Cycle Based Clustering Using Reversible Cellular Automata

Sukanya Mukherjee, Kamalika Bhattacharjee, Sukanta Das

Cycle based Clustering using Reversible Cellular Automata

Sukanya Mukherjee[1⋆], Kamalika Bhattacharjee[2], and Sukanta Das[3]

[1] Department of Computer Science and Engineering, Institute of Engineering and Management, Kolkata, West Bengal, 700091, India
sukanya.mukherjee@iemcal.com
[2] Department of Computer Science and Engineering, Indian Institute of Information Technology Ranchi, Jharkhand, 834010, India
kamalika.it@gmail.com
[3] Department of Information Technology, Indian Institute of Engineering Science and Technology, Shibpur, Howrah, West Bengal, 711103, India
sukanta@it.iiests.ac.in

Abstract. This work proposes cycle based clustering technique using reversible cellular automata (CAs) where 'closeness' among objects is represented as objects belonging to the same cycle, that is *reachable* from each other. The properties of such CAs are exploited for grouping the objects with minimum intra-cluster distance while ensuring that limited number of cycles exist in the configuration-space. The proposed algorithm follows an iterative strategy where the clusters with *closely reachable* objects of previous level are *merged* in the present level using an unique *auxiliary* CA. Finally, it is observed that, our algorithm is at least at par with the best algorithm existing today.

Keywords: Reversible Cellular Automata, Reachability, Large length cycle, Level-wise clustering, Connectivity, Silhouette score, Dunn index

1 Introduction

Clustering technique [10, 12, 13] is a well studied research topic when no class information is provided for grouping the available objects (data records) based on some closeness measuring metric. It also exploits the inherent anatomy of data objects for partitioning the dissimilar objects into separate clusters. Till date, a varied collection of well accepted algorithms for clustering [10, 12] have been developed. Such clustering techniques aim to gather alike data objects for producing 'good' clusters where intra-cluster and inter-cluster isolation, based on feature space should be lower and higher respectively. Maintaining less intra-cluster distance between any two objects intrinsically means the alike objects are *interconnected*. This interconnectivity in clustering motivates us to use cellular automata (CAs) as *natural* clusters.

In a cellular automaton (CA), the configurations within a cycle are *reachable* from one another, whereas the configurations of different cycles are *not-reachable*.

⋆ Corresponding author

Moreover, the *locality* property of CA influences that the *related* configurations are reachable. This 'reachability' is the key of clustering using CA as reachable configurations form cycle(s). Therefore, CA can act as a function that maintains bijective mapping among the configurations which are reachable or connected and gathers similar objects (configurations) into same cluster (cycle). This work reports reversible non-uniform CAs as the proposed model, where each configuration `x` (object) is reachable (connected) from the remaining configurations of that cycle (cluster) where `x` belongs to. Hereafter, if not otherwise mentioned, by a 'CA', we shall mean a reversible non-uniform CA under null-boundary condition which uses Wolfram's rules [11].

Ideally, a CA of size n can distribute 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 in one cluster or distribute among m clusters where the count of target objects is m ($\leq 2^n$) – both of which are not desirable for good clustering. Therefore, a CA is said to be *effective* for clustering if it can distribute the target objects among a *limited number of clusters*. This limited number of clusters is not necessarily the only primary objective for efficient CA based clustering – the feature based distances among the reachable configurations of each cluster should also be as minimum as possible. So, clustering can be viewed as an optimization problem where there is a trade-off between these two facets.

To incorporate this idea in CA based clustering technique, this work proposes an intelligent *arrangement* of CA rules for generating an n-cell *candidate* CA which is capable of maintaining less intra-cluster distance among objects and generates limited number of cycles. (Here, n is determined based on the number of features owned by the target objects which is always finite.) For ensuring that this CA has limited number of cycles, we use the framework of a related problem – CA with large cycles (introduced in [1]). However, to guarantee that for the candidate CA, the configurations inside a cycle maintain minimum possible *hamming distance* (as binary CA is considered for this research), we propose a scheme to select *significant* rules which contribute minimum change in cell's state value when transition occurs between configurations (Section 3.2). Next, the *proportion* of such significant rules for designing an n-cell (*non-uniform*) CA for clustering is evaluated to ensure optimal number of clusters (Section 3.3).

Definitely, generating a CA for a fixed n maintaining less intra-cluster distance and limited number of clusters is a very challenging problem; even if the desired number of clusters is given, it is difficult to determine the corresponding CA which can produce *good* clusters for the given dataset. The inherent hardness of this problem motivates us to take an iterative strategy which distributes the target objects in m clusters. In the proposed algorithm, the clusters of level i are generated by *merging* a set of clusters of level $i-1$ which are *closely reachable* (Section 4). To measure the quality (goodness) of clusters, some benchmark *cluster validation indices* (internal) [2] are used. Section 5 presents the performance analysis of our proposed cycle based clustering algorithm on some real datasets taken from ML repository (`http://archive.ics.uci.edu/ml/index.php`). Finally, we compare our proposed algorithm with some traditional benchmark clus-

tering algorithms like *centroid based clusterings*, *hierarchical clusterings* [2, 12]. Our results indicate that, performance of our CA-based clustering technique is at least as good as the best known clustering algorithm existing today.

