A P System for K-Medoids-Based Clustering

Ping Guo, Jingya Xie
College of Computer Science
Chongqing University
Chongqing,400044,China

***Abstract*—The membrane computing model, also known as the P system, is a parallel and distributed computing system. K-medoids algorithm is one of the most famous algorithms in partition-based clustering algorithms, and has been widely used in data analysis and modern scientific research. Combining the P system with the k-medoids algorithm, the maximum parallelism calculated by the P system can effectively reduce the time complexity of the k-medoids clustering algorithm. Based on this, this paper proposes a cell-like P system with promoters and inhibitors based on k-medoids clustering, and then an instance is given to illustrate the practicability and effectiveness of the P system designed.**

***Keywords*—P systems; Clustering; K-medoids-based clustering; Membrane computing; Parallel and distributed computing**

I. Introduction

Membrane computing[1,2], which is initiated by Pun in 1998, is a branch of molecular computing. The computing models in the framework of membrane computing, also called P systems, are distributed, non-deterministic and maximally parallelized[1]. P systems are inspired from the compartmental structure and the way to process chemical compounds of alive cells, cells in tissue, organs, etc. Up to now, many variants of P systems have been investigated, mainly including cell-like P systems[1,3,4], tissue-like P systems[5-7] and neural-like P systems[8,9]. P systems have been studied in many areas, such as biology[10], linguistics[11], computer science, mathematics[12], etc.[13]. Many variants are universal computationally, likewise it has been proved that P systems have the computing capacity with the equivalent of Turing machine[14,15]. Besides, more information about P systems can be found at the website of Ref[16].

Information plays an increasingly important role in modern society. Consequently, the issue of crucial importance is data analysis. Clustering is a basic and significant composition of data analysis, and it is employed as an ordinary method in modern science research[17]. However, it is not come to an agreement with the complete definition for clustering. The classic one is described as: instances in the same cluster must be similar as much as possible, instances in the different clusters must be different as much as possible and measurement for similarity and dissimilarity must be clear and have the practical meaning[18]. Generally speaking, Clustering algorithms are divided into traditional ones and modern ones, where traditional ones are based on partition[19,20], fuzzy theory[21], distribution[22,23], density[24, 25], grid[26-28], graph theory[29,30], etc. It is applied crossing communication science, computer science, biology science, etc. Clustering is also introduced to membrane computing[31-33]. For the data clustering problem, ref.[33] presents a novel clustering algorithm based on a tissue-like P system with loop structure of cells, called membrane clustering algorithm, to realize a local neighborhood topology, and proves the high efficiency and competitiveness of the proposed algorithm. To deal with self-driven clustering problem, ref.[32] proposes a membrane clustering algorithm based on a tissue-like P system with fully connected structure to solve how many clusters is the most appropriate and what does a good clustering partitioning look like at the same time. It develops an improved velocity-position model as evolution rules and proves the competitiveness of the propose algorithm either. Ref. [31] proposes the k-medoids-based consensus clustering based on a cell-like P system with inhibiters and promoters by means of introducing k-medoids algorithm and cell-like P system with inhibiters and promoters to the consensus clustering, and it is proved to be highly accurate and highly efficient. K-medoids[34] is a melioration of k-means, and these two are the most famous ones of clustering algorithm based on partition. K-medoids deals with discrete data and designates the data point most near to cluster center as medoid. This method is more robust to noise and outliers as compared to k-means due to minimizing a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances, but is suitable for small data sets because of larger calculation. However, the time complexity can be decreased by cell-like P systems with promoters and inhibiters, because it has the inherent mechanism of parallel and distributed computing. In short, the maximum parallelism of P systems contributes to improving algorithm efficiency of unsupervised learning.

In this paper, a P system Π_{kmbc} is proposed to implement a kind of k-medoids-based algorithm which is modified mildly to adjust with the evolution mechanism of P Systems to achieve clustering. The paper is organized as follows: Section II introduces the basic knowledge of the k-medoids algorithm and cell-like P system with priority and promoters; Section III proposes the design of the P system Π_{kmbc} with its k-medoids-based algorithm, definition and rules discussed in detail. Subsequently, an instance is given in section IV; and the conclusions are drawn in Section V.

II. Foundation

A. The k-medoids algorithm

The k-medoids algorithm proposed in 1987[34] is a classical partitioning algorithm of clustering related to the k-means algorithm. Both the k-medoids and the k-means algorithms are to cluster the data set of n objects into k (a known priori) clusters and to minimize the distance between points in the same cluster and a point called medoid which is designated as the center of that cluster. A medoid is a most centrally located point in the cluster.

As the K-Medoids algorithm is improved by the K-Means algorithm, it is partitioning around medoids instead of means. The K-Medoids algorithm is more robust to outliers and noise than the K-Means algorithm due to choosing medoids as centers and minimizing a sum of pairwise dissimilarities. However, the same characteristic is that the actual definition of the distance has various alternatives according to the requirement of actual problems. The smaller the sum of distances between each two data points is, and the more similar the data points in same cluster are, the more dissimilar the data points from different clusters are, the better the clustering result is.

The most representative realization of k-medoids algorithm is the Partitioning Around Medoids (PAM) algorithm. PAM uses a greedy search which may not find the optimum solution, but it is faster than exhaustive search. It works as follows[35]:

1)Initialize: select k of the n data points as the medoids.

2)Associate each data point to the closest medoid.

3)While the cost of the configuration decreases, repeat step 4).

4)For each medoid m, for each non-medoid data point o, repeat step 5) to 6).

5)Swap m and o, recalculate the sum of distances of data points to their medoid.

6)If the total cost of the configuration increased in the previous step, undo the swap.

Algorithms other than PAM have also been suggested in the literature [36,37]. Voronoi iteration method is included, which is more simple and faster than PAM. The steps of Voronoi iteration method are as follows:

1)Initialize: select k of the n data points as the medoids.

2)Repeat step 3) to 4) while the cost decreases.

3)In each cluster, make the data point that minimizes the sum of distances within the cluster the medoid.

4)Reassign each data point to the cluster defined by the closest medoid determined in the previous step.

B. Cell-like P System with Priority and Promoters

There are many variants of P systems already introduced in section 1. This paper is only related to cell-like P systems with priority and promoters. Therefore, this piece gives the basic concepts about cell-like P systems with priority and promoters.

