Genetic Algorithm with Healthy Population and Multiple Streams Sharing Information for Clustering

A. H. Beg[1], Md Zahidul Islam[1*], Vladimir Estivill-Castro[2]

1-School of Computing and Mathematics, Charles Sturt University, Panorama Avenue, Bathurst 2795, Australia
2- School of Information and Communication Technology, Griffith University, Nathan, QLD 4111, Australia

ARTICLE INFO

Keywords:
Data mining
Clustering
K-means
Genetic algorithm

ABSTRACT

Many popular clustering techniques including K-means require various user inputs such as the number of clusters *k*, which can often be very difficult for a user to guess in advance. Moreover, existing techniques like K-means also have a tendency of getting stuck at local optima. As a result, various evolutionary algorithm based clustering techniques have been proposed. Typically, they choose the initial population randomly, whereas carefully selected initial population can improve final clustering results. Hence, some existing techniques such as GenClust carefully select high quality initial population with a complexity of $O(n^2)$ which is very high. We propose a clustering technique that in addition to selecting initial population with a low complexity of $O(n)$, uses a number of new components including multiple streams, information exchange between neighboring streams, regular health improvement of the chromosomes, and mutation which also aims to improve chromosome health. We compare the proposed technique *HeMI* with five (5) existing techniques on 20 publicly available datasets in terms of two well-known evaluation criteria. We also carry out a through experimentation to investigate the usefulness of the new components of HeMI. Our experimental results demonstrate statistically significant superiority of HeMI over existing techniques, and the effectiveness of the proposed components.

1. Introduction

1.1 What is Clustering?

Nowadays, with the advancement of scientific technology and increase of information, huge amount of data can be collected [8, 11]. It is difficult for a domain expert to infer knowledge manually from the enormous amount of data. To acquire information from the huge amount of data and facilitate decision making process data mining techniques are required.

Clustering is an important and well-known technique in the area of data mining. It aims to group the similar records in one cluster and dissimilar records in different clusters [1, 4, 5, 6, 8, 9].Through clustering the hidden information can be extracted from the data that can help in decision making process [4]. In recent years, clustering has been applied in various areas such as machine learning [10, 12], image segmentation [13-15], business [16, 17], social network analysis [18], medical imaging and object detection [19-21].

1.2 Definitions and Notations

We consider a dataset D to be a two dimensional matrix/table having n records (i.e. rows) and m attributes (i.e. columns). We represent the dataset as $D = \{R_1, R_2, \ldots R_n\}$, where R_i is the ith record. The set of attributes are represented as $A = \{A_1, A_2, \ldots A_m\}$, where A_j is the jth attribute. Each record R_i has $|A|$ attributes. An attribute can be either categorical or numerical. The domain of a numerical attribute A_i is represented as $A_i = [l_i, u_i]$ where l_i is the lower limit and u_i is the upper limit of the domain of A_i. The domain of a categorical attribute A_j is represented as $A_j = \{A_j^1, A_j^2, \ldots A_j^x\}$, where A_j^k is the k-th domain value and x is the domain size of A_j.

1.3 Some Existing Techniques and their Limitations

K-means is one of the most popular techniques for clustering. In k-means, it requires a user (data miner) to define the number of clusters (k) in advance [2]. Based on the user defined number of clusters, it then randomly selects k records as initial seeds form the data set and each record of the data set is then allocated to its closest seed in order to form clusters. The seed of a cluster is then updated based on the records allocated to the cluster. The updated seed is a (real or pseudo) record where each attribute value of the updated seed is the average of all values of the attribute for all records belonging to the cluster.

The process of the record allocation/re-allocation to the clusters and updating is considered to be an iteration of K-means. The iterations continue until any of the termination conditions are met. Typically there are two terminations conditions: first, if the user defined number of maximum iterations is reached then the process terminates and second if the improvement of the objective function values of two consecutive iterations do not improve more than a user defined threshold [1-3].

* Corresponding author. Tel.: +61 2 63384214; fax: +61 2 633 84649.
E-mail addresses: abeg@csu.edu.au (A. H. Beg), zislam@csu.edu.au (Md Zahidul Islam), v.estivill-castro@ griffith.edu.au (Vladimir Estivill-Castro).

While K-means is popular for its simplicity, it has a number of well-known drawbacks [4, 22].One of the main drawbacks of K-means is its requirement of user defined number of clusters (*k*) prior to clustering. The appropriate number of clusters has influence on the quality of final clustering solution [8]. It is difficult for a user (data miner) to estimate the appropriate number of clusters in advance. Another drawback of K-means is that it has a tendency to getting stuck at local optima. Moreover, the random selection of the initial seeds is also considered to be a major drawback since it influences heavily the final clustering quality [3].

A recent technique K-means ++ [3] addresses the last drawback of K-means. It chooses only the first seed randomly. It then chooses the second seed in a probabilistic way so that the record having the highest distance with the first seed has the highest probability to be chosen as the second seed. While choosing the third seed the record having the maximum distance with its nearest seed has the highest probability. Similarly it picks the fourth seed and so on; it picks as many seeds as the user defined number of clusters. All other components of K-means++ are exactly the same as K-means. Hence, it also suffers from other drawbacks of K-means.

To overcome these limitations various methods have been proposed in the recent years, such as stochastic search [23], simulated annealing [24, 25] and genetic algorithms [4, 5, 6, 7, 9, 47, 48]. Genetic algorithms (GA) are randomized search and optimization techniques based on the concepts of Darwin's law of evolution "Survival of the fittest in natural selection" proposed by John H. Holland [26]. This algorithm simulates the biological structure of the genetic evolution process.

In recent years, many GA based clustering techniques have been proposed such as GenClust [4], AGCUK [5], GAGR [6] and so on. In GA a chromosome contains a set of genes, where a gene is a (real or pseudo) record. A gene is considered to be the center of a cluster. Therefore, a chromosome is considered to be a clustering solution. GA generally contains many iterations/generations. Each generation typically contains a number of chromosomes that are known as the population of the generation. GA consists of five main phases namely population initialization, selection, crossover, mutation and elitist operation.

Many existing techniques generate the number genes of a chromosome randomly, in the population initialization phase. They also randomly choose records as genes, instead of carefully choosing genes of a chromosome. Careful selection of genes can create an initial population containing high quality chromosomes. High quality initial population typically increases the possibility of obtaining a good clustering solution at the end of the genetic processing [4, 27, 28]. An existing technique called GenClust [4] finds high quality initial population and thereby obtains good clustering solution.

However, its initial population selection process is very complex with a complexity of $O(N^2)$, where N is the number of records in a dataset. Moreover, GenClust requires a user input on the number of radius values for the clusters in the initial population selection. It can be very difficult for a user to guess the set of radius values (i.e. radii).

1.4 Novel Components of the Proposed Technique and their Advantages

In this paper, we propose a genetic algorithm based clustering technique called "Genetic Algorithm with Healthy Population and Multiple Streams Sharing Information for Clustering (HeMI)". We now briefly introduce the novel components/properties of HeMI and their logical justifications as follows.

Following the approach of some existing techniques, HeMI also uses a high quality initial population. It does not require any user input on the radii of clusters and keeps the complexity of the initial population selection operation low, $O(N)$. It also uses a set of randomly selected chromosomes in the initial population in order to maintain both high quality and randomness, since genetic algorithms also require randomness.

It is evident from the literature [29, 30] and through our empirical analysis (carried out in this study) that the population size has a positive impact on the clustering quality. That is, a big population size is likely to contribute towards a good clustering solution. However, big population size requires high time complexity. Therefore, HeMI uses a big population in multiple streams, where each stream contains a relatively small number of chromosomes.

Various genetic operations (such as crossover and mutation) are applied on each stream in parallel. As a result HeMI is likely to produce better quality clustering solutions within reasonable time. Moreover, due to splitting the chromosomes into a number of streams and processing the splits separately HeMI exhibits higher ability to explore the solution space than the traditional approach of processing all chromosomes in a single stream. We present empirical evidence of this phenomenon where we use a single stream of 20 chromosomes, 40 chromosomes and 80 chromosomes, and four streams of 20 chromosomes.

Note that there are some existing techniques that use parallel genetic algorithms [29, 33, 34, 35] where they *divide* the total number of chromosomes into a number of parallel runs, whereas in our technique we *increase* the total number of chromosomes. The main goal of these existing techniques is to reduce the time complexity through the parallelization of the genetic algorithms, whereas the main goal of HeMI is to improve clustering results. While employing parallelization these existing techniques do not share information among the parallel streams, whereas HeMI introduces information sharing among the streams at a regular interval in order to take advantage of the multiple streams.

For a stream S_i, HeMI first identifies its neighboring streams and then spots out the best chromosome from all neighboring streams and S_i. It then replaces the worst chromosome of S_i by the best chromosome. The information sharing is carried out at a regular interval such as at every 10^{th} iteration.

For Stream 1, Stream 2 and Stream 3 are considered to be neighbors. Similarly for Stream 2, Stream 3 and Stream 4 are considered to be neighbors. While the sharing of the best chromosome from the neighbors increases the fitness of the best chromosome, it maintains the divergence among the streams. That is, had HeMI used/inserted the best chromosome out of all streams into all streams then they would have the same best chromosome in all streams.

The presence of healthy chromosomes (i.e. chromosomes with high fitness values) in a population can increase the possibility of good clustering results. Therefore, HeMI replaces the sick chromosomes (i.e. chromosomes with low fitness) by healthy chromosomes. Some of the healthy chromosomes are chosen from a pool of healthy chromosomes obtained by the initial population operation, whereas some of the healthy chromosomes are generated through the crossover operation of the existing healthy chromosomes of a generation with the hope that the crossover of two healthy chromosomes may generate new healthy chromosomes. The crossover operation aims to maintain some randomness even in the process of replacing sick chromosomes by healthy chromosomes.

Randomness is also maintained through the mutation operation which employs a division and absorption operation in sequence if they improve the quality of clustering solutions. Additionally, at the end of the division and absorption operation it also applies a random change in chromosomes. Unlike HeMI, an existing technique [5] applies either division or absorption randomly. Another existing technique [7] applies division (they call it splitting) on the largest cluster instead of the sparsest cluster, and absorption on two randomly chosen clusters. Another existing technique [9] applies division and absorption of randomly chosen clusters. Hence, HeMI has a better approach to carefully improve clustering quality through mutation while exploring unconventional solution space.

1.5 Evaluation Techniques

We evaluate our technique by comparing its performance with the performance of five high quality techniques namely AGCUK [5], GAGR [6], GenClust [4], K-means [2], K-means ++ [3]. We conduct experiments for the techniques on twenty (20) real life datasets that are available in the UCI machine learning repository [31]. The experimental results clearly indicate that the proposed technique performs significantly better than other techniques in terms of the evaluation criteria considered in this study: silhouette coefficient and DB Index. We also experimentally evaluate the usefulness of various components of HeMI.

1.6 Main Contributions of this Study

Note that HeMI is a significant improvement of our previous technique called DeRanClust that we published in a conference [32] and have never published in any journal. We consider this paper as a significant extension of the conference paper where the only similarity is the initial population selection process. The main contribution of DeRanClust was as follows:

- The initial population was selected through deterministic and randomly chosen chromosomes.

However, the original contributions of HeMI are as follows:

- The use of multiple streams (see Section 2.2.2).
- High quality initial population selection maintaining some randomness (see Section 2.2.3).
- The three steps mutation operation (see Section 2.2.7).
- The Health improvement operation (see Section 2.2.8).
- Neighbor information sharing (see Section 2.2.10).
- The Global Best Selection operation (see Section 2.2.11)
- HeMI works on datasets having numerical and/or categorical attributes.

