Clustering Using Cyclic Spaces of Reversible Cellular Automata

Sukanya Mukherjee
Department of Computer Science and Engineering
Institute of Engineering and Management
Kolkata, West Bengal, 700091, India
sukanya.mukherjee@iemcal.com

Kamalika Bhattacharjee
Department of Computer Science and Engineering
National Institute of Technology
Tiruchirappalli, Tamil Nadu, 620015, India
kamalika.it@gmail.com

Sukanta Das
Department of Information Technology
Indian Institute of Engineering Science and Technology
Shibpur, West Bengal, 711103, India
sukanta@it.iiests.ac.in

This paper introduces a cycle-based clustering technique using the cyclic spaces of reversible cellular automata (CAs). Traditionally, a cluster consists of close objects, which in the case of CAs necessarily means that the objects belong to the same cycle; that is, they are reachable from each other. Each of the cyclic spaces of a cellular automaton (CA) forms a unique cluster. This paper identifies CA properties based on "reachability" that make the clustering effective. To do that, we first figure out which CA rules contribute to maintaining the minimum intracluster distance. Our CA is then designed with such rules to ensure that a limited number of cycles exist in the configuration space. An iterative strategy is also introduced that can generate a desired number of clusters by merging objects of closely reachable clusters from a previous level in the present level using a unique auxiliary CA. Finally, the performance of our algorithm is measured using some standard benchmark validation indices and compared with existing well-known clustering techniques. It is found that our algorithm is at least on a par with the best algorithms existing today on the metric of these standard validation indices.

Keywords: reversible cellular automata; reachability; large length cycle; iterative level-wise clustering; connectivity; silhouette score; Dunn index

1. Introduction

A *cluster* is a collection of similar objects that are also close to each other. The *closeness* between any two target objects is measured by the feature-based distances between them. The clustering technique is a well-studied research topic [1–3] with the aim of effectively distributing the target objects (data records) among the clusters. A cluster also establishes an intrinsic connection among the objects. This connectivity leads us to think of cellular automata (CAs) as natural clusters.

A clustering can be represented as a bijective function $A : D \longrightarrow D$, where D is a (finite) domain of objects to be clustered. Two elements x and y are *close* to each other with respect to A if $A^{k_1}(x) = y$, $A^{k_2}(y) = x$ for some $k_1, k_2 \in \mathbb{N}$. On the other hand, two elements x, z are *not close* with respect to A if $A^k(x) \neq z$ (or, $A^{k'}(z) \neq x$) for some $k, k' \in \mathbb{N}$. In a cellular automaton (CA), a configuration space can be represented as the union of disjoint cyclic subspaces where each cycle connects configurations that are reachable from each other. This "reachability" is the key to clustering with CAs because reachable configurations form cycle(s). Therefore, CAs can act as a function that maintains a bijective mapping among the configurations (target objects) of a cycle (cluster). Two configurations (objects) are close to each other if they are reachable; that is, if they are part of the same cycle (cluster). Moreover, a change in the state of a CA cell is influenced by its neighboring cells. This locality property influences the gathering of similar objects. The number of state changes in a configuration can only affect those configurations that are in the same cycle. So the smallest change in a cell's state within a cycle guarantees that feature-based distances among the target objects of each cluster are as small as possible. Therefore, similar objects in a CA are close to each other and gathered into the same cycle. In this way, a (reversible) CA can work as a natural clustering tool.

This paper uses reversible nonuniform CAs as the proposed model. An object is represented by a set of features, so the CA used for clustering is finite. Hereafter, if not otherwise mentioned, by "CA" we mean a reversible nonuniform CA of finite size under null-boundary conditions that uses Wolfram's rules [4]. Moreover, each feature value is to be mapped into a binary string. By appending all these feature values in binary form, a binary string of length n is formed that we call a *useful configuration*. For the same object, several useful configurations can be formed if the ordering of features is changed. This reordering of features for the given objects guides us to use different CAs to cluster the same set of objects, depending on the useful configuration.

Ideally, a CA of size n distributes 2^n configurations among m cycles where m ranges from 1 to 2^n. To an extreme degree, a CA-based clustering can attract all target objects into one cluster or distribute them among m clusters where the number of target objects is m ($\leq 2^n$)—neither of which is desirable for good clustering. Therefore, a CA is said to be *effective* for clustering if the useful configurations (objects) can be distributed among an optimal number of cycles (clusters). Further, the feature-based distances among the reachable configurations of each cluster have to be as small as possible, such that the intracluster and intercluster isolation, based on feature space, are lower and higher, respectively.

Any clustering technique is driven by two influencing factors: a smaller intracluster distance and an optimal number of clusters. In this paper, our primary target is to diagnose these features. Therefore, the main objective of our basic model is to select CA rules for generating CAs that are capable of connecting the configurations that maintain the minimum possible Hamming distance (because binary CAs are considered for this paper). This is possible if we can select those rules that contribute a minimum change in the cell's state value when a transition occurs between configurations. To find significant rules, we assign some rank to a CA rule based on its intrinsic self-replicating property [5] (a cell restores its previous state). After assigning the ranks, we figure out the proportion of significant rules for designing an n-cell (nonuniform) CA for clustering. Here, n is determined based on the number of features owned by the target objects, which is always finite. However, consecutive configurations of a cycle maintain the minimum distance in the feature space if more significant rules are applied in an n-cell CA; that is, the same cycle connects similar objects but it may increase the number of cycles. So clustering can be viewed as an optimization problem where there is a tradeoff between these two facets: the number of clusters and maintaining a smaller intracluster distance among the objects. This guides us to think about an intelligent arrangement of CA rules for generating an n-cell candidate CA that is capable of maintaining a smaller intracluster distance among objects and generates an optimal number of clusters. We use the framework of a related problem to ensure that a CA has a limited number of cycles—CAs with large length cycles (introduced in [6]). A scheme is developed to select significant rules (Section 4.1) to guarantee that the configurations inside a cycle maintain the minimum possible Hamming distance for the candidate CA. Next, the proportion of significant rules for designing an n-cell (nonuniform) CA for clustering is evaluated to ensure an optimal number of clusters (Sections 4.2 and 4.3).

Definitely, generating a CA for a fixed n maintaining a smaller intracluster distance and a limited number of clusters is a very challenging problem; moreover, distributing the target objects among the desired number of clusters is difficult to ensure. The inherent hardness of this problem motivates us to take an iterative strategy that distributes the target objects into m clusters. The clusters of level i in the proposed algorithm are generated by merging a set of clusters of level

$i-1$ that are closely reachable (Section 5). To measure the quality (goodness) of clusters, some benchmark cluster validation indices (internal) [7] are used. Section 6 presents the performance analysis of our proposed cycle-based clustering algorithm on some real datasets taken from the UC Irvine Machine Learning Repository (archive.ics.uci.edu/ml/index.php). Finally, we compare our proposed algorithm with some traditional benchmark clustering algorithms such as centroid-based clusterings and hierarchical clusterings [1, 7]. Our results indicate that the performance of our CA-based clustering technique is at least as good as the best known clustering algorithms that exist today. This research also discovers that our CA-based clustering scheme has the characteristic that the arrangement of clusters remains fixed even if the ordering of the features of a given set of objects is changed. This property is verified with real datasets.

2. Basics of Cellular Automata

In this paper, we use one-dimensional three-neighborhood n-cell CAs under null boundary conditions, where each cell takes any of the states $S = \{0, 1\}$. The next state of each cell is updated following an elementary CA (ECA) rule. The present state of all cells at a given time is called the *configuration* of the CA. The evolution of a CA is determined by a global transition function G such that $G : C \to C$ where $C = \{0, 1\}^n$ represents the configuration space. Hence, if the next configuration of $x = (x_i)_{\forall i \in n}$ is y, then $y = G(x)$ where $x, y \in C$, $y = (y_i)_{\forall i \in n}$ and x_i, y_i are the present and next state values of cell i, respectively. Therefore, $y_i = \mathcal{R}_i(x_{i-1}, x_i, x_{i+1})$ where $\mathcal{R}_i$ is the rule corresponding to cell i and x_{i-1}, x_i, x_{i+1} is the *neighborhood combination* for cell i. This neighborhood combination is called the *rule min term* (RMT) and is represented by its decimal equivalent $r = 2^2 \times x_{i-1} + 2^1 \times x_i + x_{i+1}$. A rule vector $\mathcal{R} = (\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{n-1})$ of length n is used to represent any arbitrary n-cell nonuniform CA, where $\mathcal{R}_i \neq \mathcal{R}_j$ for some i and j.

Table 1 presents the rules in tabular form. Obviously, there are $2^8 = 256$ distinct rules. These rules are traditionally named by their decimal equivalents. Because we are using null boundary conditions, $x_{-1} = x_n = 0$. Therefore, $y_0 = \mathcal{R}_0(0, x_0, x_1)$ for the first cell and $y_{n-1} = \mathcal{R}_{n-1}(x_{n-2}, x_{n-1}, 0)$ for the last cell. So, for each of these terminal cells, only $2^4 = 16$ distinct rules are considered as valid. For these rules, next state values for the present states $(1, x_0, x_1)$ and $(x_{n-2}, x_{n-1}, 1)$, respectively, are undefined and marked as invalid (i) (see, for example, the first row of Table 1).

Present State	111	110	101	100	011	010	001	000	Rule
(i) next state	i	i	i	i	1	0	0	1	9 ($\mathcal{R}_0$)
(ii) bext state	1	1	1	0	1	0	0	0	232 ($\mathcal{R}_1$)
(iii) next state	0	1	0	1	1	0	1	0	90 ($\mathcal{R}_2$)
(iv) next state	i	0	i	1	i	1	i	0	20 ($\mathcal{R}_3$)

Table 1. A four-cell CA (9, 232, 90, 20).

If $\mathcal{R}_i(x_{i-1}, x_i, x_{i+1}) = x_i$ in a CA, the corresponding RMT of rule $\mathcal{R}_i$ is called a *self-replicating* RMT. If all RMTs of a configuration x are self-replicating, then its next configuration y is identical to it and they form a cycle of length one. The number of self-replicating RMTs for a rule plays a major role in cycle formation, determining the number and lengths of cycles. Let $x, y \in C$ be two configurations of the CA G. Configuration y is reachable from configuration x if there exists an $l_1 \in \mathbb{N}$ such that $G^{l_1}(x) = y$; otherwise y is not reachable from x. Similarly, if x is also reachable from configuration y, then they are reachable from each other and they are in the same cycle. These configurations x and y are also called *cyclic configurations*. Let $C_i \subseteq C = \{0, 1\}^n$ be a set of configurations such that $G^l(x) = x$, $\forall\, x \in C_i$, where $l \in \mathbb{N}$ and $|C_i| = l$. Then, any $x \in C_i$ is cyclic and reachable from all configurations of C_i. This C_i that connects l configurations is called a *cyclic space* of the CA.