There are three main components to cell-like P system: the membrane structure, objects and rules. As suggested by Fig.1, the membrane structure is a hierarchically arranged set of membranes which are usually identified by labels from a given set and divide a cell-like P system into separated regions. A membrane which does not contain any other membranes is called elementary. The membrane which contains all the other membranes is referred as the skin. Each Membrane only determines a region bordered above by itself and below by the membranes placed directly inside, if any exists. Rules are only effective in the region of the membrane they belong to. Objects which are expressed by characters or string of symbols can evolve to new objects or be transferred to new regions

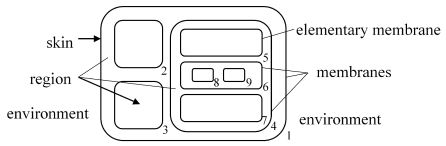

Fig. 1. The membrane structure of cell-like P system[38]

according to the rules in the membrane whose region objects appear in.

Formally, a cell-like P system (of degree $m \geq 1$) with priority and promoters is of the form

$$\Pi = (O, \mu, M_1, M_2, \cdots, M_m, i_{out}) \quad (1)$$

where,

1)O is the alphabet of objects used in Π;

2)μ is the membrane structure of Π, and degree is m;

3)M_i = (ω_i, R_i, ρ_i) defines the membrane with label i in Π (i=1,2,$\cdots$,m), where ω_i is the multiset including the initial objects in membrane i, object λ (an empty string) means there is no object in membrane i, R_i is a set of rules in membrane i, promoters present in rules, ρ_i specifies a priority relation among the rules in R_i, smaller value means higher executing priority;

4)i_{out} is appointed as output region which saves the results.

The form of rules in Π is: $(u \rightarrow v)$ or $(u \rightarrow v|_\alpha, \rho)$ or $(u \rightarrow (w)_i)$, where, $u \in O^+$, $v \in (O \times Tar)^*$, α is promoter. O^* is the finite and non-empty multiset over O, $O^+ = O^* - \{\lambda\}$, $Tar = \{here, out, in_i | 1 \leq i \leq m\}$. Objects appear in u will be consumed. If v appears in form of $(a, here)$, a will remain in the membrane where the corresponding rule is applied. If v appears in form of (a, out), a will become an object of the region immediately outside the membrane where the corresponding rule is applied. If v appears in form of (a, in_i), a will be produced in membrane i. Object δ appeared in v means dissolving the membrane where δ presents in and releasing all objects in this membrane to the region immediately outside this membrane. The second form means executing $u \rightarrow v$ with the priority of ρ when α exists in the membrane where the corresponding rule is applied. Priority and promoters can both appear in this form or only one. The third form means creating a membrane labeled i and adding w to the multiset in membrane i.

Rules are executed according to the principles of non-determinism and maximal parallelism in each membrane. Only several rules are chosen non-deterministically when more rules can possibly be applied. All rules that can be applied must be applied concurrently. These two principles are limited by reactant in a membrane. As only dealing with cell-like P systems, the rest of this paper refers to the cell-like P system as the P system for brevity.

III. The Design of P System Π_{kmbc}

This paper aims to obtain a P system Π_{kmbc} for clustering based on k-medoids method. The algorithm for Π_{kmbc} is discussed before the definition of Π_{kmbc} is designed.

A. The algorithm for Π_{kmbc}

The algorithm for Π_{kmbc} is modified from PAM and Voronoi iteration method which are mentioned in subsection A in section II.

Suppose $T = \{p_1, p_2, , p_n\}$ denotes a dataset with n data points which can be multi-dimensional vectors. This paper supposes them two-dimensional, and names the input data set $T_{kmbc} = \{p_i = (x_i^{a_i}, y_i^{b_i}) | 1 \leq i \leq n, a, b \in N^+\}$ where a_i and b_i represents the number of x_i and y_i respectively. All data points are divided into k (k$\leq$ n) clusters $C = \{C_1, C_2, , C_k\}$. Each data point belongs to and only belongs to one cluster. Default medoids are first k data points of T_{kmbc}.

According to subsection A in section II, distances between un-medoid data points and medoids need to be calculated repeatedly. In order to reduce calculation amount and realize in P systems, the algorithm for Π_{kmbc} introduces definitions of the distance matrix, and the point-medoids distances set, the point-point distances set, the sum of the point-point distances set. In addition, this paper considers squared Euclidean distance.

Suppose D_{nn} the distance matrix:

$$D_{nn} = \begin{pmatrix} 0 & d_{1,2} & \cdots & d_{1,n} \\ 0 & 0 & \cdots & d_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & d_{n-1,n-1} \\ 0 & 0 & \cdots & 0 \end{pmatrix} \quad (2)$$

where $d_{i,j}$ is the distance between p_i and p_j.

Suppose D_i the point-medoids distances set which contains k distances associated with p_i and k medoids, and it meets:

$$D_i = \{d_{j,i}, j \leq i \wedge j \in S_m\} \cup \{d_{i,j}, i \leq j \wedge j \in S_m\} \subset D \quad (3)$$

where S_m is a set of k numbers which correspond to k subscripts of k medoids, D is a set whose objects equal to all the nonzero objects of D_{nn}.

Suppose $D_i^{'}$ the point-point distances set which contains distances associated with p_i and all the other data points belonged to the same cluster with p_i, and it meets:

$$D_i^{'} = \{d_{j,i}, j \leq i \wedge j \in S_{cm}\} \cup \{d_{i,j}, i \leq j \wedge j \in S_{cm}\} \quad (4)$$

where, S_{cm} is a set of numbers which correspond to subscripts of all the data points in cluster $C_m (C_m \subset C, m \in [1, k])$. Suppose $d_i^{'}$ the sum of $D_i^{'}$.

It uses unique data point mark ε_i to replace p_i for convenient. Therefore, unique data point marks are assigned to clusters instead of data points benefited from introducing all the definitions above.

Suppose z the number of iterations, and it is assigned artificially.

Based on PAM and voronoi iteration method, all the definition introduced, the algorithm for Π_{kmbc} is mainly composed of initialization, initial assignment and iterative assignments. The algorithm flow for Π_{kmbc} is shown in Fig.2. The detailed flow of the algorithm is described as follow.

The algorithm for Π_{kmbc} computes D_{nn} and select first k of the n data points as the medoids at first. It compute D_i for each p_i in T_{kmbc} and associates each data point mark to the closest medoid according to the minimal item in D_i next. For un-medoid data point p_i, this algorithm elects the minimal item $d_{i,j}$ or $d_{j,i}$ in D_i, then assigns ε_i to the cluster whose medoid is p_j. Then iterative steps go. The first step of iterative steps is to compute $D_i^{'}$ and $d_i^{'}$ for each data point and to designate the new medoid according to the minimal $d_i^{'}$ in each cluster. That is to elect theS p_i which corresponds to the minimal $d_i^{'}$ as the new medoid in each cluster. The second step of iterative steps is to reassign each data point mark to the cluster defined by the closest medoid determined in the previous step. Moreover, the implement of assignments in iterative step are similar to that of the initial assignment. Due to the mechanism of parallel and distributed computing, all data points are processed in parallel in this algorithm.

