A hybrid clustering technique combining a novel genetic algorithm with K-Means

Md Anisur Rahman, Md Zahidul Islam *

Centre for Research in Complex Systems, School of Computing and Mathematics, Charles Sturt University, Australia

1. Introduction

Clustering is a process of grouping similar records in a cluster and dissimilar records in different clusters. It has a wide range of applications including social network analysis, software engineering, and crime detection. There are many existing clustering algorithms, out of which K-Means and Fuzzy C-Means are two commonly used techniques perhaps because of their simplicity. However, they require a user to provide the number of clusters in a data set [1–6]. Based on this user defined value, K-Means randomly selects k number of initial seeds (cluster centers) from a data set. However, in reality it can be difficult for a user to guess the number of clusters in a data set [1,7]. Therefore, clustering techniques that are capable of automatically selecting the number of clusters are highly desirable.

Moreover, K-Means is generally very sensitive to the quality of initial seeds. They can produce poor quality clustering results due to the poor quality of initial seeds [5,8,9]. Several techniques [10,11] have been proposed for finding high quality initial seeds. These techniques generally produce better clustering results than K-Means that chooses initial seeds randomly.

Additionally, K-Means generally gets stuck at local minima resulting in poor clustering results [12–15]. Another drawback of K-Means is that it does not record the quality of clusters obtained in the previous iterations and therefore, does not take advantage of good quality clusters of the previous iterations [16]. Generally the use of genetic algorithms in K-Means improves the quality of the clusters [16]. The use of genetic algorithms with K-Means can also help to avoid local minima issue of K-Means [13,14,16,17]. Typically, a genetic algorithm based clustering technique does not require a user input on the number of clusters [16]. For example, AGCUK obtains the number of clusters automatically through the clustering process [18].

AGCUK [18] randomly selects the genes and the number of genes in a chromosome, for the initial population having a user defined number of chromosomes. Each chromosome is made of two or more genes, where each gene is a cluster center (i.e. a record at the center of a cluster) and a chromosome is a clustering solution. Due to the random selection of the genes and the number of genes, chromosomes in an initial population may fail to capture the genes representing all clusters. As a result of not having the appropriate genes in the initial population the cluster quality of the final solution may not be good. This is further explained in Section 3.1.1.
Another limitation of the technique is that it does not re-arrange the genes before the crossover operation. For a single point crossover, if the genes are not properly rearranged then we may end up having useless offspring chromosomes; each having a concentration of genes representing only a subset of all clusters as explained in Section 3.1.3. GAGR [19] performs a gene rearrangement in a chromosome in order to produce better quality offspring chromosomes. However, the gene rearrangement operation used in GAGR requires the same size (i.e. the same number of genes) for both chromosomes involved in a crossover operation. As part of our novel clustering technique, we propose a gene rearrangement operation that can handle chromosomes having different sizes.

The main contributions of our novel clustering technique called GenClust, that combines our proposed genetic algorithm (GA) with K-Means, are as follows. Our technique uses a set of systematically selected chromosomes in the initial population for capturing the clusters of different size and shape. In Section 3.1.1 we present an analysis to demonstrate the advantage of the initial population selection technique and the use of the selected initial population in a GA. Unlike some existing genetic algorithms such as AGCUK [18] and GAGR [19] that select the initial genes randomly our proposed genetic algorithm finds genes both deterministically (to support having good quality genes in the initial population) and randomly (to support the exploration of ultimately good quality genes). The mixture of deterministic and randomly chosen genes is expected to be more exploratory than just deterministically chosen initial seeds in a non-genetic algorithm like CRUDAW [10]. Another main contribution of the paper is the introduction of a novel fitness function and cluster evaluation technique. The fitness function requires a low complexity due to using the distance between the cluster centers, unlike Silhouette coefficient [20] that uses the distances between the records of each pair, where one record of the pair belongs to one cluster and the other record belongs to another cluster.

Our technique also uses a novel gene rearrangement operation before a crossover. The proposed gene rearrangement algorithm (See Algorithm 1) can handle the case when two chromosomes participating in a crossover operation do not have the equal number of genes, whereas some existing techniques [19] can only handle two equal length chromosomes. Another contribution is the twin removal which eliminates a gene that is exactly same (or very similar) as another gene of the same chromosome. We argue that the existence of twins in a chromosome may reduce the diversity of the genes and unnecessarily increase complexity.

Besides, the technique integrates GA and K-Means to produce high quality clustering results. K-Means is a fast clustering algorithm which is also very commonly used [5,15]. Two major issues with K-Means are the random seed selection and the requirement of the user input on the number of clusters. Our proposed technique overcomes the issues with K-Means by using our GA that automatically selects high quality initial seeds without requiring the user input on the number of clusters. Moreover, the combination of GA and K-Means can also overcome the local minima issue of K-Means [13,14,16,17]. Since GA is less likely to get stuck at local minima it is more likely to find a rough set of clusters (close to global minima) which can then be fine tuned by K-Means [16].

Unlike most (if not all) of the existing genetic algorithms [18,19,21,22], GenClust can handle both numerical and categorical attributes. It estimates the similarity of categorical values instead of considering the similarity to be either zero (if the values are different) or 1 (if the values are same) [10,23,24]. GenClust has three main steps: 1. the systematic selection of the initial population, 2. the determination of the cluster centers through a genetic algorithm, and 3. the use of the cluster centers as initial seeds of the K-Means algorithm to finally produce the clusters.

We compare GenClust with CRUDAW [10], AGCUK [18], GAGR [19], GFCM [4] and SABC [3] on twenty (20) natural data sets for six (three internal and three external) cluster evaluation criteria namely Xie-Beni Index, SSE, our novel cluster evaluation criteria (COSEC), F-Measure, Entropy, and Purity. GenClust performs better than all five (5) existing techniques in all twenty (20) data sets for all six (6) evaluation criteria. Sign tests are carried out on the cluster evaluation results in order to verify the statistical significance of the quality improvement made by GenClust over the five existing techniques. The sign test results indicate that GenClust achieves significantly better quality clusters than the five existing techniques. We also present a complexity analysis of the techniques.

Therefore, the main contributions of the proposed technique can be summarized as follows.

- The selection of the initial population combining the deterministic and randomly chosen chromosomes.
- The gene rearrangement technique.
- The twin removal operation.
- The new fitness function.
- Our proposed technique works on a data set that has both categorical and numerical attributes.

In Section 2, we present literature of review. In Section 3, we present our novel clustering technique. Experimental results are presented in Section 4. We give the concluding remarks in Section 5.

2. Literature review

Simple K-Means (SK) requires a user input on the number of clusters (k). It then randomly selects k number of records from a data set and considers them as the initial seeds [1,12,20,24,25]. Each record of the data set is then assigned to the seed which the record has the minimum distance with. Therefore, the records of the data set are partitioned into as many mutually exclusive clusters as the number of seeds. A new seed is then computed for each partition (i.e. cluster), where a seed represents the records belonging to the partition. The value of an attribute of the seed is calculated by taking the average value of the attribute for all records belonging to the seed. All attributes are generally assumed to be numerical in the basic form of K-Means [20,22,24,26].

Once the set of new seeds is determined, the partitions (clusters) are re-computed by re-assigning the records to the seeds. A record is assigned to the seed which the record has the minimum distance with. The process of partitioning the records and seed calculation continues until the termination conditions are satisfied. Typically, a maximum number of iterations is considered as a termination condition. Additionally, a minimum absolute difference between the values of the objective function in two consecutive iterations is considered as another termination condition.

Although the very basic form of K-Means can handle only numerical attributes, there are many further developments [3,11,24,26], where they also handle categorical attributes. Sometimes, the value of a categorical attribute of a seed is computed by taking the most frequent value of the attribute among the records belonging to the partition [5,11,26]. SABC [3], which is a modified version of K-Means, can handle both numerical and categorical attributes. It uses a fuzzy seed where the fuzzy value of a categorical attribute in a seed is computed based on the frequency distribution of the categorical values of the attribute for all records belonging to the cluster. However, for a numerical attribute it uses the average value (as usual) of the attribute for all records belonging to the cluster.
While calculating the distance between a pair of records (or a record and a seed), SABC uses a weighted Euclidian distance for the numerical attributes and a weighted similarity based distance for the categorical attributes. It computes the weight of each attribute automatically (not user defined), unlike CRUDAW [10]. However, a limitation of SABC is its requirement for a user input on the number of clusters. Another limitation of SABC is the random selection of the initial seeds.

An existing technique [27] selects initial seeds deterministically based on the frequency scores of the records. The frequency score of a record is the average frequency of the categorical values of the attributes of the record, where the frequency of a categorical value is the number of records that contain the value for the attribute. The record having the maximum average frequency is considered as the first initial seed. A record is considered as the second seed, if the frequency of the record times the distance of the record with the first seed is greater than any other records that are not identified as a seed. Similarly, \( k \) initial seeds are selected. One limitation of the seed selection technique is that it requires a user to provide the number of clusters \( k \) as input, which may often be difficult for a user to predict. Moreover, the technique works on a data set that has only categorical attributes.

Like SABC, GFCM [4] also requires a user input on the number of seeds (clusters). It is a modified Fuzzy C-Means algorithm that randomly assigns a fuzzy membership degree for a pair consisting of a record and a seed (cluster). That is, for each record it assigns as many membership degrees as the user defined number of seeds; one membership degree for each seed. A membership degree represents the degree of association of a record with a seed and the sum of all membership degrees of a record is one.

Based on the first set of membership degrees and number of seeds it then calculates a set of seeds. GFCM works on mixed data sets. While calculating the seeds, the value of an attribute of a seed is calculated by considering the values of the attribute for all records (instead of the records belonging to the cluster only) and their membership degrees with the seed. While the seed contains a rigid value (single value) for a numerical attribute, it contains a fuzzy value for a categorical attribute. By a “fuzzy value”, it means that all domain values have different probabilities of being the value of the attribute.