For instance, in Figure 1, configuration 1100 is reachable from configuration 1111 but not reachable from configuration 0111. Moreover, the configurations 1100, 1110 and 1111 are reachable from one another since they form a cyclic space of length three. A CA is called

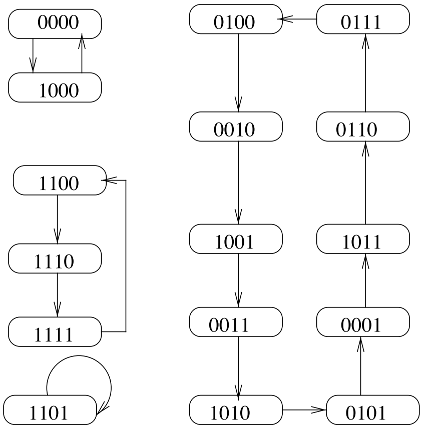

Figure 1. Transition diagram of the four-cell reversible CA (9, 232, 90, 20).

reversible if all configurations are part of some cyclic space. Figure 1 represents a reversible CA. Therefore, the configuration space of any reversible CA can be represented as a collection of cyclic spaces.

We use the methodology described in [8] to synthesize an n-cell reversible CA. For ease of reference, the table describing the class information of the participating rule $\mathcal{R}_i$, $0 \leq i \leq n-1$, is reproduced here as Table 2. The generation of an n-cell reversible CA is guided by the rule of cell i and the class information of the rule of cell $i+1$ (see [8] for more details). Example 1 illustrates the process of synthesis.

Example 1. Let us design a four-cell reversible CA. To select an arbitrary $\mathcal{R}_0$, the first column of Table 2(b) is taken into consideration. Select rule 9 as $\mathcal{R}_0$ from Table 2(b) (the third row and first column of Table 2(b)). Since the class information of the rule next to rule 9 is class III (the second column and third row of Table 2(b)), $\mathcal{R}_1$ can be any rule from the pool of CA rules in class III from Table 2(a). Select rule 232 as $\mathcal{R}_1$ (the first column of Table 2(a) for rule 232 is

Class of $\mathcal{R}_i$	$\mathcal{R}_i$	Class of $\mathcal{R}_{i+1}$
I	51, 204, 60, 195	I
	85, 90, 165, 170	II
	102, 105, 150, 153	III
	53, 58, 83, 92, 163, 172, 197, 202	IV
	54, 57, 99, 108, 147, 156, 198, 201	V
	86, 89, 101, 106, 149, 154, 166, 169	VI
II	15, 30, 45, 60, 75, 90, 105, 120, 135, 150, 165, 180, 195, 210, 225, 240	I
III	51, 204, 15, 240	I
	85, 105, 150, 170	II
	90, 102, 153, 165	III
	23, 43, 77, 113, 142, 178, 212, 232	IV
	27, 39, 78, 114, 141, 177, 216, 228	V
	86, 89, 101, 106, 149, 154, 166, 169	VI
IV	60, 195	I
	90, 165	IV
	105, 150	V
V	51, 204	I
	85, 170	II
	102, 153	III
	86, 89, 90, 101, 105, 106, 149, 150, 154, 165, 166, 169	VI
VI	15, 240	I
	105, 150	IV
	90, 165	V

Table 2. (a) Class relationship of $\mathcal{R}_i$ and $\mathcal{R}_{i+1}$.

Rules for $\mathcal{R}_0$	Class of $\mathcal{R}_1$
3, 12	I
5, 10	II
6, 9	III

(b) First rule table.

Rule Class for $\mathcal{R}_{n-1}$	Rule Set for $\mathcal{R}_{n-1}$
I	17, 20, 65, 68
II	5, 20, 65, 80
III	5,17, 68, 80
IV	20, 65
V	17, 68
VI	5, 80

(c) Last rule table.

Table 2. Rules to generate a reversible CA.

class III). Now, the third column and the row corresponding to rule 232 in Table 2(a) is class IV; therefore, $\mathcal{R}_2$ is to be selected from class IV in the first column of Table 2(a). By repeating the same process, we choose rule 90 as $\mathcal{R}_2$. Therefore, rule $\mathcal{R}_3$ is from class IV. However, because this is the last rule, we need to select this rule from Table 2(c). Let $\mathcal{R}_3$ be 20 (second column and fourth row of Table 2(c)). Therefore, the reversible CA is (9, 232, 90, 20) (see Figure 1).

Next, our aim is to explain the significance of reversible CAs as a clustering tool. First, we focus on how the cyclic spaces of reversible CAs can be used as natural clusters. Next, we figure out the significant CA rules for effective clustering. We also observe that two different CAs of the same size can identically cluster a given dataset.

3. The Mapping between Cellular Automata and Clustering

We discussed earlier that clustering can also be viewed as a bijective mapping where a set of target objects is exclusively in a unique cluster if the objects are close. Since our focus is on using CAs for clustering, each target object first needs to be mapped to a configuration. To do that, data discretization should be done effectively.

3.1 The Encoding Technique

Let $\mathbb{X} = \{X_1, X_2, \ldots, X_k\}$ be the set of target objects that are to be distributed among m clusters and $A_1, A_2, \ldots, A_p$ are p distinct attributes (features) of the objects, where each A_j represents a finite set of t values. Depending on these values, the corresponding attribute is called *quantitative* or *qualitative*. Since we use binary CAs as our tool, each object (X) is converted into a binary string x (where

$x \in \{0, 1\}^n$ and $n \in \mathbb{N}$). Now, let $X = (x^1 x^2 \ldots x^p)$ where $x^j \in A_j$. To maintain a smaller intracluster distance based on the feature space, the configurations having more similarity in their features' space (attributes' values) should be reachable from each other. The Hamming distance [1] is taken into consideration for measuring the similarity. Next, we show the role of Hamming distance in designing the encoded target objects.

If A_j is a quantitative attribute, the range can be partitioned into v closed intervals such that A_j can be represented as an ordered (either ascending or descending) list of elements. Frequency-based encoding [9] is used for each A_j; therefore, each interval can be encoded as a binary string of length $v - 1$ to maintain a minimum Hamming distance. Let the first interval be represented as a binary string such that 0^{v-1}; the next interval is represented by $0^{v-2}1$. Obviously, the Hamming distance between 0^{v-1} and $0^{v-2}1$ is one and the values represented by the first interval are closest to the second interval with respect to other intervals. To maintain such a feature in the representation of the encoded objects, the encoding function M can be designed as

$$M(a) = 0^{v-i} 1^{i-1} \tag{1}$$

where $a \in [a_{i_1}, a_{i_k}]$ and $[a_{i_1}, a_{i_k}]$ is the ith interval of A_j. In this paper, for each quantitative feature A_j, v is set to three; therefore, two bits are needed to encode any value from A_j. Using equation (1), the three intervals are represented as 00, 01 and 11.

By considering A_j as a qualitative attribute, $M(a)$ ($a \in A_j$) can be redesigned in the following manner: let $A_j = \{a_1, a_2, \ldots, a_u\}$, then each element of A_j is represented as a binary string of length u where there is only one 1 at some unique position. Let a be the ith element of A_j that has the value 1; then $M(a)$ can be represented using

$$M(a) = 0^{i-1} 1 0^{u-i}. \tag{2}$$

So the Hamming distance between any pair of elements for such a qualitative A_j is always fixed, that is, two.

Using the encoding techniques of quantitative and qualitative attributes, a target object with p features (where p_{q_n} and p_{q_l} are the count of quantitative and qualitative attributes, respectively [2]) is mapped to a configuration of an n-cell CA. Here, n can be computed as

$$n = \left((v-1) * p_{q_n}\right) + \left(u_1 + u_2 + \cdots + u_{p_{q_l}}\right)$$

where $|A_i| = u_i$ ($\forall\ i$) and A_i is a qualitative attribute. Assume that $v = 3$. In this way, $\mathbb{X}$ is mapped to a set of useful configurations for clustering that is a proper subset of C. Let C denote the set of useful configurations and $M : \mathbb{X} \to C$ such that M is surjective. Therefore, $|C| < |\mathbb{X}|$ and the clustering operation is actually performed on a smaller number of objects. Moreover, the maximum count of C can also be computed. If all the attributes are quantitative, then $n = (v-1) * p$; so $|C| \leq v^p$. Otherwise, $|C| \leq \left(v^{p_{q_n}} * \prod_{i=1}^{p_{q_l}} u_i\right)$.

Example 2. Take a hypothetical set of books, where each book is identified by three attributes: number of pages, ratings by reviewers and type of binding. The first two attributes are continuous, whereas the last one is categorical. Table 3 shows the detailed encoding scheme for 10 such objects into CA configurations. Here, the categorical (qualitative) attribute values are encoded as 01 (hard) or 10 (soft). The continuous (quantitative) attribute values are divided into three subintervals to be represented by 00, 01 and 11, respectively. For example, ratings values are divided into subintervals $[2.6, 4]$, $[4.5, 7]$ and $[8, 9.5]$ depicted by 00, 01 and 11, respectively. Therefore, each object is mapped to a six-bit string that can be shown as a configuration of a six-cell CA.

	Continous Attribute				Categorical Attribute		
Object ID	Number of Pages	Encoding	Ratings	Encoding	Binding Type	Encoding	Encoded CA
1	300	01	9	11	hard	01	011101
2	325	11	8	11	soft	10	111110
3	40	00	9.5	11	soft	10	001110
4	200	01	4	00	hard	01	010001
5	129	01	4.5	01	soft	10	010110
6	65	00	7	01	hard	01	000101
7	319	11	6.8	01	soft	10	110110
8	110	00	3	00	soft	10	000010
9	400	11	2.6	00	soft	10	110010
10	350	11	9.3	11	soft	10	111110

Table 3. Encoding a set of hypothetical books into CA configurations.

Example 2 mainly depicts the data encoding technique. Let $\mathbb{X}$ be the set of target elements such that $C_i \cap C_j = \emptyset$ ($\forall\ i, j$ and $i \neq j$). Now, each C_i can be formed using a unique cyclic space of a CA. Next, we present how the inherent cyclic structure of CAs can connect close objects naturally into the same cluster.

3.2 Natural Clusters: Cycles of Reversible Cellular Automata

Since reversible CAs can also be represented as a bijective mapping, the closeness property of clustering can be defined by the reachability property of reversible CAs: two configurations x and y are said to be *close* if $x, y \in C_i \subset \mathbb{X}$ such that $G^{l_1}(x) = y$ and $G^{l_2}(y) = x$ for some l_1, l_2. Similarly, if z is not reachable from x, then obviously, x is also not reachable from z and they are also not close to each other because $x \in C_i$ and $z \in C_j$ where $i \neq j$. In this way, a CA can place a set of close objects (configurations) into one cluster (cycle) that is not reachable from the remaining objects (configurations). Thus, the cyclic spaces of CAs can be used as natural clusters.