2 Basics of CAs

In this work, we use one-dimensional three-neighborhood n-cell CAs under null boundary condition, where each cell takes any of the states $S = \{0, 1\}$. The next state of each cell is updated following an *elementary cellular automaton* (ECA) rule. The present state of all cells at a given time is called the *configuration* of the CA. Thus, 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 $\mathtt{x} = (x_i)_{\forall i \in n}$ is $\mathtt{y}$, then $\mathtt{y} = G(\mathtt{x})$ where $\mathtt{x}, \mathtt{y} \in C$, $\mathtt{y} = (y_i)_{\forall i \in n}$ and x_i, y_i are the present and next state values of cells 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 named as the *Rule Min Term* (RMT) and represented by its decimal equivalent $r = 2^2 \times x_{i-1} + 2^1 \times x_i + x_{i+1}$. A *rule vector* $\mathcal{R} = (\mathcal{R}_0,\ \mathcal{R}_1, \cdots, \mathcal{R}_{n-1})$ of length n is used to represent any arbitrary n-cell non-uniform CA, where $\mathcal{R}_i \neq \mathcal{R}_j$ for some i and j.

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) Next state	1	0	0	1	0	1	1	0	150($\mathcal{R}_1$)
(iii) Next state	0	0	1	0	1	1	0	1	45($\mathcal{R}_2$)
(iv) Next state	i	1	i	0	i	0	i	1	65($\mathcal{R}_3$)

Table 1: An 4-cell CA $(9, 150, 45, 65)$.

Let us now present the rules in tabular form (see Table 1). Obviously, there are $2^8 = 256$ distinct rules. These rules are traditionally named by their decimal equivalents. Since we are using null boundary condition, $x_{-1} = x_n = 0$. Therefore, for the first cell, $y_0 = \mathcal{R}_0(0, x_0, x_1)$, and for the last cell, $y_{n-1} = \mathcal{R}_{n-1}(x_{n-2}, x_{n-1}, 0)$. 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, first row of Table 1).

In a CA, if $\mathcal{R}_i(x_{i-1}x_ix_{i+1}) = x_i$, the corresponding RMT of rule $\mathcal{R}_i$ is called a *self-replicating* RMT. If all RMTs of a configuration x is 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 – number of cycles and lengths of cycles. Let $C_i \subseteq C = \{0, 1\}^n$ be a set of configurations such that $G^l(\mathtt{x}) = \mathtt{x}$, $\forall \mathtt{x} \in C_i$, where $l \in \mathbb{N}$ and $|C_i| = l$. Then, all configurations $\mathtt{x} \in C_i$ are *cyclic* and *reachable* from each other. The set of l configurations which forms a single cycle is named as a *cycle space*. So, C_i is a cycle space of the CA. For instance, in Figure 1, the configurations 1000, 0111 and 0001 are reachable from one another as they form a cycle space of length 3. A CA is called *reversible*, if all configurations are part of some cycle.

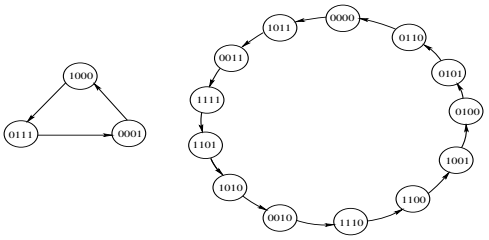

Fig. 1: Transition diagram of the 4-cell reversible CA $(9, 150, 45, 65)$.

Figure 1 represents a reversible CA. Therefore, configuration space of a CA can be represented as a collection of cycle spaces.

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

(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.

To synthesize an n-cell reversible CAs, we use the methodology described in [4]. 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 (see 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 [4] for more details). The following example illustrates the process of synthesis.

Example 1. Let us design a 4-cell reversible CA. To select an arbitrary $\mathcal{R}_0$, the first column of Table 2(b) is taken into consideration. Let rule 9 be selected as $\mathcal{R}_0$ from Table 2(b) (the third row and first column of Table 2(b)). As the class information of the next rule of rule 9 is class III (the second column and third row of Table 2(b)), therefore, $\mathcal{R}_1$ is to be anyone from the pool of CA rules of class III from Table 2(a). Let rule 150 be chosen as $\mathcal{R}_1$ (the first column of Table 2(a)

for rule 150 is class III). Now, the third column and the row corresponding to rule 150 in Table 2(a) is class II, therefore, $\mathcal{R}_2$ is to be selected from class II from Table 2(a). By repeating the same process, let us choose $\mathcal{R}_2$ as 45. Therefore, the class information for rule $\mathcal{R}_3$ is class I. However, as, this is the last rule, we need to select this rule from Table 2(c). Let $\mathcal{R}_3$ is 65 (second column and first row of Table 2(c)). Therefore, the reversible CA is $(9, 150, 45, 65)$ (see Figure 1).

3 The Mapping between CA and Clustering

This section introduces how an n-cell reversible CA can be used for clustering problem with k target objects having p distinct features. As binary CA is to be used as a tool, each target object needs to be mapped to a *configuration*. To do that, *data discretization* should be done *effectively*.

