An Improved Consensus Clustering Algorithm Based on Cell-Like P Systems With Multi-Catalysts

YUZHEN ZHAO¹, WEINING ZHANG², MINGHE SUN³, AND XIYU LIU¹

¹College of Business, Shandong Normal University, Jinan 250014, China
²College of Sciences, The University of Texas at San Antonio, San Antonio, TX 78249, USA
³College of Business, The University of Texas at San Antonio, San Antonio, TX 78249, USA

Corresponding authors: Weining Zhang (weining.zhang@utsa.edu) and Xiyu Liu (xyliu@sdnu.edu.cn)

ABSTRACT Consensus clustering algorithm, which integrates several clustering results obtained by common algorithms, can find a better result that is independent on parameter settings. However, this kind of algorithm is often designed based on simple, such as $K$-means, algorithms, which is limited by the time complexity. In this work, a P system, a novel branch of bio-inspired computing with inherent parallel and distributed computation, is combined with the consensus clustering algorithm. As a result, an improved consensus clustering algorithm is constructed using the hierarchical membrane structure and parallel evolution mechanism in a cell-like P system with multi-catalysts, where the catalysts are utilized to regulate the parallelism of objects evolution. The integration strategy of the algorithm is based on a revised PAM where only the $q$-nearest neighbors of the original medoids are considered as candidates for the new medoids. The experimental results indicate that the clustering quality of the proposed algorithm is more robust than well-known consensus clustering algorithms on data sets with noises and outliers. This work gives evidence that the effectiveness and efficiency of consensus clustering algorithms can be improved using P systems.

INDEX TERMS Consensus clustering, membrane computing, PAM, cell-like P systems.

I. Introduction

Information plays an important role in each field in modern society. Analyzing data and extracting useful information from huge amounts of data are always important topics in both research and practice. Clustering analysis, revealing relationships between data, is a class of important data analysis methods. Common clustering algorithms have three limitations: 1). The clustering results largely depend on the parameter settings and their initial values. 2). Most clustering algorithms cannot judge the actual number of clusters. 3). Different clustering algorithms may generate different clustering results.

The associate editor coordinating the review of this manuscript and approving it for publication was Weipeng Jing.

Consensus clustering gives a solution to these limitations [1]. It aggregates several basic partitionings (BPs) obtained by common algorithms and obtains the final result which is better than each of these BPs. There are two types of consensus strategies, the object co-occurrence and the median partition [2]. The object co-occurrence strategy computes the number of occurrences of a point in a certain cluster or the number of occurrences of two points belonging to the same cluster, and then determines the final cluster label of each point [3]. The relabelling and voting method [4] and the graph partitioning based method [5], [6] follow this strategy. The median partition strategy turns the consensus clustering into an optimization problem aiming to find the median partition, that is, the partition which can maximize the similarity of BPs. The kernel-based method [7] and the genetic-based method [8] follow this strategy. Consensus clustering is
very popular, and researchers have proposed many improvements since it was proposed. For high dimensional datasets, Li et al. [9] designed a random projection framework using a cumulative agreement scheme. For a large-scale dataset, Yu et al. [10] proposed a three-way algorithm based on Spark inspired by the theory of three-way decisions. For multiview datasets, Zhao et al. [11] projected incomplete multiview to a complete and unified representation in a common subspace, Liu and Yun [12] presented an interactive consensus clustering approach in which the BPs from each view interact with the final result, and Tao et al. [13] proposed a marginalized algorithm considering higher-level information such as BPs generated by a single-view. For large scale group decision-making processes, Wu and Xu [14] designed a model where the clusters could be changed and the decision-makers used fuzzy preference relations to show their preferences. For datasets with dubious data points in which many features are uninformative to clustering, Wang and Tsuchiya [15] performed sub-clustering on data points using some subset of features from data points to handle these dubious data. Considering the relationship among BPs, Rathore et al. [16] proposed a maximum relative density path accumulation algorithm. Considering the local diversity of clusters inside the same BP, Huang, Wang and Lai [17] proposed a locally weighted clustering approach using uncertainty estimation and a local weighting strategy. Considering the cluster-level selection in the selection of BPs, Nazari et al. [18] designed a framework based on cluster-level weighting. However, a significant drawback of consensus clustering is the slow running-time, and simple strategies are used in various algorithms to balance the running-time and the quality of the clustering results. How to improve algorithm quality within an acceptable time complexity is an important research topic in consensus clustering. This work addresses this problem using a bio-inspired computing method, i.e., a P system.

A P system [19] is a novel bio-inspired computational model that is inherently parallel and distributed [20]. There are three types of, i.e., cell-like [19], tissue-like [21] and neural-like [22], P systems which were inspired by a single cell, a set of multiple connected cells, and a network of nerve cells, respectively. Up to now, many variants of P systems have been proposed which abstract various biological mechanisms, such as cell-like P systems with channel states and symport/antiport rules [23], cell-like P systems with polarizations and minimal rules [24], P systems with rule production and removal [25], tissue-like P systems with evolutionary symport/antiport rules [26], spiking neural P systems with white hole neurons [27] and so on [28]–[31]. These types of P systems have been shown to be computationally complete [32]–[36]. The current research on using P systems to solve practical real-world problems takes two approaches. One approach, called the indirect membrane algorithm approach [37], [38], is to design algorithms based on certain features of a P system such as separating the whole computational areas into serval regions (membranes) and to execute these algorithms on a modern electronic computer. While this approach leads to novel algorithms for solving many complex real-world problems, the efficiency of these algorithms is still bounded by the physical limitations of electronic computer. Another approach, called the direct membrane algorithm approach [39]–[41], is to design P systems that are capable of performing algorithms for solving real-world problems, and the computation of these P systems can be executed on hardware, software or living cells as a natural evolution process. While the execution on living cells has a promising potential to significantly improve the computational efficiency, it also faces a lot of challenges such as defining the P system to solve specific real-world problems and making living cells perform the computation defined by a P system. While the construction of a bio-computer relies on the participation of biology researchers [42]–[44], the design of P systems appropriate for solving specific problems is up on computer science research and is considered in this study.

This work follows the direct membrane algorithm approach and chooses the cell-like P system with multi-catalysts (MCC-P), a specific cell-like P system, to develop and to improve a consensus clustering algorithm based on PAM (Prediction Around Medoids) [45] with the q-nearest neighbors. For easy reference, this improved consensus clustering algorithm is called the P-PCC.

A MCC-P system consists of a membrane structure, a multiset of objects and a set of evolution rules (rules for short) for each membrane. Once entering an input membrane, the objects will start evolving according to applicable rules and to improve a consensus clustering algorithm based on the computation required by the consensus clustering algorithm which is robust to noises and outliers.

This work makes the following contributions.