Example 3. Consider clustering the following data objects (configurations): 1100, 1110, 0100, 0110 and 0101. Each object is represented by a four-bit binary string, so a binary CA of size four can act as a clustering tool. We will use a four-bit CA (9, 232, 90, 20) for this purpose. This CA distributes the objects among two clusters where the first one represents $\{1100, 1110\}$ because 1100 is reachable from 1110 and 1110 is also reachable from 1100. Similarly, the second cluster is formed as $\{0100, 0110, 0101\}$.

However, by changing the CA, the same data objects can be clustered into different arrangements—the distribution of the target objects as well as the number of clusters can be changed. For instance, use the CA (9, 23, 90, 20) for clustering the same set of objects $\{1100, 1110, 0100, 0110, 0101\}$. The only change in the latter CA compared to the former is that $\mathcal{R}_1$ is changed to 23 from 232. Figure 2 explains how the clustering is done using CA (9, 23, 90, 20). In this CA, 0101 is separated from the remaining $2^n - 1$ configurations, which form another cycle. Therefore, the target objects are distributed into two clusters in the following way: the first cluster connects 1100, 1110, 0100 and 0110, whereas the second is formed by only 0101. This shows that, even though both CAs make a minimal change in the rule vector and form two clusters for the given objects, the arrangement is different. For the CA (9, 232, 90, 20), 0100 and 0110 are in the second cluster and it is separated from the other cluster consisting of $\{1100, 1110\}$, but using the CA (9, 23, 90, 20) all of these objects—1100, 1110, 0100 and 0110 are in the same cluster. Because a CA can be viewed as a unique bijective function, the connectivity among the configurations can be changed even if a small change occurs (at least one participating rule changes) in the given rule sequence of the CA. For example, let $x = x_0 x_1 \ldots x_{i-1} x_i x_{i+1} \ldots x_{n-1}$ be a configuration of an n-cell CA G_1 such that $G_1(x) = y_1$. Let $\mathcal{R}_i$ be the rule at cell i of the given CA such that $\mathcal{R}_i(x_{i-1} x_i x_{i+1}) = x$.

Consider another CA G_2 of the same size n and let $\mathcal{R}'_i$ be the i^{th} rule of G_2. If $\mathcal{R}'_i(x_{i-1}x_i x_{i+1}) = x' \neq x$, then $y_1 \neq y_2$ where $G_2(x) = y_2$. Definitely, y_1 is reachable from x in one step using G_1, whereas y_1 is not reachable from x in one step if we use G_2. However, it may be reached from x if $G_2^k(x) = y_1$ for some $k \in \mathbb{N}$. That is, x, y_1 and y_2 can be part of the same cyclic space. Since all configurations inside the same cyclic space form the same cluster, similar clustering may be formed even if different CAs are used.

Let G_1 and G_2 be two CAs and $\mathbb{X}$ be the set of target objects such that $\mathbb{X} \subset C$. Let the clustering done by G_1 and G_2 be $\mathbb{X} = C_1^1 \cup C_2^1 \cup \cdots \cup C_m^1$ and $\mathbb{X} = C_1^2 \cup C_2^2 \cup \cdots \cup C_m^2$, respectively, where C_i^j and $C_{i'}^j$ are disjoint sets for any $i \neq i'$. Now, two clusters C_i^1 and C_j^2 are called *identical clusters* if $C_i^1 = C_j^2$. If for each C_i^1 $(1 \leq i \leq m)$, there is an identical cluster C_j^2, then G_1 and G_2 are called *equivalent clustering CAs* with respect to the set of target objects $\mathbb{X}$.

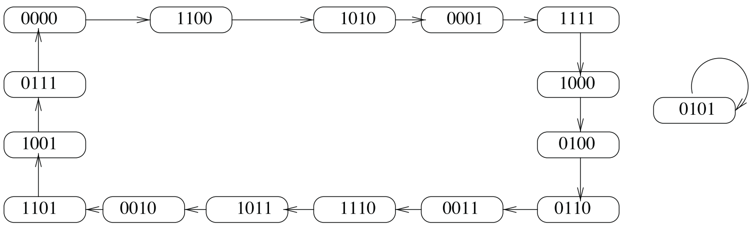

Figure 2. Transition diagram of the four-cell reversible CA (9, 23, 90, 20).

Example 4. Consider clustering the following data objects (configurations): 0100, 1001, 0001, 1011 and 0101. For G_1 use the four-bit CA (9, 232, 90, 20) where $C_1^1 = \{0100, 1001, 0001, 1011\}$ and $C_2^1 = \{0101\}$. For G_2 use the four-bit CA (9, 23, 90, 20) to get the clusters $C_1^2 = \{0100, 1001, 0001, 1011\}$ and $C_2^2 = \{0101\}$. Since $C_1^1 = C_1^2$ and $C_2^1 = C_2^2$, each pair represents identical clusters and the two CAs (9, 232, 90, 20) and (9, 23, 90, 20) are said to be equivalent clustering CAs with respect to the given set of data objects $\{0100, 1001, 0001, 1011, 0101\}$.

It is clear from the discussion that the same set of objects can be distributed among clusters in multiple ways even if different CAs are used—sometimes, in a similar arrangement among the clusters; sometimes, in the same number of clusters but different arrangements. It can also be noted that the objects can be distributed among different numbers of clusters using dissimilar arrangements. So, out of all those possible arrangements, we need to figure out which is the most appropriate reversible CA for a given dataset. Such a CA contributes to effective clusters. In the next section, the goal is to figure out the significant CA rules for designing effective clustering.

4. Clustering Using Reversible Cellular Automata

A CA is significant for effective clustering if it maintains the following necessary conditions: (*i*) all cells follow CA rules that contribute a reasonable number of self-replicating RMTs and (*ii*) the number of cycles is limited. First, the target is to identify CA rules that help reduce the intracycle distance (Hamming distance) among the configurations based on feature space by maintaining a reasonable number of self-replicating RMTs.

4.1 Cellular Automata Rules That Maintain Minimum Intracluster Distance

For our present objective, we need to select CA rules that make minimal changes during state transition. Ideally, it is desired that $f_i(x_{i-1}, x_i, x_{i+1}) = x_i$ for any value of i and any combination of $x_{i-1}x_i x_{i+1}$. The intrinsic property of such an n-cell CA is that each cell follows a special rule where all RMTs are self replicating—this is rule 204. However, an n-cell CA with only rule 204 is not effective because the number of clusters is 2^n where each object forms a unique cluster. Therefore, we want to rank a rule based on the number of its self-replicating RMTs when used in designing the rule vector of a reversible CA. Our objective is to determine significant rules and the proportion of high-ranked rules for designing clusters of objects that maintain smaller intracluster distances. To detect such rules, we rank each rule in Table 2. This rank determines how a rule can act as an influencing factor for designing a cluster with more similar (less Hamming distance) data.

Because balanced rules are used for reversible CAs, each rule follows an even number of self-replicating RMTs (0, 2, 4, 6 and 8). We rank these CA rules into five categories based on the contribution of self-replicating RMTs (the contribution is measured by the ratio of the number of self-replicating RMTs with the total number of RMTs). Table 4 presents the rank of rules depending on the number of self-replicating RMTs. Column 1 of Table 4 refers to the rank of the corresponding rule. We can see that rule 204 is ranked first, but it was mentioned earlier that if rule 204 is applied to every cell of an n-cell CA, then each target object belongs to a unique cycle. This also means that rule 204 distributes similar objects into different clusters, which is not desirable. Moreover, rule 51 is the least significant rule for clustering, with a rank of 5. If rule 51 is applied at every cell of a CA of size n, then each cluster considers only a pair of configurations with Hamming distance n. Therefore, we need to select rule 204 for as many cells as possible and the opposite strategy should be used for rule 51. However, CAs with only rules 204 and 51 are not effective for clustering. Hence, to design clusters of objects with a smaller intra-cluster distance, we take the following strategy of choosing rules for synthesizing a reversible CA:

1. Discard all rules with ranks 4 and 5 (i.e., less than four self-replicating RMTs) from Table 2. Seventeen rules are discarded by this condition: (51, 53, 58, 83, 163, 54, 57, 99, 147, 23, 43, 113, 178, 27, 39, 114, 177) reducing the rule space to 45.
2. For the $n-2$ nonterminal cell positions (cell 1 to cell $n-2$), at most 50% of rules with rank 2 (six self-replicating RMTs) are to be selected.

Rank	$\mathcal{R}_i$	Cell Position ($i \in$)	Contribution of Self-Replicating RMTs
1	12 204 68	0 $[1, n-2]$ $n-1$	100%
2	92, 172, 197, 202, 108, 156, 198, 201, 77, 142, 212, 232, 78, 141, 216, 228	$[1, n-2]$	75%
3	6, 9, 5, 10 86, 89, 101, 106, 149, 154, 166, 169, 30, 45, 75, 120, 135, 180, 210, 225, 90, 105, 150, 165, 85, 170, 102, 153, 60, 195, 15, 240 5, 20, 65, 80	0 $[1, n-2]$ $n-1$	50%
4	53, 58, 83, 163, 54, 57, 99, 147, 23, 43, 113, 178, 27, 39, 114, 177	$[1, n-2]$	25%
5	3 51 17	0 $[1, n-2]$ $n-1$	0%

Table 4. Ranking of CA rules based on the number of self-replicating RMTs.

Example 5 explains how this can be effective for maintaining a smaller intracluster distance.

Example 5. Consider a hypothetical dataset where each object is identified by three quantitative features A_1, A_2 and A_3. Since $v = 3$, $n = 6$. Set the total number of target objects to 50. After encoding, let those 50 objects be mapped to 10 useful configurations and let the configurations be: 111100, 111111, 111101, 011011, 011000, 011001, 001100, 001101, 001011 and 001001. A six-bit CA (6, 232, 60, 197, 105, 17) is designed using the given strategy to perform clustering on those 10 configurations. Based on this CA, these 10 configurations belong to six different cycles that result in the target objects being distributed over six clusters. The clustering is done in the given manner: cluster 1 contains 111100 and 111111, whereas cluster 2 keeps the configuration 111101; similarly, clusters 3 and 4 present the configurations $\{011\,011, 011\,000\}$ and $\{011001\}$, respectively. Cluster 5 refers to $\{001100, 001101\}$ and cluster 6 has the configurations 001011 and 001001. Out of these six clusters, four of them store two configurations in each cluster and each of the remaining two clusters contains only one configuration. We can observe that each cluster includes configurations where state values differ only at two positions—cells 5 and 6, meaning that target objects have close values with respect to features 1 and 2. In this way, a smaller intracluster distance is maintained in the resultant clustering.