The rest of the paper is organized as follows: in Section 2 we provide a brief literature review on some existing techniques. Section 3 describes our proposed technique. We discuss the experimental result in section 4 and in section 5 we provide the concluding remarks.

2. Our Technique

2.1 Basic Concepts

In this section we introduce the basic concepts of various components of the proposed technique called HeMI and present a logical justification of the basic concepts. One important component of HeMI is the initial population selection aiming to get high quality initial population. The basic idea here is high quality initial population is likely to result in high quality final clustering solution as evident in the literature [4, 27, 28]. HeMI uses K-means/K-means ++ many times with a set of different *k* values to explore the best *k* value for a dataset and then use the best chromosomes (obtained by the best *k* value/s) in the initial population. We carry out empirical analysis (presented in Section 3.5.2) that supports the basic concept of using high quality initial population.

Another important component of HeMI is multiple streams. It is evident from relevant literature that [29, 30] in genetic algorithm based clustering techniques bigger population size tends to increase the quality of the final clustering solution. Therefore, we realize that a population size of 80 chromosomes is more likely to produce better clustering solution than a smaller population size such as 20 chromosomes. In this study we carry out empirical analysis on this (presented in Section 3.5.1) where we can see the improvement of clustering quality with the increase of population size in an existing genetic algorithm based clustering technique called AGCUK [5] and GenClust [4].

An obvious issue related to the increase of population size is the increased time complexity. In order to reduce the time complexity HeMI uses multiple streams, where in each stream it uses smaller population size. The multiple streams can be processed in parallel to reduce the time complexity. Another advantage of using the multiple streams is better

exploration of clustering solutions. That is, if we run all chromosomes in a single stream (like traditional techniques) then we get one best chromosome from the whole population, whereas if we divide the chromosomes in multiple streams and run them independently then we get multiple best chromosomes; one best chromosome from each stream. We naturally expect this approach to get better clustering solution. The empirical analysis carried out in this study (presented in Section 3.5.1) also supports the expectation.

While running multiple streams independently we also make them help each other in achieving better clustering solutions. That is, the independent streams can exchange message at a regular interval in order to increase the clustering quality of each stream. One way we could do this is by identifying the best chromosome out of all streams and implanting the chromosome in each stream. However, in that case all streams would have the same best chromosome and would lose the diversity among them. Therefore, for each stream we first identify a set of neighboring streams and then identify the best chromosome within the neighboring streams which is then implanted into the stream. This way we can ensure that all streams will not have the same best chromosome in them. Our empirical analysis again shows (presented in Section 3.5.1) a clear evidence that the use of multiple streams with sharing information among the neighboring streams result in better clustering solutions.

Another interesting idea of HeMI is the continuous health improvement in every generation in order to ensure the presence of high quality chromosomes in each population. In each population it identifies a number of sick chromosomes and replaces them by healthy chromosomes. Some of the healthy chromosomes are obtained from the pool of high quality chromosomes created for the initial population using K-means/K-means ++ many times. Moreover, some of the healthy chromosomes are created by applying the crossover operation on pairs of good chromosomes. Again our empirical analysis indicates the effectiveness of the health improvement as presented in Section 3.5.4.

The mutation operation generally changes some chromosome randomly [4, 7, 9]. However, HeMI aims to use the mutation operation for improving the chromosome health while changing them randomly. The mutation operation in HeMI has three components: division, absorption and random change. In the division operation, it examines whether dividing the sparsest cluster into two separate clusters can improve the chromosome health. Similarly in the absorption operation, it examines whether the chromosome health can be improved by merging the two closest clusters. After the division and absorption operation it finally makes a slight change randomly. The effectiveness of this mutation operation has been empirically analyzed in Section 3.5.3.

2.2 Main Components

In this subsection we introduce the main components of the proposed technique, HeMI before we present the complete algorithm and steps of HeMI in the next subsection.

2.2.1 Component 1: Normalization

HeMI first normalizes a dataset D (See Algorithm 1) in order to weigh each attribute equally regardless of their domain sizes [4]. The normalization brings the domain range of each numerical attribute between 0 and 1. It generates a normalized attribute value $X_N = \frac{X_{Max}-\mu}{X_{Max}-X_{Min}}$, where X_{Max} is the maximum, X_{Min} is the minimum, and μ is the average domain value of the numerical attribute. The distance between records is computed using the Euclidean distance metric [37-39]. Hence, the distance between two records for a numerical attribute can vary between 0 and 1. For a categorical attribute, HeMI uses an existing technique [36] to compute the similarity *S* between two categorical values of the categorical attribute. The distance between two values of a categorical attribute $d = 1 - S$. The similarity *S* varies between 0 and 1 and hence the distance *d* also varies between 0 and 1. As a result the distance between any two records varies between 0 and 1 and all attributes have equal weight in the distance calculation.

2.2.2 Component 2: Multiple Streams

This component is an original/new contribution of HeMI that aims to take advantage of using a big population through multiple streams where each stream contains a relatively small number of chromosomes. Generally in the genetic algorithm based clustering techniques the bigger population size tends to increase the quality of final clustering solutions [29, 30]. Therefore, HeMI aims to use a big population in order to produce better clustering solution. However, big population can increase the time complexity. Thus, HeMI uses multiple streams where each stream contains a subpopulation. It can process the streams in parallel in order to reduce the time complexity. The chromosomes for each stream are generated separately through the population initialization. Various components such as crossover and mutation are applied on each stream separately.

2.2.3 Component 3: Population Initialization

This is another new/original contribution of HeMI that selects high quality chromosomes in the initial population through two phases: a deterministic phase and a random phase. The proposed technique selects 50% of the chromosomes of the initial population through the deterministic phase and the remaining 50% chromosomes through the random phase.

For the deterministic phase HeMI uses a set of predefined numbers of genes/clusters *k*. The default set of predefined *k* is {2, 3, …. 10} where the size of the set is nine. HeMI uses each element of the set as the number of clusters (*k*) for K-means/K-means++ and thus produce a clustering solution i.e. chromosome. For each element it run K-means/K-means++ five times and thus produces five chromosomes. That is it produces altogether 5 x 9 = 45 chromosomes in the deterministic phase.

Due to the use of K-means/K-means++ HeMI expects to get high quality chromosomes for a given *k* value. Since HeMI does not know the actual *k* in a dataset it explores numbers from 2 to 10. Typically the *k* value for a dataset varies between 2 and 10, which is supported by our empirical analysis on the datasets in the UCI machine learning repository [31]. In the UCI repository there are 157 datasets for which the class sizes (i.e. the domain sizes of the class attributes) have been reported. The domain size of the class attribute of a dataset is indicative to the number of clusters in the dataset. The mean and standard deviation of the class sizes of the datasets are 5.36 and 5.49, respectively. That is, the number of clusters of a data set typically varies between 2 and 10. Hence, HeMI uses the set of *k* {2, 3,…..10}, in the deterministic phase.

However, the actual *k* values in many datasets are more than 10. In order to handle such situations HeMI uses the random phase where it generates 45 chromosomes. For each chromosomes, it randomly generates the *k* value between 2 and $\sqrt{N}$(N is the number of records in a dataset) and then randomly picks *k* records to form *k* genes of the chromosome.

HeMI by default uses 20 chromosomes in the population of a generation. Therefore, it chooses the best 10 chromosomes from the 45 chromosomes generated in the deterministic phase and the best 10 chromosomes from the 45 chromosomes generated in the random phase. While the use of K-means/K-means++ helps to get high quality chromosomes the use of the random approach helps to explore the solution space through its randomness.

The best chromosome out of the 20 chromosomes of the initial population is stored separately as the best chromosome which is then used in the elitist operation later on. The DB Index [40] is used by default as the fitness function of the chromosomes throughout all steps in HeMI. A small DB Index value indicates a good clustering result and therefore, the fitness of a chromosome is computed by 1/DB [5] where the chromosome with low DB Index has a higher fitness value.

2.2.4 Component 4: Noise Based Selection

At the beginning of each generation starting from Generation 2, we carry out the Noise Based Selection [5] in order to get a new population for subsequent genetic operations such as crossover and health improvement. This component was originally proposed in the literature [5].

The chromosomes of two generations are compared pair wise. If we have twenty chromosome in the current (*i*-th) generation P_1^i, P_2^i, … P_{20}^i and twenty chromosome in the previous (*i-1*)-th generation P_1^{i-1}, P_2^{i-1}, … P_{20}^{i-1} then P_j^{i-1} and P_j^i are compared pairwise, $\forall j$. If the fitness of chromosome P_j^{i-1} and P_j^i are f_j^{i-1} and f_j^i, respectively, and $f_j^{i-1} > f_j^i$ then we choose P_j^{i-1} if $f_j^{i-1} > f_j^i + noise > 0$, otherwise we choose P_j^i. The *noise* value ranges between 1 and 0. We use this component to introduce a randomness in selecting the chromosomes of a generation, where the chances are high that the chromosome with higher fitness will be chosen (especially when the difference between the fitness values is big), but still in some cases it may not happen.

2.2.5 Component 5: Crossover Operation

HeMI performs a crossover operation on a pair of chromosomes where the chromosomes swap their segments/genes to each other and generate a pair of offspring [4, 7, 9].Typically, there are many selection criteria such as roulette wheel [9, 10, 41] rank-based wheel [7] and random selection [6] that are used to select a chromosome pair for a crossover operation. In HeMI, the best chromosome (which is currently available in the population) is chosen as the 1st chromosome of the pair. The 2nd chromosome of the pair is chosen using the roulette approach, where a chromosome P_j is chosen with a probability $T_j = (f_j / \sum_{j=1}^{|P|} f_j)$. Here, f_j is the fitness of the chromosome P_j and $|P|$ is the size of the current population. Once a pair of chromosomes is chosen it is removed from the current population. For the selection of the next pair, again the new best chromosome is chosen. The 2nd chromosome of the pair is chosen using the same process described above. The intuition behind the roulette wheel selection is to take a non-deterministic approach with high probability of choosing a pair of good chromosomes.

There are many approaches to perform crossover between a pair of chromosome such as single point [4, 9, 42], multi-point [7], arithmetic [43] and path-based crossover [6]. However, Peng et al. [42] in some experiments demonstrates that single point crossover performs better than the multi-point crossover.

Therefore, in the proposed technique the single point crossover is used where it randomly generates a crossover point for each chromosome of the pair in order to divide a chromosome into two segments and then swaps the segments between the chromosomes.

2.2.6 Component 6: Twin Removal

Two identical genes can somehow be generated in a chromosome [4]. Therefore, we use the twin removal approach [4] to remove/change the identical genes. If the length of a chromosome is more than two then while there are two identical genes we delete one of the two identical genes. Thus the length of the chromosomes is decreases by one. If the length of a chromosome is two and both the genes are identical then we randomly change one of the two identical genes in order to make sure that the genes are not identical.

2.2.7 Component 7: Three Steps Mutation Operation

This is another new contribution of HeMI that changes a chromosome using three steps/operations: division, absorption and a random change. Note that the division and absorption operations are also used in some existing techniques [5, 7, 9], but there are differences between them and HeMI as follows.

Chang et al [9] applies both division and absorption on a chromosome where clusters are chosen randomly for division and absorption, unlike HeMI that carefully chooses clusters for division and absorption. Blas et al [7] also chooses a cluster for division carefully, where the largest cluster is chosen for division. We argue that a large cluster can also be compact and may not always require to be divided into smaller clusters. Therefore, HeMI applies division on the sparsest (not largest) cluster of a chromosome. Moreover, Blas et al [7] chooses two random clusters for absorption, whereas HeMI chooses the two closest clusters for absorption.