3.1 The encoding technique

Let $\mathbb{X} = \{\mathtt{X_1}, \mathtt{X_2}, \cdots, \mathtt{X_k}\}$ be the set of target objects which are to be distributed among m clusters and $A_1, A_2, \cdots, A_p$ are p distinct attributes (features) of the objects where each A_j is a quantifiable property. This A_j is a finite set depicting the range of values of each feature has. To measure closeness among the target objects using hamming distance [12], each object ($\mathtt{X}$) is converted into a binary string $\mathtt{x}$ (where $\mathtt{x} \in \{0,1\}^n$ and $n \in \mathbb{N}$). Now, let $\mathtt{X} = (x^1 x^2 \cdots x^p)$ where $x^j \in A_j$.

If A_j is *continuous* attribute, the range can be partitioned into v disjoint subsets $A_{j1}, A_{j2}, \cdots, A_{jv}$ such that $A_{j1} \cap A_{j2} \cap \cdots \cap A_{jv} = \emptyset$. Using *frequency based* encoding [5] for each A_j, let the encoding function be $\mathtt{M} : A_j \mapsto {A_j}^1$, where ${A_j}^1$ is a finite set of discretized elements. Here, $\mathtt{M}$ is *surjective* and $|A_j| \ll |A_j^1|$. As we use binary CA, therefore, ${A_j}^1 \in \{0,1\}^i$ (where $i > 1$). In this work, we consider $v = 3$, that is, three intervals, so, two bit representation is needed to maintain a *minimum* hamming distance. Therefore, ${A_j}^1 \subset \{0,1\}^2$; we use binary strings 00, 01 and 11 to refer the intervals $[a_1, a_s]$, $[a_{s+1}, a_w]$ and $[a_{w+1}, a_t]$ respectively, where $a_1 < a_s < a_{s+1} < a_w < a_{w+1} < a_t$. However, if A_j is *categorical* attribute, then the length of the substring for encoding each element of A_j is $\log_2 \mathtt{u}$ where $|A_j| = \mathtt{u}$ and each element of A_j is represented as a way where there is only one 1 at some unique position. Therefore, the hamming distance between any pair of elements for such qualitative A_j is always fixed i.e. *two*. At a glance, to convert a target object with p features (where p_{q_n} and p_{q_l} are the count of quantitative and qualitative attributes [10]) into a configuration of n-cell CA, the following mechanism is considered: For each quantitative attribute, as *two* bits are used, therefore, $n = (2 * p_{q_n}) + (\mathtt{u_1} + \mathtt{u_2} + \cdots + \mathtt{u_{p_{q_l}}})$. If all attributes are quantitative, then $n = 2 * p_{q_n}$.

Example 2. Let us 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 ten such objects into CA configurations. Here, the categorical attribute values are encoded as 01 (hard) or 10 (soft). Whereas, the continuous attribute values are divided into three subintervals to be represented by 00, 01 and 11 respectively. For example, values in

#Object	Continuous Attribute				Categorical Attribute		Encoded CA configuration
	#Pages	Encoding	Ratings	Encoding	Binding Type	Encoding	
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

Ratings are divided into sub-intervals $[2.6, 4]$, $[4.5, 7]$ and $[8, 9.5]$ depicted by 00, 01 and 11 respectively. Therefore, each object is mapped to a 6-bit string which can be shown as a configurations of a 6-cell CA.

Hence, using the encoding function $\mathtt{M}$, the set of target objects ($\mathbb{X}$) is mapped to the configurations $C = \{0, 1\}^n$ of an n-cell CA. This work also supports *hard clustering* [3, 10] where each target object is assigned to a fixed cluster. Next, the significance of CA rules in designing the desired clusters is depicted.

3.2 CA rules to maintain minimum intra-cluster distance

For our present objective, we need to select such rules where minimum changes occur during state transition. Ideally, it means all RMTs of any configuration is *self-replicating*. 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, this CA is not *effective* as number of clusters is 2^n. Therefore, we have to restrict on the count of self-replicating RMTs for any configuration. If CA size is n, it is expected that more number of cycles possess configuration pairs that maintains as minimum as possible hamming distances, which eventually leads to enrich preservation of less intra-cluster distance. To detect such *significant* rules, we rank each rule of Table 2 based on the number of self replicating RMTs it possesses when used in designing the rule vector of a reversible CA. If all 8 RMTs (resp. the 4 valid RMTs, if used in cell 0 or cell $n-1$) are self-replicating, the rule is *ranked* 1. Similarly, a CA rule is *ranked* 2, 3, 4 or 5 depending on whether it possess 6, 4 (resp. 2 for cells 0 and $n-1$), 2 or 0 self-replicating RMTs respectively. Table 4 shows this ranking. This *rank* determines how a rule can act as an influencing factor for designing a cluster with more similar (less hamming distance) data.

