To be published at:
International Conference on Applied Mathematics, Computational Science and Systems Engineering,
Sapienza University, Italy-Rome, November 5-7, 2016.

A variant of genetic algorithm for non-homogeneous population

Najmeh Alibabaie[1], *Mohammad Ghasemzadeh* [2,*], and *Christoph Meinel* [3]

[1] Computer Department, Engineering Campus, Yazd University, Yazd, Iran
[2] Assoc. Prof. at Yazd University in Iran and Guest Researcher at HPI, Potsdam, Germany
[3] President and CEO of Hasso Plattner Institute (HPI), at Potsdam University, Potsdam, Germany

Abstract. Selection of initial points, the number of clusters and finding proper clusters centers are still the main challenge in clustering processes. In this paper, we suggest genetic algorithm based method which searches several solution spaces simultaneously. The solution spaces are population groups consisting of elements with similar structure. Elements in a group have the same size, while elements in different groups are of different sizes. The proposed algorithm processes the population in groups of chromosomes with one gene, two genes to k genes. These genes hold corresponding information about the cluster centers. In the proposed method, the crossover and mutation operators can accept parents with different sizes; this can lead to versatility in population and information transfer among sub-populations. We implemented the proposed method and evaluated its performance against some random datasets and the Ruspini dataset as well. The experimental results show that the proposed method could effectively determine the appropriate number of clusters and recognize their centers. Overall this research implies that using heterogeneous population in the genetic algorithm can lead to better results.

1 Introduction

Optimization is one of the most important topics in various fields of science and technology. In computer science, this issue is concerned with finding optimal solutions for some problems. There are exact and efficient algorithms for a subset of optimization problems, but for lot of them we need to do a comprehensive search in an exponentially large solution space, which most of times is impractical. In order to find a solution for these problems, we usually use heuristic and approximation methods. These problems include wide range of applications such as data mining, computer vision, knowledge discovering and decision support systems. The heuristic methods do not necessarily find the optimal solutions, but often they manage to find a solution which is close to the optimal solution. This can be accomplished by consuming reasonable time and processing resources. Many problems involve complex search spaces with conflicting objectives. In these cases, optimization of an objective may prevent to achieve the other objectives. Here, sometimes there are a set of optimal solutions which have no clear superiority on each other. In multi objective optimization problems we try to find a trade-off among the requested objectives.

Clustering is a data mining technique, which is used widely in solving related problems [1]. The main issue in clustering is to partition the given data, with a specified number of clusters such that minimize the total within cluster variation (TWCV) and maximize the variance between clusters. The k-means algorithm is the simplest and the most popular clustering method [2].

The initial cluster centers usually have a high efficacy on performance of the k-means algorithm. If these centers are selected randomly, the process may converge to suboptimal partitions [3]. Regardless of the applied clustering method, one of the other most important parameters is number of clusters. In non-hierarchical clustering methods like k-means, this issue corresponds to finding a proper value for k. This parameter must be determined before the clustering process starts. There is no straightforward or easy way to determine this value, because it depends on different issues like the form and distribution of data, as well as the clustering resolution expected by the user.

The ultimate goal of this research is to find the best cluster centers and the best number of clusters simultaneously. The proposed method takes advantage of a genetic algorithm in which the population consists of some sub-populations. Since each point in the k-gene solution space can be a solution candidate, we allow these points to stay in the population at the same time,

and in their corresponding sub-populations. Furthermore we let them contribute in evolution of the final solution by applying genetic operators on them. In this regard, the genetic operators are implemented in such a way that could combine individuals of different sub-populations to create new varied individuals for the sub-population groups. Individuals in a subpopulation compete with each other, and also the best individuals of each subpopulation compete with each other as well. The main idea is that we want the genetic algorithm select the best solution from non-homogeneous solutions which are being evaluated by the same criteria.

2 Literature Review

In K-Means (KM), the initial cluster centers usually have a high impact on the clustering process. If these centers are selected randomly from the dataset, then each run of KM on the same dataset may lead to a different result. In other words choosing different centroid points every time gives different clusters. Furthermore the process may converge to suboptimal partitions [3]. In order to cope this dilemma, Sawant [4] in his research, proposed to calculate the neighbourhood distance between the first data item and the other data items, and then sort the items according to the obtained distances. Afterwards he divides the arranged data items into K equal portions. As an initial candidate, the first data item of each partitioning takes part in the KM process. The above method, manages to find appropriate results with higher degree of accuracy in fewer number of steps, but since it requires computing neighbourhood distance between data items and rearranging them, it would be very slow in clustering large datasets.