GFCM is expected to obtain good quality clusters on the data sets that have fuzzy clusters, where a record may belong to more than one cluster with different probabilities for different clusters [4]. However, a limitation of GFCM is that (like SK and SABC) it also requires a user input on the number of clusters. Moreover, since GFCM randomly chooses the initial fuzzy membership degrees, a data miner may get different clustering solutions in different runs resulting in a similar problem to the case where initial seeds are selected randomly by SK and SABC.

CRUDAW [10] handles the problems caused by the random seed selection approach of SK, SABC, and GFCM by choosing the initial seeds through a deterministic process based on a user defined radius \( r_x \). It first chooses the record (of a data set), which has the densest region (the maximum number of records) within its \( r_x \) radius, as the 1st seed. It then removes the 1st seed and all records within its \( r_x \) radius. From the remaining set of records it then selects the record, which has the densest region within its \( r_x \) radius, as the 2nd seed. It continues the seed selection while the following two conditions are satisfied. Condition 1 is that there are user defined number of records in the data set even after the removal of the seeds and associated records (i.e. the records within their \( r_x \) distance). Condition 2 is that a seed has the user defined number of records within its \( r_x \) radius.

The initial seeds are then used to determine the initial fuzzy membership degrees for clustering. Although CRUDAW eliminates the issues related to the random seed selection and user input on the number of clusters, it still requires a user input on the initial cluster radius \( r_x \). The selection of \( r_x \) may impact heavily the final clustering quality. Moreover, it can be very difficult for a user to estimate a useful value for \( r_x \).

The requirement of a user input on the number of clusters or the radius of a cluster can be eliminated by using genetic algorithms for clustering. Generally, a genetic algorithm (GA) uses a random number \( (k) \) of clusters (not user defined) ranging between 2 to \( \sqrt{n} \) (\( n \) is the number of records) and thereby forms an initial clustering solution (called chromosome) having \( k \) seeds (called genes) [12,16,18]. It first creates a number of such chromosomes to form an initial population, which is also known as the 1st generation. A pair of the chromosomes of the 1st generation then participates in a crossover operation where the chromosomes exchange genes and thereby form a pair of new chromosomes [16,19,28]. All chromosomes of the 1st generation participate in the crossover operation, and a set of new chromosomes are generated.

Some of the genes of the new chromosomes are then randomly modified in the mutation operation in order to avoid a local optima [14–16,19,28]. Following the evolutionary theory of the “survival for the fittest” genetic algorithms use a fitness function in the elitism operation and finds the set of fit chromosomes for the 2nd generation. The crossover, mutation and elitism operations continue for a user defined number of iterations and finally using the fitness function the best chromosome is chosen as the clustering solution. An effective fitness function is crucial for the success of a genetic algorithm.

The application of genetic algorithm for clustering has a number of advantages over other non-genetic algorithms such as K-Means, SABC, GFCM and CRUDAW. First, it does not require a user defined number of clusters or radius of a cluster. Second, it avoids the dependency (to some extent) on a set of high quality initial seeds since it uses a big number of chromosomes in each generation. Third, it avoids getting stuck at local minima by using randomness in crossover and mutation operation, and therefore has higher chance of reaching the global minima.

The best chromosome obtained is generally a rough clustering solution [16] which is very close to the global minima. Therefore, using the genes of the best chromosome as the initial seeds of K-Means can produce a better clustering solution since K-Means can further fine tune the rough solution obtained by GA and get closer to the global solution [13,15,16,19].

However, there are some limitations of the existing genetic algorithms. The random selection of the genes in an initial population and the random number of genes in a chromosome of the initial population can result in the selection of a set of less useful genes and chromosomes. A selection of high quality chromosomes in the initial population can help a GA to finally find a better quality final solution. A combination of randomly chosen chromosomes (to support the exploration of a good solution) and deterministically chosen high quality chromosomes in the initial population can help a GA to finally obtain a better solution, as we now propose in Section 3 of this study.

Moreover, the issue of gene re-arrangement and twin removal is also crucial for a GA to finally find a good solution. Additionally, most (if not all) of the existing genetic algorithms only work on the data sets having just numerical attributes and no categorical attributes [13–15,18,19,28]. There are many data sets having only categorical attributes or both categorical and numerical attributes. Furthermore, the fitness function plays a crucial role in the success of a genetic algorithm. Therefore, it is important to have a good fitness function to improve the clustering results obtained by a...
genetic algorithm. Our proposed genetic algorithm (presented in Section 3) addresses these issues carefully.

Particle swarm optimization (PSO) clustering techniques produce optimal clustering results [29–32] using a concept called swarm, which is a collection of a number of particles. A particle is an individual (similar to a chromosome of GA) which is a clustering solution containing a number of cluster centers. Typically, a user defined number of records are randomly selected as the cluster centers that collectively form a particle. The number of particles in a swarm varies from application to application and usually varies between 20 and 40 [31]. The quality of each particle is measured through a fitness function.

In PSO there are a number of iterations. In each of the iteration a particle changes it position. That is, it changes its cluster centers/ seeds. Each particle has three properties namely current position, current velocity and personal best position i.e. the best position that the particle has visited so far. The best position of a particle represents the cluster centers that result in the best fitness of the particle. The current position of a particle is represented by the seeds. A particle changes its current position over the iterations in a velocity. Typically, the velocity depends on the difference between the current position and the personal best. The velocity also depends on the difference between the current position and the position of the particle that has the best fitness in the iteration. That is, there is a tendency of a particle to move to a new position considering its best position and the position of the particle that has the best fitness in the latest iteration. Hence, the particles pull each other toward the best solution and all of them move toward the best clustering solution over the iterations. However, a limitation of PSO based clustering techniques is that they require a user defined number of clusters to form a particle, which may be a difficult issue for a user.

Collaborative clustering techniques make use of the clustering results of multiple clustering techniques in order to produce high quality clusters. They are also known as ensemble clustering techniques [33–35]. However, the collaborative clustering techniques require more computational time than traditional clustering techniques. An existing collaborative clustering technique called Cooperative Bisecting K-Means (CBKM) [34] combines clustering results of K-Means and Bisecting K-Means [36]. A limitation of CMMK is that it requires a user defined number of clusters as an input. For a user it may be difficult to provide the correct number of clusters. Another limitation of the collaborative clustering techniques is their high computation time.

Hierarchical clustering techniques group the records of a data set into a hierarchy [20,37]. They produce a set of nested clusters and the clusters are formed like a tree structure. The tree structure of the clusters is also called a dendrogram. There are two types of hierarchical clustering namely Agglomerative Hierarchical Clustering and Divisive Hierarchical Clustering. Agglomerative hierarchical clustering is a bottom-up approach, where each individual record is first considered as a cluster [20,37]. It iteratively merges two similar clusters together. For merging agglomerative clustering uses some similarity measures. The merging process continues until it obtains one single cluster or certain conditions are satisfied.

Divisive hierarchical clustering is a top-down approach, where all the records of data set are first considered as one large cluster [20,37]. It next divides the large cluster into smaller clusters in such a way so that the most similar records are placed in same cluster. The process of division continues until each record forms a separate cluster. It is kind of a reverse process of the agglomerative hierarchical clustering. One of the limitations of many hierarchical clustering techniques is that they require high computational complexity. For example, an existing hierarchical clustering technique has complexity of $O(n^2)$ [38], whereas K-Means has the complexity of $O(n)$. Hierarchical clustering techniques a record belonging to a cluster cannot move into another cluster. They may also fail to separate the overlapping clusters [29].

3. GenClust: the proposed novel clustering technique

We first present the basic concepts in the next sub-section and then give a formal presentation of the technique.

3.1. The basic contributions of GenClust

3.1.1. A novel initial population selection technique

We assume that a data set D has a set of records $R = \{R_1, R_2, \ldots, R_n\}$ where each record has a set of attributes $A = \{A_1, A_2, \ldots, A_m\}$. An attribute $A$ can be either numerical or categorical. A chromosome $CR$ is made of a number of genes $(G_1, G_2, \ldots, G_k)$ put together in a sequence, where a gene $G_l$ has the same m attributes as any record $R_l$. For example, a chromosome $CR = (G_1, G_2, \ldots, G_k)$ has k number of genes meaning k number of clusters as a solution to the clustering problem. A gene $G_l$ is the same as a seed $S_l$, which is the center of a cluster $C_l$. A cluster $C_l$ is the set of records that have the smallest distance to the gene/seed of the cluster, and is calculated as $C_l = \{R_l : \text{dist}(R_l, G_l) \leq \text{dist}(R_l, G_j) \; \forall j \neq l\}$, where $\text{dist}(R_l, G_l)$ is the distance between a record $R_l$ and a gene $G_l$. Similarities (ranging between 0 and 1) of the values of a categorical attribute $A_l$ are calculated using an existing technique [39] and used to calculate the distance between two records $R_l$ and $R_k$. The values of a numerical attribute $A_l$ are normalized within a range/domain [0, 1]. The distance $\text{dist}(R_l, R_k)$ can range between 0 and 1, and $\text{dist}(R_l, R_k) = \sum_{l=1}^{m} \text{abs}((A_l - A_k))$, where we consider that the first N attributes are numerical (normalized), the next C attributes are categorical, and $R_{lb}$ is the ith attribute value of the ith record.

Our novel initial population selection technique selects 30 chromosomes deterministically and 30 chromosomes randomly. In the deterministic process, in order to select a chromosome we first choose a radius $r_s$ and then calculate the density of each record of the data set as follows: $\text{Density}(R_l) = |\{R_k : \text{dist}(R_l, R_k) \leq r_s ; \forall j\}|$. The record $R_l$ having the highest density (i.e. $\text{Density}(R_l) > \text{Density}(R_j) ; \forall j$) is then chosen as the first gene $G_1$, and all records $(\{R_k : \text{dist}(R_l, R_k) \leq r_s ; \forall j\})$ within the $r_s$ distance of $R_l$ are removed from the data set. We continue the selection of the subsequent genes as long as we get a record with a density greater than a user defined threshold $T$. Therefore, for an $r_s$ value we get a number of genes to form a chromosome $CR_l$.