Evidently, rule 204 is ranked *first* (8 self-replicating RMTs), but as it is already mentioned, if rule 204 is applied to every cell of an n-cell CA, then each target object belongs to an unique cycle which is not desirable for clustering. Similarly, rule 51 is the *least significant* rule for clustering (no self-replicating RMTs), as, for an n-cell CA with rule 51 as the only rule, each cluster consists of a pair of configuration with hamming distance n. Therefore, we need to select rule 204 for as many cells as possible and just 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 less intra-cluster distance, we take the following strategy of choosing rules for synthesizing a reversible CA:

1. *Discard all rules with Rank* 4 *and* 5 *(that is, less than* 4 *self-replicating RMTs) from Table 2.* `Seventeen` *rules (51, 53, 58, 83, 163, 54, 57, 99, 147, 23, 43, 113, 178, 27, 39, 114, 177) are discarded by this condition. Therefore, currently, the rule space is reduced to* 45*.*
2. *For the* $n-2$ *non-terminal cell positions (cell* 1 *to cell* $n-2$*), at most 50% rules with* rank 2 *(*6 *self-replicating RMTs) are to be selected.*

If we process in this way, the configurations with lesser hamming distances are placed on the same cycle. However, this scheme can not restrict the number of cycles. Next, we focus on the design of CA with limited number of cycles.

3.3 Designing CA with optimal number of clusters

From the above discussion, it is obvious that, the consecutive configurations of a cycle maintain minimum distance in feature space if *more* significant rules are used in an n-cell CA. That is, the same cycle connects alike objects. However, it may increase the number of cycles. So, there is a trade-off between these two aspects of clustering technique - maintaining less number of cycles (clusters) and less intra-cluster distance among the objects, that is, configurations with less hamming distances are in the same cycle. In this section, we discuss a technique to generate CAs with limited number of cycles, that is, more configurations are to be placed on the same cycle. This requirement matches with an existing problem statement, *generation of large cycle CA*, already studied in [1]. For ease of understanding, we briefly recall the idea.

Category	$\mathcal{R}_i$	Rank
Completely Dependent	90, 165, 150, 105	3
Partially Dependent	30, 45, 75, 120, 135, 180, 210, 225, 86, 89, 101, 106, 149, 154, 166, 169	3
Weakly Dependent	92,172, 197, 202, 108, 156, 198, 201, 77, 142, 212, 232, 78, 141, 216, 228,	2
	53, 58, 83, 163, 54, 57, 99, 147, 23, 43, 113, 178, 27, 39, 114, 177	4
Independent	51,	5
	85, 170, 102, 153, 60, 195, 15, 240	3
	204	1

(a) Categories of reversible CA rules

Category	$\mathcal{R}_0$	Rank	$\mathcal{R}_{n-1}$	Rank
Completely Dependent	5, 6, 9, 10	3	5, 20, 65, 80	3
Independent	3, 12	5 1	17, 68	5 1

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

Table 4: Categories of reversible CA rules on the parameter `P`.

A CA is expected to have a cycle of large length, if its rules are dependent on both of the left and right neighbors. To measure this dependence, a *parameter* (`P`), called *degree of dependence on both the neighbors*, is defined which determines how much a cell is dependent on its neighbors for updating its state. For a rule $\mathcal{R}_i$, $\mathtt{P}(\mathcal{R}_i) = \mathtt{P_r}(\mathcal{R}_i) * \mathtt{P_l}(\mathcal{R}_i)$. Here, $\mathtt{P_r}(\mathcal{R}_i)$ (resp. $\mathtt{P_l}(\mathcal{R}_i)$) 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, $\mathtt{P}(\mathcal{R}_i)$ 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* ($\mathtt{P} = 1$), *partially dependent* ($\mathtt{P} = 0.5$), *weakly dependent* ($\mathtt{P} = 0.25$) and *independent* ($\mathtt{P} = 0$) (see Table 4). It is observed that in a CA with large cycle(s), majority of the participating rules are from the *completely dependent* category, some are from the *partially dependent* category and a few are from the category of *weakly dependent*, whereas, none are from the *Independent* category. For more detailed discussion, please see [1].

Obviously, following the strategy mentioned in Section 3.2, sixteen rules from *weakly dependent* category and all rules from *independent* categories are rejected for clustering purpose as they produce more pair of configurations with high hamming distances. The remaining sixteen rules of *weakly dependent* category have *rank* 2, whereas, the rules of *completely dependent* and *partially dependent* categories have *rank* 3. In the following section, our proposed clustering technique, cycle based clustering using these CAs, is described in detail.

4 Cycle based clustering

Generating an n-cell CA with large number of cycles having less intra-cluster distance is itself a very challenging problem. Moreover, even if desired number of clusters are generated, then also it is difficult to ascertain that the produced clusters are *good* for the given dataset. To deal with this issue, this section describes a CAs based iterative algorithm which efficiently generates *desired* number of *good* clusters.

In this algorithm, more than one CA can participate in designing the clusters for a given dataset. However, any arbitrary CA is not *acceptable* as candidate; such CAs need to maintain the necessary conditions described in Section 3. Therefore, in our CA based clustering approach, the following characteristics are to be maintained:

- `Property 1:` Participating CA of size n maintains 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 $\mathtt{M}(\mathbb{X})$ be the set of *encoded* target objects where $\mathtt{M}(\mathbb{X}) \subset C$ and $\mathbb{X} = \{\mathtt{X_1}, \mathtt{X_2}, \cdots, \mathtt{X_k}\}$ is the set of target objects. These encoded target objects are named as *target configurations*. Let $|\mathtt{M}(\mathbb{X})| = k'$ where $k' \leq k$. At any level, a target configuration $\mathtt{x} \in \mathtt{M}(\mathbb{X})$ is member of a distinct cluster $\mathtt{c}$ (a set of encoded target objects) such that $|\bigcup c| = |\mathtt{M}(\mathbb{X})|$.

