A Cluster Splitting Technique by Hopfield Networks and P Systems on Simplices

Xiyu Liu¹ · Jie Xue¹

Published online: 21 January 2017

© The Author(s) 2017. This article is published with open access at Springerlink.com

Abstract In this paper we propose a new graph based P system. The main feature of graph P system is that membrane compartments spread on edges as well as on vertices. These two kinds of compartments are topologically connected with certain communication rules. A new neural computing technique is proposed to solve cluster splitting into finding the equilibrium of Hopfield neural networks. Then we use the new graph P systems to obtain stable status of the neural networks. An example is presented to show the computing efficiency of the new P system with comparison with classical genetic algorithms.

Keywords Membrane computing · Cluster analysis · Hopfield neural networks · P systems

1 Introduction

Membrane computing as one of the most promising biological computing techniques has attracted more and more researchers in natural computing in the last decade. A classical literature in this area is Păun's book [25]. Roughly speaking, there are two key issues in membrane computing, i.e., the topological structure of compartments, and the computations comprising a set of rules.

There are many categories of P systems as summarized in Păun's book [25]. In the topological point of view, the most typical P system is the cell-like P system with communication rules and, topologically a tree structure. New communication P systems solving sorting and communication models and synchronous object movement across components are proposed

Research is supported by the Natural Science Foundation of China (No. 61472231, 61170038). Other Grants: 201401202, 12YJA630152, 11CGLJ22.

✉ Xiyu Liu
xylu@sdnu.edu.cn

Jie Xue
xiaozhuzhu1113@163.com

¹ School of Management Science and Engineering, Shandong Normal University, Jinan, China

in [5,30]. Also there are P automata which are devices characterizing non-universal family of languages [26]. Another important P system is the tissue P system which is topologically a graph [25]. There are many tissue P systems such as rule applicability based communication modes [4], and trajectory P systems with evolutionary rules [3]. Spiking neural P systems are extensively studied recently by Pan and his team [17]. For example, Zeng et al. [37] study spiking neural P system with only one neuron simulating register machines, and other smaller weakly universal spiking neural P systems. There are also researches on asynchronous spiking neural P systems with promoters [35], spiking P systems to implement arithmetic operations [20], systems with fuzzy logic and firing mechanism [27], and applications with membrane algorithms [8].

P systems can be dynamic in its topological structure. By using rules that describe the movement and evolution compartments, P systems with active membranes are able to solve complicated combinatory problems in linear time [11]. Dynamic compartments are also applied in a research of Spicher [29]. There are also researches on introducing probabilities in a membrane system with respect to multisets and rules [24]. For applications of P systems we refer to [9], and the Proceedings of CMC15 [12].

For joint study of membrane computing with other optimization techniques, there are evolutionary string object P systems which introduces genetic operations such as crossover and mutation in membrane computing [31]. Alhazov in [1] generalize a splicing P system which joins the Tom Head splicing DNA system with membrane computing. By linking membrane computing with picture grammars, the so-called array P systems are proposed with objects in region as arrays applicable in syntactic analysis of images [6]. There are other works which incorporate Petri nets with spiking Neural P system with anti-spikes (SN PA systems) in order to verify system properties, soundness and simulation [22].

The second focus of this paper is cluster analysis [13,33,34,36]. For example, a graph-skeleton-based clustering method with maximal spanning trees is proposed in [14]. In order to overcome local optima problem, Amiri in [2] proposes a population-based Teaching-Learning-Based-Optimization method without controlling parameters. In [10], the authors present a randomised algorithm for time series clustering, while Chen et al. [7] focus on the prognostic effect of the immunophenotypic clusters of acute myeloid leukemia patients. Research on discriminative spectral clustering can be found in [32]. For dynamic time series clustering we refer to [15]. The Q-statistics is also applied for clustering Non-Hodgkin lymphoma (NHL) in space and time [23]. Applications of cluster analysis in genetic analysis of strawberry [28], the k -means clustering in blogs analysis [19], gene expression data mining [16], joint study of membrane computing with cluster analysis [18].

Although clustering methods are proposed extensively, new technique and new joint research are still important in this big data era due to the vast development of business intelligence and other applications. In the point of membrane computing, we still need new system structure in order to solve new and potential problems. In [18], a new P simplicial P system is proposed. The purpose of this paper is to continue exploring simplicial P systems.

In this paper we will propose a graph based P system. Topologically, this graph P system uses a graph as its data structure. Different with tissue P systems, in graph P system we construct cells (compartments) on edges rather than only on vertices. Vertex cells and edge cells are topologically connected with special communications rules. We propose a novel neural computing technique to transform cluster analysis to the equilibrium of Hopfield neural networks. Computation of the neural network is performed by graph P systems. The new technique provides a cluster splitting method which is the key process of hierarchical cluster analysis. Finally we present an example to show the computation of the new P system.

Comparison of genetic algorithm with our method is given which shows that the new method is efficient in computing complexity.