Since we normalize a data set, the maximum and minimum distance between two records can be 1 and 0, respectively. We therefore use 30 different sensible $r_s$ values (where $r_s \in r$ and $r = \{r_1, r_2, \ldots, r_{30}\}$ ranging between 0 and 1. By a “sensible $r_s$ value” we mean an $r_s$ value that can produce a number of clusters. For example, an $r_s$ value greater than or equal to 0.5 is likely to capture only one seed/gene at the center of the data set since it grabs all records within its radius. For $r_s = 0.5$, we are likely to end up having only one cluster which is the whole data set. Therefore, we choose the upper limit of the set of $r_s$ values to be a lot smaller than 0.5, which is only 0.2 in this study. The $r_s$ values used are 0.0001, 0.0005, 0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 0.009, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, and 0.2. For each $r_s$ value we get a chromosome and therefore, we get 30 chromosomes from the deterministic process as part of the initial population.

An advantage of using different $r_s$ values is demonstrated in Fig. 1. Fig. 1a shows the records (represented by dots) of a two dimensional synthetic data set. Clearly the data set has four clusters. Fig. 1b shows the seeds/genes $(G_1, G_2, \ldots, G_4)$ for a small $r_s$ value. The small $r_s$ value produces excessive seeds/genes. Fig. 1c shows the genes for a different $r_s$ value, which picks the right genes.
and therefore the right chromosome. Since the right $r_x$ value can vary from data set to data set and is generally unknown to a data miner, the use of different $r_x$ values to create different chromosomes for a GA can be useful. A chromosome (such as the one obtained from Fig. 1c) that has a high fitness (i.e. high cluster quality) can help a GA to finally obtain a high-quality clustering result.

Another advantage of using of multiple $r_x$ values to form the chromosomes is demonstrated in Fig. 2. Fig. 2a represents a two-dimensional synthetic data set that has around six clusters of different sizes and shapes. Unlike Fig. 1a, there is no single $r_x$ value that can capture the clusters of Fig. 2a. However, the use of different $r_x$ values to select different chromosomes can help us to finally get a chromosome that captures all useful genes. Figs. 2b and 2c show the genes based on two different $r_x$ values.

In Fig. 2b, for a big $r_x$ value we obtain two genes $G_{11}$ and $G_{12}$ (marked as the solid squares). The circles represent the records contributing to the density of the genes. Fig. 2c shows the eight genes obtained from the data set for a smaller $r_x$ value. Fig. 2d shows the two chromosomes (CR$_1$ and CR$_2$) obtained from the two $r_x$ values in Figs. 2b and 2c. A possible crossover operation on CR$_1$ and CR$_2$ produces two offspring chromosomes CR$_3$ and CR$_4$. The genes of CR$_3$ capture the clusters very well as shown in Fig. 2e. Note that for the purpose of the demonstration of the advantages of our novel initial population selection technique we have chosen a suitable crossover, to get the best offspring chromosome CR$_3$ in a single iteration, which we may not have in reality. However once the right genes are captured in chromosomes, we can expect that through GA operations such as crossover, mutation and elitism over a number of generations we should finally get a chromosome (like CR$_3$) with the best genes, even if the most suitable crossover (as shown in this example) does not happen.
The above examples clearly demonstrate the usefulness of the deterministic process of our initial population selection technique. We now explain the usefulness of the random process where we select another 30 chromosomes. The number of genes of a chromosome is randomly chosen between \[2, \sqrt{n}\], where \(n\) is the number of records (i.e. \(|D|\) or \(|R|\)). Each gene is a record \(R_i\), which is randomly selected from \(R\).

Fig. 2e demonstrates that although our deterministic process produces a set of genes capturing the clusters very well, it still can miss a cluster as shown in top right corner of the figure. Due to the insufficient number of records (i.e. the number of records being less than \(T\)) the deterministic process does not pick a gene to represent the records. However, the random process may still pick a gene from the top right corner. An example set of genes, selected through the random process, is shown in Fig. 3a. Generally the fitness of a chromosome selected through the random process is not expected to be high, but the chromosome can still have some useful genes like \(G_{33}\) chosen from the top right corner of Fig. 3a. Once good genes are included in the chromosomes of the initial population we can expect to finally get a chromosome with all good genes as shown in Figs. 3b and 3c.

3.1.2. Feeding the genes of the best chromosome as the initial seeds of K-Means

It is well-known that a high quality initial seeds can improve the clustering quality of the K-Means algorithm [5,11]. K-Means is also known as being computationally inexpensive [5,15]. We therefore use the best chromosome obtained from the GA as the initial seeds of K-Means in order to allow the genes/seeds to readjust through K-Means and thereby further improve the clustering quality. Fig. 3d shows the final clustering obtained from K-Means based on the best chromosome. Figs. 10 and 11 suggest that GenClust with the K-Means component achieves better result than GenClust without it.

3.1.3. Gene rearrangement

A novel gene rearrangement technique is another basic contribution of GenClust. In Fig. 4 we present two example chromosomes CR7 and CR8 which are also shown in Figs. 5a and 5b, respectively. Two offspring chromosomes CR9 and CR10 are also shown in Fig. 4. The chromosome CR9 is also shown in Fig. 5c. Due to an inappropriate arrangement of the genes in the chromosomes (CR7 and CR8), we end up with all genes from the same side of the data set in the offspring chromosomes CR9 and CR10.
In order to avoid a situation like this, the gene rearrangement of a chromosome before performing the crossover operation is crucial. Note that an existing gene rearrangement technique [19] considers that the lengths of the both chromosomes are equal and therefore is unable to rearrange the genes if the lengths of the chromosomes are not equal.

However, our technique can handle chromosomes of unequal lengths. We rearrange the genes of the inferior chromosome (out of the pair of chromosomes participating in a crossover operation) with respect to the gene arrangement of the superior chromosome. We call the superior chromosome the “reference chromosome” and the inferior chromosome the “target chromosome”.

Algorithm 1. Gene Rearrangement

If \( p = q \) then we rearrange the genes as follows.
1. For \( i = 1 \) to \( q \) DO:
   - \( G_{pi} = G_{pi}; \) abs(dist\((G_{ri}, G_{qi})\)) \(<\) abs(dist\((G_{ri}, G_{qi})\)); \( \forall i \neq k \)
   - \( CR_{k} = CR_{k} \cup G_{qi} \)
   - \( CR_{k} = CR_{k} \cup G_{qi} \)
   - END FOR.

If \( p < q \) then we rearrange the genes as follows.
1. For \( i = 1 \) to \( p \) DO:
   - \( G_{pi} = G_{pi}; \) abs(dist\((G_{ri}, G_{qi})\)) \(<\) abs(dist\((G_{ri}, G_{qi})\)); \( \forall i \neq k \)
   - \( CR_{k} = CR_{k} \cup G_{qi} \)
   - \( CR_{k} = CR_{k} \cup G_{qi} \)
   - END FOR.

If \( q < p \) then we carry out the following rearrangement.
1. For \( i = 1 \) to \( q \) DO:
   - \( G_{pi} = \text{abs}(\text{dist}(G_{ri}, G_{qi})) < \text{abs}(\text{dist}(G_{ri}, G_{qi})); \) \( \forall i \neq k \)
   - \( CR_{k} = CR_{k} \cup G_{qi} \)
   - \( CR_{k} = CR_{k} \cup G_{qi} \)
   - END FOR.

Let us assume that the reference chromosome \( CR_{r} = \{G_{r1}, G_{r2}, \ldots, G_{rp}\} \) has \( p \) number of genes and the target chromosome \( CR_{t} = \{G_{t1}, G_{t2}, \ldots, G_{tp}\} \) has \( q \) number of genes. The target chromosome is rearranged into a new chromosome \( CR_{q} = \{G_{q1}, G_{q2}, \ldots, G_{qp}\} \) that has the same length as the target chromosome. There are three possible cases: \( p = q, p < q, \) and \( p > q \), as discussed below in Algorithm 1 that explains the re-arrangement process.

We now discuss the impact of our rearrangement operation. After rearrangement the chromosome \( CR_{r} \) becomes \( CR_{11} \) as shown in Fig. 6. The offspring chromosomes (\( CR_{12} \) and \( CR_{13} \)) are also shown in Figs. 7a and 7b that gives a much better clustering result compared to Fig. 5c. Fig. 12 and Table 4 suggest better clustering results obtained by GenClust with rearrangement than without it.

3.1.4. Twin removal

If the length of a chromosome (i.e. the number of genes) is more than 2 and if there are two identical genes, we delete one of the two identical genes. The chromosome length thus decreases by one. If the length of a chromosome is 2 and both genes are identical then we randomly change one gene to make sure that the genes are not identical.

In this study we consider two genes \( G_{ji} \) and \( G_{ijk} \) identical when dist\((G_{ji}, G_{ijk})\) = 0. However, a user may choose any other small value \( s \) (such as 0.01 and 0.02) where \( G_{ji} \) and \( G_{ijk} \) can be considered identical if dist\((G_{ji}, G_{ijk})\) \( \leq s \). Two genes may not be exactly the same (i.e. dist\((G_{ji}, G_{ijk}) \neq 0 \)), but they can still be very similar to each other (i.e. dist\((G_{ji}, G_{ijk}) \) can be close to zero) making it sensible to consider only one of those in the chromosome. An imperial analysis on the impact of different \( s \) values can be an interesting future work.