Let m_i be the number of *primary* clusters at level i. For level 0, the primary clusters are $\mathtt{c_1}^0, \mathtt{c_2}^0, \cdots, \mathtt{c_{m_0}}^0$, where each cluster is a singleton set. Therefore, $k' = m_0$. In general, for any level i, the primary clusters are $\mathtt{c_1}^i, \mathtt{c_2}^i, \cdots, \mathtt{c_{m_i}}^i$. 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 as an *auxiliary* CA. This CA plays a major role in clustering. Firstly, it is needed to compute the number of *auxiliary clusters* of such a CA, in which the *target* configurations (k') *strictly belong to.*

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

Let the target configurations *strictly belong* to m' number of *auxiliary* clusters $C_1^i, C_2^i, \cdots, C_{m'}^i$ of level i. Our *second* step is to follow `property 2`, that is, *merging* the primary clusters ${\mathtt{c_1}}^{i-1}, {\mathtt{c_2}}^{i-1}, \cdots, {\mathtt{c_{m_0}}}^{i-1}$ 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 can not be merged arbitrarily; a pair of primary clusters can be merged depending on their *degree of membership of participation.*

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

The configurations of ${\mathtt{c_j}}^{i-1}$ can *strictly belong to* more than one auxiliary cluster. Similarly, C_t^i can possess target configurations from different clusters of level $i-1$. Let ${\mathtt{c_1}}^{i-1}$ and ${\mathtt{c_j}}^{i-1}$ 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 ${\mathtt{c_j}}^{i-1}$, ${\mathtt{c_1}}^{i-1}$ and ${\mathtt{c_s}}^{i-1}$ be the clusters whose members strictly belong to C_t^i. Now, clusters ${\mathtt{c_j}}^{i-1}$ and ${\mathtt{c_1}}^{i-1}$ are said to be* closely reachable *in C_t^i if $|(\mu(C_t^i, {\mathtt{c_j}}^{i-1}) - \mu(C_t^i, {\mathtt{c_1}}^{i-1}))| < |(\mu(C_t^i, {\mathtt{c_j}}^{i-1}) - (\mu(C_t^i, {\mathtt{c_s}}^{i-1}))|$.*

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 ${\mathtt{c_1}}^0, {\mathtt{c_2}}^0, \cdots, {\mathtt{c_{m_0}}}^0$ (resp. ${\mathtt{c_1}}^{i-1}, {\mathtt{c_2}}^{i-1}, \cdots, {\mathtt{c_{m_{i-1}}}}^{i-1}$) be the primary clusters of level 0 (resp. $i-1$) where the count of clusters is m_0 (resp. m_{i-1}). Also let $C_1^1, C_2^1, \cdots, C_{m'}^1$ (resp. $C_1^i, C_2^i, \cdots, C_{m'}^i$) 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 $\mu(C_t^1, {\mathtt{c_j}}^0)$ (resp. $\mu(C_t^i, {\mathtt{c_j}}^{i-1})$). For any given ${\mathtt{c_j}}^0$ (resp. ${\mathtt{c_j}}^{i-1}$), 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 ${\mathtt{c_j}}^0$ (resp. ${\mathtt{c_j}}^{i-1}$) 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 from ${\mathtt{c_j}}^0$ (resp. ${\mathtt{c_j}}^{i-1}$). Therefore, ${\mathtt{c_j}}^0$ (resp. ${\mathtt{c_j}}^{i-1}$) can merge with some of the clusters which have also participated in $C_{t_1}^1$ (resp. $C_{t_1}^i$). However, only those clusters are to be merged with ${\mathtt{c_j}}^0$ (resp. ${\mathtt{c_j}}^{i-1}$), which are closely reachable to ${\mathtt{c_j}}^0$ (resp. ${\mathtt{c_j}}^{i-1}$). Hence, a new primary cluster ${\mathtt{c_j}}^1$ (resp. ${\mathtt{c_j}}^i$) is formed as ${\mathtt{c_j}}^i = {\mathtt{c_j}}^{i-1} \cup {\mathtt{c_1}}^{i-1}$ if and only if $|(\mu(C_{t_1}^i, {\mathtt{c_j}}^{i-1}) - \mu(C_{t_1}^i, {\mathtt{c_1}}^{i-1}))| <$

$|(\mu(C^i_{t_1}, \mathtt{c_j}^{i-1}) - \mu(C^i_{t_1}, \mathtt{c_s}^{i-1}))|$, for any $s \neq l$ where $\mathtt{c_s}^{i-1}$ is another participating cluster in $C^i_{t_1}$ and $max\{\mu(C^i_t, \mathtt{c_j}^{i-1})_{(\forall t)}\} = \mu(C^i_{t_1}, \mathtt{c_j}^{i-1})$. Therefore, the newly generated cluster $\mathtt{c_j}^i$ constitutes of a set of target configurations, out of which some strictly belongs to a cluster (cycle) of the auxiliary CA of level i. This instigates to name our approach as *cycle based clustering*.