Liu et al [5] also chooses the sparsest cluster for division and two closest clusters for absorption. However, they randomly apply either division or absorption on a chromosome, regardless of the improvement of its fitness. However, HeMI applies both division and absorption on a chromosome. Division/absorption is applied only if it improves the fitness of a chromosome. Additionally after the division and absorption operation, HeMI also applies the random change operation on a chromosome based on a mutation probability in order to support the exploration nature of genetic algorithms.

In the division operation HeMI identifies the sparsest cluster C_{i} of a chromosome P_{i} and then divides the cluster C_{i} into two clusters by applying K-means on C_{i} where the value of *k* is set to 2. If the fitness of the chromosome after division P_i^d is better than the fitness of the chromosome P_{i} then P_i^d is selected, otherwise P_{i} is selected for the absorption operation. The absorption operation finds the two closest clusters C_{i} and C_{j} of the chromosome P_{i} or P_i^d (whichever is selected from the division operation), and merges them into one cluster. Thus it forms a new chromosome P_i^a. If the fitness of P_i^a is better than the fitness of P_{i} and P_i^d then P_i^a is selected, otherwise either P_{i} or P_i^d (whichever is selected from the division operation) is selected for the random change operation.

Once the division and absorption operations for all chromosomes of a population are completed then the random change operation is carried out. In the random change operation the mutation probabilities for each chromosome are computed. The mutation probability of a chromosome P_{j} is calculated using its fitness f_{j}, and the maximum fitness value $f_{\max}$ and average fitness value f_{mean} of all chromosomes in the current population. The mutation probability [6, 44] of the *j*-th chromosome is calculated as follows, where k_1 and k_2 are equal to 0.5.

$$T_j = \begin{cases} k_1 * \frac{f_{max} - f_j}{f_{max} - f_{mean}} & if\ f_j > f_{mean}, \\ k_2\,, & if\ f_j \leq f_{mean} \end{cases} \quad (1)$$

The intuition behind this is to reduce the amount of random changes on good chromosome. The T_j value for the chromosome having the best fitness is zero. The T_j value increases for the chromosome with lower fitness value. The T_j value is 0.5 for all chromosomes having fitness less than the average fitness.

If the mutation probability of a chromosome is greater than a random number (between 0 and 1) then the chromosome is selected for the random change operation, otherwise it remains unchanged. In the random change operation, a gene of the chromosome is randomly chosen where an attribute value of the gene is randomly changed to another value within its domain.

2.2.8 Component 8: Health Improvement Operation

This is an original contribution of HeMI. The aim of this component is to continuously improve the health of chromosomes in every generation in order to ensure the presence of high quality chromosomes within a population. Crossover and mutation operations are likely to improve health/fitness of some chromosomes, but they can also decrease health/fitness of some chromosomes. Therefore, after the crossover and mutation operations HeMI identifies sick chromosomes and replaces them through three phases.

In Phase 1, HeMI identifies the healthy and sick chromosomes. It sorts the chromosomes in descending order of their fitness values and identifies 50% of the chromosomes to be *sick*. For example, if there are 20 chromosomes in a population (i.e. population size = 20) then it identifies the 10 sickest chromosomes to be *sick* and the others to be *healthy*. The sick chromosomes are then removed from the population. So the population size temporarily decreases to 50% where all of them are considered to be healthy. In the following two phases 50% new chromosomes are added to bring the population size back to 100%.

In Phase 2, HeMI generates 20% new chromosomes i.e. if the original population size is 20 then it generates 4 chromosomes. For this, it first picks the healthiest 20% chromosomes from the set of *healthy* chromosomes found in Phase 1. Applying the same approach of Component 5 it then chooses pairs of chromosomes from these 20% healthy chromosomes. It next applies the crossover operation on each pair in order to generate offspring chromosomes which are then added into the population. Hence, at this stage the population size is back to 70% of the original size.

In Phase 3, HeMI adds the remaining 30% chromosomes in the population. These chromosomes are chosen from the *pool* of chromosomes that was obtained through the deterministic phase of Component 3 which are supposed to be healthy chromosomes due to the use of K-means/K-means++. Moreover, in this phase HeMI chooses the best chromosomes of the *pool*. For each of these chromosomes HeMI then randomly changes an attribute value of a gene within its original domain. These chromosomes are then added into the population to bring the population size back to 100%.

2.2.9 Component 9: The Elitist Operation

The Elitist operation keeps track of the best chromosome throughout the generations in order to ensure the continuous

Algorithm 1: HeMI

Input: A dataset D having N records and |A| attributes, where A is the set of attributes
Output: A set of cluster C
Require:
    $P_s \leftarrow \emptyset$ /* P_s is the set of initial population (20 chromosomes), initially set P_s to empty */
    $P_o \leftarrow \emptyset$ /* P_o is the set of offspring chromosomes, initially set P_o to empty*/
    $P_m \leftarrow \emptyset$ /* P_m is the set of mutated chromosomes, initially set P_m to empty*/
    $p_c \leftarrow \emptyset$ /* p_c is the set of healthy chromosomes, initially set p_c to empty*/
    $D' \leftarrow$ Normalized (D) /* normalize each attribute of the data set in the normalized data set (D') */
    $P_d \leftarrow \emptyset$ /* P_d is the set of deterministic chromosome (45 chromosomes), initially set P_d to empty */
    $P_r \leftarrow \emptyset$ /* P_r is the set of random chromosome (45 chromosomes), initially set P_r to empty */
end
for k= 1 to m **do** /* m=4, user defined number of streams, default value of m is set to 4 and k is the counter of m */
    Step 1: /* Population Initialization */
        $P_d \leftarrow$GenerateDeterministicChromosome (D') /* generate P_d through K-means, the number of initial seeds for k-means are chosen deterministically */
        $P_d \leftarrow$SelectedDeterministicChromosome (P_d) /* select 10 chromosomes (50% chromosomes of the initial population) based on fitness */
        $P_r \leftarrow$GeneratRandomChromosome (D') /* generate P_r randomly, the number of initial seeds are chosen randomly and the seeds also chosen randomly */
        $P_r \leftarrow$RandomChromosomeSet (P_r) /* select 10 chromosomes (50% chromosomes of the initial population) based on fitness */
        $P_s \leftarrow P_s \cup (P_r \cup P_d)$ /* insert P_r and P_d into P_s */
        P_b= FindBestChromosome (P_s) /* P_b is a chromosome that has the maximum fitness value in P_s */
    end
end
for j=1 to G **do** /* default G=5, G is the number of intervals of the total number of iterations and j is the counter of G */
    for k= 1 to m **do** /* default m=4, m is the user defined number of streams, and k is the counter of m */
        for t=1 to I **do** /* default I = 10, I is the user defined number of iterations for each interval and t is the counter of I */
            Step 2: /* Noised Based Selection */
                if t >1 **then**
                    P_s = NoiseBasedSelection(P_s^i, P_s^{i-1}) /* perform noise based selection between current (P_s^i) and previous (P_s^{i-1}) generation*/
                end
            end
            Step 3: /* Crossover operation */
                $P_o \leftarrow$ PerformCrossover (P_s) /*perform single point crossover on P_s and get offspring P_o*/
            end
            Step 4: /*Twin Removal */
                P_0 =Twin Removal (P_0) /*perform twin removal on P_0 and get a set of chromosome P_0 */
            end
            Step 5: /* Three Steps Mutation operation */
                while $|P_0| \geq 0$ **do**
                    $P_m \leftarrow$ DivisionOperation (P_0) /*perform division operation on P_o and get chromosome P_m */
                    $P_m \leftarrow$ AbsorptionOperation (P_m) /*perform absorption operation on P_m and get chromosome P_m */
                    $P_m \leftarrow$ RandomChangeOperation (P_m) /*perform random change operation on P_m and get chromosome P_m */
                end
            end
            Step 6: /* Health Improvement Operation */
                P_x = MajoritySelection (P_m) /* select 10 best chromosomes from P_m based on their fitness */
                P_y = MinoritySelection (P_m) /* select 4 best chromosomes from P_m and perform single point crossover and get offspring P_y */
                P_z = MidSelection (P_d) /* select 6 best chromosomes from P_d and perform random mutation and get chromosomes P_z */
                $P_c \cup (P_x \cup P_y \cup P_z)$ /* insert P_x, P_y and P_z into P_c */
            end
            Step 7: /* Elitist Operation */
                $P_b^k \leftarrow$ElitistOperation (P_c & P_b) /* apply elitist operation on P_c & P_b and find the best chromosome P_b^k */
                $P_g \leftarrow P_b^k$ /* insert P_b^k into P_g, P_g is the set of chromosomes that contains the best chromosome of each stream */
            end
        end
    end
    Step 8: /* Neighbor Information Sharing */
        for k= 1 to m **do** /* default m=4, m is the user defined number of streams and k is the counter of m */
            P_w^k = FindWorstChromosome (P_c) /* find the worst chromosome P_w^k in P_c */
            $P_w^k \leftarrow$ ReplaceWithNeighborBestChromosome (P_g) /* replace W_p^k with the best chromosome of its neighbor */
            $P_b^k \leftarrow$FindLocalBestChromosome (P_c) /* find the local best chromosome */
            $L_b \leftarrow P_b^k$ /* insert P_b^k into L_b, L_b is the set of chromosomes that contains the best chromosome of each k */
        end
    end
end
Step 9: /* Global Best Selection*/
    C $\leftarrow$FindGlobalBestChromosome (L_b) /* find the global best chromosome C from L_b*/
    Return C
end

improvement of the quality of the best chromosome found so far over the iterations. The operation is applied on a population at the end of all other operations in a generation. If the fitness of the worst chromosome P_w^i of the i-th population (i.e. the current population) is less than the fitness of the best chromosome P_b^{All} found so far from all previous generations then P_w^i is replaced by P_b^{All} in the current population. Moreover, if the best chromosome of the current population P_b^i has higher fitness than the fitness of P_b^{All} then P_b^i is copied in P_b^{All}, replacing its old value.

2.2.10 Component 10: Neighbor Information Sharing

This is a new contribution of HeMI where neighboring streams share/exchange the best chromosome among them, at a regular interval such as at every 10th generation. If a stream somehow suffers from the low quality of its best chromosome then it gets an opportunity to borrow the best chromosome from its neighboring streams.

For a stream S_i, HeMI first identifies its neighboring streams. The streams $S = \{S_1, S_2 \ldots S_{|S|}\}$ are number sequentially where $|S|$ is the user defined number of streams. The default number of streams is four in this study. For any stream S_i, the two streams $S_{i+1 \ MOD \ |S|}$ and $S_{i+2 \ MOD \ |S|}$ are considered to be the neighboring streams. The *MOD* operation ensures that the neighbors are found in a wrap up fashion where for the stream $S_{|S|}$ the neighboring streams will be S_1 and S_2.

For a stream S_i, HeMI spots out the best chromosome P_b out of its neighboring streams and S_i. It then replaces the worst chromosome of S_i by P_b, if P_b comes from a neighboring stream of S_i. The chromosomes of S_i are then sorted and the best chromosome is stored as P_b^{All} which is the best chromosome found so far for S_i, as explained in Section 2.2.9.

While the sharing of the best chromosome from the neighboring streams increases the fitness of the best chromosome S_i, it maintains the divergence among the streams since the sets of neighboring streams for any two streams S_i and S_j are different.