3.1.5. A novel fitness function

Another basic contribution of the study is a novel fitness function for use in the genetic algorithm. The same fitness function can also be used in the cluster quality evaluation. It favors a clustering solution that has compact clusters (i.e. records within each cluster are close to each other) and big separations/gaps among the clusters (i.e. records belonging to different clusters have big distances). We name the fitness function as “Compactness and Separation Measure of Clusters” (COSEC).

In order to reduce the complexity of the fitness function, it first identifies the seed \( S_{i} \) of a cluster \( C_{i} \); \( \forall j \) and calculates the distance between two seeds \( S_{i} \) and \( S_{j} \) to estimate the separation between the clusters, instead of calculating the average distance between all pairs of records \( R_{k} \in C_{i} \) and \( R_{k} \in C_{j} \) as it is done by many existing techniques such as Silhouette Coefficient [20]. It also uses the distance between the records of a cluster and its seed dist\((R_{k}, S_{i})\); \( \forall R_{k} \in C_{i} \) (instead of average distance for all records \( R_{k} \in C_{i} \)) in order to calculate the compactness of a cluster \( C_{i} \).

Let us assume that the reference chromosome \( CR_{r} = \{G_{r1}, G_{r2}, \ldots, G_{rp}\} \) has \( p \) number of genes and the target chromosome \( CR_{t} = \{G_{t1}, G_{t2}, \ldots, G_{tp}\} \) has \( q \) number of genes. The target chromosome is rearranged into a new chromosome \( CR_{q} = \{G_{q1}, G_{q2}, \ldots, G_{qp}\} \) that has the
The compactness \( \text{Comp}_j \) (see Eq. (1)) of a cluster \( C_j \), the separation \( \text{Sep}_j \) (see Eq. (2)) of a cluster \( C_j \) with all other clusters, and the overall fitness \( \text{Fitness} \) (see Eq. (3)) of a clustering solution are calculated as follows, where \(|C_j|\) is the number of records belonging to the cluster \( C_j \).

\[
\text{Comp}_j = \sum_{R_k \in H_{R_j}} \text{dist}(R_k, S_j)
\]

In that case from a chromosome we can calculate the membership degree \( l_{aj} \) of a record \( R_a \) with a seed \( S_j \) [28], and then use the membership degrees in calculating the compactness (see Eq. (4)) which in turn is then used for the fitness calculation.

\[
\text{Fitness} = \sum_{j} (\text{Sep}_j / \text{Comp}_j)^3
\]

3.2. The main steps of GenClust

We now introduce the main steps (see Algorithm 2) of GenClust as follows.

**Step 1: Initial population selection**: We prepare an initial population of the \( 2 \times |r| \) numbers of chromosomes, \(|r|\) from the deterministic process and \(|r|\) from the random process as explained before in Section 3.1.1 and Step 1 of Algorithm 2. The step needs as input a whole data set \( D \), a set of radius \( r \) and a user defined value for \( T \), which is the minimum number of records required within a radius \( (r_x) \) of a record \( R_a \). The value of \(|r|\) in this study is 30 i.e. \( r = \{r_1, r_2, \ldots, r_{30}\} \). See Step 1 of Algorithm 2. Since it is unlikely for a user to know the most appropriate \( r_x \) value for finding the best initial seeds (as explained in Section 3.1.1 and Fig. 1) our approach of using a number of different \( r_x \) values to find a number of chromosomes is useful. Moreover, our approach of using the chromosomes in a genetic algorithm is also useful to find the best set of initial seeds as explained in Section 3.1.1 and Fig. 2.

**Step 2: Selection operation**: We first sort the \( 2 \times |r| \) chromosomes in the descending order of their fitness values. We then choose the \(|r|\) number of best chromosomes from the initial population of \( 2 \times |r| \) chromosomes using our fitness function (see Section 3.1.5). A copy of the best chromosome \( CR_b \) is stored in the memory.

**Step 3: Crossover operation**: We first sort the \(|r|\) chromosomes in the descending order according to their fitness values. All chromosomes participate in the crossover operation pair by pair since for a crossover operation we need a pair of chromosomes. The best chromosome (which is available in the current population) is chosen as one chromosome of the pair. The second chromosome of the pair is chosen using the roulette wheel technique [14,18,28] where a chromosome \( CR_i \) is picked with a probability \( p(CR_i) = f(CR_i) / \sum_j f(CR_j) \). Here, \( f(CR_i) \) is the fitness of the chromosome \( CR_i \) and \(|P|\) is the size of the current population, which is \(|r|\) at the beginning. Every time a chromosome is chosen it is removed from the population of the current generation and the number decreases by 1.

Once the pair of chromosomes for crossover is selected, we then apply the gene rearrangement operation (see Sections 3.1.3 and 3.1.4). The gene pair at this stage participates in a conventional single point crossover operation [14,28], where each chromosome is divided into two parts at a random point between two genes. The left part (having one or more genes) of one chromosome joins the right part (having one or more genes) of the other chromosome to form an offspring chromosome. From the crossover of a pair of chromosomes we get a pair of offspring chromosomes which are added in the population of the next generation. The whole process

---

Fig. 4. Two chromosomes \( CR_p \) and \( CR_o \) and two offspring \( CR_b \) and \( CR_o \).

Fig. 5a. Thirteen genes in \( CR_p \).

Fig. 5b. Fourteen genes in \( CR_o \).

Fig. 5c. Fifteen genes in \( CR_b \).
of gene pair selection and crossover operation continues while we have a gene in the current population. At the end of this, we get |r| chromosomes in the population of the next generation. We then apply the twin removal operation on the new population made of the offspring chromosomes.

Step 4: Elitism operation: Elitism keeps track of the best chromosome throughout the generations and also keeps improving the quality of the population in each generation. If the fitness of the worst chromosome (i.e. the chromosome having the worst fitness among all chromosomes of the new generation) is less than the fitness of the best chromosome $C_{rb}$, then the worst chromosome is replaced by $C_{rb}$. Additionally, if the best chromosome of the new population has better fitness than $C_{rb}$ then we store a copy of the best chromosome into $C_{rb}$.

Step 5: Mutation operation: The basic idea of the mutation operation is to randomly change some of the chromosomes in order to explore different solutions. While adding random changes to the chromosomes we use a probabilistic approach where a chromosome with a low fitness has a high probability of getting a random change, and vice versa [19]. The mutation probability of each chromosome $C_{pc}$ is calculated (see Step 5 of Algorithm 2). The mutation probability of the $i$-th chromosome is calculated as follows.

$$M_i = \begin{cases} K_1 \frac{f_{\text{max}} - f_i}{f_{\text{max}} - \hat{f}}, & f_i > \hat{f} \\ K_2, & f_i \leq \hat{f} \end{cases}$$

where $K_1$ and $K_2$ are equal to 0.5, $f_{\text{max}}$ is the maximum fitness of a chromosome in $P_c$, $f$ is average fitness of the chromosomes of $P_c$, and $f_i$ is the fitness of the $i$th chromosome. Once a chromosome is selected for mutation we randomly select at attribute of each gene and modify the attribute value randomly.

After the mutation operation, we perform the twin removal operation to remove twin genes (if any) from each chromosome. At this stage we again update the chromosome $C_{rb}$ having the best fitness. If the termination condition is not met we repeat the whole process from Step 3 to Step 5, and finally at the end of $N$ iterations we get the chromosome $C_{rb}$ having the best fitness.

Step 6: K-Means: We use the genes of the best chromosome $C_{rb}$ as the initial seeds of the well-known K-Means clustering algorithm. With high quality initial seeds K-Means is expected to produce a high quality clustering solution as well. Fig. 10 empirically supports the idea as we can see that for all data sets and all evaluation criteria we get a better clustering result from GenClust with K-Means (denoted as GenClust-H in Section 4) than GenClust without K-Means (denoted as GCWoK).

The K-Means that we use in this study is a slight modification of the existing Simple K-Means [20]. We normalize all numerical values where the values for each attribute vary between 0 and 1. While calculating the distance between the categorical values of an attribute for two records we take the similarity of the values into consideration [10]. The distance between two values of a categorical attribute varies between 0 and 1. Finally, for seed calculation we consider the categorical value that has the highest frequency [11].

### 3.3. Complexity analysis

Considering the total number of records is $n$, number of attributes is $m$, maximum domain size of an attribute is $d$, maximum number of genes in a chromosome is $k$, the number of iterations of crossover and mutation is $N$, the number of iteration for K-Means is $N_0$, and number of chromosomes is $z = |r|$ the estimated complexity of GenClust is $O(nm^2z + md^2z + n^2mz + Nnzkz + \ldots)$.
For a high dimensional data set where \( m \) is very large compared to all other parameters the complexity of GenClust is \( O(m^2) \). For a very large data set where the number of records \( n \) is very large compared to all other parameters the complexity is \( O(n^2) \). The detailed complexity analysis is presented in Appendix A. We also present the detailed complexity analysis of CRUDAW [10] in Appendix A since it helps us to explain the complexity of GenClust.

In Section 4, we empirically compare GenClust with five existing techniques called AGCUK [18], SABC [3], CRUDAW [10], GAGR [19] and GFCM [4]. The complexity analysis of AGCUK and SABC was presented in the original papers [3,18]. The complexity of AGCUK is \( O(2kmnN) \) [18]. The complexity of SABC [3] is \( O(nm^2 + m^d + nkN'(m_1 + m_d)) \), where \( d \), \( m_1 \) and \( m_d \) are the average domain size of the categorical attributes, number of numerical attributes and categorical attributes, respectively. The complexity of CRUDAW is \( O(nmn + md^2 + n^2m + nnkN') \). The complexity of the well known K-Means algorithm is \( O(kmnN) \), where \( N \) is the number of iterations [40,41]. For large data sets (where \( n \gg \) any other parameter) the complexity of GenClust and CRUDAW are \( O(n^2) \) while some existing techniques such as ACCA [42], H C [38] and ACAD [43] have the complexity of \( O(n^3) \). However, for big data sets the complexity of AGCUK [18] and SABC [3] is \( O(n) \).

