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 P¨un 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[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 inhibitors and promoters by means of introducing k-medoids algorithm and cell-like P system with inhibitors and promoters to the consensus clustering, and it is proved to be highly accurate and highly efficient. K-medoids[34] is a mellioration 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 inhibitors, 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 \( \Pi_{kmbc} \) 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 \( \Pi_{kmbc} \) 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 I. 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 according to the rules in the membrane whose region objects appear in.

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

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

where,

1) O is the alphabet of objects used in \(\Pi\);

2) \(\mu\) is the membrane structure of \(\Pi\), and degree is \(m\);

3) \(M_i = (\omega_i, R_i, \rho_i)\) defines the membrane with label \(i\) in \(\Pi\) (\(i=1,2,\cdots,m\)), where \(\omega_i\) is the multiset including the initial objects in membrane \(i\), object \(\lambda\) (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, \(\rho_i\) 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 \(\Pi\) is: \((u \rightarrow v)\) or \((u \rightarrow v|\alpha, \rho)\) or \((u \rightarrow (w)_{i})\), where, \(u \in O^+, v \in (O \times Tar)^*, \alpha\) is promoter. \(O^+\) is the finite and non-empty multiset over \(O\), \(O^* = O^+ - \{\lambda\}\), \(Tar = \{here, out, in, |1 \leq i \leq m\}\). 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 \(\delta\) appeared in \(v\) means dissolving the membrane where \(\delta\) 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 \(\rho\) when \(\alpha\) 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 \(\Pi_{kmbc}\)

This paper aims to obtain a P system \(\Pi_{kmbc}\) for clustering based on k-medoids method. The algorithm for \(\Pi_{kmbc}\) is discussed before the definition of \(\Pi_{kmbc}\) is designed.
The algorithm for \( \Pi_{kmbc} \) is modified from PAM and Voronoi iteration method which are mentioned in subsection A in section II.

Suppose \( T = \{p_1, p_2, \ldots, p_n\} \) 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 \( T_{kmbc} = \{p_i = (x_i^a, y_i^b) | 1 \leq i \leq n, a, b \in N^+ \} \) 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 \( C = \{C_1, C_2, \ldots, C_k\} \). 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 \( \Pi_{kmbc} \) 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:

\[
D_{nn} = \begin{pmatrix}
0 & d_{1,2} & \cdots & d_{1,n} \\
0 & 0 & \cdots & d_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & d_{n-1,n-1} \\
0 & 0 & \cdots & 0
\end{pmatrix}
\] (2)

where \( d_{i,j} \) is the distance between \( p_i \) and \( p_j \).

Suppose \( D_1 \) the point-medoids distances set which contains \( k \) distances associated with \( p_i \) and \( k \) medoids, and it meets:

\[
D_1 = \{d_{j,i}, i \leq j \in S_m \} \cup \{d_{i,j}, i \leq j \in S_m \} \subset D \] (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 \( D_1' \) 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:

\[
D_1' = \{d_{j,i}, i \leq j \in S_{cm} \} \cup \{d_{i,j}, i \leq j \in S_{cm} \} \] (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 \( D_1' \) the sum of \( D_1' \).

It uses unique data point mark \( \varepsilon_i \) to replace \( p_i \) for convenient. Therefore, unique data point marks are assigned to clusters instead of data points benefitted 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 \( \Pi_{kmbc} \) is mainly composed of initialization, initial assignment and iterative assignments. The algorithm flow for \( \Pi_{kmbc} \) is shown in Fig. 2. The detailed flow of the algorithm is described as follow.

The algorithm for \( \Pi_{kmbc} \) computes \( D_{nn} \) and select first \( k \) of the \( n \) data points as the medoids at first. It compute \( D_1 \) for each \( p_i \) in \( T_{kmbc} \) and associates each data point mark to the closest medoid according to the minimal item in \( D_1 \) next. For un-medoid data point \( p_i \), this algorithm elects the minimal item \( d_{i,j} \) or \( d_{j,i} \) in \( D_1 \), then assigns \( \varepsilon_i \) to the cluster whose medoid is \( p_j \). Then iterative steps go. The first step of iterative steps is to compute \( D_1 \) 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 the \( S_1 \) 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.