2 Cluster Splitting by Hopfield Neural Networks

In this section, we propose a Hopfield neural network setting for classical clustering problems. Our purpose is to find a new cluster splitting method by Hopfield neural networks. First we suppose the data set to be clustered is $\Omega = \{\xi_1, \dots, \xi_N\}$ with N data points. A similarity measure ρ is defined on $\Omega \times \Omega$ which forms similarity matrix $W = [w_{ij}]_{N \times N}$ where $w_{ij} = \rho(\xi_i, \xi_j)$.

Classical clustering methods include the partition method, the hierarchical method, the model based method, etc. For hierarchical method there are two categories namely agglomerative and divisive methods, where the latter is more complicated in most occasions. In this paper our focus is the divisive clustering method. In divisive clustering analysis, one of the most difficult problem is to split a cluster into two sub-clusters with specific conditions, i.e., maximum intra-cluster similarity and minimum inter-cluster similarity. If we search all possible splitting examples, it will be NP-complete.

To simplify the problem, we need only to find an efficient method to split the data set Ω into two clusters satisfying certain optimal criteria. In order to achieve this task, we use a class label $x_i = \pm 1$ where $x_i = 1$ indicates one cluster and $x_i = -1$ indicates another split one. Then we define a vector $x = [x_1, \dots, x_N]^T$ with $x_i = \pm 1$. To specify the criteria for optimal splitting, an energy function is defined as follows

$$E = E(x) = -\frac{1}{2} \sum_{1 \leq i, j \leq N}^N w_{ij} x_i x_j \quad (2.1)$$

Now the Ω is split into two clusters $\Omega_1 = \{\xi_i | x_i = 1\}$ and $\Omega_2 = \{\xi_i | x_i = -1\}$. The energy function E in (2.1) is the sum of intra-cluster weights minus inter-cluster weights, together with a computing constant $-\frac{1}{2}$. Therefore the minimal solution of E is an optimal cluster splitting solution in the sense of our energy function.

To solve this problem, we now construct a Hopfield neural network (W, θ) with $\theta = 0$. By standard neural network theory [21], the stable state of the network corresponds to the minimal solution of Eq. (2.1). Suppose the state of the network is $x(t)$ with a discrete time variable t . Then the evolution of $x(t)$ is by the following Eq. (2.2):

$$\begin{cases} x_i(0) = x_i^0 \\ x_i(t+1) = f_i(H_i(t)) \\ H_i(t) = \sum_{k=1}^N w_{ik} x_k(t) \end{cases} \quad (2.2)$$

where the function $f(u) = \text{sgn}(u)$.

Since similarity measure is non-negative, the matrix diagonal of W is positive. By the standard neural network theory [21], the network (W, θ) is stable in asynchronous mode.

In the following, we always assume that the weight w_{ij} takes non-negative integers as its values.

3 A Graph Based P System

A basic cell-like P system is defined as a hierarchical arrangement of compartments delimited by membranes. Each compartment may contain a finite multiset of objects and a finite set of rules, as well as a finite set of other compartments. The rules perform transformation and communication operations.

In order to solve the Hopfield neural networks by membrane computing technique, we need to extend a typical P system to simplices. Simplices can be seen as a new class of data structure different with the linear structure, the tree structure, and the graph structure. In topological point of view, these structures are one dimensional. A simplex structure however, can be multi-dimensional. A tissue like P system is a system on graphs. Now we extend membrane membrane structures to edges. Then we obtain a totally new class of P system, called P system on simplices, or simplicial P systems.

A k -simplex σ is the convex hull of $k + 1$ affinely independent points $a_0, a_1, \dots, a_k$ defined as the set of points in the form $x = \sum_{i=0}^k \lambda_i a^i$ where

$$\sum_{i=0}^k \lambda_i = 1, \lambda_0, \lambda_1, \dots, \lambda_k \geq 0. \quad (3.1)$$

A simplex (cell) is uniquely indicated by its vertices and denoted as

$$\sigma = [a_0, a_1, \dots, a_k], \text{ or } \sigma = a_0 a_1 \dots a_k. \quad (3.2)$$

A face τ of σ is a simplex by a subset vertices written by $\tau < \sigma$. τ is called a hyperface if $\dim \tau = \dim \sigma - 1$, denoted by $\tau < \sigma$, while σ is called its parent. Two cells τ_1 and τ_2 are incident if $\tau_1, \tau_2 < \sigma$, and σ is called the coface of τ_1 and τ_2 . Two cells σ_1, σ_2 are called neighbors if they share a common hyperface. A simplicial complex K is a collection of simplices for which $\sigma \in K$ and $\tau < \sigma$ implies $\tau \in K$; and $\sigma_1, \sigma_2 \in K$ implies that $\sigma_1 \cap \sigma_2$ is either empty or a face of both. For a k -simplex σ , define K to be the collection of σ and all its faces. Then K is a simplicial complex, namely, a simple complex, or simply, a complex.

3.1 Membrane Structures on Graphs

Graph is a simple kind of simplicial complex. Now we will define membrane structures on graphs. Notice that these are different with tissue like P systems.