Algorithm 2. GenClust

```
Algorithm: GenClust
Input: A normalized data set \( D \), a set of user defined radius \( r \), a user defined minimum number of records \( T \), user defined number of generations/iterations \( N \).
Output: A set of clusters \( C \).

Set \( P_1 = \emptyset \). \( P_1 \) is the set of initial population (60 chromosomes), initially set \( P_1 \) to null/*.
Set \( P_2 = \emptyset \). \( P_2 \) is the set of selected chromosomes, initially set \( P_2 \) to null/*.
Set \( P_0 = \emptyset \). \( P_0 \) is the set of offsprings chromosomes, initially set \( P_0 \) to null/*.

/*Step 1: Initial Population */
FOR (i=1 to \( T \)) DO
    \( C_R \leftarrow \) DeterministicProcess \( (P_i, T) \).*Obtain chromosome \( C_R \) for a radius \( r \) for the deterministic process*/
    \( P_1 \leftarrow P_1 \cup C_R/.*\)insert chromosome \( C_R \) in \( P_1 */
END FOR

FOR (i=1 to \( T \)) DO
    \( C_R \leftarrow \) RandomProcess \( (D) \).*generate k seeds randomly and encode them together to form the chromosome \( C_R */
    \( P_1 \leftarrow P_1 \cup C_R/.*\)insert chromosome \( C_R \) in \( P_1 */
END FOR

/*Step 2: Selection Operation */
WHILE \( |P_1| \leq |T| \) DO
    \( C_{R_{\text{max}}} \leftarrow \) FindChromosomeWithMaxFitness \( (P_1) \). \( C_{R_{\text{max}}} \) is a chromosome having maximum fitness */
    \( P_2 \leftarrow P_2 \cup C_{R_{\text{max}}}/.*\)insert chromosome \( C_{R_{\text{max}}} \) in \( P_2 */
    \( P_1 \leftarrow P_1 \cup C_{R_{\text{max}}}/* remove \( C_{R_{\text{max}}} \) from \( P_1 */
END WHILE

\( C_{R_0} \leftarrow \) FindChromosomeWithMaxFitness \( (P_2) \). \( C_{R_0} \) is a chromosome having maximum fitness in \( P_2 */

FOR (g =1 to \( N \)) DO /* g counts the number of generations*/

/*Step 3: Crossover Operation */
WHILE \( |P_0| \geq 0 \) DO
    \( P = \) PickChromosomePair \( (P_0) \).*select a pair of chromosomes \( P = (P_1, P_2) \) from \( P_0 \) using roulette wheel*/
    \( P = \) RearrangeOperation \( (P) \).*rearrange one chromosome of \( P \) considering the other one as a reference chromosome */
    \( P_0 = \) Crossover \( (P) \).*after crossover between \( P_1 \) \& \( P_2 \), two offspring \( O = \{O_1, O_2\} \) are generated */
    \( P_0 = P_0 \cup O/.*\)insert offspring \( O = \{O_1, O_2\} \) in \( P_0 */
END WHILE

\( P_0 = \) TwinRemoval \( (P_0) \).*removing twin genes from \( P_0 */

/*Step 4: Elitism Operation */
IF fitness of worst chromosome of \( P_0 \) < fitness of \( C_{R_0} \)
    Replace the worst chromosome of \( P_0 \) by \( C_{R_0} \)
END IF
IF fitness of best chromosome of \( P_0 \) > fitness of \( C_{R_0} \)
    Replace the \( C_{R_0} \) by the best chromosome of \( P_0 \)
END IF

/*Step 5: Mutation Operation */
Set \( P_0 = \emptyset \). \( P_0 \) is the set of mutated chromosomes, initially set \( P_0 \) to null/*.
\( M = \) CalculateMutationProbability \( (P_0) \).*Generate a random number between \( 0 \) and \( 1 \)*/
\( M = \{M_1, M_2, \ldots, M_{mn}\} \) is a set of the mutation probability of every chromosome in \( P_0 */
FOR (i=1 to \( |P_0| \)) DO /*generate a random number between \( 0 \) and \( 1 \)*/
    \( \text{rand} = \) RandomNumber \( (1) \).*generate a random number between \( 0 \) and \( 1 \)*/
    IF \( \text{rand} < M_i \
        \( C_{R_m} = \) PerformMutation \( (C_{R_m}) \).*perform mutation on \( C_{R_m} \) chromosome & get mutated chromosome \( C_{R_m} \)*/
        \( P_0 = P_0 \cup C_{R_m} \).*insert mutated chromosome \( C_{R_m} \) in \( P_0 */
    END IF
ELSE
        \( P_0 = P_0 \cup C_{R_m} \).*insert un-mutated chromosome \( C_{R_m} \) in \( P_0 */
        \ENDELSE
END FOR

\( P_0 = \) TwinRemoval \( (P_0) \).*removing twin genes from \( P_0 */
\( C_{R_0} = \) FindChromosomeWithMaxFitness \( (P_0) \).*\( C_{R_0} \) is a chromosome having the maximum fitness*/

Return \( C */
```

Please cite this article in press as: M.A. Rahman, M.Z. Islam, A hybrid clustering technique combining a novel genetic algorithm with K-Means, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.011
4. Experimental results and discussion

4.1. The data sets and the cluster evaluation techniques

As shown in Step 6 of Algorithm 2, our proposed technique called GenClust uses hard K-Means along with a novel genetic algorithm. However, for experimentation and comparison we in this section use two versions of our proposed GenClust algorithm. In the first version (called GenClust-H) we use K-Means in Step 6 of Algorithm 2, whereas in the second version (called GenClust-F) we replace K-Means by Fuzzy C-Means [10] in Step 6 of Algorithm 2. We empirically compare the both versions of GenClust with five existing techniques called AGCUK [18], CRUDAW [10], GAGR [19], GFCM [4] and SABC [3] on twenty natural data sets that are available from the UCI machine learning repository [44] (see Table 1). The existing techniques are recent, of high quality and shown (in the literature) to be better than many other techniques. For example, AGCUK [18] and GAGR [19] are shown to be better than some other genetic algorithm based clustering techniques [13, 25, 45]. Some of the data sets that are used in this study have either only numerical or only categorical attributes whereas some others (CMC, CA, and Adult) have both numerical and categorical attributes. The domain sizes of the class attributes vary from 2 to 10. Similarly, the numbers of records vary from 214 to 32,561.

Some data sets have missing values in them. We delete all records having any missing values resulting in the Dermatology, Credit Approval, Mammographic Mass, Mushroom and Adult data sets having 358, 653, 830, 5644 and 30,162 records, respectively. The data sets in Table 1 have class attributes. However, the data set for which a clustering technique is applied generally does not have a class attribute. Therefore, we first remove the class attributes from the data sets in Table 1, before we apply clustering techniques on them. The class attributes are again used for the cluster quality evaluations based on external quality metrics such as F-measure, Entropy and Purity [20]. We evaluate and compare the clustering results based on the internal (our novel fitness function called COSEC, Xie-Beni (XB), and Sum of Square Error (SSE)), and the external (F-measure (FM), Entropy (E), and Purity (P)) evaluation criteria [20, 28].

4.2. The parameters used in the experiments

We use the value of $T$ (for both GenClust and CRUDAW) equal to 1% of the records of a data set. The number of chromosomes in the initial population of GenClust is equal to 60, and after the selection operation equal to 30. The number of iterations used for the crossover, elitism and mutation are 60 (i.e. $N = 60$ in Algorithm 2) and the number of iterations for K-Means is 50. The termination condition of K-Means is that the SSE of two consecutive iterations needs to be less than 0.005. For AGCUK [18], the number of chromosomes in the initial population and the number of iterations are 20 and 50, respectively as recommended by the study [18]. The values of $r_{\text{max}}$ and $r_{\text{min}}$ are 1 and 0, respectively as recommended by the study [18]. For CRUDAW [10], we use the fuzzy coefficient $\beta = 2.2$, and the fuzzy termination condition $e = 0.005$. The maximum number of iterations for CRUDAW is considered to be 50. For GAGR [19], the number of chromosomes in the initial population is 50 and the number of generations is 50, as recommended in the study [19]. For GFCM [4] the fuzzy coefficient $\beta = 1.3$ and fuzzy termination condition $e = 0.0001$, as recommended in the original paper [4] to obtain the best result for the technique. For SABC [3] the maximum number of iterations is considered to be 50 and a user defined threshold $\epsilon$ is considered to be 0.005.