The proposed method by Karegowda et al. [5], also the proposed method by Al-Shboul and Myaeng [3] use genetic algorithm (GA) to determine the initial values of the clusters. In both of them a population is generated by using a GA process. They apply classic KM algorithm to evaluate fitness of each candidate center. When the terminating condition of GA goes true, the chromosome with highest fitness value decides which samples will be the k-means initial cluster centers. Since KM algorithm must be run for many times, these methods are usually computationally expensive.

The research carried out by Krishna and Murty [2], and another investigation performed by Yi Lu et al. [6] are also based on using GA. These methods compared to the already mentioned methods[5,3] are faster and always converge to the global optimum. In addition they are not sensitive to the initial values. These methods maintain a population of some coded solutions and apply a single step K-means operator (KMO) to evaluate the chromosomes. The common point among all the existing methods which try to find better initial centers, is that they need to start with a given number of clusters.

Another challenge in clustering algorithms is related to deciding about number of clusters, in other words, determination of the K parameter. The algorithm proposed by Hamerly and Elkan [7] is based on a statistical test. It relies on the hypothesis that a subset of data follows a Gaussian distribution. The algorithm starts with a small number of clusters, and then increases them when needed. In each iteration of the algorithm, if any cluster appear whose data dispersion do not follow the Gaussian distribution then that cluster would be divided into two clusters. In order to determine the number of clusters, some of the recent research works rely on using GA. In this regard, Llet´ et al. [8] use the silhouette to find the appropriate number of clusters. The common point among all the existing methods which are developed to find an appropriate number of clusters is that they evaluate the concerned criteria separately for different values of k. In other words, they need to run the KM algorithm for different values of K; this process is usually very time consuming.

The research works reported by Li and Chang [9] and also Chittu and Sumathi [10] are based on introduction of modified genetic algorithms. They divide the existing population into a number of subpopulations. The chromosomes of a population have the same structure, but they use different parameters for the subpopulations. This could let information be transferred between subpopulations. In fact, migration of individuals between subpopulations along with application of genetic operators would lead to generation of new individuals. The purpose of using this modified genetic algorithm with multiple subpopulations and dynamic parameters is to make GA run faster.

3 The proposed method

One of the most important topics and applications of computer science is concerned with information clustering. The K-means algorithm is one of the most popular methods used widely for this purpose.

This algorithm gets the data set and the desired number of clusters, and then it finds the clusters and their centers. In order to find the best number of clusters, typically the user must run this algorithm for different number of clusters and evaluate the obtained results. It can be shown that this parameter may be determined automatically by using genetic algorithms. Genetic algorithm is a search algorithm inspired from nature and genetics. Unlike many other search algorithms, which perform local and greedy search, GA performs a stochastic universal search. It is mainly composed of three operators: reproduction, crossover and mutation.

In many clustering problems, the classical genetic algorithm can be applied easily and get acceptable results. But still there are some disadvantages that need special consideration. One of these disadvantages is concerned with deciding about the suitable number of clusters and their centers. In order to cope with these disadvantages we proposed a modified structure for the genetic algorithm. In fact, in this paper, we show how we can organize the population in several sub-populations and mend the relevant operators to attain an improved genetic algorithm. We accomplish this by considering a population consisting of several set of chromosomes. The structure of these sets is almost similar because these genes hold corresponding

information about the cluster centers but the number of genes in every group differs. In other words, the structure of the chromosomes is different in terms of number of constituent genes. In the proposed method, the existing population is processed in the form of several subsets which respectively contain one gene, two genes to k genes. These genes hold corresponding information about the cluster centers. New individuals are created by using the two main genetic recombination operators known as crossover and mutation. We re-implemented these operators to let them accept parent chromosomes with different size. This could lead to have variation in the population and which in turn lets better data transfer between the subsets.

Let $\overrightarrow{X_1}, \overrightarrow{X_2}, \ldots, \overrightarrow{X_N}$ be the set of n patterns and X_{nd} denoting dth feature of $\overrightarrow{X_N}$. For the existing dataset if we consider number of clusters to be k for k=1, 2,…, K, then structure of the individuals which belong to each respective sub-space would be in the form of $C_k = (\overrightarrow{C_1}, \overrightarrow{C_2}, \ldots, \overrightarrow{C_k})$, in which $\overrightarrow{C_i} = (c_1, c_2, \ldots, c_d)$. Here, the number of genes in a subpopulation is proportional to the number of clusters and the gene length is also proportional to the data dimension. In other words, the proposed method maintains a set of encoded solutions (called chromosomes). Number of chromosomes is specified by the user. Here, each allele in the chromosome represent the value of cluster center, and number of genes in a chromosome stands for number of clusters.