3. If for any primary cluster $\mathtt{c_j}^0$ (resp. $\mathtt{c_j}^{i-1}$), there is no closely reachable primary cluster in all auxiliary clusters, then the new primary cluster of level i is $\mathtt{c_j}^i = \mathtt{c_j}^{i-1}$. 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 user or if $m_i = m_{i-1}$ after a fixed number of attempts.

Algorithm 1: Cycle based clustering algorithm

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

Step 1 Set $n \leftarrow (2 * p_{q_n}) + (\mathtt{u_1} + \mathtt{u_2} + \cdots + \mathtt{u_{pq_l}})$;
foreach $j = 1$ *to* k **do** *Encode* $\mathtt{X_j}$ into an n-bit binary string ;
Let $\mathtt{M}(\mathbb{X})$ be the set of *encoded* target objects $\{\mathtt{x_1}, \mathtt{x_2}, \cdots, \mathtt{x_{k'}}\}$ where $|\mathtt{M}(\mathbb{X})| = k'$;
Step 2 Construct a set of n-cell CAs $\mathtt{R}$ from the given auxiliary CA space ;
Step 3 Set $m_0 \leftarrow k'$, $i \leftarrow 1$ and $\mathtt{z} \leftarrow 1$;
for $j = 1$ *to* m_0 **do**
  Set $\mathtt{c_j}^0 \leftarrow \{\mathtt{x_j}\}$; `// Initialize primary clusters of level 0`
Step 4 **while** $(m_i \neq m_{i-1}) || (m_i \neq m)$ **do**
  Select $\mathcal{R} \in \mathtt{R}$ and Set $\mathtt{R} \leftarrow \mathtt{R} \setminus \{\mathcal{R}\}$ `// Auxiliary CA is selected randomly at uniform without replacement`
  Generate auxiliary clusters $C^i_1, C^i_2, \cdots, C^i_{m'}$ 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 $a_{tj} \leftarrow \mu(C^i_t, \mathtt{c_j}^{i-1})$;
  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` $\mathtt{c_j}^{i-1}$
    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 $\mathtt{c_z}^i \leftarrow \mathtt{c_j}^{i-1} \cup \mathtt{c_{j'}}^{i-1}$ and $\mathtt{z} \leftarrow \mathtt{z} + 1$;
      Mark $\mathtt{c_j}^{i-1}$ and $\mathtt{c_{j'}}^{i-1}$ as *modified* ;
    Remove row t' from A;
  foreach unmodified *clusters* $\mathtt{c_y}^{i-1}$ **do**
    Set $\mathtt{c_z}^i \leftarrow \mathtt{c_y}^{i-1}$ and $\mathtt{z} \leftarrow \mathtt{z} + 1$; `// move the unmodified primary cluster(s) of previous level` i`1 to get a new primary cluster of level` i `and update cluster number`
  Set $m_i \leftarrow z$ and $\mathtt{i} \leftarrow \mathtt{i} + 1$;
Step 5 Report $\mathtt{c_1}^i$, $\mathtt{c_2}^i$, $\cdots$, $\mathtt{c_{m_i}}^i$ as the final clusters at level i and *Exit* ;

Example 3. Let us consider the `Iris` dataset (`http://archive.ics.uci.edu/ml/index.php`, see Table 5) where $\mathbb{X} = \{\mathtt{X_1}, \mathtt{X_2}, \cdots, \mathtt{X_{150}}\}$ and each object has four quantitative (p_{q_n}) and no qualitative attributes (p_{q_l}). Hence, size of the CA

Name	# of p	# of p_{q_n}	# of p_{q_l}	# of target objects	CA size (n)
Iris	4	4	0	150	8
BuddyMove	6	6	0	249	12
Wholesale Customers	8	6	2	440	16
Seed	7	7	0	210	14

Table 5: Description of real datasets used for cycle based clustering.

for this data set is $n = 2 * 4 = 8$. Now, let the desired number of clusters (m) is `two`. Using the encoding technique, we get $\mathtt{M}(\mathbb{X}) = \{\mathtt{x}_1, \mathtt{x}_2, \cdots, \mathtt{x}_{24}\}$.