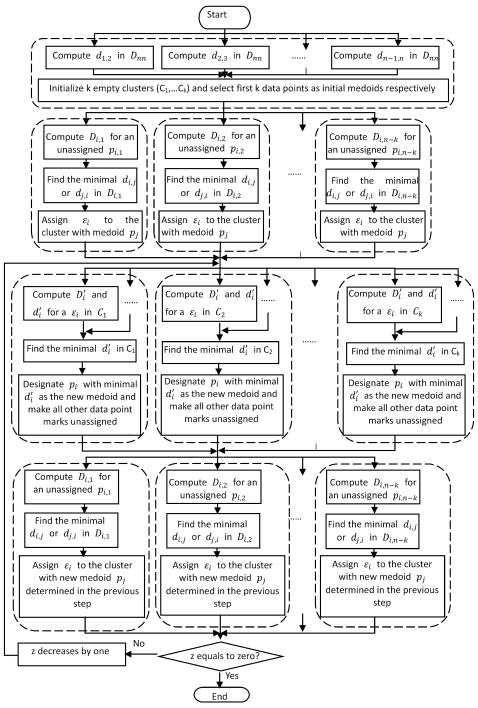

Fig. 2. The algorithm flow for Π_{kmbc}

B. The definition of Π_{kmbc}

Based on the algorithm for Π_{kmbc} which discussed in subsection A in section III, the definition of P system Π_{kmbc} is figured out and as follow:

$$\Pi_{kmbc} = (O, \mu, M_A, M_{B_i}, M_C, M_{D_{i,j}}, M_{E_i}, M_{F_j}, i_{out}) \quad (5)$$

where,

1)$O = \{x_i, y_i, x_{i,j}, a_{i,j}, d_i, d_{i,j}, h_i, g_i, t_i, t_{i,j}, \varepsilon_i, \varepsilon_{i,j}, \alpha_i, \beta_i, \gamma_i, \xi_i | 1 \leq i, j \leq n\} \cup \{s, a_0, a_1, b_1, c, c_1, c_2, d, d_1, d_2, e, z, \alpha, \alpha\prime, \alpha\prime\prime, \beta, \gamma, \delta, \eta, \theta, \lambda, \mu, \xi, \pi, \rho, \sigma, \omega\}$. This multiset includes objects which are related to data points, which control the application of the rules, and which are special in P system (such as δ, λ, s and ω). The most important objects are d_i, $d_{i,j}$, ε_i, $\varepsilon_{i,j}$ and $t_{i,j}$. Objects d_i denote the sum of distances between data point p_i to all other data points in the same cluster. Objects $d_{i,j}$ denote the distance between data point p_i

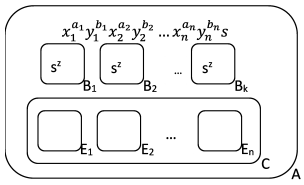

Fig. 3. The initial membrane structure of Π_{kmbc}

to the medoid whose subscript is j. Object ε_i which is unique in Π_{kmbc} denotes data point p_i or medoid p_i (and then it is called data point mark or medoid mark). Object $\varepsilon_{i,j}$ is designed to record the information that cluster i disignates the data point whose subscript is j as the medoid. Object $t_{i,j}$ is designed as transmission mark to record the information that data point p_i belongs to the same cluster with the medoid whose subscript is j. Other objects are designed to ensure the integrity of data information or as flow controller.

2)$\mu = [_A[_{B_1}]_{B_1}[_{B_2}]_{B_2}[_{B_k}]_{B_k}[_C[_{E_1}]_{E_1}[_{E_2}]_{E_2}[_{E_n}]_{E_n}$ $]_C]_A$ is the initial membrane structure of Π_{kmbc} as shown as what Fig.3 illustrates, and it will change with the evolution of Π_{kmbc}. As suggested by Fig.3, membrane A is the skin which contains all other membranes. The class membranes of B_i are designed as clusters in which data point marks distributed to them are placed and new medoids are computed. They also are storage areas of terminal results. Membrane C is designed to deposit the distance matrix. The class membranes of E_i are designed to compute the point-medoids distances set and decide where unassigned data points will belong. One membrane of E_i corresponds with one unassigned data point, and they have the same subscript. With the evolution of Π_{kmbc}, the class membranes of $D_{i,j}$ and F_j are dynamically generated in membrane A and the class membranes of B_i respectively. The class membranes of $D_{i,j}$ compute the distance matrix and are dissolved after computation. One membrane $D_{i,j}$ corresponds with one nonzero element in the distance matrix, and they have the same subscript. The class membranes of F_j compute the point-point distances set and the sum of the point-point distances set and are also dissolved after computation. The delivery of objects between other membranes is achieved in Membrane A. At the same time, information transferring or calculation flow comes true.

3)$M_{mem} = (w_{mem}, R_{mem}, \rho_{mem})$ represents a membrane of Π_{kmbc} in which $mem \in \{A, C\} \cup \{D_{i,j}, E_i, F_j | 1 \leq i \leq n, 1 \leq j \leq n\} \cup \{B_i, | 1 \leq i \leq k\}$, $w_A = T_{kmbc} \cup \{s\}$, $w_{B_i} = \{e^z\}$, $w_C = w_{D_{i,j}} = w_{E_i} = w_{F_j} = \lambda$, R_{mem} and ρ_{mem} are described in hand as R_{mem} for convenience, and more details about R_{mem} are in subsection C in section III.

4)$i_{out} = \{B_1, B_2, \dots, B_k\}$.

In addition, child membrane that dynamically generated in father membrane inherits all rules in father membrane. This paper eliminates the influence of inherited rules on child membrane when designing all the rules in Π_{kmbc}. As a result, rule sets of dynamically generated membranes only contain un-inherited rules for concision. The rules in Π_{kmbc} are elaborated in next subsection and calculation processes too.

C. The rules in Π_{kmbc}

For the sake of being understand, the rule sets R_{mem} in Π_{kmbc} are explained in order of operation flow. Of these, ρ_{mem} whose value is 1 or 2 or 3 specifies a priority relation among a rule set of R_{mem}, and smaller value means higher executing priority.