B. The definition of \( \Pi_{kmbc} \)

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

\[
\Pi_{kmbc} = (O, \mu, M_A, M_B, M_C, M_{D_{i,j}}, M_{E_i}, M_{F_j}, t_{out})
\] (5)

where,

1) \( O = \{x_i, y_i, x_{i,j}, d_{i,j}, d_i, d_{i,j}, h_i, g_i, t_i, t_{i,j}, \varepsilon_i, \varepsilon_{i,j}, \alpha_i, \beta_i, \gamma_i, \xi_i | 1 \leq i, j \leq n \} \cup \{s, d_0, a_1, b_1, c_1, c_2, d, d_1, c_2, e, z, \alpha, \omega, \alpha, \beta, \lambda, \eta, \theta, \lambda, \mu, \xi, \pi, \rho, \sigma, \omega\} \) 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 \( d, \lambda, s \) and \( \omega \)). The most important objects are \( d_{i,j}, \varepsilon_i, \varepsilon_{i,j} \) and \( h_{i,j} \). Objects \( d_{i,j} \) 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 \( p_i \)
to the medoid whose subscript is \( j \). Object \( \varepsilon_i \) which is unique in \( \Pi_{kmbe} \) denotes data point \( p_i \) or medoid \( \rho_i \) (and then it is called data point mark or medoid mark). Object \( \varepsilon_{i,j} \) is designed to record the information that cluster \( i \) designates 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) \( \beta = [B_1, B_2, B_3, B_4, E_1, E_2, E_3, E_4, E_n] \) is the initial membrane structure of \( \Pi_{kmbe} \) as shown as what Fig.3 illustrates, and it will change with the evolution of \( \Pi_{kmbe} \). 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 \( \Pi_{kmbe} \), 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 \( \Pi_{kmbe} \) in which \( mem \in \{A, C\} \cup \{D_{i,j}, E_i, F_j\}, 1 \leq i \leq n, 1 \leq j \leq k \} \), \( w_A = T_{kmbe} \cup \{s\} \), \( w_{B_i} = \{e^2\} \), \( w_C = w_{Di,j} = w_{E_i} = w_{F_j} = \lambda \), \( R_{mem} \) and \( \rho_{mem} \) are described in hand as \( R_{mem} \) for convenience, and more details about \( R_{mem} \) are in subsection C in section III.

4) \( i_{out} = \{B_1, B_2, \ldots, B_k\} \).

In addition, child membranes 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 \( \Pi_{kmbe} \). As a result, rule sets of dynamically generated membranes only contain uninherited rules for concision. The rules in \( \Pi_{kmbe} \) are elaborated in next subsection and calculation processes too.

C. The rules in \( \Pi_{kmbe} \)

For the sake of being understand, the rule sets \( R_{mem} \) in \( \Pi_{kmbe} \) are explained in order of operation flow. Of these, \( \rho_{mem} \) 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 \( \Pi_{kmbe} \), the rules in \( R_A \) associated with the preparation are:

- \( r_1 : s \rightarrow a_1 a_2 \ldots a_{n-1} \theta, 1; \)
- \( r_2 : x_i \rightarrow (x_i, here)(h_i), 1 \leq i \leq n-1; \)
- \( r_3 : a_i \rightarrow h_i, 1 \leq i \leq n; \)
- \( r_4 : y_i \rightarrow (y_i, here)(g_i, in_i, i+1), 1 \leq i \leq n-1; \)
- \( r_5 : h_i \rightarrow (h_i, g_i, in_i)\), 1 \leq i \leq n-1; \)
- \( r_6 : (x_i, here)(h_i, in_i)\), 1 \leq i \leq n-1; \)
- \( r_7 : h_i \rightarrow (h_i, g_i, in_i)\), 1 \leq i \leq n-1; \)
- \( r_8 : g_i \rightarrow (g_i, in_i)\), 1 \leq i \leq n-1; \)
- \( r_9 : (\varepsilon_i, here)(\alpha, in_i)\), 1 \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 points 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 \( r_8 \) 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 \( D_{ij} \). Other rules assist in process control.

b) Compute and deposit \( D_{nn} \)