2.2.11 Component 11: Global Best Selection

This is another contribution of HeMI. At the end of all iterations/generations each stream has a best chromosome for the stream. HeMI compares all such best chromosomes from all streams and then select the best of the best chromosomes as the final clustering solution. The genes of the best chromosome represent the cluster centers and records are allocated to their closest seeds to form the final clusters.

2.3 The HeMI Algorithm:

After introducing the main components we are now ready to present the overall algorithm of HeMI. HeMI takes a dataset D as input. It first normalizes all numerical attributes separately as explained in Section 2.2.1. HeMI uses a user defined number of multiple streams as explained in Section 2.2.2. The use of multiple streams aiming to improve clustering results is an original contribution of HeMI. The default number of multiple streams is four (4) in this study. Each stream contains a subpopulation (see Section 2.2.2) in the sense that the total number of chromosomes is equally (or as equally as possible) divided among the streams.

HeMI then generates initial chromosomes for each stream separately through its proposed Population Initialization component (see Section 2.2.3 and Step 1 of Algorithm 1). It skips the Noise Based Selection operation in the first iteration as shown in Step 2 of Algorithm 1. The Noise Based Selection operation is applied from the 2nd iteration.

The single point crossover, Twin Removal, Mutation, Health Improvement and Elitist operation are then applied sequentially. All these operations are described before (see from Section 2.2.5 to Section 2.2.9). They can also be studied in various steps (from Step 3 to Step 7) of Algorithm 1. The Mutation and Health Improvement operation are also original contributions of HeMI.

In order to take the advantage of the multiple streams HeMI then performs the Neighbor Information Sharing operation at a regular interval, which is by default 10 iterations. This operation has been explained in Section 2.2.10 and Step 8 of Algorithm 1. This is another original contribution of HeMI. At the end of all iterations HeMI applies the Global Best Selection operation in order to find the final clustering solution (see Section 2.2.11 and Step 9 of Algorithm 1). This is also an original contribution of HeMI.

3. Experimental Results and Discussion

3.1 The Datasets and the Evaluation Criteria

We empirically compare our proposed technique called HeMI with five existing techniques namely K-means [2], K-means ++ [3], GAGR [6], AGCUK [5] and GenClust [4] on twenty (20) natural datasets that are available from the UCI machine learning repository [31]. HeMI is compared with these existing techniques because they are recent and were shown to be better than many other high quality techniques [45-49].

Detailed information on the datasets is provided in Table 1. We choose datasets with wide variety. For example, some datasets (such as Glass identification) have only numerical attributes, and some datasets (such as BC) have categorical attributes. The Credit Approval (CA) dataset has 6 numerical and 9 categorical attributes. The reason why we choose most of the datasets with only numerical attributes is that all techniques (except HeMI and GenClust) that we use in this study can handle only numerical attributes.

Some datasets have low number of attributes such as Blood Transfusion (BT) that has only 4 attributes and some datasets have high number of attributes such as Dermatology (DT) that has 34 attributes. Similarly, some datasets have low number of records such as Glass identification that has 214 records and some datasets have relatively high number of records such as MGT that has 19,020 records. Also some datasets have low number of class values (i.e. low domain size of class attributes) such as BC that has only two class

values, but some datasets have high number of class values such as LF that has 36 class values.

Class values are the labels of records which show an important property of a dataset. Typically, clustering algorithms are applied on datasets that do not have any class values. Hence, we delete the class attributes from all datasets prior to any experimentation.

Some of the datasets contain missing values in them, meaning that for some records some attribute values are missing. Column 2 of Table 1 shows the total number of records of the datasets including the records that have some missing values. We first delete the records having any missing value/s. Column 3 of Table 1 shows the number of records without missing value/s. For example, the BC dataset has altogether 286 records, but 9 of them have one or more missing values. Hence, after these 9 records are deleted the dataset has 277 records without any missing values. In all experiments, we use the datasets without any missing values.

We evaluate and compare the clustering techniques based on two well-known evaluation criteria namely Silhouette Coefficient [1, 7] and DB Index [40]. Note that higher values of Silhouette Coefficient represent better clustering results, whereas lower values of DB Index indicate better clustering results.

3.2 The Parameters used in the Experiments

In the experiments on AGCUK [5], GAGR [6], GenClust [4] and HeMI we consider the population size to be 20 and number of iterations/generations to be 50. We maintain this consistency for all of these techniques in order to ensure a fair comparison among them.

In the experiments, the number of iterations of K-means/K-means ++ in HeMI is set to 50 and the number of iterations of K-means in GenClust is also set to be 50. The cluster numbers in GAGR, K-means and K-means ++ is user defined. However in order to simulate a natural scenario, the cluster number for GAGR, K-means and K-means ++ are generated randomly between 2 and $\sqrt{N}$, where N is the number of records in a dataset.

The number of iterations for K-means and K-means ++ is also set to 50 and the threshold value is set as 0.005 that was suggested in GenClust. The value of r_{max} and r_{min} for AGCUK and HeMI is set to be are 1 and 0 respectively as suggested in AGCUK.

3.3 The Experimental Setup

For each dataset, we run HeMI 20 times since it can produce different clustering results in different runs. We then present the average and standard deviation of the clustering results. We also run all other techniques AGCUK, GAGR, K-means, K-means ++ and GenClust 20 times. We present the average and standard deviation of the clustering results.

We run each of the techniques 20 times on all 20 datasets. Moreover, in order to evaluate the effectiveness of various components of HeMI we use 5 datasets where we run the techniques 20 times.

3.4 Experimental Results on All Techniques

In this section we experimentally evaluate the performance of HeMI by comparing it with K-means, K-means ++, GGAR, AGCUK and GenClust on all 20 datasets where each technique runs 20 times on each dataset. Since there are 2 datasets with categorical attributes AGCUK, GAGR, K-means and K-means++ cannot handle these datasets and therefore, these techniques are tested on 18 (instead of 20) datasets. However, HeMI and GenClust are tested on all 20 datasets.

Fig. 1a and Fig. 1b show the average and standard deviation of the Silhouette Coefficient of the clustering solutions, where HeMI achieves better results than GenClust in 20 out of 20 datasets. That is, in 20 out of 20 datasets the average Silhouette Coefficient of 20 runs of HeMI is higher than the average Silhouette Coefficient of 20 runs of GenClust. Moreover, in 17 out of 20 datasets the standard deviations of HeMI do not overlap the standard deviations of GenClust and the average Silhouette coefficient of HeMI is higher than GenClust. Note that the cases where the standard deviations of HeMI overlap with the standard deviations of other techniques are indicated with an arrow in Fig. 1a, Fig. 1b, Fig. 2a and Fig. 2b. That is, for all cases without an arrow HeMI achieves better result with no overlap of standard deviations.

HeMI achieves higher Silhouette Coefficient than AGCUK in 20 out of 20 datasets. The standard deviations of HeMI do not overlap the standard deviations of AGCUK in 19 out of 20 datasets. HeMI also achieves higher Silhouette Coefficient than K-means, K-means++ and GAGR in all 20 datasets. In 20 out of 20 datasets the standard deviations of HeMI do not overlap with the standard deviations of any of these techniques.

All bar graphs in Fig. 1a, Fig. 1b, Fig. 2a and Fig. 2b are in the same sequence: K-means, K-means++, GAGR, AGCUK, GenClust and HeMI. As we can see in Fig. 2a and Fig. 2b, HeMI achieves better clustering results (on an average) than GenClust in 19 out of 20 data sets, based on DB Index for which a lower value indicates a better result. In 18 out to 20 datasets HeMI does not have any overlap of standard deviations with the standard deviations of GenClust. Moreover, HeMI performs better than K-means, K-means ++, GAGR and AGCUK on all 20 datasets based on DB Index. In 20 out to 20 datasets HeMI does not have any overlap of standard deviations with the standard deviations of these techniques.

The right most columns of Fig. 1b and Fig. 2b show the average Silhouette Coefficient and DB Index of all techniques on all datasets. HeMI achieves clearly better results on an overage than all other techniques without any overlapping of standard deviations. We believe that this is a very strong result in order to demonstrate the superiority of HeMI over a number of recent and high quality clustering techniques. We next present some interersting results supporting the effectivenss of various proposed components of HeMI.

Table 1.
A brief description of the datasets

Dataset	No. of Records with missing	No. of Records without missing	No. of numerical attributes	No. of categorical attributes	Class size
Glass Identification (GI)	214	214	10	0	7
Breast Cancer (BC)	286	277	0	9	2
Vertebral Column (VC)	310	310	6	0	2
Ecoli (EC)	336	336	8	0	8
Leaf (LF)	340	340	16	0	36
Liver Disorder (LD)	345	345	6	0	2
Dermatology (DT)	366	358	34	0	6
Credit Approval (CA)	690	653	6	9	2
Breast Cancer Wisconsin Original (WBC)	699	683	10	0	2
Blood Transfusion (BT)	748	748	4	0	2
Pima Indian Diabetes (PID)	768	768	8	0	2
Statlog Vehicle Silhouettes (SV)	846	846	18	0	4
Mammographic Mass (MGM)	961	830	5	0	2
Bank Note Authentication (BN)	1372	1372	4	0	2
Contraceptive Method Choice (CMC)	1473	1473	9	0	3
Yeast (YT)	1484	1484	8	0	10
Image Segmentation (IS)	2310	2310	18	0	7
Wine Quality (WQ)	4898	4898	11	0	7
Page Blocks Classification (PBC)	5473	5473	10	0	5
MAGIC Gamma Telescope (MGT)	19020	19020	11	0	2

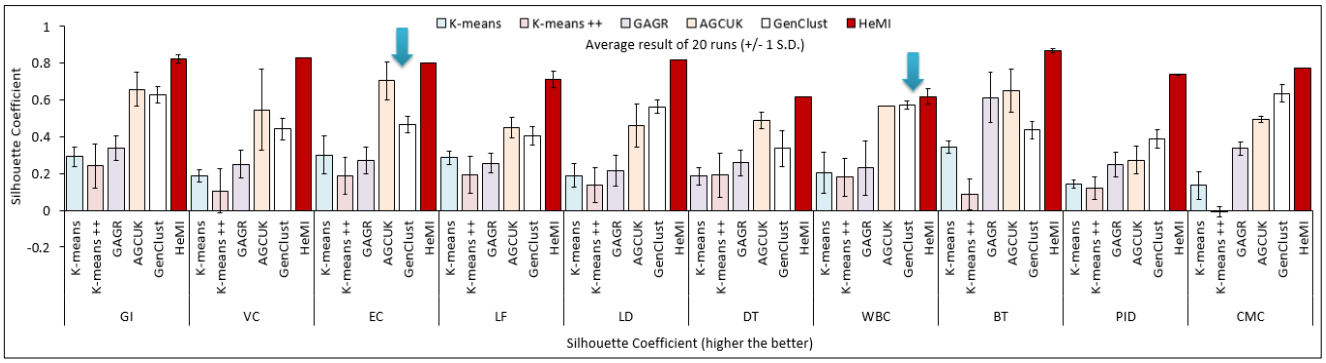

Fig. 1a. Comparative results between HeMI and other techniques on ten datasets based on Silhouette Coefficient