Initially, at level 0, there exist `twenty four` primary clusters such that ${\mathtt{c}_1}^0 = \{\mathtt{x}_1\}, {\mathtt{c}_2}^0 = \{\mathtt{x}_2\}, \cdots, {\mathtt{c}_{24}}^0 = \{\mathtt{x}_{24}\}$, where $m_0 = 24$. As $m_0 \neq m$, 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 - $C_1^0, C_2^0, C_3^0, C_4^0$. Next, we find the degree of participation of each ${\mathtt{c}_\mathtt{j}}^0$ in these clusters. As we are at level 1, $\mu(C_1^1, {\mathtt{c}_{\mathtt{j}_1}}^0) = 100\%$ $(\forall j_1 \in \{1, 3, 4, 22, 13, 14\})$, $\mu(C_2^1, {\mathtt{c}_{\mathtt{j}_2}}^0) = 100\%$ $(\forall j_2 \in \{2, 5, 6, 7, 8, 9, 10, 11, 12, 15, 17, 19, 20, 21, 23, 24\})$, $\mu(C_3^1, {\mathtt{c}_{16}}^0) = 100\%$ and $\mu(C_4^1, {\mathtt{c}_{18}}^0) = 100\%$. 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 ${\mathtt{c}_1}^0$, ${\mathtt{c}_3}^0$, ${\mathtt{c}_4}^0$, ${\mathtt{c}_{22}}^0$, ${\mathtt{c}_{13}}^0$ and ${\mathtt{c}_{14}}^0$. Similarly, C_2^1 has maximum participation of ${\mathtt{c}_2}^0$, ${\mathtt{c}_5}^0$, ${\mathtt{c}_6}^0$, ${\mathtt{c}_7}^0$, ${\mathtt{c}_8}^0$, ${\mathtt{c}_9}^0$, ${\mathtt{c}_{10}}^0$, ${\mathtt{c}_{11}}^0$, ${\mathtt{c}_{12}}^0$, ${\mathtt{c}_{15}}^0$, ${\mathtt{c}_{17}}^0$, ${\mathtt{c}_{19}}^0$, ${\mathtt{c}_{20}}^0$, ${\mathtt{c}_{21}}^0$, ${\mathtt{c}_{23}}^0$ and ${\mathtt{c}_{24}}^0$. Therefore, the newly generated primary cluster of level 1 is ${\mathtt{c}_1}^1 = {\mathtt{c}_1}^0 \cup {\mathtt{c}_3}^0 \cup {\mathtt{c}_4}^0 \cup {\mathtt{c}_{22}}^0 \cup {\mathtt{c}_{13}}^0 \cup {\mathtt{c}_{14}}^0$. Similarly, ${\mathtt{c}_2}^1$ can be generated. For the remaining two auxiliary clusters, new primary clusters are formed as ${\mathtt{c}_3}^1 = {\mathtt{c}_{16}}^0$ and ${\mathtt{c}_4}^1 = {\mathtt{c}_{18}}^0$. As, the number of primary clusters at level 1 (m_1) is $4 \neq m$, we move from level 1 to level 2.

Let, at level 2, the selected auxiliary CA is $(6, 232, 90, 90, 165, 90, 90, 20)$. 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 also, we compute the maximum participation of each primary cluster of level 1 in auxiliary cluster C_t^2, $(1 \leq t \leq 6)$. It is found that $\mu(C_1^2, {\mathtt{c}_1}^1) = 16\%$, $\mu(C_2^2, {\mathtt{c}_1}^1) = 33\%$, $\mu(C_2^2, {\mathtt{c}_2}^1) = 62\%$, $\mu(C_2^2, {\mathtt{c}_3}^1) = 100\%$, $\mu(C_3^2, {\mathtt{c}_1}^1) = 16\%$, $\mu(C_3^2, {\mathtt{c}_2}^1) = 12\%$, $\mu(C_3^2, {\mathtt{c}_4}^1) = 100\%$, $\mu(C_4^2, {\mathtt{c}_1}^1) = 33\%$, $\mu(C_5^2, {\mathtt{c}_2}^1) = 18\%$ and $\mu(C_6^2, {\mathtt{c}_2}^1) = 6\%$. Hence, we can merge ${\mathtt{c}_2}^1$ and ${\mathtt{c}_3}^1$ with respect to the closeness in the auxiliary cluster C_2^2. Similarly, ${\mathtt{c}_1}^1$ and ${\mathtt{c}_4}^1$ can be merged with respect C_3^2. Hence, the newly generated primary clusters of level 2 are ${\mathtt{c}_1}^2 = {\mathtt{c}_2}^1 \cup {\mathtt{c}_3}^1 = \{2, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 17, 19, 20, 21, 23, 24\}$ and ${\mathtt{c}_2}^2 = {\mathtt{c}_1}^1 \cup {\mathtt{c}_4}^1 = \{1, 3, 4, 22, 13, 14, 18\}$. Therefore, $m_2 = 2$. As desired number of clusters is achieved at level 2, the algorithm exists.

Hence, our algorithm can not only generate the requirement based clusters, but also it gives the direction of optimal count of the clusters. Our technique uses v auxiliary CAs if the optimal number of clusters is achieved at level v. Next, we test the competency of our proposed algorithm on real dataset.

5 Results and Discussion

This section reports the performance of our proposed cycle based clustering algorithm on some real datasets (`http://archive.ics.uci.edu/ml/index.php`)

using some benchmarks validation techniques [8, 12]. Here, we use *four* datasets **Iris, BuddyMove, Wholesale Customers, Seed** where each of them has mostly *quantitative* attributes (see Table 5 for details).