4.3. The experimental setup

On each data set, we run both versions of GenClust (GenClust-H and GenClust-F) five times each, since it can give different clustering solutions in different runs. In each run we continue 60 iterations of crossover, elitism and mutation in order to get the best chromosome (clustering solution) which is then fed into K-Means (that was explained in Step 6 of Section 3.2) for GenClust-H or Fuzzy C-Means (that was used in CRUDAW [10]) for GenClust-F to produce a further improved clustering result (see Algorithm 2). In order to assess the contribution of K-Means and Fuzzy C-Means we evaluate the cluster quality of the solutions that are obtained before K-Means or C-Means is applied (called GCWoK), and compare it with GenClust-H and GenClust-F. We also run AGCUK for five times and thereby obtain five clustering results (since it may produce different clustering results in different runs), which are then evaluated. For CRUDAW, we use five different $r$ values to obtain five different clustering results which are then evaluated. For GAGR, GFCM and SABC we produce the same number of clusters ($k$) that GenClust produces in each individual run. We cannot do this for AGCUK and CRUDAW since they produce the cluster number automatically (not user defined). Moreover, for each $k$ we

<table>
<thead>
<tr>
<th>Data sets</th>
<th>No. of records with missing</th>
<th>No. of records without missing</th>
<th>No. of numerical attributes</th>
<th>No. of categorical attributes</th>
<th>Class size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass Identification (GI)</td>
<td>316</td>
<td>314</td>
<td>10</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Ecoli (EC)</td>
<td>336</td>
<td>336</td>
<td>10</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Liver Disorders (LD)</td>
<td>345</td>
<td>345</td>
<td>10</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Dermatology (DL)</td>
<td>366</td>
<td>358</td>
<td>6</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Credit Approval (CA)</td>
<td>650</td>
<td>653</td>
<td>18</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Pima Indian Diabetes (PDM)</td>
<td>768</td>
<td>768</td>
<td>18</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Mammographic Mass (MGM)</td>
<td>961</td>
<td>830</td>
<td>5</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Statlog Vehicle Silhouettes (SVS)</td>
<td>846</td>
<td>846</td>
<td>6</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Tic-Tac-Toe (T5T)</td>
<td>958</td>
<td>958</td>
<td>0</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Contraceptive Method Choice (CMC)</td>
<td>1473</td>
<td>1473</td>
<td>2</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Yeast</td>
<td>1484</td>
<td>1484</td>
<td>8</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Image Segmentation (IS)</td>
<td>2310</td>
<td>2310</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Chess King-Rook vs. King-Pawn (CKRP)</td>
<td>3196</td>
<td>3196</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Wine Quality (WQ)</td>
<td>4898</td>
<td>4898</td>
<td>11</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Page Blocks Classification (PBC)</td>
<td>5473</td>
<td>5473</td>
<td>10</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Mushroom (MR)</td>
<td>8124</td>
<td>5644</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Pen-Based Recognition of Handwritten Digits (PBPRHD)</td>
<td>10,992</td>
<td>10,992</td>
<td>16</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>MAGIC Gamma Telescope (MGT)</td>
<td>19,020</td>
<td>19,020</td>
<td>10</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Adult</td>
<td>32,561</td>
<td>30,162</td>
<td>6</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>
run GFCM and SABC ten times and take the average clustering results to compare it with GenClust-H and GenClust-F.

4.4. The experimental results

In Table 2, we present the Xie-Beni (XB) Index of the techniques for five runs on the Pima Indian Diabetes (PID) data set. A lower XB value indicates a better clustering solution. We assign score (the higher the better) to each technique based on its average XB value, where the technique having the lowest XB value gets 7 points and the highest XB gets 1 point. Table 3 shows the scores for all evaluation techniques.

Table 2
XB Index (lower the better) of the PID data set.

<table>
<thead>
<tr>
<th>No.</th>
<th>GenClust-H</th>
<th>GenClust-F</th>
<th>CRUDAW</th>
<th>AGCUK</th>
<th>GAGR</th>
<th>GFCM</th>
<th>SABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1200</td>
<td>0.2326</td>
<td>0.3993</td>
<td>0.2934</td>
<td>0.5904</td>
<td>0.3055</td>
<td>0.5162</td>
</tr>
<tr>
<td>2</td>
<td>0.0818</td>
<td>0.3938</td>
<td>1.8809</td>
<td>0.3202</td>
<td>0.2995</td>
<td>0.3376</td>
<td>0.4415</td>
</tr>
<tr>
<td>3</td>
<td>0.0932</td>
<td>0.2390</td>
<td>2.5524</td>
<td>0.2150</td>
<td>0.5904</td>
<td>0.3055</td>
<td>0.5162</td>
</tr>
<tr>
<td>4</td>
<td>0.0892</td>
<td>0.2895</td>
<td>1.6396</td>
<td>0.3140</td>
<td>0.3975</td>
<td>0.4090</td>
<td>0.5256</td>
</tr>
<tr>
<td>5</td>
<td>0.1035</td>
<td>0.3287</td>
<td>1.0335</td>
<td>0.3499</td>
<td>0.9974</td>
<td>0.5084</td>
<td>0.3411</td>
</tr>
</tbody>
</table>

Similarly, we first calculate the scores for all evaluation criteria on fourteen numerical data sets (i.e. the data sets that have numerical attributes only); since AGCUK and GAGR can cluster data sets with just numerical attributes. Fig. 8 shows the average scores (over the fourteen data sets) where GenClust-H and GenClust-F receives higher scores than the other five techniques on all evaluation criteria. Fig. 9 shows the average scores (for all twenty data sets) of GenClust-H, GenClust-F, CRUDAW, GFCM and SABC, where the best technique receives 5 points and the worst technique receives 1 point, for each evaluation criteria and each data set. The both versions of GenClust outperform CRUDAW, GFCM and SABC in Fig. 9 as well.

In the figures from Figs. 10a–10f we present the actual average values (not the scores) of the evaluation criteria for fourteen numerical data sets. We also compare the clustering quality of GenClust-H and GenClust-F with GCWoK (GenClust without K-Means or Fuzzy C-Means) in order to assess the contribution of K-Means and Fuzzy C-Means in the GenClust algorithm. The figures from Figs. 10a–10f show that GenClust-H performs better than the other five existing techniques for all evaluation criteria. In all fourteen data sets, GenClust-F also performs better than the existing techniques based on all evaluation criteria except COSEC. Besides, GenClust-H and GenClust-F perform better than GCWoK for all evaluation criteria. The figures from Figs. 11a–11f

Please cite this article in press as: M.A. Rahman, M.Z. Islam, A hybrid clustering technique combining a novel genetic algorithm with K-Means, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.011
Fig. 10a. Average XB of the techniques for 14 data sets.

Fig. 10b. Average SSE of the techniques for 14 data sets.

Fig. 10c. Average Entropy of the techniques for 14 data sets.

Fig. 10d. Average COSEC of the techniques for 14 data sets.

Fig. 10e. Average F-measure of the techniques for 14 data sets.

Fig. 10f. Average Purity of the techniques for 14 data sets.

Fig. 11a. Average XB of the techniques for 20 data sets.

Fig. 11b. Average SSE of the techniques for 20 data sets.

Please cite this article in press as: M.A. Rahman, M.Z. Islam, A hybrid clustering technique combining a novel genetic algorithm with K-Means, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.011
present the results for all twenty data sets and again a similar outcome is observed.

We also note that the evaluation results (shown in Figs. 8–11) based on our novel fitness functions (i.e. cluster evaluation technique) COSEC are consistent with the results based on all other evaluation criteria. This indicates the effectiveness of COSEC as a fitness function and a cluster evaluation technique.

Fig. 12 and Table 4 evaluate the contribution of the gene rearrangement step in GenClust. It is clear (based on the average result of the fourteen numerical data sets) that GenClust-H performs better with re-arrangement than without it. However, it is worth noting that GenClust-H defeats the other five techniques even without K-Means and without gene rearrangement.

In Fig. 13 for each data set we present the average fitness of the best chromosome for five runs of GenClust-H. The average fitness values are plotted against the iterations. For three large data sets, we have run GenClust-H for 30 iterations instead of 60 iterations. Fig. 14 presents the average fitness for all data sets i.e. the average of the average values that are presented in Fig. 13. The drop at the 31st iteration of Fig. 14 is due to the discontinuation of the GenClust-H iterations for the three large data sets. It is clear that the fitness of the best chromosomes steadily increases for most data sets except CRKRP and MR. Therefore, GenClust-H is capable of producing better quality clusters with a higher number of iterations. We use only 60 iterations in this study in order to be consistent with the experiments in the literature [18,19] and get the results relatively quickly. Besides, GenClust-H achieves better results than the five existing techniques even with 60 iterations. With a higher number of iterations GenClust-H is expected to achieve a better result.

A unique property of GenClust (i.e. GenClust-H and GenClust-F) is the combination of chromosomes that are selected from a deterministic and a random process (see Step 1 and Step 2 of Algorithm 2). We examine the 30 best chromosomes in the first iteration/generation to explore whether they come from the deterministic process or random process. Fig. 15 shows that both the deterministic and the random process contribute a reasonable number of chromosomes (on an average for five runs) in the initial population of 30 chromosomes. This indicates the usefulness of both deterministic and random process in our algorithm.

In Fig. 15, we observe that the MR, CMC and CA data sets have higher number of chromosomes from the random process. We also remember that the improvements of the fitness values over the iterations for the MR and CA data sets (see Fig. 13) are very low. Therefore, it appears that the deterministic process fails to pick useful chromosomes for these data sets. This is perhaps a property of these data sets.

GenClust-H uses an initial population of 60 chromosomes where 30 of them are chosen through a deterministic process and the remaining 30 of them are chosen randomly. Conventional GA techniques use only the randomly chosen chromosomes. Therefore, we now empirically evaluate the effectiveness of our approach of using chromosomes by both random and deterministic process. We implement a modification of GenClust-H where the initial population consists of 30 randomly created chromosomes and call it GenClust-H-Random. The clustering quality of GenClust-H-Random is then compared with GenClust-H on the PID, WQ and Yeast data sets. The experimental results are presented in Figs. 16–18 that show the scores achieved by the techniques. The figures clearly demonstrate the superiority of GenClust-H suggesting the usefulness of our approach in the initial population selection.

In Table 5 we present the results on the average execution time (for five runs on each data set) required by the techniques. In the experiments we use three different machines: M1, M2 and M3. M1 has Intel (R) Core (TM) i5 CPU M430 @ 2.27GHz and 4 GB of
RAM, M2 has Intel (R) Core (TM) i5-2500 CPU @ 3.30GHZ and 8 GB RAM, and the shared machine M3 has 4×8 core Intel E7-8837 Xeon processors, 256 GB of RAM, and 23 TB of disk storage. The both versions of GenClust are computationally more expensive than the existing techniques. Genetic algorithms are generally time expensive. Moreover, GenClust uses 60 iterations (generations) compared to 50 iterations of AGCUK and GAGR. It also uses computationally expensive operations such as the deterministic process for an initial population, gene rearrangement, and twin removal. Note that, the complexity of GenClust-H and GenClust-F are the same since all steps except Step 6 (see Algorithm 2) are the same for both techniques. Additionally, Step 6 has the same complexity of $O(n)$ for both K-Means (used in GenClust-H) and Fuzzy C-Means (used in GenClust-F) [40,41]. When the complexity of GenClust is $O(n^3)$ (see Appendix A), some existing clustering techniques (not used in this study) such as ACCA [42], HC [38] and ACAD [43] have the complexity $O(n^3)$.