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

- \( r_1 : h_i \rightarrow a_0, 1; \)
- \( r_2 : a_0 \rightarrow a_1, 1; \)
- \( r_3 : g_i \rightarrow b_1, 1; \)
- \( r_4 : a_1 \rightarrow h_i, 1; \)
- \( r_5 : a_1 \rightarrow c, 2; \)
- \( r_6 : h_j \rightarrow c, 2; \)
- \( r_7 : b_1 \rightarrow g_j, 1; \)
- \( r_8 : b_1 \rightarrow d, 2; \)
- \( r_9 : g_j \rightarrow d, 2; \)
- \( r_{10} : a \rightarrow \alpha^{k}, 2; \)
- \( r_{11} : \alpha \rightarrow \alpha_{\alpha}, 2; \)
- \( r_{12} : \alpha \rightarrow e_2, 1; \)
- \( r_{13} : d \rightarrow d_1, 2; \)
- \( r_{14} : c_1 \rightarrow d_1, 1; \)
- \( r_{15} : e_2 \rightarrow \alpha_{\alpha}, 1; \)
- \( r_{16} : d_1 \rightarrow d, 1; \)
- \( r_{17} : d_2 \rightarrow \alpha_{\alpha}, 1; \)
- \( r_{18} : \alpha \rightarrow \beta, 2; \)
- \( r_{19} : c_1 \rightarrow \lambda, 1; \)
- \( r_{20} : d_1 \rightarrow \lambda, 1; \)
- \( r_{21} : \beta \rightarrow (\delta, here)(\eta, out), 1; \)

With the specific flow controller, rules in membrane \( D_{ij} \) start being executed. Rules of type \( r_{2}-r_{9} \) 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 \( D_{ij} \) contains objects \( d_{ij} \) 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 \( d_{ij} \) to membrane \( C \):

\[
r_{10} : d_{ij} \rightarrow (d_{ij}, inC)_{|\eta}, 1 \leq i \leq n, 1 \leq j \leq n, i \leq j;
\]

c)Initialize k medoids

The rules in \( R_A \) associated with the initialization of k medoids are:

\[
r_{11} : \varepsilon_i \rightarrow ((\varepsilon_i, inB_j)_{|\eta}, 1 \leq i \leq k; \]

\[
r_{12} : \theta \rightarrow (\alpha, inC)_{|\eta}, 1;
\]

\[
r_{13} : \eta \rightarrow \lambda, 1;
\]

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 \( E_i \):

\[
r_{1} : d_{ij} \rightarrow a_{ij} \rightarrow \epsilon_{ij}, 1 \leq i \leq n, 1 \leq j \leq n; \]

\[
r_{2} : d_{ij} \rightarrow (d_{ij}, here) \rightarrow (d_{ij}, inE_i)_{|\epsilon_i}, 2 \leq i \leq n, 1 \leq j \leq n; \]

\[
r_{3} : d_{ij} \rightarrow (d_{ij}, here) \rightarrow (d_{ij}, inE_i)_{|\epsilon_i}, 2 \leq i \leq n, 1 \leq j \leq n; \]

\[
r_{4} : \varepsilon_i \rightarrow (\varepsilon_i, inall)_i, 1 \leq i \leq n; \]

\[
r_{5} : \alpha \rightarrow (\beta, here)_i(\alpha, inall)_i, 1; \]

\[
r_{6} : a_{ij} \rightarrow (d_{ij}, \beta)_i, 1 \leq i \leq n, 1 \leq j \leq n; \]

Thus, each membrane of type \( E_i \) contains k medoid marks and k objects of type \( d_{ij} \) or \( d_{ji} \), 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 \( E_i \):

\[
r_{1} : d_{ij} \rightarrow d_{j,i}, 1 \leq j \leq n; \]

\[
r_{2} : d_{j,i} \rightarrow d_{j,i}, 1 \leq j \leq n; \]

\[
r_{3} : \alpha \rightarrow \beta, 1; \]

\[
r_{4} : \varepsilon_i \rightarrow \eta_{ij}, 1; \]

\[
r_{5} : \gamma \rightarrow \gamma, 1; \]

\[
r_{6} : \delta \rightarrow \theta, 1; \]

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 \( d_{ij} \) for all \( j \) existed in the membrane until there is no \( d_{ij} \) left for a certain \( j \), then turn the \( \varepsilon_j \) corresponding to the certain \( j \) to \( t_{ij} \). As consequence, transmission mark \( t_{ij} \) means that un-medoid data point \( p_i \) belongs to the cluster with the medoid \( p_j \) and rule of type \( r_{9} \) is used to move \( t_{ij} \) 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 \( r_8, r_6, r_7 \) 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:

\[
r_{7} : \beta t_{ij} \rightarrow \gamma t_{ij}, 1 \leq i \leq n, 1 \leq j \leq n;
\]

\[
r_{8} : d_{ij} \rightarrow (d_{ij}, here)(d_{ij}, out)_{|\gamma}, 1 \leq i \leq n, 1 \leq j \leq n;
\]

\[
r_{9} : t_{ij} \rightarrow (t_{ij}, out)_{|\gamma}, 1;
\]

\[
r_{10} : \gamma \rightarrow (\mu, out), 1;
\]