Suppose $G = (V, E)$ is a graph where V is the set of vertices and E the set of edges. Define a simplicial complex K as the collection of V and E . Each simplex $\sigma \in K$ is called a membrane. A membrane in E is maximal since it is not a face of another simplex in K . Evidently, vertices are zero-dimensional cells and hence are elementary membranes. Next we will identify K with the graph G .

If $\tau < \sigma$ is a face of σ , then we say that σ is the parent of τ . If σ_1, σ_2 are incident, we say there is an upper link (channel) between σ_1, σ_2 . If σ_1, σ_2 are neighbors, we say there is a lower link between them. An upper link is denoted by $(\sigma_1, \sigma_2)^\tau$ where τ is their (common) parent, while lower link is written as $(\sigma_1, \sigma_2)_\tau$ where τ is their common hyperface. Upper link is also written by $(\sigma_1, \sigma_2)^U$, or, simply (σ_1, σ_2) , while lower link is denoted by $(\sigma_1, \sigma_2)_L$.

Definition 1 A P system on a graph G , called a graph P system, with antiport and symport rules is a construct

$$\Pi = (m, O, \omega_1, \dots, \omega_m, R_1, \dots, R_m, \\ ch, F(i, j)|_{(i,j) \in ch}, i_0)$$

where m is the number of cells labeled with $1, 2, \dots, m$, O is the alphabet, $\omega_1, \dots, \omega_m$ are initial strings over O of multisets, $R_1, \dots, R_m$ are symport and antiport rules associated with the m membranes, $ch \subseteq \{(i, j)_R, (i, j)_L : i, j \in \{1, \dots, m\}\}$ is the set of links, $F(i, j)$ is a finite set of antiport and/or symport rules associated with the link $(i, j) \in ch$.

3.2 Rules of Graph P Systems

Now we describe rules in graph P systems. For the purpose of this paper, there are mainly four types of communication rules. Each type of rules contains four special operators *out*, *in*, *up* and *down*.

First suppose there exists an upper link $(\tau_1, \tau_2)^\sigma$. A rule in the form of $[(x, y), up]^U$ means that the multiset x and y from τ_1 and τ_2 transform into some z and go up to their parent σ . Second, for two cells $\tau < \sigma$, antiport rule in τ in the form of $[u, out; v, in | \tau < \sigma]$ means that exchanging multiset u inside membrane τ with the multiset v outside (both in σ). Thirdly, symport rule $[u, out | \tau < \sigma]$ sends multiset u out of τ to σ . Symport rule $[u, in | \tau < \sigma]$ works similarly and takes u from σ into τ .

Combining these rules together we define upper link rule in $R^\sigma(\tau_1, \tau_2)$ in the following forms

$$[(x, y), up; (\alpha, \beta), in]^U \longrightarrow [z, in; \gamma, down] \quad (3.3)$$

$$[u, out; v, in | \tau_1 < \sigma], [u, out; v, in | \tau_2 < \sigma] \quad (3.4)$$

Operations of Eq. (3.3) are listed here. Multiset x from cell τ_1 and y from τ_2 are removed transformed into z to parent cell σ , while simultaneously γ of σ is moved and transformed into α in τ_1 and β in τ_2 . Operations of Eq. (3.4) is to move u out to σ while taking v from σ into τ_1 , or τ_2 .

A lower link rule in $R_\sigma(\tau_1, \tau_2)$ has the following forms

$$[(x, y), down; (\alpha, \beta), in]_L \longrightarrow [z, in; \gamma, up] \quad (3.5)$$

$$[u, out; v, in | \sigma < \tau_1], [u, out; v, in | \sigma < \tau_2] \quad (3.6)$$

Equation (3.5) means that we move x from cell τ_1 and y from τ_2 to z into their hyperface σ , while simultaneously moving γ from σ to α up to τ_1 and β up to τ_2 . Equation (3.6) is similar to Eq. (3.5).

3.3 Configurations and Computations

Now we describe the configurations and computations of the proposed graph P systems. For our purpose, change of membrane structure is not involved. A *configuration* of a graph P system is the state of the system described by specifying the objects and rules associated to each membrane. The initial state is called *initial configuration*. Therefore, the multisets represented by ω_i , $1 \leq i \leq d$ in Π , constitute the initial configuration of the system.

The system evolves by applying rules in membranes and this evolution is called *computation*. A computation starts with the multisets specified by $\omega_1, \dots, \omega_m$ in its m cells. During each time unit, rules are applied in a cell. If no rule is applicable for one cell, then no object change occurs in it. The system evolves synchronously for all cells.

When the system has reached a configuration in which no rule is any longer applicable, we say that the computation *halts*. A configuration is *stable* if, even if some rules are still applicable, their application does not change the object content of the membranes. The computation is successful if and only if it halts, or it is stable. The result of a halting/stable

Table 1 An alphabet list

$a_1, \dots, a_N$	a, a_+, a_-	b_+, b_-, c
α, α_s	$\alpha_+, \alpha_-, \alpha_0$	β, γ, γ_+
π, π_0	π_c, π_h	π_σ

computation is the number described by the multiplicity of objects present in the cell i_0 in the halting/stable configuration.