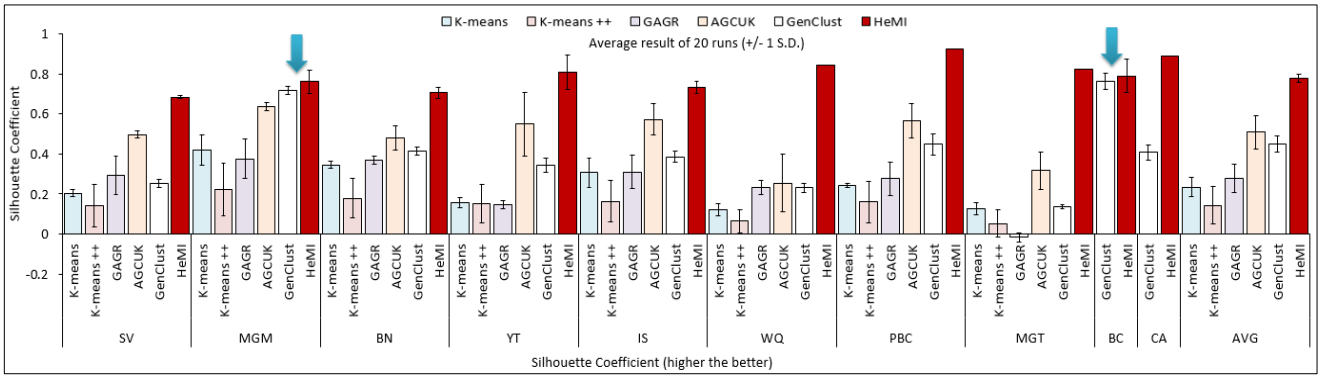

Fig. 1b. Comparative results between HeMI and other techniques on ten datasets based on Silhouette Coefficient

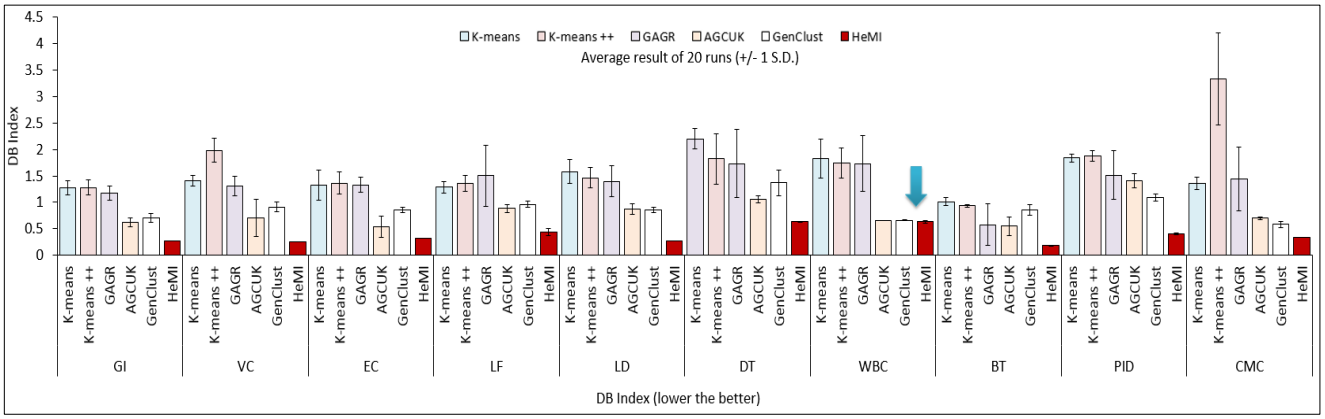

Fig. 2a. Comparative results between HeMI and other techniques on ten datasets based on DB Index

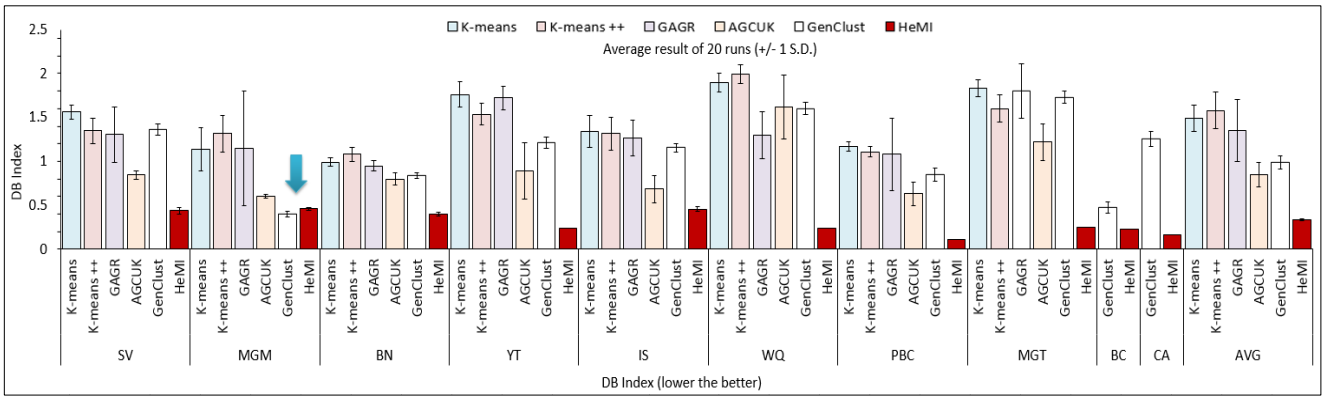

Fig. 2b. Comparative results between HeMI and other techniques on ten datasets based on DB Index

3.5 An Analysis of the Impact of Various Properties of HeMI

We now explore the effectiveness of some novel properties/components of HeMI in the following subsections. For the experiments, we add a component of HeMI to an existing technique called AGCUK, and investigate its impact on AGCUK.

3.5.1 An Analysis of the Impact of Multiple Streams that Exchange Information

An important contribution of HeMI is its multiple streams that share/exchange information at a regular interval. Due to having multiple streams HeMI can accommodate more chromosomes than existing techniques. Hence, in this section we carry out experiments to first investigate whether higher number of chromosomes improves the clustering results. We next investigate the impact of exchanging information among the streams at a regular interval. The results justify the usefulness of the components.

Table 2 demonstrate that AGCUK achieves better clustering results (in terms of Silhouette Coefficient and DB Index) when it uses 40 chromosomes instead of 20 chromosomes and 80 chromosomes instead of 40 chromosomes. We run 50 iterations as usual. Average results of 20 runs are presented in the tables.

Table 3 present clustering results obtained by AGCUK with 80 chromosomes in single stream and AGCUK with 80 chromosomes equally divided among 4 streams called *AGCUK with Multiple Streams*. In this case the streams do not exchange information. We pick the best clustering result of 4 streams at the end of 50 iterations. It clearly shows that multiple streams help AGCUK to achieve better results.

Table 4 compare clustering results obtained by AGCUK with 4 streams that *do not exchange information* and AGCUK with 4 streams called *AGCUK with Neighbor Exchange* that *exchange information* among neighbors regularly at 10 iterations. We can clearly see the impact of information exchange on the final clustering results.

The total number of chromosomes and iterations in all versions of AGCUK are same, but still AGCUK with multiple streams that exchange information achieves better results than others. This clearly indicates the effectiveness of the component.

Similar experiments are then carried out on another existing technique called GenClust. The results are consistent with AGCUK and indicate the effectiveness of the proposed component. See Table 5.

Table 2
Comparative results between AGCUK, AGCUK with 40 Population and AGCUK with 80 Population

Dataset	DB Index (lower the better)			Silhouette Coefficient (higher the better)		
	AGCUK	AGCUK with 40 Population	AGCUK with 80 Population	AGCUK	AGCUK with 40 Population	AGCUK with 80 Population
PID	1.40	**1.30**	1.32	0.27	**0.31**	0.29
BT	0.54	0.53	**0.47**	0.64	0.63	**0.66**
GI	0.62	**0.53**	0.54	0.65	**0.70**	0.69
LD	0.87	0.82	**0.79**	0.46	0.50	**0.51**
BN	0.79	0.76	**0.67**	0.47	0.49	**0.55**
Average	0.84	0.78	**0.75**	0.49	0.52	**0.54**

Table 3
Comparative results between AGCUK with 80 Population and AGCUK with Multiple Streams

Dataset	DB Index (lower the better)		Silhouette Coefficient (higher the better)	
	AGCUK with 80 Population	AGCUK with Multiple Streams	AGCUK with 80 Population	AGCUK with Multiple Streams
PID	1.32	**1.30**	0.29	**0.31**
BT	0.47	**0.43**	0.66	**0.69**
GI	0.54	**0.40**	0.69	**0.78**
LD	0.79	**0.77**	0.51	**0.55**
BN	0.67	**0.68**	0.55	0.55
Average	0.75	**0.71**	0.54	**0.57**

Table 4
Comparative results between AGCUK with Multiple Streams and AGCUK with Neighbor Exchange

Dataset	DB Index (lower the better)		Silhouette Coefficient (higher the better)	
	AGCUK with Multiple Streams	AGCUK with Neighbor Exchange	AGCUK with Multiple Streams	AGCUK with Neighbor Exchange
PID	1.30	**1.25**	0.31	**0.33**
BT	**0.43**	0.44	0.69	**0.70**
GI	0.40	0.40	**0.78**	0.74
LD	0.77	**0.46**	0.55	**0.67**
BN	0.68	**0.53**	0.55	**0.63**
Average	0.71	**0.61**	0.57	**0.61**

Table 5
Comparative results between GenClust, GenClust with Multiple Streams and GenClust with Neighbor Exchange

Dataset	DB Index (Lower the better)			Silhouette Coefficient (higher the better)		
	GenClust	GenClust with Multiple Streams	GenClust with Neighbor Exchange	GenClust	GenClust with Multiple Streams	GenClust with Neighbor Exchange
PID	1.11	1.03	**0.95**	0.37	0.42	**0.47**
BT	0.89	0.80	**0.77**	0.41	0.44	**0.47**
GI	0.71	0.69	**0.65**	0.62	0.63	**0.65**
LD	0.85	0.83	**0.75**	0.56	0.57	**0.63**
BN	0.85	0.82	**0.76**	0.41	0.42	**0.46**
Average	0.88	0.83	**0.77**	0.47	0.49	**0.53**

Table 6
Comparative results between HeMI, AGCUK with Neighbor Exchange and GenClust with Neighbor Exchange

Dataset	DB Index (lower the better)			Silhouette Coefficient (higher the better)		
	AGCUK with Neighbor Exchange	GenClust with Neighbor Exchange	HeMI	AGCUK with Neighbor Exchange	GenClust with Neighbor Exchange	HeMI
PID	1.25	0.95	**0.40**	0.33	0.47	**0.73**
BT	0.44	0.77	**0.17**	0.70	0.47	**0.86**
GI	0.40	0.65	**0.26**	0.74	0.65	**0.82**
LD	0.46	0.75	**0.26**	0.67	0.63	**0.81**
BN	0.53	0.76	**0.39**	0.63	0.46	**0.70**
Average	0.61	0.77	**0.29**	0.61	0.53	**0.78**

Table 7
Comparative results between AGCUK and AGCUK with HeMI Population

Dataset	DB Index (lower the better)		Silhouette Coefficient (higher the better)	
	AGCUK	AGCUK with HeMI Population	AGCUK	AGCUK with HeMI Population
PID	1.40	**1.29**	0.27	**0.30**
BT	0.54	**0.48**	0.64	**0.66**
GI	0.62	**0.57**	0.65	**0.70**
LD	0.87	0.87	0.46	0.45
BN	0.79	**0.75**	0.47	**0.50**
Average	0.84	**0.79**	0.49	**0.52**