1)Initialization

a)Preparation

In P system Π_{kmbc}, the rules in R_A associated with the preparation are:

$r_1 : s \to \alpha_1\alpha_2 \dots \alpha_{n-1}\theta, 1;$
$r_2 : x_i \to (x_i, here)(h_i)_{D_{i,i+1}} \dots (h_i)_{D_{i,n}}|_{\alpha_i}, 1, 1 \leq i \leq n-1;$
$r_3 : \alpha_i \to \beta_i, 1, 1 \leq i \leq n;$
$r_4 : y_i \to (y_i, here)(g_i, in_{Di,i+1}) \dots (g_i, in_{Di,i+n})|_{\beta_i}, 1, 1 \leq i \leq n-1;$
$r_5 : \beta_i \to \gamma_{i+1}, 1, 1 \leq i \leq n-1;$
$r_6 : x_i \to (x_i, here)(h_i, in_{D1,i}) \dots (h_i, in_{Di-1,i})|_{\gamma_i}, 1, 2 \leq i \leq n;$
$r_7 : y_i \to (y_i, here)(g_i, in_{D1,i}) \dots (g_i, in_{Di-1,i})|_{\gamma_i}, 1, 2 \leq i \leq n;$
$r_8 : \gamma_i \to (\varepsilon_{i-1}, here)(\alpha, in_{D1,2}) \dots (\alpha, in_{Dn-1,n})|_{\gamma_i}, 1, 2 \leq i \leq k+1;$
$r_9 : \gamma_i \to \lambda, 1, k+2 \leq i \leq n;$

Rule of type r_1 is used to start the system, rules of types r_2-r_9 is used to create the class membranes of $D_{i,j}$ which are to compute D_{nn} for preparing initialization and to move objects which are data objects and flow controllers to these membranes. After rules of type r_2 and r_4 are applied, each membrane of type $D_{i,j}$ is created finished and contains objects who has the same subscript with the first subscript of itself. After rules of type r_6 and r_7 are applied, each membrane of type $D_{i,j}$ contains objects who has the same subscript with the second subscript of itself. Rule of type r_8 moves specific flow controllers to membrane $D_{i,j}$. Other rules assist in process control.

b)Compute and deposit D_{nn}

Rule set $R_{D_{i,j}}$ is associated with computation of D_{nn}, and each subscript i and j in $R_{D_{i,j}}$ equals to the first and second subscript of membrane $D_{i,j}$ respectively:

$r_1 : h_i \to a_0, 1;$ $r_2 : a_0 \to a_1, 1;$
$r_3 : g_i \to b_1, 1;$ $r_4 : a_1h_i \to \lambda, 1;$
$r_5 : a_1 \to c, 2;$ $r_6 : h_j \to c, 2;$
$r_7 : b_1g_j \to \lambda, 1;$ $r_8 : b_1 \to d, 2;$
$r_9 : g_j \to d, 2;$ $r_{10} : \alpha^k \to \alpha, 1;$
$r_{11} : \alpha \to \alpha^{'}\alpha^{''}, 2;$ $r_{12} : c \to c_1c_2, 1;$
$r_{13} : d \to d_1d_2, 1;$ $r_{14} : c_1 \to c_1d_{i,j}|_{c_2}, 1;$
$r_{15} : c_2\alpha^{'} \to \alpha^{'}, 1;$ $r_{16} : d_1 \to d_1d_{i,j}|_{d_2}, 1;$
$r_{17} : d_2\alpha^{''} \to \alpha^{''}, 1;$ $r_{18} : \alpha^{'}\alpha^{''} \to \beta, 2;$
$r_{19} : c_1 \to \lambda|_{\beta}, 1;$ $r_{20} : d_1 \to \lambda|_{\beta}, 1;$
$r_{21} : \beta \to (\delta, here)(\eta, out), 1;$

With the specific flow controller, rules in membrane $D_{i,j}$ start being executed. Rules of type r_4-r_9 are used to turn the differences of the first and second dimensions of the two data points to objects type of c and d. After rules of type r_{12}-r_{17} are applied for limited times, each membrane of type $D_{i,j}$ contains objects $d_{i,j}$ with the quantities of squared c and squared d.

Rule of type r_{21} means dissolving the membrane and send a specific flow controller which reflects the end of computation out.

The rule in R_A associated with depositing D_{nn} moves objects of type $d_{i,j}$ to membrane C:
$r_{10} : d_{i,j} \to (d_{i,j}, in_C)|_\eta, 1, 1 \le i \le n-1, 2 \le j \le n, i \le j;$

c)Initialize k medoids

The rules in R_A associated with the initialization of k medoids are:
$r_{11} : \varepsilon_i \to (\varepsilon_i, in_{B_i})(\varepsilon_i, in_C)|_\eta, 1, 1 \le i \le k;$
$r_{12} : \theta \to (\alpha, in_C)|_\eta, 1;$
$r_{13} : \eta \to \lambda, 1;$

After rule of type r_{11} is applied, each membrane B_i contains a data point mark who has the same subscript with the membrane. Other rules assist in process control.

2)Initial assignment

Rules of type r_{10}-r_{12} in R_A are used to move objects associated with computing D_i and the minimal item in D_i to membrane C . After that, rules of type r_1-r_6 in R_C are used to copy these objects into appointed membranes of type E_i:
$r_1 : d_{i,j} \to a_{i,j}|_{\varepsilon_i \varepsilon_j}, 1, 1 \le i \le n, 1 \le j \le n;$
$r_2 : d_{i,j} \to (d_{i,j}, here)(d_{i,j}, in_{E_j})|_{\varepsilon_i}, 2, 1 \le i \le n, 1 \le j \le n;$
$r_3 : d_{i,j} \to (d_{i,j}, here)(d_{i,j}, in_{E_i})|_{\varepsilon_j}, 2, 1 \le i \le n, 1 \le j \le n;$
$r_4 : \varepsilon_i \to (\varepsilon_i, in_{all}), 1, 1 \le i \le n;$
$r_5 : \alpha \to (\beta, here)(\alpha, in_{all}), 1;$
$r_6 : a_{i,j} \to d_{i,j}|_\beta, 1, 1 \le i \le n, 1 \le j \le n;$

Thus, each membrane of type E_i contains k medoid marks and k objects of type $d_{i,j}$ or $d_{j,i}$ which are all the items of D_i. Rule set R_{E_i} is associated with the computation of the minimal item in D_i, and each subscript i in R_{E_i} equals to the subscript of membrane E_i:
$r_1 : d_{i,j} \to d_j, 1, 1 \le j \le n;$ $r_2 : d_{j,i} \to d_j, 1, 1 \le j \le n;$
$r_3 : \alpha \to \beta, 1;$ $r_4 : \varepsilon_j d_j \to \eta \varepsilon_j, 1;$
$r_5 : \varepsilon_j \beta \to \gamma t_{i,j}|_\eta, 2, 1 \le j \le n;$ $r_6 : \beta \to \gamma, 3;$
$r_7 : \varepsilon_j \to \lambda|_\gamma, 1;$ $r_8 : d_j \to \lambda|_\gamma, 1;$
$r_9 : t_{i,j} \to (t_{i,j}, out)|_\gamma, 1;$ $r_{10} : \eta \to \lambda|_\gamma, 1;$
$r_{11} : \gamma \to \lambda, 1;$

When p_i is a un-medoid data point, rules of type r_4 and r_5 in R_{E_i} are used to consume d_j for all j existed in the membrane until there is no d_j left for a certain j, then turn the ε_j corresponding to the certain j to $t_{i,j}$. As consequence, transmission mark $t_{i,j}$ means that un-medoiod data point p_i belongs to the cluster with the medoid p_j, and rule of type r_9 is used to move $t_{i,j}$ out. Furthermore, rules of type r_7-r_{11} in R_{E_i}) are used to empty the objects in the membrane for the convenience of the next computation.

