence the system outputs a no. Similar to Example 1, since there is no local subset whose sum is equal to the target sum, until time step 17, Π_1 will produce all possible sum it can obtain from input instance $\{1\}$. That is s_0, s_1. When the β counter is introduced, it will wait for two time steps. Since no communication occurs between the two components, i.e., there is no local sum from Π_1 and Π_2 that when added is equal to $T = 4$, then with β_2, the system outputs a no.

ACKNOWLEDGEMENTS

This work is a result of the CS 199 research of Justin Granda and Samuel Jose, with adviser Kelvin Buño, from the Algorithms and Complexity lab of the Department of Computer Science, UP Diliman.

K. Buño is thankful to the support from the Atty. Raul C. Villanueva Professorial Chair since 2019 until present.

REFERENCES

[1] Adorna, H.N., Pan, L., Song, B.: On distributed solution to sat by membrane computing. Int. J. Comput. Commun. Control **13**(3), 303–320 (2018)

[2] Buño, K.C., Cabarle, F.G.C., Calabia, M.D., Adorna, H.N.: Solving the n-queens problem using dp systems with active membranes. Theoretical Computer Science **736**, 1 – 14 (2018). https://doi.org/https://doi.org/10.1016/j.tcs.2017.12.013

[3] Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: Solving subset sum in linear time by using tissue p systems with cell division. In: Mira, J., Álvarez, J.R. (eds.) Bio-inspired Modeling of Cognitive Tasks. pp. 170–179. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)

[4] Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A uniform family of tissue p systems with cell division solving 3-col in a linear time. Theoretical Computer Science **404**(1), 76 – 87 (2008). https://doi.org/https://doi.org/10.1016/j.tcs.2008.04.005, membrane Computing and Biologically Inspired Process Calculi

[5] Gheorghe, M., Ipate, F., Dragomir, C.: A kernel p system. Membrane Computing, Tenth Brainstorming Week, BWMC (01 2012)

[6] Gheorghe, M., Ipate, F., Konur, S.: Solutions to the subset sum and partition problems using kernel p systems. Annals of Bucharest University, Computer Science pp. 37–46 (2015)

[7] Macababayao, I.C.H., Amores, E.M.L., Hernandez, N.H.S., Cabarle, F.G.C.: Deterministic and Uniform Solutions to NP-Complete Problems using Numerical P Systems with Lower Thresholds. pp. 253–272. Pre-proc. 18th International Conference on Membrane Computing (CMC18), 24 to 28 July 2017, University of Bradford, U.K. (2017)

[8] no, K.C.B., Cabarle, F.G.C., Torres, J.G.Q.: Spiking neural dp systems: Balance and homogeneity. Philippine Computing Journal **14**(2), 1–10 (2020)

[9] Păun, A.: On p systems with active membranes. In: Antoniou, I., Calude, C.S., Dinneen, M.J. (eds.) Unconventional Models of Computation, UMC'2K. pp. 187–201. Springer London, London (2001)

[10] Păun, G.: Introduction to Membrane Computing, pp. 1–42. Springer Berlin Heidelberg, Berlin, Heidelberg (2006). https://doi.org/10.1007/3-540-29937-8_1

[11] Păun, Gh., Pérez-Jiménez, M.J.: Tissue p systems with cell division. International Journal of Computers Communications & Control **3**(3), 295–303 (2008). https://doi.org/10.15837/ijccc.2008.3.2397

[12] Păun, G., Păun, R.: Membrane computing and economics: Numerical p systems. Fundam Inf **73**(1,2), 213–227 (Apr 2006)

[13] Păun, G., Pérez-Jiménez, M.J.: Solving problems in a distributedway in membrane computing: dp systems. International Journal of Computers Communications & Control **5**(2), 238–250 (2010). https://doi.org/10.15837/ijccc.2010.2.2478

[14] Zeng, X., Zhang, X., Pan, L.: Homogeneous spiking neural p systems. Fundamenta Informaticae **97**, 275–294 (2009). https://doi.org/10.3233/FI-2009-200, 1-2