An important aim of the genetic algorithms is to explore various solutions and thereby reach the global minima as the final solution. We empirically check the exploration ability of our technique as follows. For every record $R_i$ of a data set we find its closest gene $G_j$ and the distance $d_{ij}$ between $R_i$ and $G_j$. We then calculate the
average distance \( d = \sum_{i=1}^{\mid R \mid} d_i/R \), where \( R \) is the number of records in a data set. A lower average distance \( d \) indicates the existence of a gene close to every record \( R_i \) and therefore an evidence of greater exploration through the genes. Fig. 19a shows the average distance for CRUDAW [10], AGCUK [18] and GenClust on the PID data set [44]. Note that, the initial population is the same for both versions of GenClust (GenClust-H and GenClust-F). Each time we run GenClust we may get different number of genes (not chromosomes) due to the existence of the random process. The X axis of Fig. 19a shows the number of genes for five runs. We produce the same number of genes from AGCUK and compare the average distance. Fig. 19a shows that GenClust achieves lower average distance than AGCUK and CRUDAW. Fig. 19b shows the scores of the techniques for three data sets (PID, LD, and SVS). GenClust performs better than the existing techniques.

Although our proposed techniques (such as GenClust-H) achieve better results than a well known and commonly used technique called K-Means, the complexities of our techniques and K-Means are \( O(n^2) \) and \( O(n) \), respectively. Therefore, we now carry out an empirical analysis where we allow K-Means to run \( n \) times and then we pick the best result out of the \( n \) runs of K-Means. This allows K-Means to use an equivalent complexity of \( O(n^2) \) and therefore we can make a fare comparison between our technique called GenClust-H and K-Means.

We perform empirical analyses on the PID and LD data sets. The PID and LD data sets have 768 and 345 records, respectively. We therefore run SK (i.e. Simple K-Means) on the PID and LD data sets 768 and 345 times, respectively and pick the best clustering results of SK. That is, for a data set we calculate the XB and SSE for each run of K-Means and then pick the best XB and SSE value. Note that the best XB and SSE values may not be obtained from the same run of K-Means. Also note that we run GenClust-H five times on each data set and present the best, average and worst results in Tables 6 and 7. It is clear from the tables that even the worst results of GenClust-H are better than the best results of K-Means in both data sets. It is needless to say that the worst or the average results of K-Means are worse than even the worst result of GenClust-H.
4.5. Statistical analysis

We now carry out some tests to evaluate the statistical significance of the superiority of the results (i.e. XB, SSE, COSEC, F-measure, Entropy and Purity) obtained by GenClust-H and GenClust-F over the results obtained by the existing techniques. We observe that the results do not follow a normal distribution, and their variances are different. The essential conditions for

Table 5
Overall average computational times (in minutes) of the techniques for all data sets.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>GenClust-H</th>
<th>GenClust-F</th>
<th>CRUDAW</th>
<th>AGCUK</th>
<th>GAGR</th>
<th>GFCM</th>
<th>SABC</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>PID</td>
<td>324.4293</td>
<td>324.8626</td>
<td>0.1370</td>
<td>8.4588</td>
<td>22.6528</td>
<td>8.2294</td>
<td>0.2003</td>
<td>M1</td>
</tr>
<tr>
<td>LD</td>
<td>64.4048</td>
<td>64.4415</td>
<td>0.0315</td>
<td>1.8022</td>
<td>3.7509</td>
<td>1.0009</td>
<td>0.0263</td>
<td>M1</td>
</tr>
<tr>
<td>BT</td>
<td>41.4222</td>
<td>41.4522</td>
<td>0.0882</td>
<td>2.5578</td>
<td>3.3137</td>
<td>0.7643</td>
<td>0.0564</td>
<td>M1</td>
</tr>
<tr>
<td>SVS</td>
<td>113.7751</td>
<td>114.1849</td>
<td>0.3883</td>
<td>11.0243</td>
<td>20.6648</td>
<td>5.4247</td>
<td>0.5470</td>
<td>M1</td>
</tr>
<tr>
<td>MR</td>
<td>134.7162</td>
<td>134.8428</td>
<td>0.0510</td>
<td>NA</td>
<td>NA</td>
<td>0.1060</td>
<td>0.8321</td>
<td>M2</td>
</tr>
<tr>
<td>CA</td>
<td>65.3263</td>
<td>65.6956</td>
<td>0.1151</td>
<td>NA</td>
<td>NA</td>
<td>0.0662</td>
<td>1.1134</td>
<td>M1</td>
</tr>
<tr>
<td>CMC</td>
<td>377.5297</td>
<td>381.5297</td>
<td>0.1763</td>
<td>NA</td>
<td>NA</td>
<td>11.2118</td>
<td>0.9584</td>
<td>M1</td>
</tr>
<tr>
<td>GI</td>
<td>37.3626</td>
<td>38.0486</td>
<td>0.0220</td>
<td>1.1625</td>
<td>2.2144</td>
<td>0.2047</td>
<td>0.0146</td>
<td>M1</td>
</tr>
<tr>
<td>EC</td>
<td>30.8369</td>
<td>33.1471</td>
<td>0.0259</td>
<td>0.6823</td>
<td>3.0356</td>
<td>0.2404</td>
<td>0.0040</td>
<td>M1</td>
</tr>
<tr>
<td>DL</td>
<td>199.5601</td>
<td>198.7597</td>
<td>0.0601</td>
<td>2.7414</td>
<td>23.2343</td>
<td>1.6003</td>
<td>0.0612</td>
<td>M1</td>
</tr>
<tr>
<td>MM</td>
<td>106.8060</td>
<td>107.1517</td>
<td>0.0961</td>
<td>2.3785</td>
<td>7.1028</td>
<td>1.3140</td>
<td>0.0659</td>
<td>M1</td>
</tr>
<tr>
<td>TTT</td>
<td>80.2439</td>
<td>81.9895</td>
<td>0.0538</td>
<td>NA</td>
<td>NA</td>
<td>0.3227</td>
<td>0.1684</td>
<td>M1</td>
</tr>
<tr>
<td>Yeast</td>
<td>248.7372</td>
<td>249.7457</td>
<td>0.2403</td>
<td>9.7196</td>
<td>17.3795</td>
<td>3.2181</td>
<td>0.1564</td>
<td>M1</td>
</tr>
<tr>
<td>WQ</td>
<td>437.9335</td>
<td>464.3465</td>
<td>3.0952</td>
<td>71.3376</td>
<td>60.6106</td>
<td>11.2468</td>
<td>1.1739</td>
<td>M2</td>
</tr>
<tr>
<td>CKRKP</td>
<td>479.2124</td>
<td>480.2190</td>
<td>0.4364</td>
<td>NA</td>
<td>NA</td>
<td>1.9075</td>
<td>1.3118</td>
<td>M2</td>
</tr>
<tr>
<td>PBC</td>
<td>489.6242</td>
<td>490.6173</td>
<td>0.2911</td>
<td>15.4500</td>
<td>169.2838</td>
<td>3.7765</td>
<td>0.5542</td>
<td>M2</td>
</tr>
<tr>
<td>PBRHD</td>
<td>383.9472</td>
<td>519.7663</td>
<td>5.8002</td>
<td>92.6616</td>
<td>71.0694</td>
<td>93.9565</td>
<td>1.9735</td>
<td>M3</td>
</tr>
<tr>
<td>MGT</td>
<td>436.3258</td>
<td>459.2743</td>
<td>43.9574</td>
<td>319.9201</td>
<td>267.5094</td>
<td>4.0192</td>
<td>1.4559</td>
<td>M3</td>
</tr>
</tbody>
</table>

Fig. 17. The score of GenClust-H based on Mixed and Random Initial Population on the WQ data set.

Fig. 18. The score of GenClust-H based on Mixed and Random Initial Population on the Yeast data set.
parametric tests are normal distribution of the values and the same variance of two series of values [46].

Since the results do not satisfy the conditions for parametric tests, we carry out a non-parametric sign test [46,47]. The right tailed sign test is carried out for the significance level \( \alpha = 0.05 \) (i.e. 95% significance level). In Fig. 20, we present the sign test results of GenClust-H (compared with the existing techniques) based on fourteen (14) numerical data sets. The first five bars in Fig. 20 show the \( z \)-values (test statistic values) for GenClust-H and the five existing techniques while the sixth bar shows the \( z\text{-ref} \) value. If a \( z \)-value is greater than the \( z\text{-ref} \) value then the results obtained by GenClust-H can be considered to be significantly better than the results obtained by the existing technique. For \( \alpha = 0.05 \) and degree of freedom \( df = 13 \) (we use 14 numerical data sets), the \( z\text{-ref} \) value is 1.7709. Fig. 20 shows that GenClust-H results are significantly better than the five existing techniques for all six evaluation criteria based on the fourteen (14) numerical data sets.

In Fig. 21 we present the similar sign test results for GenClust-F on the fourteen (14) numerical data sets. Figs. 22 and 23 show the sign test results on all twenty (20) data sets for GenClust-H and GenClust-F, respectively. Since AGCUK and GAGR do not work on categorical attributes we exclude them Figs. 22 and 23. The figures indicate that GenClust-H achieves significantly better clustering results than all five existing techniques for all data sets in terms of all six evaluation criteria. GenClust-F also achieves similar results except that its clustering quality is not significantly better than the clustering quality of CRUDAW in terms of the XB values. However, for all other cases it achieves significantly better results than the existing techniques.