4 Rules and Computations of Graph P Systems

In this section we consider a neural network $(W, 0)$ as in Sect. 2 and a P system on a complete graph $G = (V, E)$ with N nodes. Notice that we always assume that the elements of W are non-negative integers. We use an integer N_i to denote the sum

$$N_i = \sum_{j=1}^N w_{ij} - w_{ii} \quad (4.1)$$

Consider the P system

$$\Pi = (O, \{\omega_i, R_i\}_{1 \leq i \leq N}, \{\omega_{ij}, F_{i,j}\}_{(i,j) \in ch}, i_0) \quad (4.2)$$

where the number of edges (links) are $m = \frac{1}{2}N(N-1)$, ω_i, R_i are initial configurations and rules of (on) node membranes σ_i , and $\omega_{ij}, F_{i,j}$ are initial configurations and rules of (on) edge membranes σ_{ij} , $i_0 = 1$ indicates the output node. The alphabet is shown in Table 1.

For initial configurations of node cells, we set

$$\begin{cases} \omega_1 = \{a_1\} \cup \{\alpha_s, \alpha^{N-1}, \alpha_0^{N_1}, \gamma_+^{N-1}\} \\ \omega_i = \{a_i\} \cup \{\alpha, \beta\}, \quad i > 1 \end{cases} \quad (4.3)$$

where the symbol $a_i = a_+$ or a_- . As initialization we set $a_i = a_+$ or a_- randomly. The initial configuration on edges are empty strings $\omega_{ij} = \{\lambda\}$.

4.1 Rules with Partial Ordering

All the rules are categorized as several groups. The first group is annihilation rules acting on node cells

$$F_0 = \begin{cases} r_{00} : b_+ b_- \rightarrow \lambda \\ r_{01} : [\sigma_i, \sigma_1] \rightarrow [\sigma_i, \sigma_1] (\alpha_0 \alpha_+, \lambda) \rightarrow (b_+, c) \\ r_{02} : [\sigma_i, \sigma_1] \rightarrow [\sigma_i, \sigma_1] (\alpha_0 \alpha_-, \lambda) \rightarrow (b_-, c) \\ r_{03} : \sigma_i \rightarrow \sigma_i | \beta \alpha_+ \alpha_+ \rightarrow \beta \alpha_+ \succ \\ \quad [\sigma_i, \sigma_1] \rightarrow [\sigma_i, \sigma_1] | (\beta \alpha_+, \lambda) \rightarrow (\lambda, c) \\ r_{04} : \sigma_i \rightarrow \sigma_i | \beta \alpha_- \alpha_- \rightarrow \beta \alpha_- \succ \\ \quad [\sigma_i, \sigma_1] \rightarrow [\sigma_i, \sigma_1] | (\beta \alpha_-, \lambda) \rightarrow (\lambda, c) \end{cases} \quad (4.4)$$

We also have a set of rules acting on node cells.

$$F_{01} = \begin{cases} r_{011} : [\sigma_i, \sigma_1] \rightarrow [\sigma_i, \sigma_1] \\ (\alpha_s, \lambda) \rightarrow (b_+^{w_{ii}} \gamma, c) \end{cases} \quad (4.5)$$

The second group is edge rules.

$$F_1 = \begin{cases} r_{11} = \{a_+a_+ \rightarrow b_+\} \\ r_{12} = \{a_+a_- \rightarrow b_-\} \\ r_{13} = \{a_-a_- \rightarrow b_+\} \end{cases} \quad (4.6)$$

$$F_1^{ij} = \begin{cases} r_{14} : \sigma_{ij} \rightarrow [\sigma_i, \sigma_j] \\ \quad b_+a^{w_{ij}} \rightarrow [\alpha_+^{w_{ij}}, \alpha_+^{w_{ij}}] \\ r_{15} : \sigma_{ij} \rightarrow [\sigma_i, \sigma_j] \\ \quad b_-a^{w_{ij}} \rightarrow [\alpha_-^{w_{ij}}, \alpha_-^{w_{ij}}] \end{cases} \quad (4.7)$$

The third group is rules on nodes by links.

$$F_2 = \begin{cases} r_{21}^U : [\sigma_i, \sigma_j] \rightarrow [\sigma_i, \sigma_{ij}, \sigma_j] \\ \quad (\alpha\gamma_+, \alpha) \rightarrow (\gamma, a_i a_j a^{w_{ij}}, \lambda) \\ r_{22}^U : [\sigma_i, \sigma_j] \rightarrow [\sigma_i, \sigma_{ij}, \sigma_j] \\ \quad (\alpha, \alpha\gamma_+) \rightarrow (\lambda, a_i a_j a^{w_{ij}}, \gamma) \\ \quad (i \neq j) \end{cases} \quad (4.8)$$

Finally we consider controlling rules. The fourth group of rules control the active node of the neural network. These rules act on node cells.