Dataset	Algorithm	Connectivity		Dunn Index		Silhouette Score	
		Score	m	Score	m	Score	m
Iris	Hierarchical	0.0000	2	0.3389	2	0.6867	2
	K-means	6.1536	2	0.1365	4	0.6810	2
	DIANA	6.1536	2	0.1302	5	0.6810	2
	PAM	3.9623	2	0.1235	5	0.6858	2
	SOTA	11.5016	2	0.0582	5	0.6569	2
	Proposed Algorithm	**0.0000**	**2**	**0.3389**	**2**	**0.6867**	**2**
BuddyMove	Hierarchical	2.9290	2	0.3146	2	0.4764	2
	K-means	30.0881	2	0.0193	5	0.3492	3
	DIANA	27.1242	2	0.0476	6	0.3020	2
	PAM	41.6647	2	0.0178	3	0.3819	3
	SOTA	44.4111	2	0.0666	3	0.3134	2
	Proposed Algorithm	**2.9289**	**2**	**0.3146**	**2**	**0.4763**	**2**
Wholesale Customers	Hierarchical	2.9290	2	0.3853	2	0.7957	2
	K-means	35.2032	2	0.0900	6	0.3492	3
	DIANA	34.3694	2	0.0870	6	0.3020	2
	PAM	38.3802	2	0.0511	5	0.3310	4
	SOTA	43.8266	2	0.0061	2	0.3134	2
	Proposed Algorithm	**3.7329**	**2**	**0.0508**	**2**	**0.5257**	**2**
Seed	Hierarchical	8.7861	2	0.1089	6	0.5248	2
	K-means	21.3698	2	0.0855	3	0.5229	2
	DIANA	19.1714	2	0.0743	6	0.5218	2
	PAM	20.6762	2	0.0788	5	0.5175	2
	SOTA	16.0179	2	0.0566	4	0.5049	2
	Proposed Algorithm	**3.6**	**2**	**0.09**	**2**	**0.5288**	**2**

Table 6: Comparison of clustering techniques based on internal validation indices for each of the available datasets of Table 5.

To measure the performance of our proposed cycle based clustering algorithm on these datasets, we use the package **clValid** in **R**, using the implementation available in the package [2]. Table 6 records the performance of this algorithm. In Table 6, columns III & IV, V & VI and VII & VIII represent the optimal scores and the optimal number of clusters (m) of the validation indices - *connectivity* [9], *silhouette score* [7] and *Dunn index* [6] respectively. Our algorithm gives the optimal score in each of the internal validation indices for the *Iris* dataset in just 2 levels using the CA (10, 75, 166, 105, 105, 166, 150, 20) for level 0 and CA (6, 166, 165, 154, 105, 165, 165, 65) for level 1. Similarly, the optimal results for the *BuddyMove* dataset is found in 4 levels. For *Wholesale* dataset, the optimal result on connectivity, Dunn index and silhouette score are observed in 2, 5 and 3 levels respectively. Whereas, for the *Seed* dataset, the optimal score is recorded in just 1 level for Dunn index, whereas in 3 and 7 levels for silhouette score and connectivity respectively.

To compare the performance of our algorithm, these datasets are tested on five benchmark clustering algorithms – *K-means* (centroid based clustering) [2], *hierarchical* (agglomerative hierarchical clustering) [2], *DIANA* (divisive hierarchical clustering) [2], *PAM (Partitioning around medoids)* (centroid based clustering) [2] and *SOTA (Self-organizing tree algorithm)* (unsupervised network

with a divisive hierarchical clustering) [2] using the implementation in **R** [2]. Table 6 also reports result of this comparison. It can be observed that, among the existing algorithms, performance of *hierarchical* algorithm is best with respect to all datasets. However, our algorithm performs at par with this algorithm, and even beats it for the **Seed** dataset. Therefore, from this table, it can be concluded that our algorithm can easily compete with the *highly* efficient benchmark algorithms and performance of our proposed cycle based clustering algorithm is at least as good that of the best algorithm existing today.

References

1. S. Adak, S. Mukherjee, and S. Das. Do There Exist Non-linear Maximal Length Cellular Automata? A Study. In *Proceedings of* 13^{th} *International Conference on Cellular Automata for Research and Industry (ACRI)*, pages 289–297, 2018.
2. G. Brock, V. Pihur, S. Datta, and S. Datta. clValid: An R Package for Cluster Validation. *Journal of Statistical Software*, 25(4):1–22, 2008.
3. F. Carvalho, Y. Lechevallier, and F. Melo. Partitioning hard clustering algorithms based on multiple dissimilarity matrices. *Pattern Recognition*, 45(1):447–464, 2012.
4. S. Das. *Theory and Applications of Nonlinear Cellular Automata In VLSI Design.* PhD thesis, Bengal Engineering And Science University, Shibpur, India, 2006.
5. J. Dougherty, R. Kohavi, and M. Sahami. Supervised and Unsupervised Discretization of Continuous Features. In *Machine Learning Proceedings 1995*, pages 194–202. Elsevier, 1995.
6. J. C. Dunn. Well Separated Clusters and Fuzzy Partitions. *Journal on Cybernetics*, 4:95–104, 1974.
7. J. C. Dunn. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.
8. V. Estivill-Castro. Why so many clustering algorithms – a position paper. *ACM SIGKDD Explorations Newsletter*, 4, 2002.
9. J. Handl, J. Knowles, and D. B. Kell. Computational Cluster Validation in Post-Genomic Data Analysis. *Bioinformatics*, 21(15):3201–3212, 2005.
10. A.K. Jain, M.N Murthy, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):165–193, 1999.
11. S. Wolfram. *Theory and applications of cellular automata.* World Scientific, Singapore, 1986. ISBN 9971-50-124-4 pbk.
12. D. Xu and Y. A Tian. A Comprehensive Survey of Clustering Algorithms. *Annals of Data science*, 2:165–193, 2015.
13. R. Xu and D. Wunsch. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.