1) A revised PAM is proposed in which only the q-nearest neighbors of the original medoids are considered as candidates of a possible replacement for the original medoids, which improves the computational efficiency. This medoid replacement approach is used as a BP integration strategy of the consensus clustering algorithm which is inherently parallel and distributed [20]. There are three types of, i.e., cell-like [19], tissue-like [21] and neural-like [22], P systems which were inspired by a single cell, a set of multiple connected cells, and a network of nerve cells, respectively. Up to now, many variants of P systems have been proposed which abstract various biological mechanisms, such as cell-like P systems with channel states and symport/antiport rules [23], cell-like P systems with polarizations and minimal rules [24], P systems with rule production and removal [25], tissue-like P systems with evolutionary symport/antiport rules [26], spiking neural P systems with white hole neurons [27] and so on [28]–[31]. These types of P systems have been shown to be computationally complete [32]–[36]. The current research on using P systems to solve practical real-world problems takes two approaches. One approach, called the indirect membrane algorithm approach [37], [38], is to design algorithms based on certain features of a P system such as separating the whole computational areas into serval regions (membranes) and to execute these algorithms on a modern electronic computer. While this approach leads to novel algorithms for solving many complex real-world problems, the efficiency of these algorithms is still bounded by the physical limitations of electronic computer. Another approach, called the direct membrane algorithm approach [39]–[41], is to design P systems that are capable of performing algorithms for solving real-world problems, and the computation of these P systems can be executed on hardware, software or living cells as a natural evolution process. While the execution on living cells has a promising potential to significantly improve the computational efficiency, it also faces a lot of challenges such as defining the P system to solve specific real-world problems and making living cells perform the computation defined by a P system. While the construction of a bio-computer relies on the participation of biology researchers [42]–[44], the design of P systems appropriate for solving specific problems is up on computer science research and is considered in this study.

This work follows the direct membrane algorithm approach and chooses the cell-like P system with multi-catalysts (MCC-P), a specific cell-like P system, to develop and to improve a consensus clustering algorithm based on PAM (Prediction Around Medoids) [45] with the q-nearest neighbors. For easy reference, this improved consensus clustering algorithm is called the P-PCC.

A MCC-P system consists of a membrane structure, a multiset of objects and a set of evolution rules (rules for short) for each membrane. Once entering an input membrane, the objects will start evolving according to applicable rules and to improve a consensus clustering algorithm based on the computation required by the consensus clustering algorithm which is robust to noises and outliers.

This work makes the following contributions.

1) A revised PAM is proposed in which only the q-nearest neighbors of the original medoids are considered as candidates of a possible replacement for the original medoids, which improves the computational efficiency. This medoid replacement approach is used as a BP integration strategy of the consensus clustering algorithm which is inherently parallel and distributed.
major factors affecting the performance, of the P-PCC.
If realized in a living cell, the P-PCC will be much more efficient than implemented on an electronic computer.

The remaining of this article is organized as follows. Section II briefly reviews the basic concepts of consensus clustering and the cell-like P systems with multicatalysts. In Section III, the P-PCC algorithm is presented. In Section IV, the simulation results are reported and analyzed. Conclusions are drawn and future works are outlined in Section V.

II. PRELIMINARIES

In this section, some key concepts about consensus clustering are reviewed and the concepts of a MCC-P system are introduced. For more detail, the readers can refer to [1], [47], [48].

A. CONSENSUS CLUSTERING

Consensus clustering is a computing framework that can aggregate the results obtained by several algorithms and produce a better final result.

As shown in Fig. 1, let \( X = \{x_1, x_2, \ldots, x_n\} \) denote a dataset with \( n \) data points. Arbitrary clustering algorithms are used \( p \) times to cluster \( X \) into \( K_1, K_2, \ldots, K_p \) clusters respectively. Generally, different algorithms or the same algorithm with different parameters are chosen to capture different information about the dataset. The corresponding clustering results \( \pi_1, \pi_2, \ldots, \pi_p \) are called BPs where \( \pi_i = \{L_{pi}(x_1), \ldots, L_{pi}(x_n)\} \) with \( L_{pi}(x_j) \in [1, 2, \ldots, K_i] \) for \( 1 \leq i \leq p \) and \( 1 \leq j \leq n \) is a label vector, the element \( L_{pi}(x_j) \) indicates that the data point \( x_j \) belongs to cluster \( L_{pi}(x_j) \) in clustering algorithm \( i \). The consensus function aggregates the BPs and obtains the final result \( \pi \) with clustering quality typically better than that of the best BP. The clustering quality can be evaluated by using the normalized Rand index and the normalized mutual information, among others. Note that consensus clustering may take two forms of input. In the first form, the BPs are the input and the original dataset cannot be accessed. In the second form, the original dataset is the input.

The P-PCC takes the BPs as the input, and uses the object co-occurrence strategy. The details of the method are as follows [48].

The BPs are transformed into a binary dataset \( X^{(2)} = \{x_i^{(2)} | i = 1, 2, \ldots, n\} \):

\[
x_i^{(2)} = \begin{cases} x_{i,1}^{(2)}, \ldots, x_{i,p}^{(2)}, & \text{with} \\
x_{i,1}^{(2)}, \ldots, x_{i,j}^{(2)}, x_{i,i_k}, & \text{and} \\
x_{i,j}^{(2)} = \begin{cases} 1, & \text{if } x_i \text{ belongs to cluster } j \text{ in BP } \pi_i \\
0, & \text{otherwise.} \end{cases}
\end{cases}
\]

The revised PAM is used as the consensus function. The final result can be obtained by clustering this new binary dataset \( X^{(2)} \).

B. MCC-P SYSTEMS

A cell-like P system is a distributed and parallel computational model that mimics the biological behavior of a single living cell. A living cell consists of one or more membranes, such as the plasma membrane and the membranes surrounding the internal organelles, which separate the inner space of the cell into relatively independent regions. These membranes may form a nested structure, in which some membranes may contain one or more other membranes. Each membrane may contain some chemical elements and the type, the number, and the location of these elements are controlled by some biological processes, such as chemical reactions and transmembrane transport. The biological process may be influenced by catalysts. Chemical elements can enter the cell through a specific membrane, and evolve under biological regulations by changing their types, numbers, and locations (e.g., moving among membranes). Evolution can occur simultaneously for different chemical elements.

To model the biological behavior of a living cell, a cell-like P system uses a hierarchical structure of regions to represent the nested membrane structure, a multiset of objects to represent chemical elements, and a set of rules to mimic the regulation of the biological processes. In this work, a novel variant of the cell-like P system, called the MCC-P system represented by \( \Pi \) of degree \( m \), is defined as follows,

\[
\Pi = (O, \mu, w_0, R, \rho, i_{in}, i_{out}),
\]

where:

- \( O \) is the alphabet, i.e., a set of symbols that denotes various types of objects in \( \Pi \).
- \( \mu \) is a nested set of \( m \) pairs of matching square brackets, serving as a serialized representation of a hierarchical structure of \( m \) membranes. Each pair of [ ], for \( i = 1, \ldots, m \), is called a membrane. For example, \([]\) represents a membrane structure in which membrane 1 contains membranes 2 and 3.
- \( w_0 = \{w_{1,0}, w_{2,0}, \ldots, w_{m,0}\} \) is the set of objects initially in each membrane, where \( w_{1,0} \) is the initial set of objects in membrane 1.
- \( R = \{R_1, \ldots, R_m\} \) is a set of rules, where \( R_i \) is the set of rules for membrane \( i \). Each rule is in the form of \( r_{ij} : u_Q \rightarrow v \), where \( u \) is a multiset of symbols from \( O, v \) is a multiset of tuples from \( (\alpha, \text{tar}) | \alpha \in O, \text{tar} \in \{\text{here, out, in}\} \), and \( Q \) is a list of catalysts, i.e., promoters and inhibitors represented by special symbols in \( O \). A rule can associate with multiple catalysts, different catalysts are separated by commas. A rule is enabled when any catalyst is satisfied. A catalyst can contain either or both promoters and inhibitors. A catalyst is satisfied only when a promoter but no inhibitor appears. To distinguish promoters and inhibitors, a symbol \( \neg \) is placed before an inhibitor. If a rule does not need a catalyst, \( Q \) can be omitted.
- \( \rho \) is a partial ordering relationship of the rules, where \( r_1 > r_2 \) means the priority of \( r_1 \) is higher than that of
$r_2$, and $r_1 = r_2$ means the priorities of $r_1$ and $r_2$ are the same. Rules with higher orders have higher priorities for activation.