Table 6

<table>
<thead>
<tr>
<th>Evaluation criteria</th>
<th>GenClust-H (Best)</th>
<th>GenClust-H (Average)</th>
<th>GenClust-H (Worst)</th>
<th>SK (Best)</th>
<th>SK (Average)</th>
<th>SK (Worst)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xie-Beni Index (lower the better)</td>
<td>0.0818</td>
<td>0.0976</td>
<td>0.1200</td>
<td>0.1801</td>
<td>0.4004</td>
<td>0.9243</td>
</tr>
<tr>
<td>SSE (lower the better)</td>
<td>26.6837</td>
<td>27.7240</td>
<td>30.2891</td>
<td>48.3042</td>
<td>57.0238</td>
<td>84.9774</td>
</tr>
</tbody>
</table>

Table 7

<table>
<thead>
<tr>
<th>Evaluation Criteria</th>
<th>GenClust-H (Best)</th>
<th>GenClust-H (Average)</th>
<th>GenClust-H (Worst)</th>
<th>SK (Best)</th>
<th>SK (Average)</th>
<th>SK (Worst)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xie-Beni Index (lower the better)</td>
<td>0.1015</td>
<td>0.1225</td>
<td>0.1486</td>
<td>0.1746</td>
<td>0.5127</td>
<td>1.1574</td>
</tr>
<tr>
<td>SSE (lower the better)</td>
<td>9.1777</td>
<td>10.4803</td>
<td>12.3430</td>
<td>18.0916</td>
<td>21.9359</td>
<td>30.0220</td>
</tr>
</tbody>
</table>

Fig. 20. Sign test of GenClust-H on the fourteen numerical data sets.
5. Conclusion

The proposed technique aims to achieve better quality clusters without requiring any user inputs such as the number of clusters \( k \). GenClust uses the proposed GA to avoid the user input on \( k \) while achieving high quality clusters. Moreover, it uses 30 default radii values \( r \) ranging from 0.0001 to 0.2, where the domain of \( r \) value is \([0, 1]\). The minimum number of records within the radius of a seed \( T = 1\% \) by default. Since the radii values and \( T \) are interrelated it makes sense to vary one of them while keeping the other one constant. Note that these parameters are used by our genetic algorithm to select the initial population only. The selected population then goes through the genetic operations including crossover, mutation, gene rearrangement, twin removal and fitness evaluation. Finally the genes of the chromosome with the best fitness are used as the initial seeds of K-Means that produces the final clusters. In the experimentation of this study we use the default \( T \) (i.e. \( T = 1\% \)) and \( r \) values for all twenty (20) datasets and find that GenClust achieves better clustering results than all existing techniques used in the experimentation. These results further justify our argument in favor of using the default \( T \) and \( r \) values. Therefore, through the experimentation we demonstrate that GenClust achieves better clustering results without requiring any user input whatsoever.

The basic contributions of the paper are: 1. the novel approach for the selection of an initial population based on the deterministic and random process, 2. the novel gene rearrangement technique, 3. the use of K-Means for further improvement of clustering results obtained from the genetic algorithm, 4. the twin removal operation, and 5. the novel fitness function. Some logical and empirical justifications of the basic components have been presented. Extensive experimentations on twenty natural data sets have been carried out. Experimental results indicate a statistically significant

Please cite this article in press as: M.A. Rahman, M.Z. Islam, A hybrid clustering technique combining a novel genetic algorithm with K-Means, Knowl. Based Syst. (2014), http://dx.doi.org/10.1016/j.knosys.2014.08.011
superiority (according to the sign test analysis) of GenClust over five existing techniques that are very recent and of high quality. The results also demonstrate that GenClust with K-Means and gene rearrangement produces better results than GenClust without the operations. To pay the price of high quality clustering GenClust requires higher execution time than the techniques used in this study. Our future research plans include the reduction of time complexity for GenClust.

The high complexity of the genetic algorithms including GenClust can cause problems for clustering data sets with huge number of records. A possible solution to this problem can be as follows: (1) take a random sample of a manageable number of records (as many as possible) into a sample data set, (2) apply GenClust on the sample data set and get the best chromosome, (3) use the genes of the best chromosome as the initial seeds of the K-Means algorithm which is applied on the whole (not just the sample) data set. Due to the low complexity of K-Means its application on the sample data set and get the best chromosome, (3) use the genes of the best chromosome as the initial seeds of the K-Means algorithm. The complexity to normalize all the values of a numerical attribute is $O(nm)$. The complexity for calculating the degrees of the values of all attributes is therefore $O(znkm)$. Therefore, the complexity for normalizing $m$ attributes is $O(nm^2)$. The overall complexity is then $O(nm^2 + md^2)$.

A.1.2.2. Complexity of record to record distance. The complexity to calculate distance between two records is $O(m)$. For a data set having $n$ records, CRUDAW produces a $n \times n$ distance matrix. For each record, the distances with all other records are calculated. Therefore, the complexity for record to record distance calculation is $O(n^2m)$.

A.1.2.3. Complexity of density calculation. Density of a record $R_i$ is the number of records within its $r_s$ radius. The complexity of calculating the density of all $n$ records using the $n \times n$ distance matrix is $O(n^2)$.

A.1.2.4. Complexity of seed selection. Once the densities are calculated, the complexity to find the record with the highest density (i.e. the first seed) is $O(n)$. The first seed and all records within the $r_s$ radius of the first seed are then removed with a complexity $O(n)$. If we consider that there are at most $k$ seeds ($k$ is a small number) then the total complexity for the seed selection step is $O(n)$.

A.1.3. Fuzzy C-Means

Finally the seeds are given to a Fuzzy C-Means algorithm as input to calculate the final clusters. If the number of iterations is $N$ and number of clusters is $k$ then the complexity of the Fuzzy C-Means algorithm is $O(nmk^2N)[40,41]$. Therefore, finally the total complexity of CRUDAW is $O(nm^2 + md^2 + n^2m + nmk^2N)$. If $n$ is very large compared to all other parameters then the complexity is $O(n^2m)$.

A.2. Complexity of GenClust

A.2.1. Step 1: Initial population

GenClust produces a user defined number of chromosomes deterministically and the same number of chromosomes randomly. For every $r_s$ value it creates a chromosome in the same way as CRUDAW finds the set of initial seeds for the $r_s$ value. Therefore, the complexity for $z$ number of deterministic chromosomes is $O(z(nm^2 + md^2 + n^2m))$. The complexity for $z=\lceil r_s \rceil$ number of random chromosomes is $O(\lceil r_s \rceil k)$, where $k$ is the maximum number of genes in each chromosome.

A.2.2. Step 2: Selection operation

The fitness of each chromosome is calculated using the COSEC function. It computes the distance between all pairs of seeds. If there are $k$ number of seeds the complexity is $O(k\text{choose}2m) = O(k^2m)$. COSEC also computes the distances between each record and its closest seed with a complexity $O(nkm)$. Therefore, the complexity of the fitness calculation for each chromosome is $O(nkm + k^2m)$.

If there are $z$ number of chromosomes the complexity is $O(znkm + zk^2m)$. Once the fitness values of the $z$ chromosomes are computed we need to sort them in descending order for finding the $\lceil \frac{z}{2} \rceil$ best chromosomes. The complexity for this is $O(z^2)$. Therefore, the total complexity of selection operation is $O(znkm + zk^2m + z^2)$.

A.2.3. Step 3: Crossover operation

For the crossover operation, pairs of chromosomes are selected (from the $z$ chromosomes) where the 1st chromosome of each pair is the one with the best fitness among the available chromosomes and the 2nd chromosome is selected by the roulette wheel technique [14,18,28]. In the roulette wheel technique we need to calculate the probability for every chromosome to be chosen for a crossover operation. There will be $\lfloor \frac{z}{2} \rfloor$ crossover operations altogether. Therefore, the complexity of the crossover operation is $O(z^2)$.

In the crossover operation GenClust uses a rearrangement of genes for the target chromosome based on the reference
chromosome. The rearrangement complexity is $O(k^2 m)$ if there are $k$ genes in a chromosome and $m$ attributes in the data set. For the twin removal from a chromosome, GenClust compares the genes of the chromosome. If we have $k$ genes in a chromosome then the complexity is $O(k^2 m)$. Therefore, for $z$ chromosomes the complexity of twin removal is $O(k^2 mz)$.

A.2.4. Step 4: Elitism operation

After the crossover operation we calculate the fitness of all $z$ chromosomes again with the complexity of $O(z k m^2 + z k^2 m)$. In the elitism operation the complexity to identify the best and the worst chromosomes of a generation is $O(z)$ for $z$ chromosomes.

A.2.5. Step 5: Mutation operation

In the mutation operation we need to compute the maximum fitness and the minimum fitness for calculating the mutation probability (see Eq. (5)). The complexity for the calculation is $O(z)$ if there are $z$ chromosomes in the population. If a chromosome is chosen for mutation then an attribute value (randomly chosen) of each and every gene of the chromosome is changed. The complexity of this is $O(k)$ if the number of gene in a chromosome is $k$. Therefore, the complexity of the Mutation operation is $O(kz)$. Additionally, for $z$ chromosomes the complexity of twin removal is $O(k^2 mz)$ as computed before in Step 3.

If there are $N$ iterations then Step 2, Step 3, Step 4 and Step 5 will be repeated $N$ times while the initial population will be selected only once. Therefore, the total complexity of the steps will be $O(nmz^2 + mdz^2 + rmz^2 + N(nmkz + mkz^2 + z^2))$.


If the number of iterations in K-Means is $N$ the number of attributes is $m$ and number of clusters is $k$ then the complexity for Hard K-Means is $O(nmkN^2)$ [40,41].

Therefore, the overall complexity of GenClust (with K-Means) for a high dimensional data set where $m$ is very large compared to all other parameters the complexity of GenClust is $O(mz^2)$. Similarly, the overall complexity of GenClust for a very large data set where the number of records $n$ is very large compared to all other parameters the complexity is $O(n^2)$.

References