Table 8
Comparative results between AGCUK and AGCUK with HeMI Mutation

Dataset	DB Index (lower the better)		Silhouette Coefficient (higher the better)	
	AGCUK	AGCUK with HeMI Mutation	AGCUK	AGCUK with HeMI Mutation
PID	1.40	**0.84**	0.27	**0.53**
BT	0.54	**0.23**	0.64	**0.83**
GI	0.62	**0.32**	0.65	**0.79**
LD	0.87	**0.32**	0.46	**0.78**
BN	0.79	**0.60**	0.47	**0.52**
Average	0.84	**0.46**	0.49	**0.69**

Table 9
Comparative results between HeMI and HeMI without Mutation

Dataset	DB Index (lower the better)		Silhouette Coefficient (higher the better)	
	HeMI	HeMI without Mutation	HeMI	HeMI without Mutation
PID	**0.40**	0.46	**0.73**	0.69
BT	**0.17**	0.23	**0.86**	0.82
GI	**0.26**	0.28	**0.82**	0.80
LD	**0.26**	0.31	**0.81**	0.78
BN	**0.39**	0.47	**0.70**	0.66
Average	**0.29**	0.35	**0.78**	0.75

Table 10
Comparative results between HeMI and HeMI without Health Improvement Operation

Dataset	DB Index (lower the better)		Silhouette Coefficient (higher the better)	
	HeMI	HeMI without Health Improvement Operation	HeMI	HeMI without Health Improvement Operation
PID	**0.40**	0.80	**0.73**	0.50
BT	**0.17**	0.18	0.86	0.86
GI	**0.26**	0.28	**0.82**	**0.81**
LD	**0.26**	0.27	**0.81**	0.80
BN	**0.39**	0.44	**0.70**	0.68
Average	**0.29**	0.39	**0.78**	0.73

Since we see a clear evidence of improvement in AGCUK and GenClust with the inclusion of multiple streams that exchange information, we now compare them with HeMI in order to investigate the effectiveness of other components of HeMI.

Table 6 present that HeMI achieves better clustering result than *GenClust with multiple streams that exchange information* in 5 out of 5 datasets based on Silhouette Coefficient and DB Index. HeMI performs better than *AGCUK with multiple streams that exchange information* in all 5 datasets based on Silhouette Coefficient and DB Index. Moreover, the average clustering result of 5 datasets based of HeMI on Silhouette Coefficient and DB Index shows the clear domination of HeMI over *AGCUK* and *GenClust with multiple streams that exchange information*. All results presented in the tables are average of 20 runs.

3.5.2 An Analysis of the Impact of the Population Initialization

In order to evaluate the effectiveness of our proposed population initialization we incorporate this component (see Component 3 in Section 2.2.3) with an existing technique called AGCUK, and then see how the component impacts AGCUK.

We generate 20 initial chromosomes through the proposed component. We then use these chromosomes in AGUCK as the initial population and run AGCUK for 50 iterations on 5 data sets. We call this as *AGCUK with HeMI population.*

Table 7 clearly indicates that *AGCUK with HeMI population* achieves better clustering result compared to AGCUK according to the Silhouette Coefficient and DB Index. The average clustering result of *AGCUK with HeMI*

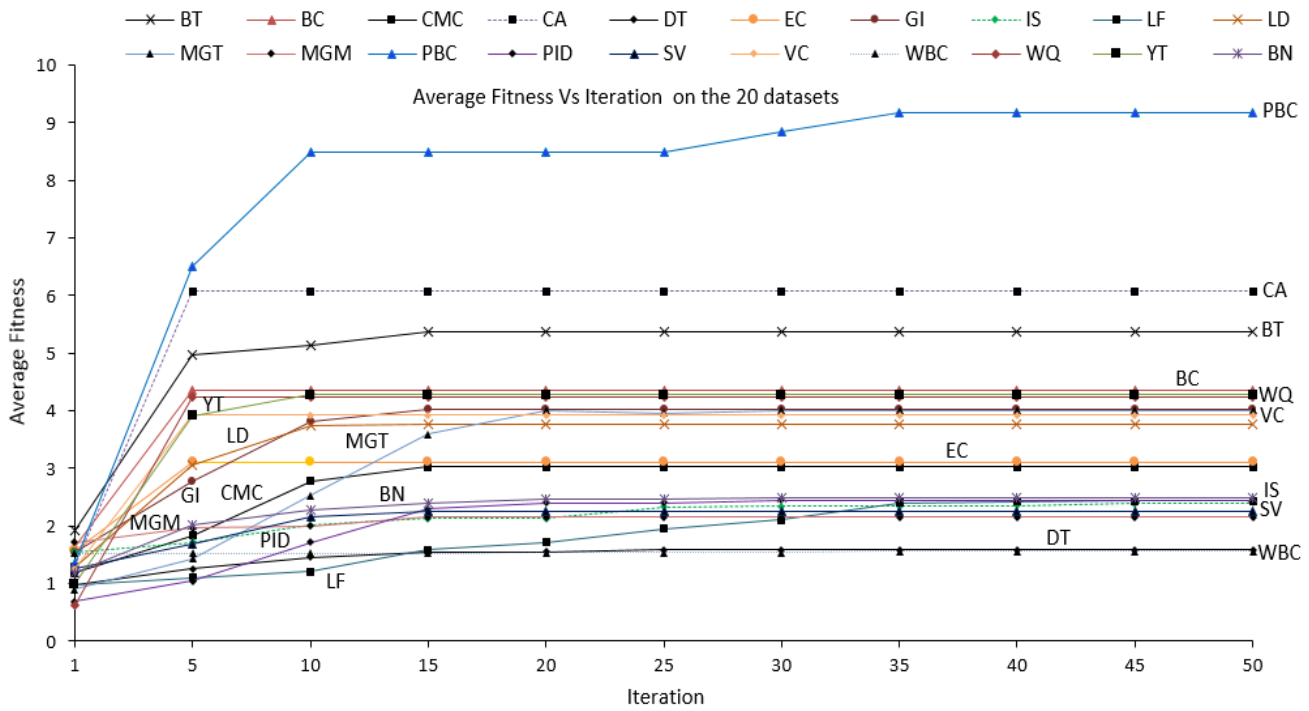

Fig. 3. Average Fitness versus Iteration. Each line represents the average fitness of the best chromosome of 5 consecutive runs of HeMI on a data set

population on 5 datasets is also better than AGCUK in terms of the both evaluation criteria.

3.5.3 *An Analysis of the Impact of the Mutation Operation*

In order to evaluate the effectiveness of the proposed mutation operation of HeMI we incorporate the component (see Section 2.2.7) with AGCUK by replacing its own mutation operation. We call this version of AGCUK as *AGCUK with HeMI mutation*. We run both AGCUK and *AGCUK with HeMI mutation* for 50 iterations on 5 data sets.

Table 8 shows that *AGCUK with HeMI mutation* achieves better clustering result compared to AGCUK in 5 out of 5 datasets according to both Silhouette Coefficient and DB Index.

We also extend this analysis by introducing a version of the proposed HeMI where we remove its mutation operation (let us call this version to be *HeMI without mutation*) and then compare this with complete HeMI.

We run both HeMI and *HeMI without mutation* for 50 iterations. Table 9 shows that HeMI achieves better clustering results than *HeMI without mutation* in 5 out of 5 datasets based on Silhouette Coefficient and DB Index. This clearly indicates the effectiveness of the proposed mutation operation used in HeMI.

3.5.4 *An Analysis of the Impact of Health Improvement*

We also explore the effectiveness of the health improvement operation (see Section 2.2.8 for Component 8) of HeMI. In Table 10 we present the experimental results of HeMI comparing with a different version of HeMI called *HeMI without health improvement operation* that is exactly same as HeMI except that it does not have Component 8 in it. We run both HeMI and *HeMI without health improvement operation* for 50 iterations on 5 datasets.

We can see in Table 10 that HeMI achieves better clustering results than *HeMI without health improvement operation* based on Silhouette Coefficient and DB Index.

3.5.5 *An Analysis of Improvement in Chromosomes over the Iterations.*

In Fig. 3 we present the average fitness (in terms of DB Index, where Fitness = *1/DB Index*) values of the best chromosomes over 5 separate runs of HeMI. We run HeMI 5 times and then present the average fitness of these 5 runs. Average fitness values are plotted against the iterations, for all 20 datasets. Most of the datasets achieve a rapid improvement within first 5 to 10 iterations, and then continues to steadily increase over the iterations. This is also clear from Fig. 4 that presents the grand average fitness of the best chromosomes over all 20 datasets, instead of each dataset separately. In Fig. 5 we present the average fitness of the best chromosome of HeMI, AGCUK and HeMI with a single stream on the PID dataset. Both HeMI and AGCUK use the same fitness function (DB Index [40]) to calculate the fitness of a chromosome. Average fitness values of the best chromosomes of HeMI are always higher than those of HeMI with a single stream and AGCUK, clearly indicating the effectiveness of various components of HeMI including its multiple streams.

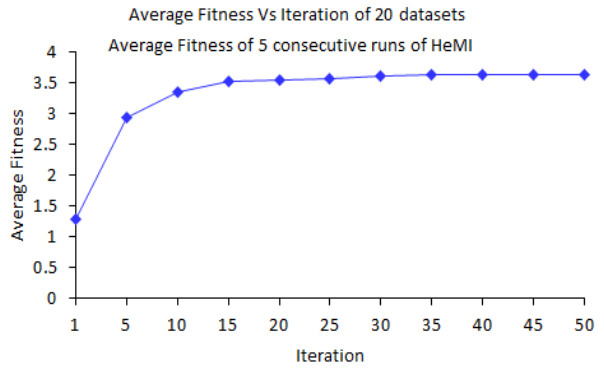

Fig. 4. Average Fitness (best chromosome) versus Iteration over the 20 datasets

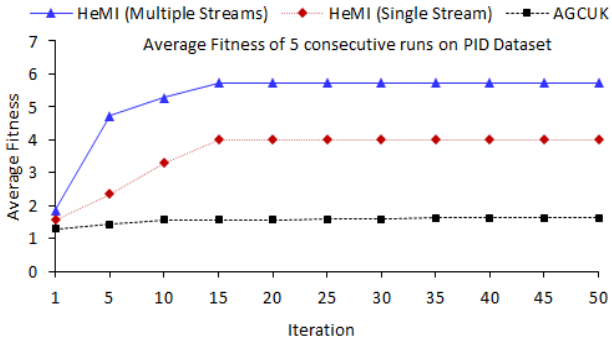

Fig. 5. Average Fitness (best chromosome) versus Iteration. Each line represents the average fitness of 5 consecutive runs on PID data set

3.6 Statistical Analysis

We now analyze the results by using a statistical sign test [51, 52] on all 20 datasets for all 20 runs to evaluate the superiority of the results (Silhouette Coefficient and DB Index) obtained by HeMI over the results obtained by the existing techniques. We observe that the results do not follow a normal distribution and thus do not satisfy the conditions for a parametric test. Hence, we perform a non-parametric sign test on the Silhouette Coefficient and DB Index as shown in Fig. 6a, Fig. 6b, Fig. 7a and Fig. 7b. The first five bars for each dataset in Fig. 6a, Fig. 6b, Fig. 7a and Fig. 7b show the z-values (test statistics) values for HeMI and the five existing techniques while the sixth bar shows the z(ref.) value. If the z value is greater than the z(ref.) value then the results obtained by HeMI are significantly better than the results of existing techniques.