- $i_{in}$ is the label of the input membrane where objects enter the system $\Pi$.
- $i_{out}$ is the label of the output membrane where the objects are stored showing the computation result.

The computation starts in the initial state when objects are placed into the input membrane. The system then evolves into new states in steps. At each step, one or more enabled rules for membranes are activated and fired in a non-deterministic maximally parallel fashion. Specifically, a rule $uQ \rightarrow v$ in a membrane is enabled if 1) at least one catalyst listed in $Q$ is satisfied in the membrane; 2) if every object specified in $u$ presents in the membrane and 3) no rule with a higher priority is enabled in the membrane. In the rule, $u$ is called the activation set of the rule. When an enabled rule is activated, it will be fired, which will consume the objects in $u$ and create and transfer objects according to $v$. If $v$ contains an $(a, \text{here})$, then object $a$ will be created and kept in the current membrane (note that $\text{here}$ can be omitted). If $v$ contains an $(a, \text{out})$, then object $a$ will be created and moved from the current membrane into the immediate parent membrane (the one immediately outside the current membrane). If $v$ contains an $(a, \text{in})$, then object $a$ will be created and moved into one of the immediately nested membranes selected at random. In a special case, a specific inner membrane, say membrane $j$, can be specified by $(a, \text{in } j)$. At any time in any membrane, an object may be listed in the activation sets of different rules, but it can only stay in one of those sets. It is undetermined about which set it should be in. Furthermore, there are many ways to group objects to form an activation set for a rule, and it is undetermined how objects are grouped. No matter how these objects are grouped, during each step of the computation, as many enabled rules as possible are activated and fired within their respective membranes which means no further rule can be enabled, and the firing of all the rules in all membranes will take place at the same time, without any particular order. This style of computation is so-called non-deterministic maximally parallel computation and is a characteristic of a P system. After a step of the computation, the system is transferred into a new state, and in the next step, another set of rules may be activated and fired. The computation halts when no more rule is enabled.

At that point, the specified set of objects remaining in the output membrane represents the computation results.

It is emphasized that while it is possible to implement the computation of a cell-like P system in a conventional computer (with or without parallel programming), the real power of the non-deterministic maximally parallel computation can only be fully unleashed in a novel biocomputing environment.

### III. A PAM-BASED CONSENSUS DIRECT MEMBRANE CLUSTERING ALGORITHM

In this section, the improved consensus clustering algorithm is developed based on cell-like P systems with multi-catalysts in the form of direct membrane algorithm (P-PCC). The PAM based on the $q$-nearest neighbors is first constructed as the BPs aggregation strategy of the P-PCC. The whole P-PCC algorithm with the form of MCC-P system is then specified. The computation process of the P-PCC is also given to show its usability and effectiveness. The flowchart of proposed P-PCC algorithm is shown in Fig. 2.

![Flow chart of the proposed P-PCC algorithm.](image)

**A. PAM BASED ON THE $q$-NEAREST NEIGHBORS**

PAM is a classical $K$-medoids clustering algorithm, while the $K$-medoids clustering algorithm is an improvement of the $K$-means algorithm. The $K$-medoids clustering algorithm is...
more robust to noises and outliers than the K-means algorithm [45].

For a dataset \( X = \{ x_1, x_2, \ldots, x_n \} \) with \( n \) data points such that \( x_i \in \mathbb{R}^m \), PAM clusters the data points into \( K \) clusters \( C = \{ C_1, C_2, \ldots, C_K \} \) with the following properties:
1) \( C_k \neq \emptyset \), for \( 1 \leq k \leq K \);
2) \( \bigcup_{k=1}^{K} C_k = X \);
3) \( C_k \cap C_{k'} = \emptyset \), for \( k' \neq k'' \) and \( 1 \leq k', k'' \leq K \).

The absolute-deviation criterion in the following is used to measure the clustering quality:

\[
E = \sum_{k=1}^{K} \sum_{x_i \in C_k} \text{dist}(x_i, o_k),
\]

where \( x_i \) is data point \( i \) in the dataset, \( o_k \) is the medoid of cluster \( C_k \), \( \text{dist}(x_i, o_k) \) is the absolute deviation between \( x_i \) and \( o_k \) representing the dissimilarity, and \( E \) is the sum of the absolute deviations.

The aim of the PAM algorithm is to partition the dataset \( X \) into \( K \) clusters with the minimum value of \( E \). The steps of PAM are as follows:

1) Randomly select \( K \) data points as the initial medoids.
2) Assign each remaining data point \( i \) to cluster \( k' \) such that \( \text{dist}(x_i, o_{k'}) = \min \{ \text{dist}(x_i, o_k) \mid k = 1, \ldots, K \} \).
3) Randomly select a non-medoid point \( o'_k \) from \( C_k \) as a candidate medoid for \( C_k \).
4) Compute \( E' \) using (1) when \( o'_k \) temporarily replaces \( o_k \) as the new medoid of \( C_k \).
5) Replace \( o_k \) by \( o'_k \) if \( E' < E \).
6) Repeat Steps 2 to 5 until none of the medoids can be replaced to further reduce the value of \( E \).

This work extends PAM in two ways. First, flexibility is introduced to specify different distance metrics dynamically. The PAM originally does not take alternative distance measures when computing \( E \) using (1). However, using different distance measures according to the characteristics of specific applications and datasets may result in improved clustering quality. Furthermore, a robust measure should work well when no external information is provided. In the experiments, four distance measures are used, including the Euclidean \( d_{\text{eucl}} \), the KL-divergence \( d_{\text{kl}} \), the cosine \( d_{\text{cos}} \) and the \( L_p \) distance \( d_p \) (see Table 1), as well as their corresponding normalized distances \( d_{\text{eucl}}, d_{\text{kl}}, d_{\text{cos}} \) and \( d_{\text{lp}} \).

Second, only the \( q \)-nearest neighbors of \( o_k \) are considered when selecting a new candidate medoid. The corresponding distance measure used to compute \( E \) is used to measure the closeness between a data point and \( o_k \) for this purpose. The conventional PAM randomly selects a non-medoid data point \( o'_k \) as a candidate medoid, and the absolute-deviation criterion \( E' \) of the whole dataset is computed. For large data sets, the computational time is long limiting the performance of the PAM algorithm. Furthermore, when \( o'_k \) and \( o_k \) are compared, only the data points within the current cluster \( C_k \) are considered. In deciding if the current medoid should be replaced, only the values of \( \sum_{x_i \in C_k} \text{dist}(x_i, o'_k) \) and \( \sum_{x_i \in C_k} \text{dist}(x_i, o_k) \) are compared. Since the data points and the medoid in each cluster change during the computation, every data point has the opportunity to become the candidate medoid. This modification to the algorithm does not affect the clustering quality of PAM. These two changes can improve the computational efficiency of the PAM algorithm.