$$F_3 = \begin{cases} r_{31}^{\rightarrow} : \\ \quad [\sigma_1, \sigma_2] \rightarrow [\sigma_1, \sigma_2] | (\pi_c \gamma^N, \lambda) \rightarrow \\ \quad (\alpha\beta, \alpha_s \alpha^{N-1} \alpha_0^{N_2} \gamma_+^{N-1}) \\ \quad \sigma_i \rightarrow \sigma_i (i > 2) | a_i \rightarrow a_i \alpha\beta \\ r_{32}^{\rightarrow} : \\ \quad [\sigma_i, \sigma_{i+1}] \rightarrow [\sigma_i, \sigma_{i+1}], 1 < i < N | \\ \quad (\gamma^N \pi_c, \lambda) \rightarrow (\alpha\beta, \alpha_s \alpha^{N-1} \alpha_0^{N+1} \gamma_+^{N-1}) \\ \quad \sigma_j \rightarrow \sigma_j (j \neq i, i+1) | a_j \rightarrow a_j \alpha\beta \end{cases} \quad (4.9)$$

Now we discuss computation and halting conditions. The main idea is using a symbol π_0 to count the number of running nodes in one cycle and compute the number of stable nodes. When the number of stable nodes equals N , then the whole network is stable. We can send a symbol π_h to output node σ_1 . Notice that rules are executed in an order.

$$F_4 = \begin{cases} r_{41}^{\rightarrow} : \sigma_i \rightarrow \sigma_i | b_+b_+ \rightarrow b_+ \succ r_{420} \\ r_{42}^{\rightarrow} : \sigma_i \rightarrow \sigma_i | b_-b_- \rightarrow b_- \succ r_{420} \\ r_{420} : \sigma_i \rightarrow \sigma_i | \\ \quad b_+a_+ \rightarrow a_+\pi, b_-a_- \rightarrow a_-\pi, \\ \quad b_+a_- \rightarrow a_+, b_-a_+ \rightarrow a_- \\ r_{43}^{\rightarrow} : [\sigma_i, \sigma_1] \rightarrow [\sigma_i, \sigma_1] \\ \quad (\pi, \lambda) \rightarrow (\lambda, \pi_0) \\ r_{44}^{\rightarrow} : \sigma_1 \rightarrow \sigma_1 | \pi_0^N c \rightarrow \pi_h(\text{halt}) \\ r_{45}^{\rightarrow} : \sigma_1 \rightarrow \sigma_1 | cc \rightarrow c \succ \\ \quad [\sigma_1, \sigma_i] \rightarrow [\sigma_1, \sigma_i] | (\text{continue}) \\ \quad (\pi_0^x c, \gamma^N) \rightarrow (\pi_\sigma, \gamma^N \pi_c), x < N \end{cases} \quad (4.10)$$

where loop controlling rules are listed as F_5 . If we found a symbol π_h in σ_1 , then the computation is complete.

$$F_5 = \left\{ \begin{array}{l} r_{51}^{\rightarrow} : \\ [\sigma_N, \sigma_1] \rightarrow [\sigma_N, \sigma_1] | (\gamma^{N-1} \pi_c, \lambda) \\ \rightarrow (\alpha \beta, \alpha_s \alpha^{N-1} \alpha_0^{N_1} \gamma_+^{N-1}) \\ \sigma_i \rightarrow \sigma_i (i \neq 1, N) | a_i \rightarrow a_i \alpha \beta \end{array} \right. \quad (4.11)$$

The total number of loops is written in the symbol π_σ .

4.2 Configurations and Computations

Now we explain the process of computation in detail. We use $i_0 = 1$ to note the output cell. For initial configuration, the multisets of edges are $\{\lambda\}$ and the node cells have the following multisets.

$$\{a_1 \alpha_s \alpha^{N-1} \alpha_0^{N_1} \gamma_0^{N-1}, a_2 \alpha \beta, \dots, \alpha_N \alpha \beta\} \quad (4.12)$$

The the running node of the neural network is $i = 1$. To illustrate the process more clearly, we assume that the current running node is $i = 2$ after the first round of running for $i = 1$. In the first node cell, the symbol π_σ notes the number of running rounds for the network. Hence the current configuration is

$$\{a_1 \alpha \beta \pi_\sigma, a_2 \alpha_s \alpha^{N-1} \alpha_0^{N_2} \gamma_0^{N-1}, a_3 \alpha \beta, \dots, \alpha_N \alpha \beta\} \quad (4.13)$$

Now for each $1 \leq i \leq N$ and $i \neq 2$ we run the rules in F_2 and the node cells are

$$\{a_1 \beta \pi_\sigma, a_2 \alpha_s \alpha_0^{N_2} \gamma^{N-1}, a_3 \beta, \dots, \alpha_N \beta\} \quad (4.14)$$

At the same time the edge cells σ_{2i} for $1 \leq i \leq N$ are

$$\{a_2 a_1 a^{w_{21}}, \lambda, a_2 a_3 a^{w_{23}}, \dots, a_2 a_N a^{w_{2N}}\} \quad (4.15)$$