When p_i is a medoid, there is no need to compute, but some objects are contained in membrane E_i after the execution of rules of type r_2-r_5 in R_C. For the convenience of the next computation, rules of type r_3, r_6, r_7 and r_{11} in R_{E_i} are executed.

As transmission marks are contained in membrane C, rules of type r_7-r_{10} in R_C are used to move them out:
$r_7 : \beta t_{i,j} \to \gamma t_{i,j}, 1, 1 \le i \le n, 1 \le j \le n;$
$r_8 : d_{i,j} \to (d_{i,j}, here)(d_{i,j}, out)|_\gamma, 1, 1 \le i \le n, 1 \le j \le n;$
$r_9 : t_{i,j} \to (t_{i,j}, out)|_\gamma, 1;$
$r_{10} : \gamma \to (\mu, out), 1;$

Rule of type r_{14} in R_A is used to send a flow controller of type α to each membrane B_i:
$r_{14} : \mu \to (\alpha, in_{B_1}) \dots (\alpha, in_{B_k}), 1;$

With the flow controllers, k marks of type $\varepsilon_{i,j}$ which reflect that the ith cluster has a medoid with subscript j right now are copied into membrane A due to the execution of rule of type r_1 in R_{B_i}:
$r_1 : \alpha \varepsilon_j \to (\xi \varepsilon_{i,j}, out)(\varepsilon_j, here)|_\alpha, 1, 1 \le j \le n;$

Finally, there are objects of type $t_{i,j}$ and $\varepsilon_{i,j}$ in membrane A. That is to say, all the objects for initial assignment are completely prepared. Rules of type r_{15}- r_{18} in R_A are associated with initial assignment and preparation for the following step:
$r_{15} : d_{i,j} \to (d_{i,j}, in_{B_1}) \dots (d_{i,j}, in_{B_k}), 1, 1 \le i, j \le n;$
$r_{16} : t_{i,j} \to (\varepsilon_i, in_{B_m})|_{\varepsilon_{mj}}, 1, 1 \le i, j \le n, 1 \le m \le k;$
$r_{17} : \xi^{k-1} \to \lambda, 1;$
$r_{18} : \xi \to (\beta, in_{B_1}) \dots (\beta, in_{B_k}), 2;$
$r_{19} : \varepsilon_{i,j} \to \lambda, 1, 1 \le i \le n, 1 \le j \le n;$

Rule of type r_{16} is used to send data point marks to the certain membranes of type B_i under the premise of $t_{i,j}$ and $\varepsilon_{i,j}$. As a result, initial assignment is achieced. Meanwhile, D_{nn} and flow controller β are send to each membrane B_i due to rules of type r_{16} and r_{18}.

3)Iterative assignment

a)Update medoids

As membranes of type F_j are dynamically generated in each membrane B_i, the rules in R_{B_i} associated with it are:
$r_2 : \varepsilon_j \to (\varepsilon_j, here)(t_j)_{F_j}|_\beta, 1, 1 \le j \le n;$
$r_3 : \beta \to \lambda, 1;$
$r_4 : d_{p,q} \to (x_{p,q}, in_{all})|_\gamma, 1, 1 \le p \le n, 1 \le q \le n;$
$r_5 : \gamma \to (\sigma, in_{all})(\eta, here), 1;$
$r_6 : \eta \to \theta, 1;$

With the flow controller β, rule of type r_2 is used to generate membranes F_j each of which corresponds to a data point in the cluster. Rule of type r_4 is used to move D_{nn} and a flow controller σ to all the membranes F_j. Therefore, each membrane F_j contains all the objects associated with computing $D_i^{'}$ and $d_i^{'}$.

Each subscript j in R_{F_j} equals to the subscript of membrane F_j, and all rules in R_{F_j} are as follow:
$r_1 : x_{i,j} \to d_j|_{t_j}, 1, 1 \le i \le n;$
$r_2 : x_{j,i} \to d_j|_{t_j}, 1, 1 \le i \le n;$
$r_3 : x_{p,q} \to \lambda, 2, 1 \le p \le n, 1 \le q \le n;$
$r_4 : t_j \to \lambda|_\sigma, 1;$
$r_5 : \sigma \to \delta, 1;$

In each membrane F_j, rules of type r_1-r_2 turn objects of type x_{ij} to d_j where the first or second subscript of x_{ij} and the subscript of d_j equals to that of F_j. Thus, the quantities of d_j equals to $d_i^{'}$ which is the sum of $D_i^{'}$. Rule of type r_5 is used to dissolve the membrane and release objects of type d_j.

Rules of type r_7-r_8 in R_{B_i} are executed repeatedly to figure the object ξ_j which points to the minimal d_j:
$r_7 : \varepsilon_j d_j \to \varepsilon_j, 1, 1 \le j \le n;$ $r_8 : \varepsilon_j \theta \to \mu \xi_j, 2, 1 \le j \le n;$

That is to say, p_i whose subscript equals to that of ξ_j will be the new medoid. The rest rules in R_{B_i} are:
$r_9 : d_j \to \lambda|_\mu, 1, 1 \le j \le n;$
$r_{10} : \xi_j \to \varepsilon_j|_\mu, 1, 1 \le j \le n;$
$r_{11} : \varepsilon_j \to \lambda|_{\mu e}, 1, 1 \le j \le n;$
$r_{12} : e\mu \to \pi, 1;$
$r_{13} : \mu \to \pi, 2;$
$r_{14} : \pi\varepsilon_j \to (\varepsilon_j, here)(\pi\varepsilon_j e, out), 1, 1 \le j \le n;$

When the number of iterations left does not equal to zero, rules of type r_9-r_{13} are executed. Rules of type r_{10}-r_{11} are used to designate the new medoid and eliminate other data points in the cluster. Rule of type r_{12} is used to reduce the number of iterations by one.

When the number of iterations left equals to zero, rules of r_9-r_{10} and r_{13} are executed. Rule of type r_{10} is used to designate the new medoid. Each membrane B_i contains data point marks when system stops because rule of type r_{11} is not executed and there is no iterations left.