The steps of the improved PAM algorithm based on the \( q \)-nearest neighbors are as follows:

1) Generate the \( q \)-nearest neighbors of each data point in the data set.
2) Randomly select \( K \) data points as the initial medoids.
3) Assign each non-medoid data point \( i \) to the cluster \( k' \) such that \( \text{dist}(x_i, o_{k'}) = \min \{ \text{dist}(x_i, o_k) \mid k = 1, \ldots, K \} \).
4) Randomly select a non-medoid point \( o'_k \) from \( C_k \) as a candidate medoid for \( C_k \).
5) Compute \( E' \) using (1) when \( o'_k \) temporarily replaces \( o_k \) as the new medoid of \( C_k \).
6) Replace \( o_k \) by \( o'_k \) if \( E' < E \).
7) Repeat Steps 4 to 6 until \( \sum_{x_i \in C_k} \text{dist}(x_i, o_k) \) cannot be reduced. The final \( o_k \) is the best medoid in \( C_k \).
8) Repeat Steps 3 to 7 until none of the medoids can be replaced.

### Table 1. Four distance measures.

| Distance measure | \( \text{dist}(x_i, o_k) \) |
|------------------|-------------------------|
| Euclidean        | \[ \| x_i - o_k \|_2 = \left( \sum_{v=0}^{m-1} (x_{iv} - o_{kv})^2 \right)^{\frac{1}{2}} \] |
| KL-divergence    | \[ \sum_{v=0}^{m-1} x_{iv} \log \frac{x_{iv}}{o_{kv}} \] |
| Cosine           | \[ \sqrt{\sum_{v=0}^{m-1} x_{iv}^2} \sqrt{\sum_{v=0}^{m-1} o_{kv}^2} \] |
| \( L_p \)        | \[ \| x_i - o_k \|_p = \left( \sum_{v=0}^{m-1} (x_{iv} - o_{kv})^p \right)^{\frac{1}{p}} \] |

**B. P-PCC WITH THE FORM OF MCC-P SYSTEM**

This section describes a specific MCC-P system to develop the P-PCC and discusses the rules in each membrane for performing the computation.

It is common knowledge that in a cell-like P system the number of membranes is inversely proportional to the number of evolution rules. In the design of the MCC-P system, a trade-off is made between the number of membranes and the number of rules in a way similar to the space-time trade-off in conventional algorithms. Hence, the MCC-P system has one membrane for each cluster, plus three additional membranes: one for data preprocessing (obtaining the \( q \)-nearest neighbors.
for each data point), one for storing intermediate results during the computation, and one for storing the final result.

\[ \pi = (O, \mu, w_0, R, \rho, i_{in}, i_{out}) \]

The details of each component in this definition are provided in the following.

The alphabet \( O \) contains objects (i.e., symbols) that denote various types of information about the data and status of the computation. Specifically, \( O = \{a_i, \hat{U}_{ij}, U_{ij}, D_{ij}, c, c_{ij}, c'_{ij}, A_{ki}, A'_{ki}, B_{ki}, a_i, d_i, b_k, \xi, \eta, \theta, a_{ik}, s, s', \sigma, \rho, d, \delta, \sigma, e_i, \tau \} \) for \( 1 \leq i, j \leq n \) and \( 1 \leq k \leq K \), where \( a_i \) denotes data point \( x_i \), \( \hat{U}_{ij} \), \( U_{ij} \) and \( D_{ij} \) denote one unit of distance between data points \( x_i \) and \( x_j \), \( c \) denotes a neighbor of a medoid, \( c_{ij} \) indicates that \( x_j \) is a neighbor of \( x_i \), \( c'_{ij} \) indicates that the operation on \( c_{ij} \) has a neighbor of a medoid, \( A_{ki} \), \( A'_{ki} \) and \( B_{ki} \) indicate that \( x_k \) is the medoid of cluster \( k \), \( a_i \) denotes the candidate medoid, \( d_i \) denotes a neighbor of data point \( x_i \), \( b_k \) is an auxiliary to choose the medoid of cluster \( k \), \( \xi \) indicates that a data point has been assigned to a certain cluster, \( \psi \) indicates that the medoid of a class does not change, \( \alpha \) indicates that all the \( K \) clusters have been formed, \( \theta \) indicates that not all the \( K \) clusters have been formed, \( a_{ik} \) indicates that data point \( x_i \) belongs to cluster \( k \), \( s \) denotes one unit of distance among data points in a certain cluster and the medoid, \( s' \) indicates one unit of distance among data points in a certain cluster and the candidate medoid, \( \sigma \) indicates that the current candidate medoid cannot reduce the absolute deviation, \( \eta \) indicates that the current candidate medoid cannot reduce the absolute deviation, \( d \) indicates that an iteration in a certain cluster ended, all \( \delta, \delta', \sigma, e_i \) and \( \tau \) are auxiliaries to control the computational process of the system and \( \lambda \) denotes the empty set.

For any object \( s \in O \) and any integer \( f > 0 \), \( s' \) denotes \( f \) copies of \( s \).

The membrane structure is defined as \( \mu = \{ [\emptyset], [1], [2], \ldots, [K], [K+1]\} \) and is shown graphically in Fig. 3.

The set of initial objects \( w_0 \) is as follows. For \( i = 0, 1, \ldots, K \), membrane \( i \) contains nothing, i.e., \( w_i = 0 = \lambda \), and membrane \( K + 2 \) contains \( w_{K+2} = (\beta_1, \beta_2, \ldots, \beta_K, \delta^K) \).

The input and output membranes are membrane 0, i.e., \( i_{in} = 0 \), and membrane \( K + 1 \) (i.e., \( i_{out} = K + 1 \)), respectively.

The set of rules consists of three subsets: \( R_0 \) is the set of rules for membrane 0; \( R_{K+2} \) is the set of rules for membranes \( K + 2, \ldots, K \), and \( R_k \), for \( 1 \leq k \leq K \), is the set of rules for membranes \( k \). The rules in \( R_k \), for \( 1 \leq k \leq K \), are similar. The partial ordering of the rules is given by \( \rho : \{ r_{ij} \} : (ij = 0, 1, \ldots, K, K + 1, \ldots, K + 2, k) \rightarrow (r_{K+2} > r_{K+3} > \cdots > r_{K+2} > 0) \).

Note that in the MCC-P system, the range for parameters \( i, i_1, \ldots, i_k, j_1, j_2, \ldots, j_{n-1} \) is 1, \( \ldots, n \) and the range for \( k, k_1, k_2 \) is 1, \( \ldots, K \) unless explicitly indicated otherwise.

The rules for membrane 0 consist of

\[
\begin{align*}
\rho & : r_{0,1} : \hat{U}_{ij} \rightarrow D_{ij} U_{ij} \\
\rho & : r_{0,2} : (d_i) - D_{ij} \rightarrow c_{ij} \rightarrow c_{ij} \\
\rho & : r_{0,3} : (S_{n-1})_{c_i} \rightarrow \lambda \\
\rho & : r_{0,n+1} : (a_i) - a_i \rightarrow (a_i, \text{in } K + 2) \\
\rho & : r_{0,n+2} : (U_{ij}) - d_i \rightarrow (U_{ij}, \text{in } K + 2) \\
\rho & : r_{0,n+3} : (c_{ij}) - c_{ij} \rightarrow (c'_{ij}, \text{in } K + 2) \\
\end{align*}
\]

The computation of this system starts by placing objects \( a_i, \hat{U}'_{ij}, c_{ij} \) and \( d_{ij} \) for \( 1 \leq i, j \leq n \) into membrane 0.