Example 5 shows that the above-mentioned scheme can guarantee distributing similar objects into the same cluster but it cannot restrict the number of clusters (cycles). Next, we focus on the design of CAs with a limited number of cycles.

4.2 Designing Cellular Automata with Optimal Number of Clusters

It is obvious from the discussion so far that the consecutive configurations of a cycle maintain the minimum distance in feature space if more significant rules are used in an n-cell CA. That is, the same cycle connects similar objects. However, it may increase the number of cycles. So, there is a tradeoff between these two aspects of the clustering technique—maintaining a smaller number of cycles (clusters) and a smaller intracluster distance among the objects; that is, configurations with smaller Hamming distances are in the same cycle. In this section, we discuss a technique to generate CAs with a limited number of cycles; that is, more configurations can be placed on the same cycle. This requirement matches the problem of generating CAs with a large number of cycles, as studied in [6]. For ease of understanding, we briefly recall the idea.

A CA is expected to have a large cycle length if its rules depend on both the left and right neighbors. To measure this dependence, a parameter P, called the *degree of dependence on both of the neighbors*, is defined to determine how much a cell depends on its neighbors for updating its state. For a rule $\mathcal{R}_i$, $P(\mathcal{R}_i) = P_r(\mathcal{R}_i) * P_l(\mathcal{R}_i)$. Here, $P_r(\mathcal{R}_i)$ (resp. $P_l(\mathcal{R}_i)$) is the *degree of right (resp. left) dependence*, defined as the ratio of the number of combinations of values of x_i and x_{i-1} (resp. x_{i+1}) for which the next state function on x_i depends on x_{i-1} (resp. x_{i+1}). Evidently, $P(\mathcal{R}_i)$ can take values 0, 0.25, 0.5 or 1. Based on these values, the rules of reversible CAs are classified into four categories: completely dependent, partially dependent ($P = 0.5$), weakly dependent ($P = 0.25$), and independent ($P = 0$) (see Table 5). It is observed that in a CA with large cycle lengths, a majority of the participating rules are from the completely dependent category, some are from the partially dependent category and a few are from the weakly dependent category, whereas none are from the independent category. See [6] for a more detailed discussion.

Category	$\mathcal{R}_i$
completely dependent	90, 165, 150, 105
partially dependent	30, 45, 75, 120, 135, 180, 210, 225, 86, 89, 101, 106, 149, 154, 166, 169
weakly dependent	92, 172, 197, 202, 108, 156, 198, 201, 77, 142, 212, 232, 78, 141, 216, 228, 53, 58, 83, 163, 54, 57, 99, 147, 23, 43, 113, 178, 27, 39, 114, 177
independent	51, 85, 170, 102, 153, 60, 195, 15, 240, 204

(a) Categories of reversible CA rules.

Category	$\mathcal{R}_0$	$\mathcal{R}_{n-1}$
completely dependent	5, 6, 9, 10	5, 20, 65,80
independent	3, 12	17, 68

(b) Categories of $\mathcal{R}_0$ and $\mathcal{R}_{n-1}$.

Table 5. Categories of reversible CA rules on the parameter P.

Obviously, following the strategy mentioned in Section 4.1, 16 rules from the weakly dependent category and all 10 rules from the independent category are rejected for clustering purposes because they produce more pairs of configurations with high Hamming distances. So the remaining 36 (for cell 1 to $n-2$) rules are significant for designing CAs that result in effective clustering. These are the 16 rules of the weakly dependent category with rank 2 and the rules of completely dependent and partially dependent categories with rank 3. In the next section we proceed to our clustering technique.

4.3 Cycle-Based Clustering

As already discussed, the clusters are nothing but the cycles of CAs, so we call our CA-based clustering technique *cycle-based* clustering. Such clustering not only maintains the cycles of the configurations with smaller Hamming distances, but also ensures that the number of cycles in the CA does not grow exponentially with CA size. Moreover, the synthesis scheme of generating significant CAs also works with a reduced ruleset. The steps of our proposed clustering technique using such a CA are stated as follows:

- Step 1. Based on the types of attributes (equation (1) for quantitative and equation (2) for qualitative), the target objects of $\mathbb{X}$ are encoded into the set of n-bit useful configurations that is denoted by $\mathcal{C}$.
- Step 2. Randomly choose an arbitrary CA $\mathcal{R}$ of size n that maintains rules at all cells from a set of rank 3 that are either completely dependent or partially dependent and at most one rule from the weakly dependent category with rank 2.
- Step 3. Let the set of remaining objects (configurations) to be clustered be $\mathcal{C}'$. Initially, $\mathcal{C}' = \mathcal{C}$. Set $k = 1$.
- Step 4. Let C_k represent those configurations that are close to x_i such that $C_k \subset \mathcal{C}'$ and $x_i \in \mathcal{C}'$. Set $C_k = C_k \cup x_i$ and $\mathcal{C}' = \mathcal{C}' \setminus C_k$. Increment k by 1.
- Step 5. Continue step 4 until all the configurations are clustered such that $\mathcal{C}' = \emptyset$ and $C_1 \cap C_2 \cap \cdots \cap C_m = \emptyset$, where m is the number of clusters.

We illustrate the process using an example.

Example 6. Consider the Iris dataset from UCI Machine Learning repository (archive.ics.uci.edu/ml/index.php, see Table 6) where $\mathbb{X} = \{X_1, X_2, \ldots, X_{150}\}$; that is, $|\mathbb{X}| = 150$. There are four quantitative attributes p_{q_n}; therefore, n can be computed as $n = (2 * p_{q_n}) = 2 * 4 = 8$ and using the encoding function (mentioned in equation (1)), each target object is mapped to a useful configuration and ultimately, these 150 are represented by only 24 configurations. Here, C can be represented as $\{x_1, x_2, \ldots, x_{24}\}$, where,

$x_1 = 00110000, x_2 = 10111011, x_3 = 00000000, x_4 = 11001110,$
$x_5 = 10110000, x_6 = 10101111, x_7 = 00100000, x_8 = 11111011,$
$x_9 = 11001010, x_{10} = 10111111, x_{11} = 10001011,$
$x_{12} = 10001110, x_{13} = 00001010, x_{14} = 10001010,$
$x_{15} = 11101010, x_{16} = 11001111, x_{17} = 10101010,$
$x_{18} = 11111111, x_{19} = 11111010, x_{20} = 11101110,$
$x_{21} = 11101111, x_{22} = 10101011, x_{23} = 10001111$ and
$x_{24} = 00001011.$

Name	p	p_{q_n}	p_{q_l}	Target Objects	CA Size (n)	Objects with Missing Terms
Iris	4	4	0	150	8	0
User Knowledge Modeling	5	5	0	403	10	0
BuddyMove	6	6	0	249	12	0
Seed	7	7	0	210	14	0
StoneFlakes	8	8	0	79	16	6
Heart failure clinical records	12	12	0	299	19	0
Wholesale Customers	8	6	2	440	16	0

Table 6. Description of real datasets used for the proposed CA-based clustering technique.

Consider $\mathcal{R}$ as (10, 45, 156, 86, 90, 165, 150, 65) by following the strategy mentioned in step 2. Initially, $C' = C$ and let $m = 1$. Let $x_1 = 00\,110\,000 \in C'$ be taken to figure out which remaining configurations from C' are close to x_1. Out of 23 configurations, only configuration 00100000 is close to x_1. Therefore, $C_1 = \{x_1, x_7\}$ and C' is updated to $C' \backslash C_1$ such that $|C'| = 22$. Next, $x_2 = 10111011$ is chosen from C'; we can see that the configurations 00000000, 00001010, 10101111 and 10001011 are close to x_2. Therefore, cluster C_2 consists of five configurations $\{x_2, x_3, x_{13}, x_6, x_{11}\}$ and C' is updated to $C' \backslash C_2$ such that $|C'| = 17$. In the same way, the remaining 17 configurations of C' are distributed among two clusters such that one cluster is $C_3 = \{x_{19}, x_9, x_{15}, x_{17}, x_4, x_{21}, x_{23}, x_{18}, x_{24}, x_{16}, x_{12}, x_{22}, x_{10}\}$ and another is $C_4 = \{x_5, x_8, x_{14}, x_{20}\}$. Therefore, these 150 target objects represented by 24 configurations are distributed among four clusters by our algorithm.

Example 6 confirms that our clustering technique not only maintains a smaller intracluster distance, but it is also capable of distributing the target objects among a limited number of clusters. Therefore, the cycle-based clustering is an effective technique for clustering. However, sometimes, there is a requirement that the target objects be distributed among a desired number of clusters. To deal with that issue, we modify our proposed technique by introducing the use of multiple CAs; the revised technique is an iterative level-wise approach that uses multiple levels and cycle-based clustering at every level.

5. Iterative Cycle-Based Clustering for a Desired Number of Clusters

It has already been established that CAs can perform effective clustering using their cyclic space, but sometimes it is difficult to find a CA that can distribute the objects into a desired number of clusters. To solve this problem, we opt for a level-wise iterative clustering technique where at every level the number of clusters is reduced to reach either the desired number of clusters or the optimal number of clusters for the given dataset. So, we reformulate our technique for clustering using the candidate CAs. It was already reported that any arbitrary CA is not acceptable as a candidate; the participating CAs must follow the conditions discussed in Sections 4.1 and 4.2:

- Property 1. Participating CAs of size n maintain rules at all cells from a subset of rank 3 and at most one rule from rank 2.
- Property 2. Our technique converges by merging the clusters. To do that, a hierarchy of levels has to be maintained.
- Property 3. Only closely reachable clusters of level $i-1$ are to be merged to generate the updated clusters of level i.

Let $M(\mathbb{X})$ be the set of encoded target objects where $M(\mathbb{X}) \subset C$ and $\mathbb{X} = \{X_1, X_2, \ldots, X_k\}$ is the set of target objects. For any particular ordering of the features, these encoded target objects are a set of useful configurations C. Let $|M(\mathbb{X})| = |C| = k'$ where $k' \leq k$. At any level, a useful configuration $x \in C$ is a member of a distinct cluster c (a set of encoded target objects) such that $|\bigcup c| = |C|$.