Recombination (or crossover) is the process of generating new items (offsprings) by exchanging a part (or some parts) between the chosen individuals (parents). Crossover is made with the hope that new chromosomes will contain good parts of each parent chromosomes and therefore the new chromosomes would constitute a better generation. In order to have heterogeneous populations in next generations, the genetic operators must be able to accept and also create heterogeneous individuals. Therefore we implemented the crossover operator in such a way that it could accept parent chromosomes, C_i and C_j, with different lengths (different number of genes). A parent chromosome is selected from the fittest individuals with probability P_b or it is selected randomly from the population with probability $1 - P_b$. The number of genes in an offspring is proportional to the number of genes of its parents. As shown in Fig. 1., If C_i is one of the parents with L_1 genes and C_j is the other parent with L_2 genes and $L_1 < L_2$, then number of genes of offspring O_1 will be L_1 and the number of genes of offspring O_2 will be L_2 genes. L_1 genes of each offspring are created by a weighted average of similar genes of the parents. The remaining $L_2 - L_1$ genes of the longer offspring is the same as the genes of the longer parent that were not similar to any genes of the shorter parent.

The **mutation** operation randomly changes the value of a gene in the offspring. Mutation is applied after the crossover with a low probability, called the mutation probability. This operator is used to prevent premature convergence to local optima. If $\overrightarrow{C_i}$ is an encoded solution, it will change one or more genes with probability P_m. According to evolutional theory, mutation takes place in a way that new generation would be more perfect than the former generation.

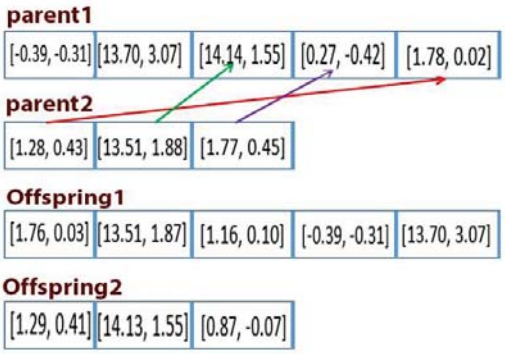


Fig. 1. Crossover of parents with different number of genes.

In the proposed method, mutation may be applied to individuals in two ways. 1) One or some genes in a chromosome are changed with probability P_{m1}, 2) Number of genes in a chromosome are changed with probability P_{m2}.

That means one or more genes are inserted into chromosome structure or are deleted from it. The allele of these genes will be set randomly. In this way, the concerned genetic operators generate the new population of every generation by combining different subpopulations of the former generation.

Selection is operator that selects a solution from the current population for the next population according to its fitness value. Selection of individuals with higher fitness to form the next generation lets the population converge toward the generation which include the desired solutions of the search space. Already, in order to speed up the convergence process,

A **K-means operator** (KMO), known as "one step of the classical K-means algorithm" [6] has been introduced. We implemented a modified version of KMO to find the closest cluster to every point.

The **silhouette value** for each point is a measure of how similar that point is to points in its own cluster compared to points in other clusters [8]. It is defined as:

$$S(i) = \frac{b(i) - w(i)}{Max\{b(i), w(i)\}} \quad (1)$$

with

$$b(i) = \min_k \{ B(i,k) \} \quad (2)$$

Where w(i) is the average distance from the ith point to the other points in its own cluster, and B(i, k) is the average distance from the ith point to points in another cluster k. The value of the parameter S(i) is in the range of [-1, +1].

This measure ranges from +1, indicating points that are very distant from neighboring clusters, through 0, indicating points that are not distinctly in one cluster or another, to −1, indicating points that are probably assigned to the wrong cluster [8]. If number of clusters is equal to number of objects, then for every point a_i we would have w(i) = 0 and s(i) = 1. In order to prevent generation of singleton clusters, we set S(i) = 0. In order to improve computational performance, we used the

average distance of data in a cluster to cluster center as an approximation to w(i) and used $b(i) = \min_k\{ D(i,k)\}$ as an approximation for b(i).