1) FINDING \( q \)-NEAREST NEIGHBORS OF EACH DATA POINT

According to the partial ordering, rule \( r_{0,1} \) fires first, which transforms each \( \hat{U}_{ij} \) into one \( D_{ij} \) and \( U_{ij} \). Since there are \( f_{ij} \) copies of \( \hat{U}_{ij} \), this serves to gather the distance information for each data point \( x_i \). Let \( D_i = \{D_{ij}\} \) and \( U_i = \{U_{ij}\} \).

Rules \( r_{0,2}, \ldots, r_{0,n} \) are used to generate the \( q \)-nearest neighbors of each data point.

Rule \( r_{0,3} : (S_{n-1}) \rightarrow \lambda \) is enabled only when the promoter, a copy of \( c_i \), presents in membrane 0. For any data point \( x_i \), the set \( S_{n-1} = \{D_{i1} \ldots D_{i(i-1)} D_{ij(i+1)} \ldots D_{in}\} \) contains one unit of distance between \( x_i \) and the other \( n - 1 \) data points. Firing \( r_{0,3} \) once will consume one unit of these distances. Because of the non-deterministic maximally parallel computation fashion, if a rule is activated by multiple copies of the activation set \( u \), the rule fires multiple times, once for each copy of the activation set \( u \), at the same time. After \( r_{0,3} \) fires multiple times, all sets \( S_{n-1} \) are consumed, and at least one type of object, say \( D_{ij} \), will no longer be present in membrane 0, indicating that \( x_i \) is the nearest neighbor of \( x_j \). Hence, rule \( r_{0,3} \) identifies the nearest neighbor for all data points.

As soon as all copies of \( D_{ij} \) are removed by the firing of \( r_{0,3} \) and \( x_j \) has not been marked as the nearest neighbor of \( x_i \), rule
1. **Rules for Initial Medoids Assignment**

   - **Rule 1** ($r_{K+1}$): $a_i \rightarrow (a_i, \text{out})$
   - **Rule 2** ($r_{K+2}$): $(\theta A_{k1}B_{k1}) \rightarrow (\psi A_{k1}, \text{out})$
   - **Rule 3** ($r_{K+3}$): $A'_{k1} \rightarrow A_{k1}B_{k1}$
   - **Rule 4** ($r_{K+4}$): $(\theta a_i c_{ij}) \rightarrow o_j c'_{ij}$
   - **Rule 5** ($r_{K+5}$): $(U_{ij}a_ia_\theta a_{\theta j}) \rightarrow e_j U_{ij} s'$
   - **Rule 6** ($r_{K+6}$): $s' \rightarrow \lambda$
   - **Rule 7** ($r_{K+7}$): $(s' y o_j A_{kj}) \rightarrow \theta A_{kj} \sigma$
   - **Rule 8** ($r_{K+8}$): $s' o_j A_{kj} \rightarrow \theta A_{kj} \eta$

2. **Randomly Selecting Initial Medoids**

   Rule $r_{K+2,1}$ is activated as soon as objects are moved into membranes $K + 2$ by rule $r_{0,n+1}$. When it is fired, the distance information $U_{ij}$ and the $q$-nearest neighbor information $c_{ij}$ are placed into membranes 1 to $K$, with a copy in membrane $K + 2$.

   Rule $r_{K+2,2}$ is used to select a random medoid for each cluster. Since there is exactly one object $b_k$, for $1 \leq k \leq K$, and exactly one $a_i$ for each data point, the activation of $r_{K+2,2}$ for any pair of $i$ and $k$ is a random event. Once activated, this rule creates a unique copy of $A_{k}$ which indicates that the medoid of cluster $k$ is $x_i$, i.e., $o_k = x_i$. Let $A = \{A_{k}\}$ be the objects describing the current set of medoids. Note that $r_{K+2,2}$ and $r_{K+2,1}$ will be activated in the same computational step.

3. **Assigning Non-Medoid Data Points to Clusters**

   Rules $r_{K+2,3}, \ldots, r_{K+2,6}$ are used to assign data points to the clusters.

   Rule $r_{K+2,4}$ is enabled for data points $i$ and $j$ only if 1) $x_i$ is a medoid of cluster $k$ (indicated by promotor $A_{ki}$), 2) $x_j$ is a non-medoid data point (indicated by promotor $aj$), and 3) the rule has not been fired before in this loop (indicated by promotor $\delta^K$). The firing of this rule once creates one copy of $D_{jk}$ to denote a unit of distance between $x_j$ and the medoid of cluster $k$ (i.e., $o_k$), an object $\delta'$ to denote that this rule has been fired, and an object $\sigma$ to be used as a promoter in rule $r_{K+2,5}$. Because of the maximally parallel nature of the $P$ system, the firing of rule $r_{K+2,4}$ finds the distance between each non-medoid data point and the $K$ medoids at one step.

   Similar to rule $r_{0,1+2}$, the firing of $r_{K+2,6}$ will consume a unit of distance $D_{j1}D_{j2} \ldots D_{jK}$ between non-medoid $x_j$ and medoid of each cluster. As a result, the object denoting the shortest distance will be removed completely. If this object is $D_{jk}$, its absence in membrane $K + 2$ will enable rule $r_{K+2,5}$, which will move $a_j$ into membrane $k$ when fired, resulting in data point $x_i$ being assigned to cluster $k$.

   Once all data points are assigned to their nearest clusters, rule $r_{K+2,7}$ fires to reset membrane $K + 2$ back to the initial state.

   Rule $r_{K+2,8}$ is used to detect if any medoid has been replaced and trigger the computational process in membranes 1 to $K$. If any medoid in the $K$ clusters is replaced, object $\theta$ is sent to each of the membranes 1, 2, $\ldots$, $K$; otherwise, if none of the medoids in the $K$ clusters is replaced, which means the $K$ clusters have been formed, object $\omega$ instead of $\theta$ is sent to each of the membranes 1, 2, $\ldots$, $K$. At the same time, $A'_{k1}$ is sent to membrane $k$ to trigger further processing.

   Rule $r_{K+2,9}$ is enabled when the whole clustering process ends. This rule is fired to move object $a_{ik}$, which is sent by rule $r_{k,1}$, to the output membrane $K + 1$ to record the assignment of data point $x_i$ to cluster $k$.

Each of the rules for the remaining membranes consists of:

   - **Rule 1**: $r_{k,1}$: $(a_i) \rightarrow (a_{ik}, \text{out})$
   - **Rule 2**: $r_{k,2}$: $(\theta A_{j1}B_{k1}) \rightarrow (\psi A_{k1}, \text{out})$
   - **Rule 3**: $r_{k,3}$: $A'_{k1} \rightarrow A_{k1}B_{k1}$
   - **Rule 4**: $r_{k,4}$: $(\theta a_i c_{ij}) \rightarrow o_j c'_{ij}$
   - **Rule 5**: $(U_{ij}a_ia_\theta a_{\theta j}) \rightarrow e_j U_{ij} s'$
   - **Rule 6**: $s' \rightarrow \lambda$
   - **Rule 7**: $(s' y o_j A_{kj}) \rightarrow \theta A_{kj} \sigma$
   - **Rule 8**: $s' o_j A_{kj} \rightarrow \theta A_{kj} \eta$
4) REDETERMINING THE BEST MEDOID IN EACH CLUSTER

The best medoid is determined among the current medoid and all its q-nearest neighbors in each cluster in the corresponding membrane.

Rule $r_{k,3}$ is activated after object $A'_{ki}$ is moved into membrane $k$ by rule $r_{k,2}$, which transforms each $A'_{ki}$ into one $A_{ki}$ and one $B_{ki}$, where $A_{ki}$ stores the current medoid and $B_{ki}$ stores the original medoid in this iteration.