No matter if the number of iterations left equals to zero or not, rule of type r_{14} is executed to copy the new medoid out and send object e which means one iteration.

b)Reassignment

The rest rules in R_A are:
$r_{20} : \varepsilon_i \to (\varepsilon_i, in_C)|_\pi, 1, 1 \le i \le n;$ $\quad r_{21} : \pi^k \to \pi, 1;$
$r_{22} : \pi \to (\alpha, in_C), 2;$ $\quad r_{23} : e^k \to e, 1;$
$r_{24} : e^z \to \omega, 2;$ $\quad r_{25} : \omega \to (\omega, out), 1;$

When the number of iterations left does not equal to zero, rules of type r_{20}-r_{23} are executed. Rules of type r_{20} and r_{22} are used to move new medoid marks and the flow controller to membrane C. Then, reassignment gets start with flow controllers playing their roles and the specific flow of reassignment is identical with that of initial assignment.

When the number of iterations left equals to zero, rule of type r_{24} reaches the conditions of usage, and rules of type r_{20}-r_{25} are executed. Rule of type r_{24} is used to create ω which has the meaning of end of the system. Rule of type r_{25} is used to send ω out membrane A to end the system, and the execution of rules of type r_{20}-r_{22} does not impact the clustering results.

D. Complexity Analysis

In this subsection, the time cost in the worst case of Π_{kmbc} is analyzed according to the algorithm flow and the operation flow of rules. It is assumed that executing a rule costs a slice.

In initialization, preparation will be done in four slices. Computing and Depositing D_{nn} starts at the third slice of the previous part and will be done in $7 + Max\_data$ slices. Initialization of k medoids will be done at the last slice of the previous part. It needs $11 + Max\_distance\_sum$ slices to achieve initial assignment. In iterative assignment, updating medoids starts at the last slice of the previous part and will be done in $5 + Max\_distance\_sum$ slices. Reassignment will be done in $11 + Max\_distance\_sum$ slices. The end of Π_{kmbc} will cost one slice. In summary, while the number of iterations is z, the cost of Π_{kmbc} is $20 + 16z + Max\_data + (2z + 1) * Max\_distance\_sum$ slices at most. Obviously, Π_{kmbc} reduces the time complexity of k-medoids algorithm. In above, Max_data is the maximum of all the dimensions of all the data points, and $Max\_distance\_sum$ is the maximum of $d_i^{'}$ mentioned in subsection A in section III.

IV. Calculate Instance

In this section, an instance is given to show how to achieve k-medoids-based clustering in P system Π_{kmbc}. Let the number of iterations equals to 4, a data set containing 15 data points is clustered to 3 clusters in this instance. The data set to be processed is as follow: $T_{ins} = \{(1,2),(2,8),(2,7),(2,4),(2,2),(3,7),(3,2),(4,3),(5,6),(6,8),(6,7),(6,5),(7,9),(7,8),(7,7)\}$. And the data points in T_{ins} in turn correspond to data point marks form ε_1 to ε_{15}.

The initialization gets starts as rules mentioned in subsection 1) in subsection C in section III are applied. The configuration of Π_{kmbc} after initialization is shown in Fig.4 and initial three medoid marks are ε_1, ε_2 and ε_3. Then, rules mentioned in subsection 2) in subsection C in section III are applied to achieve initial assignment. At this time, the configuration of Π_{kmbc} becomes as shown in Fig.5, and the temporary clustering results are $\{\varepsilon_1, \varepsilon_4, \varepsilon_5, \varepsilon_7, \varepsilon_8\}$, $\{\varepsilon_2, \varepsilon_{10}, \varepsilon_{13}, \varepsilon_{14}\}$, $\{\varepsilon_3, \varepsilon_6, \varepsilon_9, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{15}\}$. In the following, there are 4 iterations each one of which includes almost all of the rules mentioned in subsection 3) in subsection C in section III. After the first, second, third and fourth iteration, the configurations of Π_{kmbc} are as shown in Fig.6 to Fig.9. For convenience, define two strings in Fig.4-Fig.9:

$$S_T = \{x_1^1 y_1^2 x_2^2 y_2^8 x_3^2 y_3^7 x_4^2 y_4^4 x_5^2 y_5^2 x_6^3 y_6^7 x_7^3 y_7^2 x_8^4 y_8^3 x_9^5 y_9^6 x_{10}^6 y_{10}^8 x_{11}^6 y_{11}^7 x_{12}^6 y_{12}^5 x_{13}^7 y_{13}^9 x_{14}^7 y_{14}^8 x_{15}^7 y_{15}^7\},$$

$$\begin{aligned} S_D = \{ & d_{1,2}^{37} d_{1,3}^{26} d_{1,4}^5 d_{1,5}^1 d_{1,6}^{29} d_{1,7}^4 d_{1,8}^{10} d_{1,9}^{32} d_{1,10}^{61} d_{1,11}^{50} d_{1,12}^{34} d_{1,13}^{85} \\ & d_{1,14}^{72} d_{1,15}^{61} d_{2,3}^1 d_{2,4}^{16} d_{2,5}^{36} d_{2,6}^2 d_{1,7}^{37} d_{2,8}^{29} d_{2,9}^{13} d_{2,10}^{16} d_{2,11}^{17} d_{2,12}^{25} \\ & d_{2,13}^{26} d_{2,14}^{25} d_{2,15}^{26} d_{3,4}^9 d_{3,5}^{25} d_{3,6}^1 d_{3,7}^{26} d_{3,8}^{20} d_{3,9}^{10} d_{3,10}^{17} d_{3,11}^{16} d_{3,12}^{20} \\ & d_{3,13}^{29} d_{3,14}^{26} d_{3,15}^{25} d_{4,5}^4 d_{4,6}^{10} d_{4,7}^5 d_{4,8}^5 d_{4,9}^{13} d_{4,10}^{32} d_{4,11}^{25} d_{4,12}^{17} d_{4,13}^{50} \\ & d_{4,14}^{41} d_{4,15}^{34} d_{5,6}^{26} d_{5,7}^1 d_{5,8}^5 d_{5,9}^{25} d_{5,10}^{52} d_{5,11}^{41} d_{5,12}^{25} d_{5,13}^{74} d_{5,14}^{61} d_{5,15}^{50} \\ & d_{6,7}^{25} d_{6,8}^{17} d_{6,9}^5 d_{6,10}^{10} d_{6,11}^9 d_{6,12}^{13} d_{6,13}^{20} d_{6,14}^{17} d_{6,15}^{16} d_{7,8}^2 d_{7,9}^{20} d_{7,10}^{45} \\ & d_{7,11}^{34} d_{7,12}^{18} d_{7,13}^{65} d_{7,14}^{52} d_{7,15}^{41} d_{8,9}^{10} d_{8,10}^{29} d_{8,11}^{20} d_{8,12}^8 d_{8,13}^{45} d_{8,14}^{34} \\ & d_{8,15}^{25} d_{9,10}^5 d_{9,11}^2 d_{9,12}^2 d_{9,13}^{13} d_{9,14}^8 d_{9,15}^5 d_{10,11}^1 d_{10,12}^9 d_{10,13}^2 \\ & d_{10,14}^1 d_{10,15}^2 d_{11,12}^4 d_{11,13}^5 d_{11,14}^2 d_{11,15}^1 d_{12,13}^{17} d_{12,14}^{10} d_{12,15}^5 \\ & d_{13,14}^1 d_{13,15}^4 d_{14,15}^1 \}. \end{aligned}$$