In Fig. 6 a and Fig. 6 b we present the sign test of HeMI compared with the existing techniques on 20 datasets in terms of Silhouette Coefficient, where HeMI significantly performs better than other techniques on 19 out of 20 datasets. Fig. 7a and Fig. 7b show the statistical significance of HeMI compared with other existing techniques based on DB Index, where HeMI significantly performs better than other techniques on 19 out 20 datasets. We carry out the statistical analysis at $z > 1.96$, $p < 0.025$ and right-tailed in terms of Silhouette Coefficient and DB Index. Note that the cases where we have a lower z value than z(ref.) are indicated with arrows in Fig. 6a and Fig. 7b.

3.7 An Analysis on the use of K-means++ instead of K-means in HeMI

The HeMI algorithm allows us to use any light weight clustering techniques for the initial population including K-means and K-means++. In our experiments so far we used K-means for the initial population and we see that HeMI clearly outperforms all other existing techniques used in this study. Table 11 indicate that HeMI with K-means++ for the initial population achieves better clustering results than HeMI with K-means. Hence, we are confident that HeMI with K-means++ will win against other existing techniques even more strongly.

3.8 Complexity Analysis

In this section we estimate and present the complexity of HeMI and compare it with the complexities of the existing techniques used in this study. The main factors involving the complexity of HeMI are number of records n in a dataset D, number of attributes m in D, number of genes k in a chromosome, number of chromosomes z in a population of a stream, number of iterations N' of k-means and number of iterations N of HeMI. Out of these factors we consider that n, m, k and z can be much bigger than others and hence we compute the complexity in terms of them.

For the initial population HeMI uses K-means to get a number of deterministic chromosomes, the complexity of which is $O(nmkz)$. It also randomly selects some chromosomes, for which the complexity is $O(kz)$. The fitness function is DB index which has a complexity of $O(nmkz)$.

Once fitness is computed the noising selection requires pairwise comparison which can be done in $O(z)$ complexity. The crossover operation requires roulette wheel for which we need $O(z^2)$ complexity. For the twin removal we need $O(mk^2z)$ complexity. In the mutation operation, complexities for the division, absorption and random change are $O(nmkz)$, $O(mkz)$ and $O(z)$, respectively.

Complexities for Phase 1, Phase 2 and Phase 3 of the Health Improvement component are $O(nmkz)$, $O(z)$ and $O(z)$, respectively. The elitist operation has a complexity of $O(z)$ once the fitness is calculated with the cost of $O(nmkz)$. Information exchange among neighboring streams requires $O(z)$ complexity. Similarly, the global best selection also requires $O(nmkz)+O(z)$ complexity. Hence, the overall complexity of HeMI is $O(nmk^2z^2)$. With respect to n and m (the two most significant factors) it has a linear complexity $O(nm)$. The complexity of K-means, K-means ++, AGCUK, GAGR and GenClust are $O(nm)$ [2], $O(nm)$ [3], $O(nm)$ [5], $O(nm)$ [6] and $O(nm^2 + n^2m)$ [4] respectively.

3.9 Comparison Between HeMI and Multiple Runs of K-means

Although the complexity of K-means and HeMI in terms of n and m are the same, i.e. $O(nm)$, we realize that HeMI may require higher execution time than K-means. For example, it will require executing the distance calculation

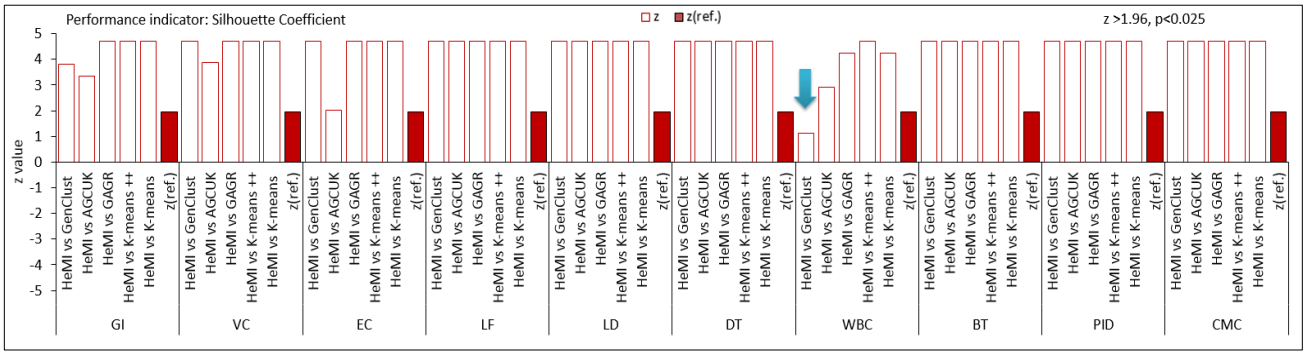

Fig. 6 a. Sign test of HeMI based on Silhouette Coefficient on ten datasets

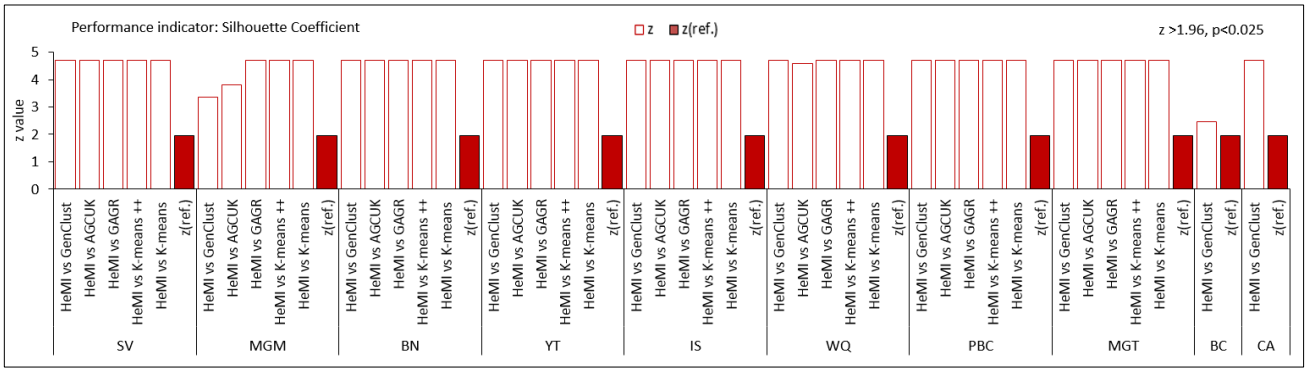

Fig. 6 b. Sign test of HeMI based on Silhouette Coefficient on ten datasets

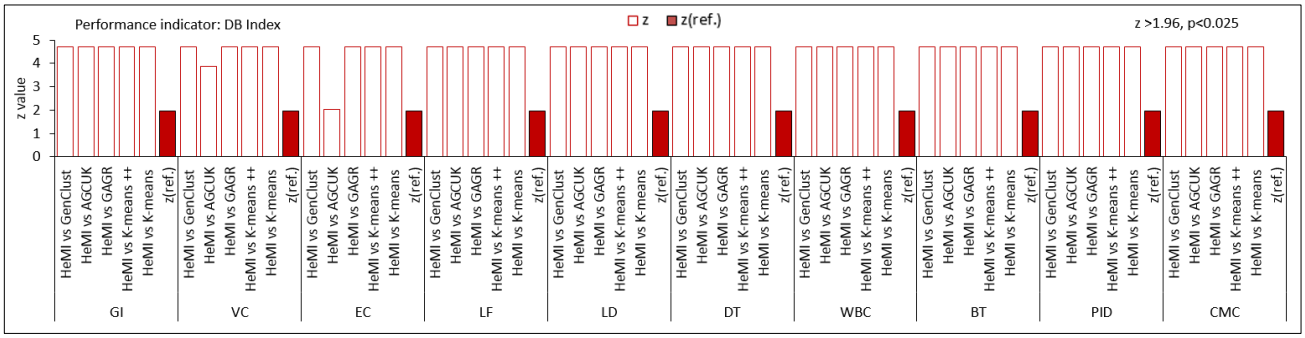

Fig. 7 a. Sign test of HeMI based on DB Index on ten datasets

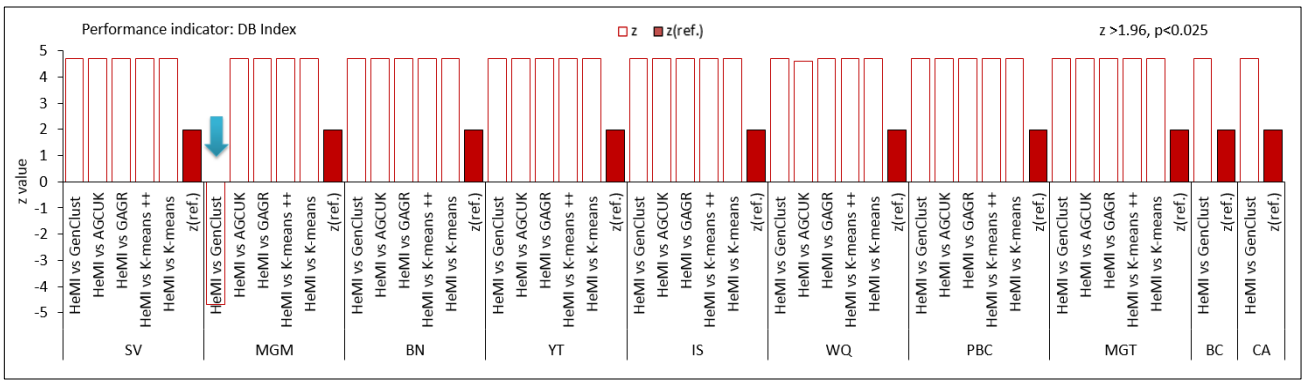

Fig. 7 b. Sign test of HeMI based on DB Index on ten datasets

Table 11
Comparative results between HeMI and HeMI with K-means++

Dataset	DB Index (lower the better)		Silhouette Coefficient (higher the better)	
	HeMI	HeMI with K-means ++	HeMI	HeMI with K-means ++
PID	0.40	**0.17**	0.73	**0.86**
BT	0.17	0.17	0.86	**0.87**
GI	0.26	0.24	0.82	**0.83**
LD	0.26	0.26	0.81	**0.82**
BN	0.39	**0.38**	0.70	**0.71**
Average	0.29	**0.24**	0.78	**0.81**

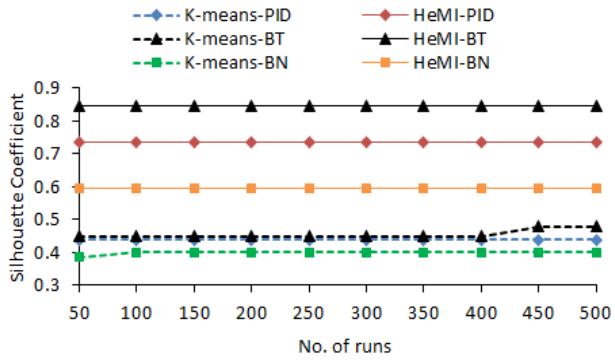

Fig. 8. Comparative result between HeMI and K-means

more frequently than K-means. Therefore, by the time we run HeMI once we can perhaps run K-means multiple times.

We compute that K-means requires executing the distance calculation approximately *nmki* times, where *i* is the number of iterations and *k* is the number of seeds. Considering 50 iterations (i.e. *i = 50*) for K-means, it requires the distance calculation *50nmk* times.