Next we run the rules in F_{01} and the edge cells remains the same while the node cells are

$$\{a_1 \beta \pi_\sigma c, a_2 \alpha_0^{N_2} \gamma^N b_+^{w_{22}}, a_3 \beta, \dots, \alpha_N \beta\} \quad (4.16)$$

Next we run rules in F_1 on edges, and all the strings change to b_+ of b_- which is noted by $b_{+/-}$.

$$\{b_{+/-} a^{w_{21}}, \lambda, b_{+/-} a^{w_{23}}, \dots, b_{+/-} a^{w_{2N}}\} \quad (4.17)$$

The next step is to run F_1^{ij} on edges and the edge cells change into empty multisets and these states remain till the end of this round.

$$\{\lambda, \lambda, \lambda, \dots, \lambda\} \quad (4.18)$$

Meanwhile the node cells are

$$\{a_1 \beta \pi_\sigma c \alpha_{+/-}^{w_{12}}, a_2 \alpha_0^{N_2} \gamma^N b_{+/-}^{w_{22}} \alpha_{+/-}^{w_{12}} \alpha_{+/-}^{w_{23}} \dots \alpha_{+/-}^{w_{2N}}, a_3 \beta \alpha_{+/-}^{w_{23}}, \dots, \alpha_N \beta \alpha_{+/-}^{w_{2N}}\} \quad (4.19)$$

Now we run rules $\{r_{01}, r_{02}, r_{03}, r_{04}, r_{00}\}$ in F_0 and count the number of c in σ_1 .

$$\{a_1 \pi_\sigma c^{N+N_2}, a_2 \gamma^N b_{+/-}, a_3, \dots, \alpha_N\} \quad (4.20)$$

Now we compare the resulting $b_{+/-}$ with a_2 and then

$$\{a_1 \pi_\sigma c^{N+N_2} \pi_0^x, a_2 \gamma^N, a_3, \dots, \alpha_N\} \quad (4.21)$$

Actually in one running we only obtain one possible correct node which is noted by the symbol π_0 . If after all the N nodes are processed we obtain π_0^N correct nodes, then the network is stable. We then send a symbol π_h to the output node the computation is complete. Otherwise we send a continuing symbol π_c to σ_1 and continue to the next step. The function of rules F_3 and F_5 is to control the running of current node to the next node.

The next figure (Fig. 1) illustrate the whole computation process when $N = 4$ and $i = 2$.

5 Example and Discussion

In this section we will give an example taken in UCI data set to show the cluster splitting technique by graph P systems. Then we will present some comparison with genetic algorithms. Experiments show that computation by Hopfield neural networks is rather efficient especially in comparison with genetic algorithms.

The Wisconsin breast cancer data set consists 699 samples with 9 attributes. The samples are classified as two categories, the malignant and the benign. The data is shown in Fig. 2 with an added noise of $[0, 0.5]$. For the purpose of this paper, we choose attribute 2 and 6 and accumulate the data samples using the grid technique. Then the data set $\Omega = \{\xi_1, \dots, \xi_N\}$ is shown in Fig. 3 with $N = 100$ data points. The number of raw data points in ξ_i is denoted by $m(\xi_i)$.

Now we define a similarity measure on the data set. For $\xi_i, \xi_j \in \Omega$, let $L(\xi_i, \xi_j)$ be the shortest path length as shown in Fig. 3. Here we always take the unit length between two nodes as one. Similarity measure is defined as follows:

$$\rho(\xi_i, \xi_j) = \left[f(m(\xi_i), m(\xi_j)) \frac{L_0}{1 + L^2(\xi_i, \xi_j)} + 0.5 \right] \quad (5.1)$$

where L_0 is a parameter, and the operator $[\cdot]$ denote the largest integer not exceeding it. The function $f(\cdot, \cdot)$ is as follows:

$$f(u, v) = \min\{|u|, |v|\} \frac{1}{1 + |u - v|} \quad (5.2)$$

In the plane, if we define the coordinates of ξ_i as (p_i, q_i) . Then

$$L(\xi_i, \xi_j) = |p_i - p_j| + |q_i - q_j|, \quad (5.3)$$

To reduce the global effect of edges, we also define another similarity measure.

$$\rho_r(\xi_i, \xi_j) = \left[f(m(\xi_i), m(\xi_j)) \frac{L_0 \delta(\xi_i, \xi_j)}{1 + L^2(\xi_i, \xi_j)} + 0.5 \right] \quad (5.4)$$

The additional parameter $\delta(\xi_i, \xi_j)$ is defined by

$$\delta(\xi_i, \xi_j) = \begin{cases} 1, & \text{if } \max\{|p_i - p_j|, |q_i - q_j|\} \leq 3 \\ 0, & \text{else.} \end{cases} \quad (5.5)$$

Next we analyze the computation with similarity measure $\rho(\xi_i, \xi_i)$. Then we obtain the weight matrix $W = [W_1, W_2, W_3, W_4]^T$ as shown in equations (5.6)(5.7) (5.8)(5.9) (5.10)(5.11) (5.12)(5.13), where each subsequent two equations compose of of the matrices W_1, W_2, W_3, W_4 each, in such a way as $W_i = [A, B]^T$.