As suggested by Fig.6, the temporary clustering results are $\{\varepsilon_1, \varepsilon_4, \varepsilon_5, \varepsilon_7, \varepsilon_8\}$, $\{\varepsilon_{10}, \varepsilon_{11}, \varepsilon_{13}, \varepsilon_{14}, \varepsilon_{15}\}$ and $\{\varepsilon_2, \varepsilon_3, \varepsilon_6, \varepsilon_9, \varepsilon_{12}\}$ after the first iteration. As suggested by Fig.7, the temporary clustering results are $\{\varepsilon_1, \varepsilon_4, \varepsilon_5, \varepsilon_7, \varepsilon_8\}$, $\{\varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}, \varepsilon_{14}, \varepsilon_{15}\}$ and $\{\varepsilon_2, \varepsilon_3, \varepsilon_6, \varepsilon_9\}$ after the second iteration. As suggested by Fig.8, the temporary clustering results are $\{\varepsilon_1, \varepsilon_4, \varepsilon_5, \varepsilon_7, \varepsilon_8\}$, $\{\varepsilon_9, \varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}, \varepsilon_{14}, \varepsilon_{15}\}$ and $\{\varepsilon_2, \varepsilon_3, \varepsilon_6\}$ after the third iteration. As suggested by Fig.9, the temporary clustering results are $\{\varepsilon_1, \varepsilon_4, \varepsilon_5, \varepsilon_7, \varepsilon_8\}$, $\{\varepsilon_9, \varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}, \varepsilon_{14}, \varepsilon_{15}\}$ and $\{\varepsilon_2, \varepsilon_3, \varepsilon_6\}$ after the fourth iteration. Finally, data set T_{ins} is clustered into $\{(1,2),(2,4),(2,2),(3,2),(4,3)\}$, $\{(5,6),(6,8),(6,7),(6,5),(7,9),(7,8),(7,7)\}$ and $\{(2,8),(2,7),(3,7)\}$ as suggested by the scatter plot in Fig.10.

V. Conclusion

This paper proposes a P system Π_{kmbc} based on k-medoids to achieve clustering in a shorter time. The algorithm for Π_{kmbc} is modified to fit this cell-like P system with promoters and inhibitors. In this P system, a hierarchically arranged

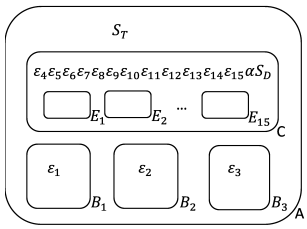

Fig. 4. The configuration of Π_{kmbc} after initialization

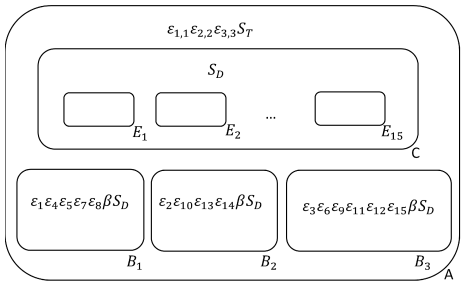

Fig. 5. The configuration of Π_{kmbc} after initial assignment

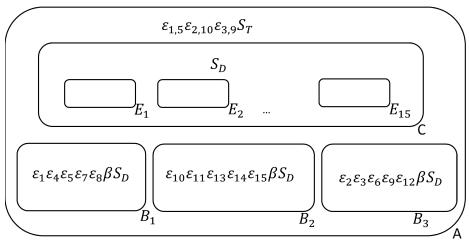

Fig. 6. The configuration of Π_{kmbc} after the first iteration

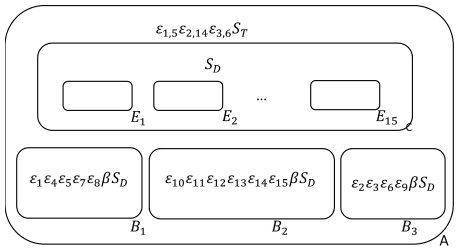

Fig. 7. The configuration of Π_{kmbc} after the second iteration

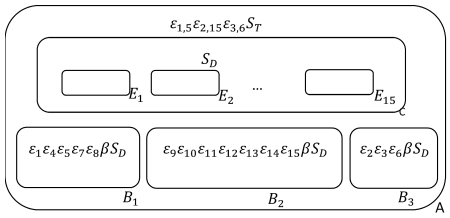

Fig. 8. The configuration of Π_{kmbc} after the third iteration

structure and numerous rules are designed to bring the parallel and distributed computing mechanism into play. An instance is given to illustrate the practicability and effectiveness of the

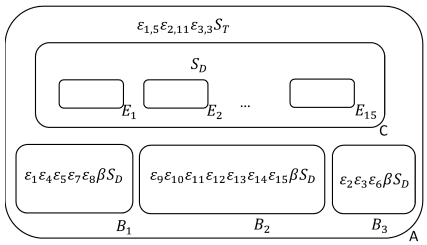

Fig. 9. The configuration of Π_{kmbc} after the fourth iteration

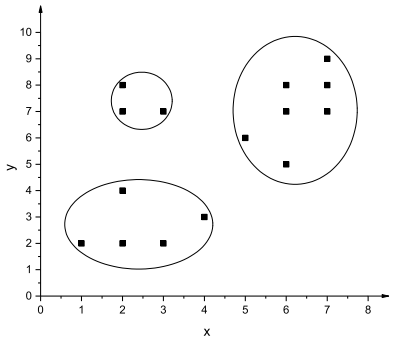

Fig. 10. Clustering results in Π_{kmbc}

P system designed. However, space complexity of Π_{kmbc} is a bit high for purpose of lower time complexity. Our future work includes simplifying the membrane structure and rules to decrease the space complexity and optimizing the algorithm for clustering to enhance learning efficiency.

REFERENCES

[1] Gheorghe, and Păun. *Computing with Membranes.* Journal of Computer and System Sciences. 61.1(2000):108-143.