On the other hand, HeMI requires executing the distance calculation approximately *nmkzi* (for initial population) + *8nmkzG* (for genetic operations) times, where *z* is the number of chromosomes, *i* is the number of iteration in K-means and *G* is the number of generations. Considering, *i=50*, *z =20* and *G = 50*, HeMI requires the distance calculation *9000nmk* times. That is, HeMI requires the distance calculation $(\frac{9000nmk}{50nmk})$ = 180 times more than K-means. As a result, by the time we can run HeMI once we can run K-means approximately 180 times.

Therefore, in this section we run K-means up to 500 times and pick *the best result* out of these 500 runs. We also run HeMI 20 times and pick *the worst result* out of the 20 runs. Finally in Fig. 8, we compare *the best result* of K-means with *the worst result* of HeMI on three randomly chosen datasets. The top three lines in Fig. 8 show the worst result of HeMI on three datasets and the bottom three lines represent the best results of K-means at different runs starting from 50 to 500 for the same three datasets. The results clearly indicate that K-means cannot beat HeMI (in terms of the Silhouette Coefficient) even if K-means runs 500 times.

CONCLUSION

In this paper we greatly extend our conference paper [32] that we have never published in any journal. The only overlapping contribution between this paper and our conference paper [32] is the initial population selection process. However, this paper has a number of useful contributions including multiple streams, mutation operation, health improvement operation, neighbor information exchange, and global best selection operation in addition to the initial population selection.

We evaluate the proposed technique (HeMI) by comparing its clustering quality with five existing techniques namely K-means [2], K-means ++ [3], GAGR [6], AGCUK [5] and GenClust [4] on twenty (20) natural datasets that are publicly available from the UCI machine learning repository [31] in terms of two well-known evaluation criteria: Silhouette Coefficient and DB Index. Our experimental results indicate a statistically significant superiority of HeMI over the existing techniques. We also carry out thorough investigation to evaluate major components of HeMI one by one. It is evident that all of these components have positive impact on the final clustering quality.

We also present a complexity analysis which shows that HeMI has a complexity of *O(n)*, where *n* is the number of records in a dataset. Note that like HeMI, GenClust also uses an initial population selection approach in order to get a good quality initial population. However, the approach used in GenClust requires a complexity of $O(n^2)$.

Our future research plan includes more investigation on the results in order to discover knowledge from the datasets through clustering and then evaluate the quality of the knowledge discovered by our technique and some existing techniques. We plan to propose clustering techniques that will improve the quality of discovered knowledge.

References

[1] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, first ed., Pearson Addison Wessley, 2005.

[2] S. P. Lloyd, Least squares quantization in PCM, IEEE Transactions on Information Theory. 28 (1982) 129-13.

[3] D. Arthur, & S. Vassilvitskii, k-means++: The Advantages of Careful Seeding, SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, 2007, pp.1027-1035.

[4] M. A. Rahman, & M. Z. Islam, A hybrid clustering technique combining a novel genetic algorithm with K-Means, Knowledge-Based Systems. 71 (2014) 345-365.

[5] Y. Liu, X. Wu, Y. Shen, Automatic clustering using genetic algorithms, Applied Mathematics and Computation. 218 (2011) 267-1279.

[6] D. Chang, X. Zhang, C. Zheng, A genetic algorithm with gene rearrangement for K-means clustering, Pattern Recognition. 42 (2009) 1210-1222.

[7] L. E. A. Blas, S. S. Sanz, S. J. Fernández, L. C. Calvo, J. D. Ser, J. A. P. Figueras, A new grouping genetic algorithm for clustering problems, Expert Systems with Applications. 39 (2012) 9695- 9703.

[8] R.J. Kuo, Y.J. Sy, Z. Y. Chen, F.C. Tien, Integration of particle swarm optimization and genetic algorithm for dynamic clustering, Information Sciences. 195 (2012) 124-140.

[9] D. Chang, Y. Zhao, C. Zheng, X. Zhang, A genetic clustering algorithm using a message-based similarity measure, Expert Systems with Applications. 39 (2012) 2194-2202.

[10] A. Mukhhopadhyay, & U. Maulik, Towards improving fuzzy clustering using support vector machine: Application to gene expression data, Pattern Recognition. 42 (2009) 2744-2763.

[11] G. Bello-Orgaz, J. J. Jung, & D. Camacho, Social big data: Recent achievements and new challenges, Information Fusion. doi: http://dx.doi.org/10.1016/j.inffus.2015.08.005

[12] G. Gan, Application of data clustering and machine learning in variable annuity valuation. Insurance, Mathematics and Economics. 53 (2013) 795-801.

[13] C. B. N. Li, K. Chui, S. Chang, S. H. Ong, Integrating spatial fuzzy clustering with level set methods for automated medical image segmentation, Computers in Biology and Medicine. 41 (2011) 1-10.

[14] F. Zhao, J. Fan, H. Liu, Optimal-selection-based suppressed fuzzy c-means clustering algorithm with self-tuning non local spatial information for image segmentation, Expert Systems with Applications. 41(2014) 4083-4093.

[15] W. Cai, S. Chen, D. Zhang, Fast and robust fuzzy c-means clustering algorithms incorporating local information for image segmentation, Pattern Recognition. 40 (2007) 825-838.

[16] S. Montani, & G. Leonardi, Retrieval and clustering for supporting business process adjustment and analysis, Information Systems. 40(2014) 128-141.

[17] M.-Y. Chen, A hybrid ANFIS model for business failure prediction utilizing particle swarm optimization and subtractive clustering, Information Sciences. 220 (2013) 180-195.

[18] M. Girvan, & M.E.J. Newman, Community Structure in Social and Biological Networks, Proceedings of the National Academy of Sciences of the United States of America. 99 (2002) 7821-7826.

[19] G. Stockman, & L.G. Shapiro, Computer Vision. (1st ed.), Prentice-Hall, ISBN 0-13-030796-3, New Jersey, 2001,pp. 279-325.

[20] S. R. Kannan, S. Ramathilagam, A. Sathya, R. Pandiyarajan, Effective fuzzy c-means based kernel function in segmenting medical images, Computers in Biology and Medicine. 40 (2010) 572-579.

[21] F. Masulli, & A. Schenone, A fuzzy clustering based segmentation system as support to diagnosis in medical imaging, Artificial Intelligence in Medicine, 1998, pp. 129-147.

[22] A. K. Jain, Data clustering: 50 years beyond K-Means, Pattern Recognition Letters. 31 (2010) 651-666.

[23] C. G. E. Boender, A. H. G. Rinnooy Kan, G. T. Timmer, L. Stougie, A stochastic method for global optimization, Mathematical Programming. 22 (1982) 125-140.

[24] S. Bandyopadhyay, U. Maulik, M. K. Pakhira, Clustering using simulated annealing with probabilistic redistribution, International Journal of Pattern Recognition and Artificial Intelligence. 15 (2001) 269- 285.

[25] Z. Güngör, & A. Ünler, K-harmonic means data clustering with simulated annealing heuristic, Applied Mathematics and Computation. 184 (2007) 199- 209.

[26] J.H. Holland, Adaptation in Natural and Artificial Systems, Univ. of Michigan Press, Ann Arbor, MI. 1975.

[27] P. A. D. Gomez, & D. F. Hougen, Initial Population for Genetic Algorithms: A Metric Approach, International Conference on Genetic and Evolutionary Methods, GEM 2007, Las Vegas, Nevada, USA. 2007, pp. 25-28.

[28] D. E. Goldberg, V. Deb, K. J. Clark, Genetic Algorithms, Noise, and the Sizing of Populations, Complex Systems, 6 (1992) 333-362.

[29] J. Strasburga, C.G. Martelb, V. Alexandrov, Parallel genetic algorithms for stock market trading rules, Procedia Computer Science. 9 (2012) 1306 -1313.

[30] H. Pourvaziri, & B. Naderi, A hybrid multi-population genetic algorithm for the dynamic facility layout problem, Applied Soft Computing. 24 (2014) 457- 469.

[31] UCI Machine Learning Repository. <http://archive.ics.uci.edu/ml/datasets.html/> (accessed 22.06.13).

[32] A.H. Beg, & M.Z. Islam, Clustering by Genetic Algorithm-High Quality Chromosome Selection for Initial Population. 10th IEEE conference on Industrial Electronics and Applications, Auckland, New Zealand. 2015, pp. 129-134.

[33] Y. Y. Liu, & S. Wang, A scalable parallel genetic algorithm for the Generalized Assignment Problem, Parallel Computing. 46 (2015) 98-119.

[34] M. Moore, An accurate parallel genetic algorithm to schedule tasks on a cluster, Parallel Computing. 30 (2004) 567–583.

[35] J. Kumar, R. T. Richard Mills, F. M. Hoffman, W. W. Hargrove, Parallel k-Means Clustering for Quantitative Ecoregion Delineation Using Large Data Sets, Procedia Computer Science. 4 (2011) 1602–1611.

[36] H. Giggins, & L.Brankovic, VICUS - A Noise Addition Technique for Categorical Data, Proceedings of the Tenth Australasian Data Mining Conference,Sydney, Australia. 2012, pp. 139-148.

[37] J. Han, & M. Kamber, Data Mining Concepts and Techniques (2nd ed.). San Francisco: Morgan Kaufmann. 2006.

[38] J. Schulz, Minkowski distance, 2008. <http://www.code10.info/index.php?option=com_content&view=article&id=61:article> (accessed 15.05.2013).

[39] Teknomo, K. (2006a). Jaccard's Coefficient. <http://people.revoledu.com/kardi/tutorial/Similarity/Jaccard.html> (accessed 15.05.2013).

[40] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 1 (1997) 224–227.

[41] U. Maulik, & S. Bandyopadhyay, Genetic algorithm-based clustering technique. Pattern Recognition, 33 (2000)1455-1465.

[42] P. Peng, O. Addam, M. Elzohbi, S. T. Özyer, A. Elhajj, S. Gao, Y. Liu, T. Özyer, M. Kaya, M. Ridley, J. Rokne, R. Alhajj, Reporting and analyzing alternative clustering solutions by employing multi-objective genetic algorithm and conducting experiments on cancer data, Knowledge-Based Systems. 56 (2014)108-122.

[43] X. Yana, Y. Zhua,W. Zoua, L. Wang, A new approach for data clustering using hybrid artificial bee colony algorithm, Neurocomputing. 97 (2012) 241-250.

[44] M. Srinivas, L. M.Patnaik, Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transactions on Systems, Man and Cybernetics. 24 (1994) 656-66.

[45] S. Bandyopdhyay, & U. Maulik, An evolutionary technique based on K-means algorithm for optimal clustering in RN, Information Sciences. 146 (2002) 221-237.

[46] C.C. Lai, A novel clustering approach using hierarchical genetic algorithms, Intelligent Automation & Soft Computing. 11 (2005) 143-153.

[47] H.J. Lin, F.W. Yang, Y.T. Kao, An efficient GA-based clustering technique, Tamkang J. Sci. Eng. 8 (2005) 113–122.

[48] S. Bandyopadhyay, U. Maulik, Nonparametric genetic clustering: comparison of validity indices, IEEE Trans. Syst. Man Cybern. Part C – Appl. Rev. 31 (2001) 120-125.

[49] C.A. Murthy, N. Chowdhury, In search of optimal clusters using genetic algorithms, Pattern Recognition Lett. 17 (1996) 825-83.

[50] L. Rokach, & O. Maimon, Clustering methods." Data mining and knowledge discovery handbook. Springer US, 2005, pp. 321-352.

[51] M.F. Triola, Elementary Statistics, eighth ed., Addison Wesley Longman Inc., 2001.

[52] R. Mason, D. Lind, W. Marchal, Statistics: An Introduction, fifth ed., Brooks/Cole Publishing Company, 1998.