Let m_i be the number of primary clusters at level i. For level 0, the primary clusters are $c_1^0, c_2^0, \ldots, c_{m_0}^0$, where each cluster is a singleton set. Therefore, $k' = m_0$. In general, for any level i, the primary clusters are $c_1^i, c_2^i, \ldots, c_{m_i}^i$. To form these primary clusters of level i from level $i-1$, a CA of size n is selected uniformly random without replacement from a pool of candidate CAs maintaining Property 1. This process is maintained at every level i. Such a CA is named an *auxiliary* CA. This CA plays a major role in clustering. First, we need to compute the number of auxiliary clusters of such a CA, which the target configurations k' strictly belong to.

Definition 1. Let x be a useful configuration and $G : C \mapsto C$ be an auxiliary CA. If $x \in C_j$ where $C_j \subset C$ is a cyclic space of G, then x strictly belongs to the auxiliary cluster C_j.

Let the useful configurations strictly belong to m' number of auxiliary clusters $C_1^i, C_2^i, \ldots, C_{m'}^i$ of level i. Our second step is to follow Property 2, that is, merge the primary clusters $c_1^{i-1}, c_2^{i-1}, \ldots, c_{m_0}^{i-1}$ of level $i-1$ using these auxiliary clusters to get the resultant primary clusters of level i where $m_i \leq m_{i-1}$. However, these clusters cannot be merged arbitrarily; a pair of primary clusters can be merged depending on their degree of membership of participation.

Definition 2. Let c_j^{i-1} be a primary cluster of level $i-1$ where $\left|c_j^{i-1}\right| = v_j$. Let C_t^i be an auxiliary cluster of level i. The degree of membership of participation of c_j^{i-1} in C_t^i, denoted by $\mu\left(C_t^i, c_j^{i-1}\right)$, is defined as the availability of configurations of c_j^{i-1} in C_t^i. It is computed as $v_j' \,/\, v_j$ where v_j' refers to the count of useful configurations from primary cluster c_j^{i-1} in auxiliary cluster C_t^i.

The configurations of c_j^{i-1} can strictly belong to more than one auxiliary cluster. Similarly, C_t^i can possess useful configurations from different clusters of level $i-1$. Let c_l^{i-1} and c_j^{i-1} be two primary clusters of level $i-1$. These two clusters may be merged if they are necessarily closely reachable (Property 3).

Definition 3. Let c_j^{i-1}, c_l^{i-1} and c_s^{i-1} be the clusters whose members strictly belong to C_t^i. Now, clusters c_j^{i-1} and c_l^{i-1} are said to be closely reachable in C_t^i if

$$\left|\left(\mu\left(C_t^i, c_j^{i-1}\right) - \mu\left(C_t^i, c_l^{i-1}\right)\right)\right| < \left|\left(\mu\left(C_t^i, c_j^{i-1}\right) - \left(\mu\left(C_t^i, c_s^{i-1}\right)\right)\right|.$$

Therefore, for every C_t^i, we can get pairs of closely reachable clusters. The degree of participation plays a vital role for selecting the closely reachable clusters, which are then merged. Next, we discuss the algorithm in detail.

1. Let $c_1^0, c_2^0, \ldots, c_{m_0}^0$ (resp. $c_1^{i-1}, c_2^{i-1}, \ldots, c_{m_{i-1}}^{i-1}$) be the primary clusters of level 0 (resp. $i-1$) where the count of clusters is m_0 (resp. m_{i-1}). Also let $C_1^1, C_2^1, \ldots, C_{m'}^1$ (resp. $C_1^i, C_2^i, \ldots, C_{m'}^i$) be the auxiliary clusters of level 1 (resp. i) where the count of auxiliary clusters is m'. For all t, $1 \leq t \leq m'$, compute $\mu(C_t^1, c_j^0)$ (resp. $\mu(C_t^i, c_j^{i-1})$). For any given c_j^0 (resp. c_j^{i-1}), find the auxiliary cluster of level 1 (resp. i) in which it has maximum participation, that is, its degree of participation is maximum. Obviously, for some value of t, maximum participation of c_j^0 (resp. c_j^{i-1}) is in C_t^1 (resp. C_t^i).

2. Let $C_{t_1}^1$ (resp. $C_{t_1}^i$) be the auxiliary cluster having maximum configurations belonging in c_j^0 (resp. c_j^{i-1}). Therefore, c_j^0 (resp. c_j^{i-1}) can merge with some of the clusters that have also participated in $C_{t_1}^1$ (resp. $C_{t_1}^i$). However, only those clusters are to be merged with c_j^0 (resp. c_j^{i-1}) that are closely reachable to c_j^0 (resp. c_j^{i-1}). Hence, a new primary cluster c_j^1 (resp. c_j^i) is formed as $c_j^i = c_j^{i-1} \cup c_l^{i-1}$ if and only if $\left|\left(\mu\left(C_{t_1}^i, c_j^{i-1}\right) - \mu\left(C_{t_1}^i, c_l^{i-1}\right)\right)\right| < \left|\left(\mu\left(C_{t_1}^i, c_j^{i-1}\right) - \mu\left(C_{t_1}^i, c_s^{i-1}\right)\right)\right|$, for any $s \neq l$ where c_s^{i-1} is another participating cluster in $C_{t_1}^i$ and

$\max\{\mu(C_t^i, c_j^{i-1})_{(\forall t)}\} = \mu(C_{t_1}^i, c_j^{i-1})$. Therefore, the newly generated cluster c_j^i constitutes a set of useful configurations, out of which some strictly belong to a cluster (cycle) of the auxiliary CA of level i.

3. If for any primary cluster c_j^0 (resp. c_j^{i-1}) there is no closely reachable primary cluster in all auxiliary clusters, then the new primary cluster of level i is $c_j^i = c_j^{i-1}$. Therefore, $m_i \leq m_{i-1}$.

4. The algorithm stops when we reach the optimal number of clusters (m). The test of optimality is determined either by arriving at the desired number of clusters given by the user or if $m_i = m_{i-1}$ after a fixed number of attempts.

```
Input:  A set of target objects $\mathbb{X} = \{X_1, X_2, \ldots, X_k\}$, number of quantitative
        and qualitative attributes $p_{q_n}$ and $p_{q_l}$ respectively, optimal number
        of clusters ($m$) and an auxiliary CA space
Output: The clusters $\{c_1^v, c_2^v, \ldots, c_m^v\}$

Step 1. Set $n \leftarrow (2 * p_{q_n}) + (u_1 + u_2 + \cdots + u_{p_{q_l}})$ ;
        foreach $j = 1$ to $k$ do Encode $X_j$ into an $n$-bit binary string ;
        Let $M(\mathbb{X})$ be the set of encoded target objects $\{x_1, x_2, \ldots, x_{k'}\}$ where
          $|M(\mathbb{X})| = k'$ ;
Step 2. Construct a set of $n$-cell CAs $R$ from the given auxiliary CA space ;
Step 3. Set $m_0 \leftarrow k'$, $i \leftarrow 1$ and $z \leftarrow 1$ ;
        for $j = 1$ to $m_0$ do
              Set $c_j^0 \leftarrow \{x_j\}$ ; // Initialize primary clusters of level 0
Step 4. while $(m_i \neq m_{i-1}) \,||\, (m_i \neq m)$ do
        Select $\mathcal{R} \in R$ and Set $R \leftarrow R \setminus \{\mathcal{R}\}$ // Auxiliary CA is selected randomly
          at uniform without replacement
        Generate auxiliary clusters $C_1^i, C_2^i, \ldots, C_{m'}^i$ for the CA $\mathcal{R}$ ;
        Initialize a matrix $A[a_{tj}]_{m' \times m_{i-1}}$ to 0 ;
        for $t = 1$ to $m'$ do
              for $j = 1$ to $m_{i-1}$ do Set $a_{tj} \leftarrow \mu(C_t^i, c_j^{i-1})$
        foreach $j = 1$ to $m_{i-1}$ do
              // For each of the primary clusters of previous level
              Let $a_{t'j}$ = maximum of $a_{tj}$ where $1 \leq t \leq m'$; // Find the
                    auxiliary cluster with maximum participation of $c_j^{i-1}$
              for $(j_1 = 1$ to $m_{i-1})$ && $(j_1 \neq j)$ do
                    Find $a_{t'j'}$ = maximum of $a_{tj_1}$ such that $a_{t'j'} \neq 0$;
              if no such $a_{t'j'}$ exists then continue ;
              else
                    Set $c_z^i \leftarrow c_{j^{i-1}} \cup c_{j'}^{i-1}$ and $z \leftarrow z + 1$ ;
                    Mark $c_j^{i-1}$ and $c_{j'}^{i-1}$ as modified ;
              Remove row $t'$ from $A$ ;
```

```
    foreach unmodified clusters $c_y^{i-1}$ do
        Set $c_z^i \leftarrow c_y^{i-1}$ and $z \leftarrow z+1$ ; // move the unmodified primary
            cluster(s) of previous level $i-1$ to get a new primary clus-
            ter of level $i$ and update cluster number
    Set $m_i \leftarrow z$ and $i \leftarrow i+1$ ;
Step 5. Report $c_1^i, c_2^i, \ldots, c_{m_i}^i$ as the final clusters at level $i$ and
    Exit ;
```

Algorithm 1. Iterative level-wise cycle-based clustering.

Example 7. Consider the Iris dataset (archive.ics.uci.edu/ml/index.php, see Table 6) where $\mathbb{X} = \{X_1, X_2, \ldots, X_{150}\}$ and each object has four quantitative p_{q_n} and no qualitative attributes p_{q_l}. Hence, the size of the CA for this dataset is $n = 2 * 4 = 8$. Now, let the desired number of clusters m be two. Using the encoding technique, we get $M(\mathbb{X}) = \{x_1, x_2, \ldots, x_{24}\}$.

Initially, 24 primary clusters exist at level 0 such that $c_1^0 = \{x_1\}, c_2^0 = \{x_2\}, \ldots, c_{24}^0 = \{x_{24}\}$, where $m_0 = 24$. Since $m_0 \neq m$, we select an auxiliary CA $\mathcal{R}$ from the set of candidate CAs (satisfying Property 1) uniformly random without replacement. Let $\mathcal{R} = (9, 169, 150, 150, 165, 105, 165, 20)$. This CA generates four auxiliary clusters: $C_1^0, C_2^0, C_3^0, C_4^0$ (see Example 6).