[2] Chengdu. *A Survey of Membrane Computing as a New Branch of Natural Computing.* Chinese Journal of Computers. 33.2(2010):208-214.

[3] Singh, Garima, and K. Deep. *A new membrane algorithm using the rules of Particle Swarm Optimization incorporated within the framework of cell-like P-systems to solve Sudoku.* Elsevier Science Publishers B. V. 2016.

[4] Chun, and Xiaolong. *Uniform Solution to QSAT by P Systems with Proteins.* Chinese Journal of Electronics. 21.4(2012):667-672.

[5] Pan Linqiang, and Pérez-Jiménez,Mario J. *"Computational complexity of tissue-like P systems.* Journal of Complexity. 26.3(2010):296-315.

[6] Gao, Tong, X. Liu, and L. Wang. *An Improved PSO-Based Clustering Algorithm Inspired by Tissue-Like P System.* International Conference on Data Mining and Big Data Springer, Cham, 2018:325-335.

[7] Zhang, Xing Yi, et al. *Tissue P systems with cell separation: attacking the partition problem.* Science China(Information Sciences) 54.2(2011):293-304.

[8] Wu, Tingfang, Z. Zhang, and L. Pan. *On Languages Generated by Cell-Like Spiking Neural P Systems.* IEEE Transactions on Nanobioscience 15.5(2016):455-467.

[9] Song, Tao, et al. *Small Universal Spiking Neural P Systems with Anti-Spikes.* Journal of Computational & Theoretical Nanoscience 10.4(2013):999-1006.

[10] Frisco, Pierluigi, M. Gheorghe, and M. J. Prez-Jimnez. *Applications of Membrane Computing in Systems and Synthetic Biology.* Emergence Complexity & Computation 7.09(2014):S624.

[11] D. Sburlan. *Membrane Computing Insights. Permitting and Forbidding Contexts.* Ovi Up, Constanta, Romania, 2012.

[12] Gabriel Ciobanu.*Proceedings 6th Workshop on Membrane Computing and Biologically Inspired Process Calculi.* Computer Science 1-2(2012):3-4.

[13] Ciobanu, Gabriel, G. Păun, and M. J. Prez-Jimnez. *Applications of Membrane Computing.* Theoretical Computer Science 287.1(2005):73–100.

[14] Gazdag, Zsolt, G. Kolonits, and M. A. Gutirrez-Naranjo. *Simulating Turing Machines with Polarizationless P Systems with Active Membranes.* Membrane Computing -, International Conference, Cmc 2014, Prague, Czech Republic, August 20-22, 2014, Revised Selected Papers 2014:229-240.

[15] Alhazov, Artiom, et al. *Space complexity equivalence of P systems with active membranes and Turing machines.* Theoretical Computer Science 529.6(2014):69-81.

[16] The P Systems Website. http://ppage.pssystems.eu, 2013-01.

[17] Xu, Dongkuan, and Y. Tian. *A Comprehensive Survey of Clustering Algorithms.* Annals of Data Science 2.2(2015):165-193.

[18] Jain, Anil K, and R. C. Dubes. *Algorithms for clustering data.* Technometrics 32.2(1988):227-229.

[19] Boley, Daniel, et al. *Partitioning-based clustering for Web document categorization.* Decision Support Systems 27.3(1999):329-341.

[20] Velmurugan, T. , and T. Santhanam. *A Survey of Partition based Clustering Algorithms in Data Mining: An Experimental Approach.* Information Technology Journal 10.3(2011).

[21] Ayed, Abdelkarim Ben, M. B. Halima, and A. M. Alimi. *Survey on clustering methods: Towards fuzzy clustering for big data.* Soft Computing and Pattern Recognitio IEEE, 2015:331-336.

[22] Preheim,P. Sarah, et al. *Distribution-Based Clustering: Using Ecology To Refine the Operational Taxonomic Unit.* Applied & Environmental Microbiology 79.21(2013):6593-6603.

[23] Jiang, Bin, et al. *Clustering Uncertain Data Based on Probability Distribution Similarity.* IEEE Transactions on Knowledge & Data Engineering 25.4(2013):751-763.

[24] Loh, Woong Kee, and Y. H. Park. *A Survey on Density-Based Clustering Algorithms.* Ubiquitous Information Technologies and Applications. 2014:775-780.

[25] Bhuyan, Rupanka, and S. Borah. *A Survey of Some Density Based Clustering Techniques.* National Conference on Advancements in Information, Computer and Communication 2013.

[26] Ilango, M. R. , and D. V. Mohan. *A Survey of Grid Based Clustering Algorithms.* International Journal of Engineering Science & Technology 2.8(2010).

[27] Nemade, P. , et al. *Survey on grid clustering approach with intuitionistic fuzzy histogram.* International Journal of Pharmacy & Technology 8.4(2016):25475-25482.

[28] Wang, Ying Hong, et al. *A Grid-Based Clustering Routing Protocol for Wireless Sensor Networks.* Energy Procedia 11(2013):602-609.

[29] Nascimento, Mari C. V. . *Spectral methods for graph clustering A survey.* European Journal of Operational Research 211.2(2011):221-231.

[30] Gallier, Jean. *Spectral Theory of Unsigned and Signed Graphs. Applications to Graph Clustering: a Survey.* 2016.

[31] Liu, Xiyu, Y. Zhao, and W. Sun. *K-Medoids-Based Consensus Clustering Based on Cell-Like P Systems with Promoters and Inhibitors.* Bio-Inspired Computing - Theories and Applications, Springer Singapore, 2016:95-108.

[32] Peng, Hong, J. Wang, and P. Shi. *An automatic clustering algorithm inspired by membrane computing.* Pattern Recognition Letters 68. P1 (2015):34-40.

[33] Peng, Hong, et al. *Membrane Clustering: A Novel Clustering Algorithm under Membrane Computing.* (2014).

[34] Kaufmann, Leonard, and P. J. Rousseeuw. *Clustering by Means of Medoids.* Statistical Data Analysis based on the L1-norm & Related Methods North-Holland, 1987:405-416.

[35] Nikam, B. Valmik, V. J. Kadam, and B. B. Meshram. *Image Compression Using Partitioning Around Medoids Clustering Algorithm.* International Journal of Computer Science Issues 8.6(2011):399-401.

[36] Hastie, Trevor, et al. *The Elements of Statistical Learning.* Springer New York, 2009.

[37] Park, Hae Sang, and C. H. Jun. *A simple and fast algorithm for K-medoids clustering.* Expert Systems with Applications 36.2(2009):3336-3341.

[38] Ping Guo, et al. *Evaluating Logical Expressions by Membrane Systems.* Chinese Journal of Electronics 23.2(2014):278-283.