Fig. 1 A flow chart computation process of a graph P system

Fig. 2 Wisconsin breast cancer data with perturbation. Attribute 2 and 6

Fig. 3 Data nodes of the Wisconsin breast cancer data

[illegible]

(5.7)

(5.8)

(5.9)

(5.11)

(5.12)

[illegible]

The graph displays the energy curve of the proposed algorithm. The Y-axis represents Energy, ranging from -5000 to -1000. The X-axis represents Running Time, ranging from 0 to 600. The energy starts at approximately -1500 at time 0, drops sharply to about -4500 by time 20, and then stabilizes around -4700 after time 50.

Fig. 4 Energy curve with maximum running time 500

Fig. 5 Energy curve with maximum running time 1000

Fig. 6 Running time comparison of GA with MC

Next we compare the graph P system with traditional genetic algorithms (GA) to see the computation efficiency. First we set GA population size as 10. Mutation probability is $p = 1$ with one-point mutation. The next figure (Fig. 6) shows the running time comparison result. Here we set the maximum running time is 100. The energy comparison is shown in Fig. 7.

Now we change some parameters of GA. Let the population size be 100. We choose two individuals each time to be mutated. When the maximum running time is 500, we obtain the energy comparison as shown in Fig. 8. The figure shows clearly that the proposed method in this paper has better performance than GA.

Finally we present another experiment with parameters as the previous one. To make it more clearly we draw the energy comparison in the tail part. Notice that we only use decimal

Fig. 7 Energy comparison of GA with MC

Fig. 8 Energy comparison of GA with MC with running time 500

part of the energy value and multiply these values by 1000. Again we obtain a better result (Fig. 9).

6 Conclusion

In this paper we present a new membrane computing model called the graph P system. Topologically, the new system extends traditional P system to more complicated topological structure. In some sense, this means a new membrane computing model is set up with new data structure, namely, simplex structure. The main ingredients of the new structure is that

Fig. 9 Energy comparison with tail part (decimal_part*1000).

topological data structure turns from one dimensional structure to multidimensional structure. In the simplest case, the graph case namely, we add connections between nodes and edges. This will induce new paradigm of rules in association with this new connections which will hopefully solve more complicated problems by simulating new operations of multisets. However, computing ability of the new P system is still an open problem.

We also present a new solution to clustering splitting problem which is the most difficult task in hierarchical cluster analysis by the Hopfield neural networks. Surprisingly the new method works in a very efficient mode than traditional heuristic optimization method such as genetic algorithms. Computation of the neural networks are done by the proposed graph P system which is new in itself. Finally we will point out that the graph based P system, or more generally the simplex P system, is a promising new direction of membrane computing. We expect new research on this area with more powerful applications.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Alhazov A, Rogozhin Y, Verlan S (2010) A small universal splicing P system. In: Gheorghe M, Hinze T, Păun G, Rozenberg G, Salomaa A (eds) Membrane computing. CMC 2010. Lecture notes in computer science, vol 6501. Springer, Berlin, pp 95–102
2. Amiri B (2012) Application of teaching-learning-based optimization algorithm on cluster analysis. J Basic Appl Sci Res 2:11795–11802
3. Annadurai S, Kalyani T, Dare VR, Thomas DG (2008) Trajectory P system. Prog Nat Sci 18:611–616
4. Bernardini F, Freund R (2006) Tissue P systems with communication modes, WMC 7. LNCS 4361:170–182