Next, we find the degree of participation of each c_j^0 in these clusters. Since we are at level 1, $\mu(C_1^1, c_{j_1}^0) = 100\%$ $(\forall j_1 \in \{1, 3, 4, 22, 13, 14\})$, $\mu(C_2^1, c_{j_2}^0) = 100\%$ $(\forall j_2 \in \{2, 5, 6, 7, 8, 9, 10, 11, 12, 15, 17, 19, 20, 21, 23, 24\})$, $\mu(C_3^1, c_{16}^0) = 100\%$ and $\mu(C_4^1, c_{18}^0) = 100\%$. So we can merge the closely reachable primary clusters of level 0 to form the primary clusters of level 1. Here, auxiliary cluster C_1^1 has maximum (and equal) participation of primary clusters $c_1^0, c_3^0, c_4^0, c_{22}^0, c_{13}^0$ and c_{14}^0. Similarly, C_2^1 has maximum participation of $c_2^0, c_5^0, c_6^0, c_7^0, c_8^0, c_9^0, c_{10}^0, c_{11}^0, c_{12}^0, c_{15}^0, c_{17}^0, c_{19}^0, c_{20}^0, c_{21}^0, c_{23}^0$ and c_{24}^0. Therefore, the newly generated primary cluster of level 1 is

$$c_1^1 = c_1^0 \cup c_3^0 \cup c_4^0 \cup c_{22}^0 \cup c_{13}^0 \cup c_{14}^0.$$

Similarly, c_2^1 can be generated. For the remaining two auxiliary clusters, new primary clusters are formed as $c_3^1 = c_{16}^0$ and $c_4^1 = c_{18}^0$. Since the number of primary clusters at level 1 (m_1) is $4 \neq m$, we move from level 1 to level 2.

At level 2, let the selected auxiliary CA be (6, 232, 90, 90, 165, 90, 90, 20). This CA generates six auxiliary clusters, $C_1^2, C_2^2, C_3^2, C_4^2, C_5^2$

and C_6^2. Like level 1, here we also compute the maximum participation of each primary cluster of level 1 in auxiliary cluster C_t^2, $(1 \le t \le 6)$. It is found that $\mu(C_1^2, c_1^1) = 16\%$, $\mu(C_2^2, c_1^1) = 33\%$, $\mu(C_2^2, c_2^1) = 62\%$, $\mu(C_2^2, c_3^1) = 100\%$, $\mu(C_3^2, c_1^1) = 16\%$, $\mu(C_3^2, c_2^1) = 12\%$, $\mu(C_3^2, c_4^1) = 100\%$, $\mu(C_4^2, c_1^1) = 33\%$, $\mu(C_5^2, c_2^1) = 18\%$ and $\mu(C_6^2, c_2^1) = 6\%$. Hence, we can merge c_2^1 and c_3^1 with respect to the closeness in the auxiliary cluster C_2^2. Similarly, c_1^1 and c_4^1 can be merged with respect to C_3^2. Hence, the newly generated primary clusters of level 2 are

$$c_1^2 = v_2^1 \cup c_3^1 = \{2, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 17, 19, 20, 21, 23, 24\}$$

and

$$c_2^2 = c_1^1 \cup c_4^1 = \{1, 3, 4, 22, 13, 14, 18\}.$$

Therefore, at this level, $m_2 = 2$. Since the desired number of clusters is already achieved, the algorithm exits.

Hence, our algorithm can generate the required clusters and also gives a direction for an optimal number of clusters. Our technique uses v auxiliary CAs if the optimal number of clusters is achieved at level v. Note also that the optimal number of clusters can be achieved using a set of v_1 CAs, v_2 CAs and so on. Now, our interest is to figure out that set of bijective functions (CAs) that gives the best-quality clusters among those possible in the set of CAs. An extensive experiment is performed in the next section on real datasets using these CAs and the results are reported.

6. Results and Discussion

This section reports the performance of our proposed iterative cycle-based clustering algorithm on some real datasets (archive.ics.uci.edu/ml/index.php). To check the quality of the clusters generated using our algorithm, we use the benchmark validation indices silhouette score, Dunn index and connectivity [1, 10]. The set of bijective functions (auxiliary CAs) with the smallest connectivity score and highest Dunn and silhouette index scores generates the best quality clusters. Next, we introduce the real datasets used in this paper.

We use seven datasets: Iris, User Knowledge Modeling, Buddy-Move, Seed, StoneFlakes, Heart failure clinical records and Wholesale Customers. Each of them has mostly quantitative attributes. Table 6 reports the details about these datasets. In this table, column 1 gives the names of the datasets and columns 2, 3 and 4 refer to the total number of attributes p, number of quantitative attributes p_{q_n} and the count of qualitative attributes p_{q_l}, respectively. Column 5 shows the number of target objects that are to be distributed among m clusters using an n-cell CA. Here, n (see column 6 for reference) is computed based on p_{q_n} and p_{q_l} ($n > p$) (see Section 3). Column 7 refers to the number of objects with missing terms for each dataset.

For analyzing the performance of our iterative clustering technique, we first need to check whether the target objects are distributed among the desired number of clusters; thereafter, the cluster quality is evaluated based on the validation indices scores that can be computed using the package clValid in R (see [7] for a detailed description of the package). In this paper, the desired number of clusters is two. Section 6.1 reports the results of these experiments, taking a fixed ordering of features for each of the datasets.

6.1 Experiment on Real Datasets

Table 7 presents the effectiveness of our technique. This effectiveness is ensured based on validation indices scores and if each of the datasets is distributed into the same desired number of clusters, which is two.

In Table 7, column 3 refers to the optimal scores of the validation indices—connectivity, silhouette score and Dunn index. Columns 4 and 5 represent the number of levels used to get the optimal results for each dataset and the corresponding ordered list of auxiliary CAs where any CA_i is exclusively used for a particular level i. For example, the optimal scores for all three validation indices in the Iris dataset are achieved while clustering is completed in level 2. Moreover, the same set of auxiliary CAs is used for all three cases. The best scores are also found for all three indices using the same set of auxiliary CAs in the BuddyMove dataset. For the other datasets, the same set of CAs was used to produce the best quality clusters with respect to two validation indices.

Dataset	m	Optimal Score	Number of Levels	Auxiliary CA
Iris	2	Silhouette=0.6867, Dunn=0.3389, Connectivity=0.0000	2	CA_0: (10, 75, 166, 105, 105, 166, 150, 20) CA_1: (6, 166, 165, 154, 105, 165, 165, 65)

Table 7. (*continues*)

Dataset	m	Optimal Score	Number of Levels	Auxiliary CA
User Knowledge Modeling	2	Silhouette=0.186714, Dunn=0.1508426, Connectivity= 3.667857	3	CA_0: (6, 166, 105, 165, 165, 150, 154, 150, 165, 65) CA_1: (10, 180, 169, 165, 90, 90, 165, 165, 149, 80) CA_2: (5, 180, 165, 225, 86, 105, 90, 105, 154, 80)
			4	CA_0: (6, 169, 90, 169, 90, 169, 150, 150, 90, 80) CA_1: (9, 228, 154, 105, 105, 89, 90, 106, 105, 65) CA_2: (9, 105, 135, 172, 90, 165, 90, 90, 90, 20) CA_3: (5, 45, 89, 90, 169, 165, 101, 150, 90, 20)
Buddymove	2	Silhouette=0.4763, Dunn=0.3146, Connectivity=2.9289	4	CA_0: (10, 135, 197, 150, 169, 105, 105, 101, 105, 105, 86, 80) CA_1: (9, 150, 30, 198, 154, 150, 165, 165, 90, 105, 105, 86, 80) CA_2: (9, 166, 105, 150, 101, 150, 90, 90, 105, 86, 105, 20) CA_3: (6, 105, 30, 202, 90, 150, 105, 105, 105, 166, 105, 65)
Seed	2	Silhouette=0.528, Dunn=0.09, Connectivity=3.91	2	CA_0: (9, 228, 169, 165, 101, 90, 90, 90, 154, 150, 150, 105, 105, 20) CA_1: (5, 105, 89, 105, 105, 105, 150, 150, 150, 150, 165, 165, 165, 20)
			4	CA_0: (5, 120, 197, 150, 86, 165, 106, 165, 166, 105, 90, 105, 166, 5) CA_1: (5, 210, 165, 180, 202, 165, 150, 101, 90, 90, 105, 165, 165, 65) CA_2: (10, 120, 90, 105, 172, 150, 86, 150, 105, 154, 150, 165, 165, 65) CA_3: (5, 120, 172, 165, 150, 149, 150, 150, 169, 150, 105, 166, 105, 20)

Table 7. (*continues*)

Dataset	m	Optimal Score	Number of Levels	Auxiliary CA
StoneFlakes	2	Silhouette=0.376, Dunn=0.1805, Connectivity=3.5123	3	CA_0: (6, 86, 165, 149, 165, 89, 165, 90, 90, 101, 105, 90, 150, 169, 150, 65) CA_1: (10, 135, 154, 165, 165, 150, 165, 90, 105, 106, 165, 154, 150, 150, 154, 5) CA_2: (10, 105, 166, 150, 150, 149, 150, 150, 165, 165, 154, 150, 90, 105, 86, 80)
			4	CA_0: (6, 166, 105, 105, 89, 90, 165, 165, 166, 150, 150, 150, 90, 106, 150, 20) CA_1: (6, 166, 150, 165, 165, 105, 105, 90, 169, 105, 90, 150, 89, 105, 165, 65) CA_2: (5, 210, 92, 105, 149, 90, 150, 105, 90, 165, 105, 106, 105, 165, 165, 65) CA_3: (5, 165, 92, 150, 165, 90, 169, 105, 105, 89, 105, 90, 150, 89, 150, 20)
Wholesale Customer	2	Silhouette=0.5257, Connectivity= 3.7329, Dunn= 0.0508	3	CA_0: (10, 45, 149, 105, 165, 90, 150, 101, 90, 149, 165, 106, 90, 150, 150, 65) CA_1: (9, 78, 165, 105, 90, 165, 105, 90, 90, 86, 105, 105, 154, 150, 90, 65) CA_2: (9, 154, 165, 150, 90, 150, 150, 90, 150, 86, 105, 90, 150, 86, 150, 20)
			2	CA_0: (6, 149, 150, 90, 150, 105, 90, 90, 165, 150, 90, 169, 165, 90, 105, 20) CA_1: (5, 210, 150, 228, 86, 105, 90, 105, 89, 90, 105, 105, 165, 90, 90, 20)
			3	CA_0: (6, 216, 149, 150, 150, 169, 150, 150, 106, 105, 90, 90, 90, 150, 169, 5) CA_1: (6, 89, 150, 165, 90, 90, 165, 90, 150, 101, 150, 105, 89, 90, 90, 5) CA_2: (10, 150, 197, 165, 90, 165, 90, 150, 154, 165, 106, 105, 165, 105, 101, 5)

Table 7. (*continues*)

Dataset	m	Optimal Score	Number of Levels	Auxiliary CA
Heart failure clinical records	2	Silhouette=0.6885, Dunn=0.0945, Connectivity=3.1718	2	CA_0: (9, 154, 90, 166, 105, 105, 154, 105, 150, 106, 150, 90, 150, 154, 90, 86, 165, 90, 5) CA_2: (9, 212, 165, 105, 166, 165, 101, 105, 150, 169, 150, 150, 105, 150, 90, 90, 105, 154, 80)
			2	CA_0: (5, 135, 166, 165, 165, 105, 165, 90, 90, 90, 165, 105, 89, 105, 150, 165, 150, 90, 20) CA_1: (9, 149, 150, 105, 169, 90, 89, 150, 90, 150, 106, 90, 89, 90, 166, 165, 86, 105, 20)

Table 7. Performance of our clustering algorithm on available datasets (see Table 6).