- If membrane $k$ contains no $a_i$, rule $r_{k,2}$ is activated to send $\psi, A_{ki}$ and $\delta$ out.
- If membrane $k$ contains object $a_i$, rules $r_{k,4}$ to $r_{k,7}$ are used multiple times to find the best medoid.

**Selection of a non-medoid point $o'_k$ in $C_k$ from the q-nearest neighbors of the original medoid $o_k$ as a candidate medoid**

Rule $r_{k,4}$ is enabled and activated by $B_{ki}$ generated in rule $r_{k,3}$ as a promoter. One neighbor of the original medoid is chosen at random as a candidate medoid, and object $a_j$ is transformed into $o_j$ to mark the candidate medoid. At the same time, $c_{ij}$ is transformed into $c'_{ij}$ which indicates that $a_j$ does not need to be chosen again. If a certain $a_j$ is not in this membrane, $c_{ij}$ is also transformed into $c'_{ij}$ which indicates that $a_j$ does not need to be chosen again but for a different reason from that just discussed above.

**Computation of $\sum_{i \in C_k} dist(x_i, o'_k)$**

Rule $r_{k,5}$ is enabled for data points $i, j$ and $t$ only if: 1) $x_i$ is a medoid of cluster $k$ (indicated by promoter $A_{ki}$), 2) $x_j$ is a candidate medoid of cluster $k$ (indicated by promoter $o_j$), 3) $x_j$ is neither a medoid nor a candidate medoid in the cluster (indicated by $a_j$), and 4) the rule has not been fired before this loop (indicated by inhibitor $e_j$). The firing of this rule once creates one copy of $s'$ representing one unit of distance between $x_i$ and $x_j$, one copy of $s$ representing one unit of distance between $x_i$ and $x_j$, and an object $e_j$ indicating that this rule has been fired. Because of the maximally parallel nature of the P system, the firing of rule $r_{k,5}$ calculates and stores the values of $\sum_{i \in C_k} dist(x_i, o'_k)$ and $\sum_{i \in C_k} dist(x_i, o_k)$ in the form of the numbers of $s'$ and $s$, respectively, in one step.

**Replacement of $o_k$ with $o'_k$ if**

$$\sum_{i \in C_k} dist(x_i, o'_k) < \sum_{i \in C_k} dist(x_i, o_k)$$

Rule $r_{k,6}$ is activated to remove the same number, determined by whichever is smaller, of copies of $s$ and $s'$, and then rules $r_{k,7}$ and $r_{k,8}$ are used to compare the two medoids. If all copies of object $s$ are consumed, which indicates that the candidate medoid cannot reduce the absolute deviations, rule $r_{k,7}$ is enabled to transform $o_j$ back into $a_j$ and generate an object $\sigma$ which indicates that the medoid is not changed. Otherwise, rule $r_{k,8}$ is enabled to transform $A_{ki}$ into $A'_{ki}$ which indicates that $x_j$ is set as the new medoid, and to generate an object $\eta$ which indicating that the medoid has been changed.

Rules $r_{k,9}, \ldots, r_{k,13}$ are used to reset membrane $k$ to its initial state and trigger the computational process in membrane $K + 2$.

Once the best medoid is found, rule $r_{k,9}$ fires and generates an object $\tau$ to enable $r_{k,10}$. Rule $r_{k,10}$ is used to transform $c'_{ij}$ back to $c_{ij}$ and remove auxiliary objects $e_i$. If the membrane does not contain object $\eta$ meaning that the medoid is not changed, the first part of $r_{k,11}$ is activated. The first part of $r_{k,11}$ transforms $\sigma$ and $\tau$ into $d$ to be used to enable rules $r_{k,12}$ and $r_{k,13}$, and into $\psi$ to be sent out to show that the medoid in this membrane is not changed. Otherwise, the rest of $r_{k,11}$ is activated to remove $\sigma$ and $\eta$, and to transform $\tau$ into $d$. Rule $r_{k,12}$ is enabled by $d$, as a promoter, to send $a_i$ out. Similarly, rule $r_{k,13}$ is enabled by $d$ to send $\delta$ out to indicate that the computation process in this membrane during this iteration is over, and to send $A_{ki}$ out to mark the updated medoid.

**C. COMPUTATIONAL COMPLEXITY**

The time complexity of the P-PCC is $O(16l + 2q)$, where $l$ is the number of iterations of PAM based on the q-nearest neighbors. While the time complexity of KCC is $O(in K)$, where $l$ is the number of iterations of the K-means clustering approach, $n$ is the size of the dataset, $r$ is the number of BPs and $K$ is the true number of clusters. Because the values of the number of iterations and $q$ are usually small, P-PCC can improve the time efficiency.

**D. AN EXAMPLE**

An example is used to show how the system works. Let $X = \{ (1, 1), (1, 2), (1, 1.5), (1.5, 1.5) \}$, $p = 10$, $K_1, K_2, \ldots, K_p = 2$ and $q = 2$. The K-means algorithm is used as the BP generation algorithm using the squared Euclidean distance. Then $X$ is transformed to $X^{(2)} = \{ x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, x_4^{(2)} \}$, where $x_1^{(2)} = < 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0 >$, $x_2^{(2)} = < 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0 >$, $x_3^{(2)} = < 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0 >$, and $x_4^{(2)} = < 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0 >$. The distance matrix $D$ is constructed by computing the squared Euclidean distance between data points in $X^{(2)}$, and multiplying the distance by 100 and rounding off, thus getting natural numbers.

$$D = \begin{pmatrix}
0 & 283 & 200 & 245 \\
283 & 0 & 200 & 245 \\
200 & 200 & 0 & 141 \\
245 & 245 & 141 & 0
\end{pmatrix}$$
The computational process is shown in the Appendix. The four data points are partitioned into two clusters by the MCC-P system constructed for the P-PCC.

IV. EXPERIMENTS AND ANALYSES

In this section, the performance of the P-PCC is demonstrated using publically available datasets. The major factors affecting the performance of the P-PCC are also examined.

A. EXPERIMENT SETUP

Fifteen datasets from the UC Irvine Machine Learning Repository [46] are used in the experiments. Table 2 shows the basic characteristics of these datasets. The normalized Rand index \( R_n \) is used to measure the clustering quality. A larger \( R_n \) value means a higher clustering quality. The BPs \( \pi_i (1 \leq i \leq p) \) are generated by the MATLAB \textit{Kmeans} function using the squared Euclidean distance. To generate the BPs, two strategies, random parameter selection (RPS) and random feature selection (RFS), are used. RPS randomizes the number of clusters \( K_i \) in a given interval, and RFS randomizes the selected features. The default settings are as follows. The parameter \( K \) for the P-PCC is set to the actual number of clusters \( K_n \). The number of BPs \( p \) is set to 100. The weight of each BP \( w_i \) is set to the same value 1. The default generation strategy is RPS, and the interval for the number of clusters is set to \([K_{min}, \sqrt{n}]\). In RFS, the number of clusters is set to \( K_n \), and two features are selected randomly from all the features. All experiments were simulated by MATLAB R2019b running on a Windows 7 platform of the 64-bit edition. The PC has an Intel Core i7-5500U 2.4 GHz CPU and 8 GB of RAM.

| dataset            | #objects | #features | #clusters |
|--------------------|----------|-----------|-----------|
| Breast_w           | 683      | 9         | 2         |
| Ecoli              | 332      | 7         | 6         |
| Iris               | 150      | 4         | 3         |
| Dermatology        | 358      | 33        | 6         |
| Wine               | 178      | 13        | 3         |
| Frosnmfcc          | 7195     | 22        | 4         |
| Cardiotocography   | 2126     | 21        | 10        |
| Avila              | 10430    | 10        | 11        |
| Abalone            | 4177     | 7         | 28        |
| Seeds              | 210      | 7         | 3         |
| Alcohol QCM Sensor Dataset | 125 | 10 | 5 |
| User Knowledge Modeling | 258 | 5 | 4 |
| Vertebral Column   | 310      | 6         | 2         |
| Somerville Happiness Survey | 143 | 7 | 2 |
| Glass Identification| 214      | 10        | 6         |

B. CLUSTERING QUALITY

The clustering results, measured by \( R_n \), obtained by the P-PCC using different distance measures are shown in Table 3. Clustering results in Table 3 show that the P-PCC has the best performance on nine out of the fifteen datasets using the normalized KL-divergence distance measure. Therefore, the normalized KL-divergence is the most robust distance measure, and is set as the default distance measure if no external information is provided.