Applying genetic operators can generate two identical cluster centers in a chromosome. In other to stop illegal solutions caused by $b_i = 0$, we consider $b_i = \max_k B(i,k)$ instead. In fact, this idea will prevent these individuals to transfer into the next generation. The main steps of the proposed method are shown in Fig. 2. The essential difference between our method and the classical genetic algorithm is in regard to using the populations with heterogeneous individuals and the changings we made to the basic operators. The new operators can accept heterologous individuals and structure of the mutation operator output won't be necessarily the same as the input structure.

1. Determine the maximum number of clusters(Kmax)
2. Initialize the heterogeneous population (generate individuals for subpopulations k=1, 2,.., Kmax).
3. Evaluate individuals using the fitness function.
4. Repeat until terminating condition is met:
   a) Select the parent chromosomes from the subpopulations and apply the crossover operation.
   b) Apply the mutation operation (change genes value or insert genes into or delete genes from the chromosomes).
   c) Evaluate new individuals using the fitness function (needed to decide who can transfer to the next population).
   d) Select individual for next generation (replace low fit chromosomes with new high fit chromosomes).
   e) Update crossover probability and mutation probability by their adjustment functions.
5. Extract number of clusters and their centroid from the fittest chromosome.

Fig. 2. The main steps of the proposed method.

4 Implementation

We implemented the proposed method on a microcomputer in MATLAB. Although in the employed development environment there is special toolbox for running some optimization and genetic algorithms, but since our proposed method uses a different structure for its chromosomes and some changes were needed in the genetic operator functions, we couldn't use the available toolbox, therefore we implemented the required procedures using the basic commands and facilities that were available in the MATLAB software environment. The pseudo-code of our modified genetic algorithm along with its crossover and mutation operators are shown in Fig. 3.

```
%==  The modified Genetic Algorithm  ===
t = 0;
Initialize Ps(t);
Evaluate Ps(t);
Until ( termination condition is met )
   {
     t=t+1;
     Select Ps(t) from Ps(t - 1);
     Crossover individual pairs of Ps(t);
     Apply Mutation on new individuals of Ps(t);
     Evaluate fitness on new individuals of Ps(t);
     Adjust Probability Parameters;
   }
```

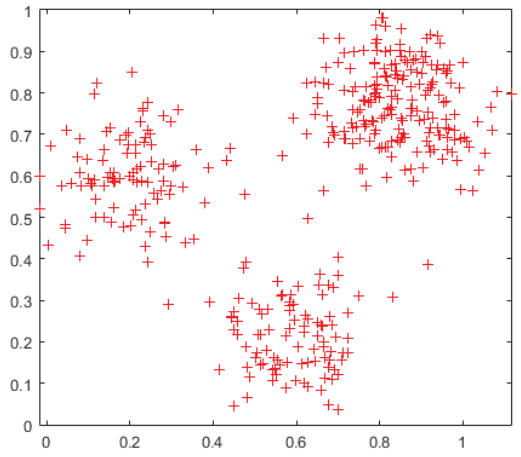
```
mutation()
 { for i=1 to k
    For j=1 to data.dimension
     { d=a number selected randomly from dataset
       If (rand()<mutation_rate)
           p[k][dimension_j]=d;
     }
   If (rand()<mutation_rate)
      newk=randi(maxk);
    If (newk<=k)
      P=p[1:newk][all_dimension]
    Else
     {d=(newk-k)*dimension-number, randomly
         selected from dataset
          P[1: k][:]= p[1: k][:]
          P[k:newk][:]= d
       }
   }
```

```
Crossover ()
{  %% P1.length=k1 & P2.length=k2 & k1<k2
   For i=1 to k1
  {  Index=find P2[j] similar to P1[i]
     O1=w*P1+(1-w)*P2
     O2[1:k1]=w*P1+(1-w)*P2
  }
   %%  wi is similar to any genes of P1
   O2[k1+1:k2]= P2[wi]
}
```

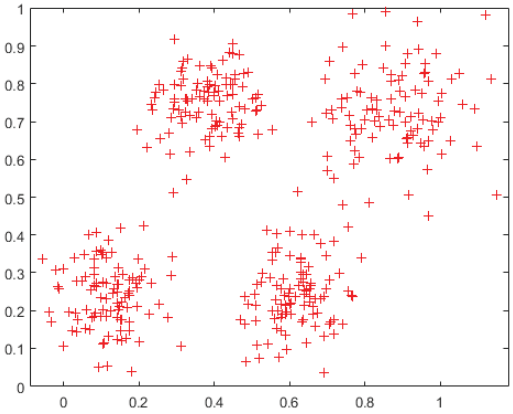
Fig. 3. Pseudo-code of modified genetic algorithm along with its crossover and mutation operators.