Next, we show that our algorithm can produce similar clusters even if the order of the features is changed. This claim is verified by extensive experimentation on the real datasets used here.

6.2 Experimentation on Datasets after Reordering Features

Every target object is represented by its p feature values if it owns p distinct features $A_1, A_2, \ldots, A_p$. Generally, an object is independent of the order of its features. But because CAs are used, the mapping of a target object to a useful configuration can be changed if the features are in a different order. However, if we can get two sets of auxiliary CAs R^1 and R^2 that cluster the given set of target objects in a similar fashion irrespective of the order of the elements, then we call the set of auxiliary CAs R^1 and R^2 *equivalent auxiliary CA spaces*. Tables 8 and 9 show that our iterative level-wise clustering algorithm also supports the reordering of features.

Let us first take the Iris dataset where each object is represented by the attributes A_1, A_2, A_3 and A_4. Since we use CAs, each object is mapped to a configuration. Until now, the experiment on this dataset has used the feature ordering $\langle A_1, A_2, A_3, A_4 \rangle$. However, by changing the ordering of the features, we can get $4! = 24$ distinct combinations. Some of them are $\langle A_2, A_3, A_1, A_4 \rangle$, $\langle A_3, A_4, A_2, A_1 \rangle$, and so on. Evidently, when the order of features is changed, the corresponding useful configuration is also changed. Therefore, another set of CAs can be used for clustering that set of useful configurations. The desired number of clusters is two. Table 8 reports the result of this experiment on all possible orderings. From Table 8, it can be concluded that the count of clusters and the arrangement of clusters remain the same even after altering the order of the features. Table 8 also shows that several different sets of auxiliary CAs exist that produce the same clusters. For instance,

$$R^1 = \{(5, 165, 101, 165, 89, 165, 106, 80)\}$$

$$R^2 = \{(9, 90, 90, 101, 150, 105, 106, 80), \\ (9, 141, 86, 90, 150, 105, 165, 20)\}$$

and

$$R^3 = \{(10, 150, 108, 105, 165, 166, 105, 65), \\ (6, 165, 165, 78, 106, 165, 90, 80)\}$$

are three sets of auxiliary CAs for the given order of features $\langle A_1, A_2, A_3, A_4\rangle$, $\langle A_1, A_2, A_4, A_3\rangle$ and $\langle A_3, A_4, A_2, A_1\rangle$, respectively. These R^1, R^2 and R^3 form equivalent clustering of the objects; hence these are equivalent auxiliary CA spaces. Moreover, it is observed that each of them produces the best clusters with the same Silhouette score, Dunn index and Connectivity value.

Order of Features	Set of Auxiliary CAs	Level
$\langle A_1, A_2, A_3, A_4\rangle$	CA_0: (5, 165, 101, 165, 89, 165, 106, 80)	1
$\langle A_1, A_2, A_4, A_3\rangle$	CA_0: (9, 90, 90, 101, 150, 105, 106, 80)	2
	CA_1: (9, 141, 86, 90, 150, 105, 165, 20)	
$\langle A_1, A_3, A_2, A_4\rangle$	CA_0: (10, 105, 156, 86, 105, 150, 166, 80)	2
	CA_1: (10, 180, 169, 165, 86, 150, 90, 65)	
$\langle A_1, A_4, A_2, A_3\rangle$	CA_0: (10, 165, 101, 105, 90, 105, 106, 80)	2
	CA_1: (5, 90, 105, 149, 150, 165, 90, 20)	
$\langle A_2, A_1, A_3, A_4\rangle$	CA_0: (10, 105, 198, 150, 90, 106, 150, 20)	2
	CA_1: (9, 166, 90, 169, 105, 105, 166, 80)	
$\langle A_2, A_1, A_4, A_3\rangle$	CA_0: (6, 86, 90, 165, 150, 105, 166, 80)	3
	CA_1: (9, 150, 120, 198, 165, 165, 106, 80)	
	CA_2: (10, 90, 169, 165, 169, 150, 90, 65)	
$\langle A_2, A_3, A_1, A_4\rangle$	CA_0: (10, 75, 166, 90, 166, 90, 106, 80)	3
	CA_1: (9, 89, 90, 165, 90, 106, 150, 20)	
	CA_2: (6, 78, 150, 165, 86, 90, 89, 80)	
$\langle A_2, A_4, A_1, A_3\rangle$	CA_0: (5, 135, 105, 90, 142, 150, 106, 80)	2
	CA_1: (9, 86, 165, 86, 165, 105, 105, 65)	

Table 8. (*continues*)

Order of Features	Set of Auxiliary CAs	Level
$\langle A_3, A_1, A_2, A_4\rangle$	CA_0: (10, 150, 172, 90, 165, 105, 169, 5)	4
	CA_1: (10, 180, 90, 165, 150, 101, 105, 65)	
	CA_2: (6, 106, 150, 150, 106, 90, 86, 80)	
	CA_3: (10, 30, 172, 105, 105, 105, 165, 65)	
$\langle A_3, A_1, A_4, A_2\rangle$	CA_0: (6, 216, 150, 90, 169, 105, 90, 65)	3
	CA_1: (10, 150, 156, 89, 165, 150, 105, 65)	
	CA_2: (6, 212, 90, 90, 150, 106, 105, 65)	
$\langle A_3, A_2, A_1, A_4\rangle$	CA_0: (6, 105, 120, 105, 101, 90, 106, 80)	3
	CA_1: (6, 169, 90, 150, 105, 150, 166, 80)	
	CA_2: (9, 232, 90, 105, 165, 150, 90, 65)	
$\langle A_3, A_2, A_4, A_1\rangle$	CA_0: (6, 105, 120, 105, 101, 90, 106, 80)	3
	CA_1: (6, 169, 90, 150, 105, 150, 166, 80)	
	CA_2: (9, 232, 90, 105, 165, 150, 90, 65)	
$\langle A_3, A_4, A_1, A_2\rangle$	CA_0: (10, 30, 92, 105, 105, 105, 165, 65)	3
	CA_1: (6, 216, 105, 105, 165, 105, 105, 80)	
	CA_2: (10, 120, 198, 169, 165, 105, 150, 65)	
$\langle A_3, A_4, A_2, A_1\rangle$	CA_0: (10, 150, 108, 105, 165, 166, 105, 65)	2
	CA_1: (6, 165, 165, 78, 106, 165, 90, 80)	
$\langle A_4, A_1, A_2, A_3\rangle$	CA_0: (6, 90, 212, 150, 106, 90, 89, 5)	3
	CA_1: (10, 180, 149, 105, 150, 86, 105, 65)	
	CA_2: (5, 105, 154, 165, 101, 165, 166, 80)	
$\langle A_4, A_1, A_3, A_2\rangle$	CA_0: (10, 30, 172, 105, 166, 165, 101, 80)	2
	CA_1: (9, 228, 150, 150, 105, 101, 150, 65)	
$\langle A_4, A_2, A_1, A_3\rangle$	CA_0: (6, 228, 86, 105, 150, 149, 105, 65)	4
	CA_1: (6, 90, 232, 90, 150, 154, 150, 65)	
	CA_2: (10, 120, 86, 90, 150, 165, 169, 5)	
	CA_3: (9, 78, 166, 105, 105, 86, 150, 65)	
$\langle A_4, A_2, A_3, A_1\rangle$	CA_0: (6, 212, 150, 105, 90, 150, 105, 20)	2
	CA_1: (10, 210, 149, 90, 89, 105, 90, 20)	
$\langle A_4, A_3, A_2, A_1\rangle$	CA_0: (6, 216, 105, 165, 90, 150, 165, 20)	1
$\langle A_4, A_3, A_1, A_2\rangle$	CA_0: (10, 120, 166, 150, 150, 165, 105, 20)	2
	CA_1: (10, 120, 166, 150, 150, 165, 105, 20)	

Table 8. Performance on reordering of features of Iris dataset where $m = 2$; for any feature ordering (column 1), there exist auxiliary CAs (column 2) that generate the set of clusters with same scores of validation indices (Silhouette= 0.6867, Dunn=0.3389, Connectivity=0.0000) as given in Table 7.

However, it is not always necessary that the exact same cluster be formed for every possible reordering. They can be almost similar, which can be indicated by having nearly the same score in the validation indices (see the result of the Seed dataset in Table 9). For example, for the BuddyMove dataset in Table 9, each of three distinct feature orders can be clustered using equivalent auxiliary CA spaces giving exactly the same value on the three validation indices. Whereas, for the StoneFlakes dataset, equivalent auxiliary CA spaces can be found with respect to Dunn index and Connectivity for the two orderings $\langle A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\rangle$ and $\langle A_4, A_5, A_6, A_7, A_8, A_1, A_2, A_3\rangle$. Now, we can compare our algorithm with some existing well-known techniques.