Rule of type \( r_{14} \) in \( R_A \) is used to send a flow controller of type \( \alpha \) to each membrane \( B_i \):

\[
r_{14} : \mu \rightarrow (\alpha, inB_i) \rightarrow \ldots (\alpha, inB_i), 1;
\]

With the flow controllers, \( k \) marks of type \( \varepsilon_{ij} \) which reflect that the \( i \text{-th} \) cluster has a medoid with subscript \( j \) right now are copied into membrane \( A \) due to the execution of rule of type \( r_{15} \) in \( R_B \):

\[
r_{15} : \varepsilon_{ij} \rightarrow (\varepsilon_{ij}, out)(\varepsilon_{ij}, here)_{|\alpha}, 1 \leq j \leq n;
\]

Finally, there are objects of type \( t_{ij} \) and \( \varepsilon_{ij} \) in membrane \( A \). That is to say, all the objects for initial assignment are completely prepared. Rules of type \( r_{15} \) in \( R_A \) are associated with initial assignment and preparation for the following step:

\[
r_{16} : t_{ij} \rightarrow (\varepsilon_{ij}, inB_m)_{|\varepsilon_{ij}}, 1 \leq i, j \leq n;
\]

\[
r_{17} : \xi \rightarrow (\beta, inB_i) \rightarrow (\beta, inB_i), 2;
\]

\[
r_{18} : \varepsilon_{ij} \rightarrow \lambda, 1 \leq i \leq n, 1 \leq j \leq n;
\]

Rule of type \( r_{16} \) is used to send data point marks to the certain membranes of type \( B_i \) under the premise of \( t_{ij} \) and \( \varepsilon_{ij} \). As a result, initial assignment is achieved. Meanwhile, \( D_{nn} \) and flow controller \( \beta \) 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:

\[
r_{2} : \varepsilon_j \rightarrow (\varepsilon_j, here)(\varepsilon_j, F_j)_{|\beta}, 1 \leq j \leq n;
\]

\[
r_{3} : \beta \rightarrow \lambda, 1;
\]

\[
r_{4} : d_{p,q} \rightarrow (x_{p,q}, inall)_{|\gamma}, 1 \leq p \leq n, 1 \leq q \leq n;
\]

\[
r_{5} : \gamma \rightarrow (\sigma, inall)_{|\eta, here}, 1;
\]

\[
r_{6} : \eta \rightarrow \theta, 1;
\]

With the flow controller \( \beta \), 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 \( \sigma \) to all the membranes \( F_j \). Therefore, each membrane \( F_j \) contains all the objects associated with computing \( D_j \) and \( d_{ij} \).

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:

\[
r_{1} : x_{ij} \rightarrow d_{ij} \rightarrow (d_{ij}, 1 \leq i \leq n;
\]

\[
r_{2} : x_{j,i} \rightarrow d_{ij} \rightarrow (d_{ij}, 1 \leq i \leq n;
\]

\[
r_{3} : x_{p,q} \rightarrow \lambda, 2 \leq p \leq n, 1 \leq q \leq n;
\]

\[
r_{4} : t_{ij} \rightarrow \lambda_{|\sigma}, 1;
\]

\[
r_5 : \sigma \rightarrow \delta, 1;
\]

In each membrane \( F_j \), rules of type \( r_1-r_2 \) turn objects of type \( x_{ij} \) to \( d_{ij} \) where the first or second subscript of \( x_{ij} \) and the subscript of \( d_{ij} \) equals to that of \( F_j \). Thus, the quantifies of \( d_{ij} \) equals to \( d_{ij} \) which is the sum of \( D_j \). Rule of type \( r_5 \) is used to dissolve the membrane and release objects of type \( d_{ij} \).