The clustering qualities of the P-PCC and another \( K \)-means-based consensus clustering algorithm (KCC) [48] are compared in Fig. 4, where the normalized KL-divergence is chosen as the distance measure. As can be seen, the clustering qualities of the P-PCC and the KCC on twelve out of the fifteen datasets are almost the same, while the P-PCC obtained better results on the Wine, Seed, and Some. datasets.

In summary, the P-PCC performs at least as well as the KCC in terms of clustering quality, and is robust to datasets with noises and outliers.

C. FACTORS AFFECTING THE CLUSTERING RESULTS

1) NUMBER OF NEAREST NEIGHBORS \( q \)

Experiments are performed to investigate the effects of the number of nearest neighbors \( q \) on the quality and on the efficiency of P-PCC. The results using for datasets are shown in Fig. 5. The range of \( q \) is set to 1, 2, \ldots, \( n - 1 \) and the value of \( R_n \) and the running time are used to measure the results. The results show that for all tested cases except for Iris with \( q = 2 \), \( R_n \) is almost not affected by the choice of \( q \). On the other hand, the running time increases in decreasing speeds in all cases as \( q \) increases. Based on the experimental results, \( q = \min(\sqrt{n}, 20) \) is used, which gives overall better results, in this work to report the experimental results.

2) NUMBER OF BPs

The number of BPs \( p \) is set to 10, 20, \ldots, 90, and the experiments are repeated for 100 times for each value of \( p \). The clustering results are shown in Fig. 6 for four datasets. As shown in Fig. 6, the clustering quality improves as the value of \( p \) increases, and becomes stable when \( p \) reaches to about 50.

3) QUALITY AND DIVERSITY OF BPs

The Ecoli, Iris, Wine and Frogsmfcc datasets are used for illustration. The frequency distributions of the BPs with different \( R_n \) values of these datasets are shown in Fig. 7. As can be seen, the clustering qualities of most BPs, represented by \( R_n \), are low. For the Ecoli dataset, only 13% of the BPs have \( R_n \) values above 0.4, and the average of the \( R_n \) values of all BPs is 0.3084. Similarly, for the Wine dataset, no BP has a \( R_n \) value above 0.4, and the average of the \( R_n \) values of all BPs is 0.2134; and for the Frogsmfcc dataset, only 5% of the BPs have \( R_n \) values above 0.4, and the average of the \( R_n \) values of all BPs is 0.1158. The qualities of the BPs of the Iris dataset are better with 60% of the BPs having \( R_n \) values above 0.4, and the average of the \( R_n \) values of all BPs is 0.4596.

To reveal the relationship between the clustering quality of the P-PCC and the quality/diversity of the BPs, the BPs are first sorted according to their \( R_n \) values in descending order. The BPs with larger \( R_n \) values are then gradually removed to see the changes in the \( R_n \) values of the P-PCC. The results are shown in Fig. 8. As can be seen from Fig. 8,
the clustering quality drops when the relatively high quality BPs are removed, which is more evident in the Iris, Wine and Frogsmfcc datasets, indicating that the clustering qualities of the P-PCC mainly depend on the high quality BPs. The BPs with smaller $R_n$ values are also gradually removed, and the changes in the $R_n$ values of the P-PCC are shown in Fig. 9. As can be seen from Fig. 9, the clustering quality basically remains unchanged, indicating that the clustering qualities
of the P-PCC are more affected by the qualities than by the diversities of the BPs.

For the Ecoli dataset, only 13 BPs have $R_n$ values above 0.4. However, the clustering quality starts dropping when 60 BPs with larger $R_n$ values are removed, indicating that the diversity can uphold the clustering quality when the qualities of all BPs are relatively low.

Furthermore, when all BPs are used, the $R_n$ values of the P-PCC on the Ecoli, Iris, Wine and Frogsmfcc datasets are 0.4095, 0.7302, 0.4024 and 0.5152 respectively, which are basically consistent with the best $R_n$ value of all BPs. These results indicate that the P-PCC can take advantage of the high quality BPs even if the percentage of the high quality BPs is very low.

4) THE GENERATION STRATEGY OF BPs

Different generation strategies of BPs are used to improve the clustering quality. In the following, the results of two such strategies are reported. One strategy uses a different interval for the number of clusters and the other uses only a selected number of features for each data set.

For large scale datasets, the interval of the number of clusters $[K_a, √n]$ is too large. When dealing with these large scale datasets, this interval is set to $[2, 2K_a]$. The clustering results, measured by $R_n$, obtained by different distance measures are shown in Table 4. Results in Table 4 show that the clustering quality of the Breast_w dataset is substantially improved, with the average values of $R_n$ increased from 0.1944 to 0.8465, while the clustering qualities of the other datasets are almost the same as the previous results. These results indicate that the interval of the number of clusters $[2, 2K_a]$ can improve the clustering quality of large-scale datasets sometimes.

For data with multi-features, i.e., with 10 or more features, the RFS strategy is adopted. Four datasets are used for illustration in Fig. 10. The clustering qualities of the four datasets with multi-features are improved. The value of $R_n$ of the Dermatology dataset reaches 0.7972 when 7 features are selected, while it is only 0.0381 when all features are used. Similarly, the value of $R_n$ of the Wine dataset reaches 0.8000 when 2 features are selected, while it is only 0.3711 when all features are used; the value of $R_n$ of the Frogsmfcc dataset reaches 0.5747 when 11 features are selected, while it is only 0.5152 when all features are used; the value of $R_n$ of the Avila dataset reaches 0.1025 when 1 feature is selected, while it is only 0.0662 when all features are used.

V. CONCLUSIONS

This work developed an improved PAM-based consensus clustering algorithm using the MCC-P system, called the P-PCC, which is robust to noises and outliers and is able to achieve an acceptable time complexity. Experimental results on fifteen datasets show that the P-PCC performs well for clustering. The results show it is viable to improve conventional clustering algorithms by using the parallel mechanism of membrane computing models.

There has been a research focus on reducing the amount of the computing resources in use. It is of interest to find ways to reduce the amount of computing resources used in the P-PCC.