Dataset	Order of Features	Set of Auxiliary CAs	Level	Scores of Indices

BuddyMove	$\langle A_1, A_2, A_3, A_4, A_5, A_6\rangle$	CA$_0$: (10, 210, 156, 106, 90, 106, 150, 150, 105, 165, 86, 80)
		CA$_1$: (10, 165, 201, 154, 165, 169, 90, 154, 150, 165, 165, 20)
		CA$_2$: (5, 135, 86, 105, 105, 165, 90, 89, 150, 150, 105, 5)
		CA$_3$: (5, 30, 89, 90, 105, 165, 86, 165, 101, 165, 166, 80)
		CA$_4$: (9, 166, 165, 169, 105, 90, 150, 89, 150, 105, 149)
			5	Silhouette= 0.4763
				
				Dunn= 0.3146
				
				Connectivity =2.9289

	$\langle A_2, A_4, A_3, A_1, A_6, A_5\rangle$	CA$_0$: (9, 86, 105, 90, 105, 149, 165, 86, 105, 90, 165, 65)	1	
	$\langle A_3, A_4, A_1, A_2, A_5, A_6\rangle$	CA$_0$: (5, 135, 172, 150, 154, 105, 150, 90, 150, 105, 89, 80) CA$_1$: (6, 90, 105, 120, 92, 90, 165, 90, 150, 106, 105, 65) CA$_2$: (9, 149, 90, 101, 90, 106, 90, 90, 105, 105, 105, 80)	3	
StoneFlakes	$\langle A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\rangle$	CA$_0$: (6, 86, 165, 149, 165, 89, 165, 90, 90, 101, 105, 90, 150, 169, 150, 65) CA$_1$: ((10, 135, 154, 165, 165, 150, 165, 90, 105, 106, 165, 154, 150, 150, 154, 5)) CA$_2$: (5, 30, 166, 165, 165, 150, 90, 65)	3	Silhouette= 0.376

Table 9. (*continues*)

Dataset	Order of Features	Set of Auxiliary CAs	Level	Scores of Indices
StoneFlakes	$\langle A_4, A_5, A_6, A_7, A_8, A_1, A_2, A_3\rangle$	CA_0: (6, 142, 105, 101, 165, 166, 90, 169, 105, 105, 166, 90, 89, 90, 86, 80) CA_1: (6, 89, 105, 165, 90, 105, 154, 165, 150, 165, 169, 165, 89, 150, 165, 65)	2	Silhouette= 0.3377
	$\langle A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8\rangle$	CA_0: (6, 166, 105, 105, 89, 90, 165, 165, 166, 150, 150, 150, 90, 106, 150, 20) CA_1: (6, 166, 150, 165, 165, 105, 105, 90, 169, 105, 90, 150, 89, 105, 165, 65) CA_2: (5, 210, 92, 105, 149, 90, 150, 105, 90, 165, 105, 106, 105, 165, 165, 65) CA_3: (5, 165, 92, 150, 165, 90, 169, 105, 105, 89, 105, 90, 150, 89, 150, 20)	4	Dunn= 0.1805 Connectivity =3.5123
	$\langle A_4, A_5, A_6, A_7, A_8, A_1, A_2, A_3\rangle$	CA_0: (9, 166, 150, 165, 105, 149, 165, 90, 90, 150, 150, 165, 165, 105, 86, 5)	1	
Seed	$\langle A_1, A_2, A_3, A_4, A_5, A_6, A_7\rangle$	CA_0: (9, 228, 169, 165, 101, 90, 90, 90, 154, 150, 150, 105, 105, 20) CA_1: (5, 105, 89, 105, 105, 105, 150, 150, 150, 150, 165, 165, 165, 20)	2	Silhouette= 0.5288
	$\langle A_1, A_2, A_5, A_6, A_7, A_3, A_4\rangle$	CA_0: (10, 225, 198, 149, 150, 90, 105, 89, 150, 90, 90, 90, 165, 65) CA_1: (6, 77, 165, 90, 150, 106, 150, 90, 150, 150, 150, 90, 90, 65) CA_2: (9, 169, 90, 106, 150, 105, 150, 105, 105, 89, 105, 105, 154, 80) CA_3: (9, 101, 105, 150, 149, 105, 165, 105, 169, 105, 165, 165, 90, 65)	4	Silhouette= 0.5282

Table 9. (*continues*)

Dataset	Order of Features	Set of Auxiliary CAs	Level	Scores of Indices
Seed	$\langle A_1, A_2, A_3, A_4, A_5, A_6, A_7\rangle$	CA_0: (5, 120, 197, 150, 86, 165, 106, 165, 166, 105, 90, 105, 166, 5) CA_1: (5, 210, 165, 180, 202, 165, 150, 101, 90, 90, 105, 165, 165, 65) CA_2: (10, 120, 90, 105, 172, 150, 86, 150, 105, 154, 150, 165, 165, 65) CA_3: (5, 120, 172, 165, 150, 149, 150, 150, 169, 150, 105, 166, 105, 20)	4	Dunn=0.09 Connectivity =3.91
	$\langle A_1, A_2, A_5, A_6, A_7, A_3, A_4\rangle$	CA_0: (5, 75, 154, 150, 90, 105, 106, 150, 165, 90, 105, 166, 150, 65) CA_1: (9, 86, 105, 105, 166, 165, 154, 90, 166, 150, 90, 165, 165, 65) CA_2: (6, 86, 150, 105, 89, 90, 101, 90, 105, 90, 105, 150, 165, 65)	3	Dunn=0.08 Connectivity =3.76

Table 9. Performance of other datasets on different ordering of features where $m = 2$.

6.3 Comparison of Clustering Techniques on the Same Dataset

To judge the efficiency of our iterative clustering model, we need to compare the results of the validation indices achieved by our technique with the existing benchmark clustering algorithms. Here, we use five benchmark clustering algorithms: K-means (centroid-based clustering) [7], hierarchical (agglomerative hierarchical clustering) [7], DIANA (divisive hierarchical clustering) [7], PAM (partitioning around medoids) (centroid-based clustering) [7] and SOTA (self-organizing tree algorithm) (unsupervised network with a divisive hierarchical clustering) [7] using the implementation in R [7]. Table 10 reports the results of this comparison. From the experiments, it can be noted that, among the existing algorithms, the optimal silhouette score for the StoneFlakes dataset is found by K-means, DIANA and SOTA. Whereas the best connectivity score for the Heart failure clinical records dataset is achieved by using the K-means and DIANA algorithms. However, the hierarchical algorithm forms clusters most effectively by overall performance on all datasets. Further, Table 10 shows that the validation indices scores obtained by our algorithm can compete with the best scores obtained by the benchmark algorithms. Hence, our algorithm is one of the best algorithms existing today for clustering any kind of dataset.

Dataset	Algorithm	Connectivity	Dunn Index	Silhouette Score
Iris	Hierarchical	0.0000	0.3389	0.6867
	K-means	6.1536	0.0765	0.6810
	DIANA	6.1536	0.0765	0.6810
	PAM	3.9623	0.0811	0.6858
	SOTA	11.5016	0.0349	0.6569
	CA-based clustering	**0.0000**	**0.3389**	**0.6867**
User Knowledge Modeling	Hierarchical	3.0540	0.1970	0.2589
	K-means	50.3321	0.0943	0.2063
	DIANA	68.7329	0.0297	0.2066
	PAM	102.8329	0.0537	0.1934
	SOTA	64.1060	0.0321	0.2042
	CA-based clustering	**3.6678**	**0.1508**	**0.1867**
BuddyMove	Hierarchical	2.9290	0.3146	0.4764
	K-means	35.2032	0.0485	0.3079
	DIANA	34.3694	0.0658	0.3020
	PAM	38.3802	0.0485	0.3055
	SOTA	44.4111	0.0518	0.3134
	CA-based clustering	**2.9289**	**0.3146**	**0.4763**
Seed	Hierarchical	8.7861	0.1065*	0.5248
	K-means	21.3698	0.0548	0.5229
	DIANA	19.1714	0.0544	0.5218
	PAM	20.6762	0.0404	0.5175
	SOTA	16.0179	0.0361	0.5049
	CA-based clustering	**3.91**	**0.09**	**0.528**
StoneFlakes	Hierarchical	2.9290	0.3176*	0.2937
	K-means	4.9488	0.1735	0.4875
	DIANA	4.9488	0.1735	0.4875
	PAM	7.9762	0.1693	0.4827
	SOTA	4.9488	0.1735	0.4875
	CA-based clustering	**3.5123**	**0.1805**	**0.376**
Heart failure clinical records	Hierarchical	5.7536	0.1506	0.7862
	K-means	1.8635	0.0079	0.5829
	DIANA	1.8635	0.0079	0.5829
	PAM	3.5020	0.0036	0.4630
	SOTA	5.0972	0.0035	0.5302
	CA-based clustering	**3.1718**	**0.0945**	**0.6885**
Wholesale Customers	Hierarchical	2.9290	0.3853	0.7957
	K-means	30.0881	0.0178	0.5115
	DIANA	27.1242	0.0313	0.5806
	PAM	41.6647	0.0167	0.3819
	SOTA	43.8266	0.0061	0.3699
	CA-based clustering	**3.7329**	**0.0508**	**0.5257**

Table 10. Comparison of clustering techniques based on internal validation indices for each of the available datasets of Table 6. Here, $m = 2$ for all entries except those marked with * for which $m = 6$.

References

[1] D. Xu and Y. A Tian, "A Comprehensive Survey of Clustering Algorithms," *Annals of Data Science*, **2**(2), 2015 pp. 165–193. doi:10.1007/s40745-015-0040-1.

[2] A. K. Jain, M. N. Murty and P. J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, **31**(3), 1999 pp. 264–323. doi:10.1145/331499.331504.

[3] R. Xu and D. Wunsch II, "Survey of Clustering Algorithms," *IEEE Transactions on Neural Networks*, **16**(3), 2005 pp. 645–678. doi:10.1109/TNN.2005.845141.

[4] S. Wolfram, *Theory and Applications of Cellular Automata: Including Selected Papers, 1983–1986*, Singapore: World Scientific, 1986.

[5] N. Naskar, S. Adak, P. Maji and S. Das, "Synthesis of Non-uniform Cellular Automata Having Only Point Attractors," in *Proceedings of* 11th *International Conference on Cellular Automata for Research and Industry (ACRI 2014)*, Kraków, Poland, 2014 (J. Was, G. C. Sirakoulis and S. Bandini, eds.), Cham, Switzerland: Springer, 2014 pp. 105–114. doi:10.1007/978-3-319-11520-7_12.

[6] S. Adak, S. Mukherjee and S. Das, "Do There Exist Non-linear Maximal Length Cellular Automata? A Study," in *Proceedings of* 13th *International Conference on Cellular Automata for Research and Industry (ACRI 2018)*, Como, Italy, 2018 (G. Mauri, S. El Yacoubi, A. Dennunzio, K. Nishinari and L. Manzoni, eds.) Cham, Switzerland: Springer, 2018 pp. 289–297. doi:10.1007/978-3-319-99813-8_26.

[7] G. Brock, V. Pihur, S. Datta and S. Datta, "clValid: an R Package for Cluster Validation," *Journal of Statistical Software*, **25**(4), 2008 pp. 1–22. doi:10.18637/jss.v025.i04.

[8] S. Das, "Theory and Applications of Nonlinear Cellular Automata in VLSI Design," Ph.D. thesis, Bengal Engineering and Science University, Shibpur, India, 2006.

[9] J. Dougherty, R. Kohavi and M. Sahami, "Supervised and Unsupervised Discretization of Continuous Features," in *Machine Learning Proceedings 1995: Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA, 1995 (A. Prieditis and S. Russell, eds.), San Francisco: Morgan Kaufmann, 1995 pp. 194–202. doi:10.1016/B978-1-55860-377-6.50032-3.

[10] V. Estivill-Castro, "Why So Many Clustering Algorithms: A Position Paper," *ACM Special Interest Group on Knowledge Discovery in Data Explorations Newsletter*, **4**(1), 2002 pp. 65–75. doi:10.1145/568574.568575.