5. Ceterchi R, Martín-Vide C (2003) P systems with communication for static sorting, GRLMC Report 26. In: Cavaliere M, Martín-Vide C, Paun G (eds) Proceedings of the brainstorming week on membrane computing. Rovira i Virgili University, pp 101–117
6. Chandra PH and Saroja Theerds Kalavathy SM (2013) Array P systems with hybrid teams. In: Proceedings of 7th international conference on bio-inspired computing: theories and applications, (BIC-TA 2012), advances in intelligent systems and computing 201, pp 239–249
7. Chen CY, Chou WC, Tsay W, Tang JL, Yao M, Huang SY, Tien HF (2013) Hierarchical cluster analysis of immunophenotype classify AML patients with NPM1 gene mutation into two groups with distinct prognosis. *BMC Cancer* 13:1–9
8. Chen Z, He J, Zheng Y, Song T, Deng Z (2016) An optimized feedforward decoupling PD register control method of roll-to-roll web printing systems. *IEEE Trans Autom sci Eng* 13:74–283
9. Christinal HA, Díaz-Pernil E, Real P (2010) P systems and computational algebraic topology. *Math Comput Model* 52:1982–1996
10. Darkins R, Cooke EJ, Ghahramani Z, Kirk PDW, Wild DL, Savage RS (2013) Accelerating Bayesian hierarchical clustering of time series data with a randomised algorithm. *PLoS ONE* 8:59795
11. Frisco P, Govana G, Leporati A (2012) Asynchronous P systems with active membranes. *Theor Comput Sci* 429:74–86
12. Gheorghe M, Sosík P, Vavrečková Š et al (2014). In: Proceedings of the 15th international conference on membrane computing, Prague, Czech Republic, pp 1–377
13. Han J, Kamber M (2002) Data mining, concepts and techniques. Higher Education Press; Morgan Kaufmann Publishers, Beijing
14. Huang RZ, Yu G, Wang ZJ, Zhang J, Shi LX (2013) Dirichlet process mixture model for document clustering with feature partition. *IEEE Trans Knowl Data Eng* 25:1748–1759
15. Ji M, Xie F, Ping Y (2013) A dynamic fuzzy cluster algorithm for time series. *Abstr Appl Anal* 183410
16. Lim HJ, Oh YS, Lim JS, Dong HL (2012) Microarray gene expression data mining: cluster analysis and applications. *Eng Technol* 70:373–375
17. Linqiang Pan, Xiangxiang Zeng, Xinyi Zhang (2012) Spiking neural P systems with weighted synapses. *Neural Process Lett* 35:13–27
18. Liu XY and Xue Alice (2012) Communication P systems on simplicial complexes with applications in cluster analysis, *Discret Dyn Nat Soc* 415242
19. Liu EZF, Lin CH, Chen FY, Peng PC (2012) Cluster analysis of adolescent blogs. *Turk Online J Educ Technol* 11:69–79
20. Liu X, Li Z, Liu J, Liu L, Zeng XX (2015) Implementation of arithmetic operations with time-free spiking neural P systems. *IEEE Trans Nanobiosci* 14:617–624
21. Liu X, Liu H (2008) Artificial neural networks and particle swarm optimization. Beijing University of Posts and Telecommunications Press, Beijing
22. Matta VP, Krithivasan K, Garg D (2011) Modelling and analysis of spiking neural P systems with anti-spikes using Pnet lab. *Nano Commun Netw* 2:141–149
23. Nordsborg BR, Meliker JR, Kjær A (2013) Ersbøll, G.M. Jacquez, and O. Raaschou-Nielsen, Space-time clustering of non-Hodgkin Lymphoma using residential histories in a Danish case-control study. *PLoS ONE* 8:60800
24. Obtułowicz A, Păun G (2003) (In search of) probabilistic P systems. *BioSystems* 70:107–121
25. Păun G, Rozenberg G, Salomaa A (2010) Membrane computing. Oxford University Press, New York
26. Păun G, Pérez-Jiménez MJ (2012) P automata revisited. *Theor Comput Sci* 454:222–230
27. Peng H, Wang J, Pérez-Jiménez MJ, Wang H, Shao H, Wang T (2013) Fuzzy reasoning spiking neural P system for fault diagnosis. *Inf Sci* 235:106–116
28. Singh SR, Lal S, Ahmed N, Srivastava KK, Kumar D, Jan N, Amin A, Malik AR (2013) Determination of genetic diversity in strawberry (*Fragaria ananassa*) using principal component analysis (PCA) and single linkage cluster analysis (SLCA). *Afr J Biotechnol* 12:3774–3782
29. Spicher A, Michel O, Cieslak M, Giavitto JL, Prusinkiewicz P (2008) Stochastic P systems and the simulation of biochemical processes with dynamic compartments. *BioSystems* 91:458–472
30. Verlan S, Bernardini F, Gheorghe M, Margenstern M (2008) Generalized communicating P systems. *Theor Comput Sci* 404:170–184
31. Xiao JH, Zhang XY, Xu J (2012) A membrane evolutionary algorithm for DNA sequence design in DNA computing. *Chin Sci Bull* 57:698–706
32. Yang Y, Yang Y, Shen HT, Zhang YC, Du XY, Zhou XF (2013) Discriminative nonnegative spectral clustering with out-of-sample extension. *IEEE Trans Knowl Data Eng* 25:1760–1771
33. Yu J, Tao D, Wang M (2012) Adaptive hypergraph learning and its application in image classification. *IEEE Trans Image Process* 21(7):3262–3272

34. Yu J, Hong R, Wang M, You J (2014) Image clustering based on sparse patch alignment framework. *Pattern Recognit* 47(11):3512–3519
35. Yuan Z, Zhang Z (2007) Asynchronous spiking neural P system with promoters. In: Xu M, Zhan Y, Cao J, Liu Y (eds) *Advanced parallel processing technologies. APPT 2007. Lecture notes in computer science*, vol 4847. Springer, Berlin, pp 693–702
36. Yu J, Yang X, Gao F, Tao D (2016) Deep multimodal distance metric learning using click constraints for image ranking. In: *IEEE transactions on cybernetics*, pp 1–11
37. Zeng XX, Lu C, Pan LQ (2010) A weakly universal spiking neural P system. *Math Comput Model* 52:1940–1946