For further research, it is of also interest to use some other membrane computing models, such as the spiking neural
P systems (SN P systems) [49], to improve the efficiency of consensus clustering. SN P systems are inspired by the mechanism of the neurons that communicate by transmitting spikes. The cells in SN P systems are neurons that have only one type of objects called spikes, which is easier to control in biological experiments. Furthermore, some other clustering algorithms, even data mining algorithms, such as spectral clustering, support vector machines, and genetic
### TABLE 5. The computational process of the example.

| step | membrane 0 | membrane 1 | membrane 2 | membrane 3 |
|------|------------|------------|------------|------------|
| 0    |            |            |            |            |
| 1    |            |            |            |            |
| 2    |            |            |            |            |
| 3    |            |            |            |            |
| 4    |            |            |            |            |
| 5    |            |            |            |            |
| 6    |            |            |            |            |
| 7    |            |            |            |            |
| 8    |            |            |            |            |
| 9    |            |            |            |            |
| 10   |            |            |            |            |
| 11   |            |            |            |            |
| 12   |            |            |            |            |
| 13   |            |            |            |            |
| 14   |            |            |            |            |
| 15   |            |            |            |            |
| 16   |            |            |            |            |

Y. Zhao et al.: Improved Consensus Clustering Algorithm Based on Cell-Like P Systems
algorithms [50], may also be improved by using parallel evolution mechanisms and tree membrane structures [19]. The use of evolutionary computing in the genetic algorithms proved to be effective in solving several problems [51]–[53]. It is also of interest to combine membrane computing models with evolutionary computing to improve the effectiveness and efficiency of problem solution algorithms.

APPENDIX
See Table 5.

REFERENCES
[1] A. Strehl and J. Ghosh, “Cluster ensembles—A knowledge reuse framework for combining multiple partitions,” J. Mach. Learn. Res., vol. 3, no. 12, pp. 583–617, Dec. 2002.

[2] S. Vega-Pons and J. Ruiz-Shulcloper, “A survey of clustering ensemble algorithms,” Int. J. Pattern Recognit. Artif. Intell., vol. 25, no. 3, pp. 337–372, May 2011.

[3] C. Zhong, L. Hu, X. Yue, T. Luo, Q. Fu, and H. Xu, “Ensemble clustering based on evidence extracted from the co-association matrix,” Pattern Recognit., vol. 92, pp. 1–19, Jan. 2019.

[4] J. Yang, L. Sun, and Q. Wu, “Constraint projections for semi-supervised spectral clustering ensemble,” Concurrency Comput., Pract. Exper., vol. 31, no. 20, Oct. 2019, Art. no. e5359.

[5] D. Huang, C.-D. Wang, J.-S. Wu, J.-H. Lai, and C.-K. Kwoh, “Ultra-scalable spectral clustering and ensemble clustering,” IEEE Trans. Knowl. Data Eng., vol. 32, no. 6, pp. 1212–1226, Jun. 2020.

[6] Z. Tao, H. Liu, S. Li, Z. Ding, and Y. Fu, “Robust spectral ensemble clustering via rank minimization,” ACM Trans. Knowl. Discovery Data, vol. 13, no. 1, pp. 1–25, Jan. 2019.

[7] S. Vega-Pons, J. Correa-Morris, and J. Ruiz-Shulcloper, “Weighted partition consensus via kernels,” Pattern Recognit., vol. 43, no. 8, pp. 2712–2724, Aug. 2010.
X. Wang, T. Song, P. Zheng, M. Jiang, and X. Zeng, “A parallel workflow pattern modeling using spiking neural p systems with colored spikes,” IEEE Trans. Nanobiosci., vol. 17, no. 4, pp. 474–484, Oct. 2018.

T. Song, S. Pang, S. Hao, A. Rodriguez-Patón, and P. Zheng, “A parallel image skeletonizing method using spiking neural p systems with weights,” Neuro Process. Lett., vol. 50, no. 2, pp. 1485–1502, Oct. 2019.

T. Song, P. Zheng, M. L. Dennis Wong, and X. Wang, “Design of logic gates using spiking neural p systems with homogeneous neurons and astrocyte-like control,” IEEE Trans., vol. 372, pp. 380–391, Dec. 2016.

J. Baumgardner, K. Acker, O. Adefuye, S. T. Crowley, W. DeLoache, J. O. Dickson, L. Heard, A. T. Martens, N. Morton, M. Ritter, A. Shoecraft, J. Treece, M. Unzicker, A. Valencia, M. Waters, A. Campbell, L. J. Heyer, J. L. Poet, and T. T. Eckdahl, “Solving a Hamiltonian Path Problem with a bacterial computer,” J. Biol. Eng., vol. 3, no. 1, pp. 11, Dec. 2009.

Y. Sakakibara, H. Nakagawa, Y. Nakashima, and K. Yugi, “Implementing in vivo cellular automata using toggle switch and inter-bacteria communication mechanism,” in Proc. 2nd Int. Conf. Bio-Inspired Models Netw. Inf. Comput., 2007, pp. 322–325.

R. Weiss, S. Basu, S. Hooshangi, A. Kalmbach, D. Karig, R. Mehreja, and J. Netravali, “Genetic circuit building blocks for cellular computation, communications, and signal processing,” Natural Comput., vol. 2, no. 1, pp. 47–84, 2003.

L. Kaufman and P. J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis. Hoboken, NJ, USA: Wiley, 2009.

D. Dua and C. Graff, UCI Machine Learning Repository. Accessed: 2017. [Online]. Available: http://archive.ics.uci.edu/ml
WEINING ZHANG received the B.Eng. degree in computer science and engineering from the University of Electronic Science and Technology of China, in 1982, and the M.Sc. and Ph.D. degrees in computer science from The University of Illinois at Chicago, in 1985 and 1988, respectively. He has been with the Department of Computer Science, The University of Texas at San Antonio, since Fall 1999. Prior to that, he was with the Department of Mathematics and Computer Science, University of Lethbridge, Alberta, Canada, from 1989 to 1999. His research interests include database systems, data mining, data privacy, cloud computing, social computing, soft computing, and nature-inspired computing.

MINGHE SUN received the Bachelor of Science degree in metallurgy from Northeastern University, China, in 1982, the M.B.A. degree from The Chinese University of Hong Kong, in 1987, and the Ph.D. degree in business administration with a major in management science and information technology and a minor in operations management from the University of Georgia, in 1992. He has been working with UTSA since he received his doctorate. His research interests include mathematical programming and related areas. His work has appeared in almost all major journals in the management science/operations research/decision sciences area including Management Science, Operations Research, Decision Sciences, Transportation Science, INFORMS Journal on Computing, the European Journal of Operational Research, Decision Support Systems, the Journal of Heuristics, Annals of Operations Research, and Networks, Computers and Operations Research, and so on. He received the prestigious Decision Sciences Institute Elwood S. Buffa Doctoral Dissertation Competition Award, in 1993, and the Decision Sciences Institute Best Theoretical/Empirical Research Paper Award, in 2003 and 2006. At UTSA, he teaches courses in management science/operations management and business statistics at the undergraduate, MBA, and doctoral levels. He received the University of Texas System Chancellor’s Council Outstanding Teaching Award, in 1999.

XIYU LIU received the B.S., M.S., and the Ph.D. degrees in science from Shandong University, Jinan, in 1984, 1987, and 1990, respectively. From 1990 to 1993, he was a Lecturer with Shandong Normal University. From 1993 to 1997, he was an Associate Professor. Since 1997, he has been a Professor. His research interests include membrane computing and data mining. Dr. Liu is a Taishan Scholar of Management Science and Engineering and the Vice President of Computer Education Research Association of China Higher Normal Universities and the Shandong Computer Society.

* * *

VOLUME 8, 2020

154517