Rules of type \( r_7-r_{10} \) in \( R_B \) are executed repeatedly to figure the object \( \xi_j \) which points to the minimal \( d_{ij} \):

\[
r_7 : \varepsilon_j d_{ij} \rightarrow \varepsilon_j, 1 \leq i \leq j \leq n; \]

\[
r_8 : \varepsilon_j \theta \rightarrow \mu \varepsilon_j, 2 \leq j \leq n;
\]
That is to say, $p_i$ whose subscript equals to that of $\xi_j$ will be the new medoid. The rest rules in $R_{B_{c}}$ are:

- $r_{9}: d_{j} \rightarrow \lambda|\mu|, 1 \leq j \leq n$;
- $r_{10}: \xi_j \rightarrow e_j|\mu|, 1 \leq j \leq n$;
- $r_{11}: e_j \rightarrow \lambda|\mu|, 1 \leq j \leq n$;
- $r_{12}: e \mu \rightarrow \pi, 1$;
- $r_{13}: \mu \rightarrow \pi, 2$;
- $r_{14}: \pi e_j \rightarrow (e_j, here)(\pi e, out), 1 \leq j \leq n$;

When the number of iterations left does not equal to zero, rules of type $r_{9}$-$r_{11}$ 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 $B_{c}$ contains data point marks when system stops because rule of type $r_{13}$ 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:

- $r_{20}: e \xi \rightarrow (e_i, in|\xi|, e_i, out), 1 \leq i \leq n$;
- $r_{21}: \pi k \rightarrow \pi, 1$;
- $r_{22}: \pi \rightarrow (\alpha, in|\alpha|, 2)$;
- $r_{23}: e^k \rightarrow e, 1$;
- $r_{24}: e^2 \rightarrow \omega, 2$;
- $r_{25}: \omega \rightarrow (\omega, out), 1$;

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 $\omega$ which has the meaning of end of the system. Rule of type $r_{25}$ is used to send $\omega$ 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 $\Pi_{kmedbc}$ 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 $\Pi_{kmedbc}$ will cost one slice. In summary, while the number of iterations is $z$, the cost of $\Pi_{kmedbc}$ is $20 + 16z + Max_{data} + (2z + 1) * Max_{distance\_sum}$ slices at most. Obviously, $\Pi_{kmedbc}$ 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 INSTANC

In this section, an instance is given to show how to achieve k-medoids-based clustering in P system $\Pi_{kmedbc}$. 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 $e_1$ to $e_{15}$.

The initialization gets starts as rules mentioned in subsection 1) in subsection C in section III are applied. The configuration of $\Pi_{kmedbc}$ after initialization is shown in Fig.4 and initial three medoid marks are $e_1$, $e_2$ and $e_3$. Then, rules mentioned in subsection 2) in subsection C in section III are applied to achieve initial assignment. At this time, the configuration of $\Pi_{kmedbc}$ becomes as shown in Fig.5, and the temporary clustering results are as Fig.6 and Fig.7, and the temporary clustering results are as Fig.6 and Fig.7.

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 $\Pi_{kmedbc}$ are as shown in Fig.6 to Fig.9. For convenience, define two strings in Fig.4-Fig.9:

$S_T = \{(1, 2), (2, 8), (2, 7), (2, 4), (2, 2), (3, 7), (3, 2), (4, 3), (5, 6), (6, 8), (6, 7), (6, 5), (7, 9), (7, 8), (7, 7)\}$

As suggested by Fig.6, the temporary clustering results are $\{e_1, e_4, e_5, e_7, e_8\}$, $\{e_{10}, e_{11}, e_{13}, e_{14}, e_{15}\}$ and $\{e_2, e_3, e_6, e_9, e_{12}\}$ after the first iteration. As suggested by Fig.7, the temporary clustering results are $\{e_1, e_4, e_5, e_7, e_8\}$, $\{e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}\}$ and $\{e_2, e_3, e_6, e_9\}$ after the second iteration. As suggested by Fig.8, the temporary clustering results are $\{e_1, e_4, e_5, e_7, e_8\}$, $\{e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}\}$ and $\{e_2, e_3, e_6\}$ after the third iteration. As suggested by Fig.9, the temporary clustering results are $\{e_1, e_4, e_5, e_7, e_8\}$, $\{e_6, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}\}$ and $\{e_2, e_3, e_6\}$ 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 $\Pi_{kmedbc}$ based on k-medoids to achieve clustering in a shorter time. The algorithm for $\Pi_{kmedbc}$ is modified to fit this cell-like P system with promoters and inhibitors. In this P system, a hierarchically arranged

www.ijacsathesai.org 46 | Page
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 P system designed. However, space complexity of $\Pi_{kmbe}$ 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