5 Experimental results

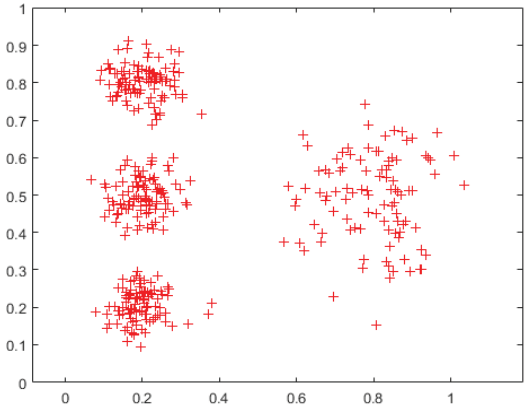
In order to evaluate the proposed method, we examined its performance against four different datasets [8, 3]. The first three datasets are generated by mathematical models. These datasets are shown in Fig. 4. They include some samples consisting of three or four clusters with different distributions, the other database is known as Ruspini data [11].



a. Dataset 1



b. Dataset 2



c. Dataset 3

Fig. 4. Random dataset distribution.

Already experimental results and official analysis have shown that silhouette value can also be used efficiently for higher dimensions [8]. We evaluate our proposed method against these data sets which consist of two-dimensional data, but since we have applied the silhouette value, the obtained results would also apply to dataset with higher dimensions. In fact, we chose the two-dimensional datasets for capability of easier presentation and spatial perception.

Dataset 1 consists of 300 points gathered around 3 clusters. The points are scattered with a radius of 0.2 around 3 specific points (0.2, 0.6), (0.6, 0.2) and (0.8, 0.8). They have same values in their boundary points.

Dataset 2 Consists of 400 points scattered around 4 specific points with a radius of 0.2, the points are (0.125, 0.25), (0.625, 0.25), (0.375, 0.75) and (0.875, 0.75). The points two and four have horizontal interleaving on the boundary. This dataset is to be clustered into 4 clusters.

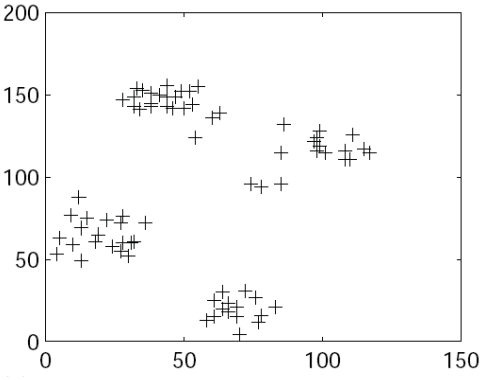
Dataset 3 Consists of 400 points scattered around 4 points with a radius of 0.3 for the first 3 centers and radius 0.4 for the last center. Points are (0.2, 0.2), (0.2, 0.5), (0.2, 0.8) and (0.8, 0.5). The point of the last cluster seems to be isolated except for the points between the cluster center and the other three cluster; the first three clusters have common boundary points. This dataset is to be clustered into 4 clusters.

Dataset 4 is the well-known Ruspini data [11]. This dataset consists of 75 samples, with two features for each sample. This dataset is frequently used in evaluation of clustering methods. Fig. 5-a shows the initial raw dataset. It is clear that this dataset consists of four clusters (Fig. 5-b). If we consider number of clusters k=5, then k-means would give the clustering shown in Fig. 5-c. Depending on the initial centers, for k=5, we may get another form of clustering like the one shown in Fig. 5-d; this time the bold circles are the objects which have been separated to form the fifth cluster.

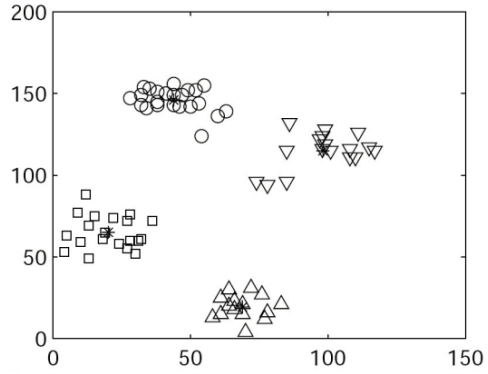
Usually, in order to find the best number of clusters and corresponding centers for a clustering problem using k-means, it is usually required to run it for different number of clusters and for different initial centers.

In addition, if these best values are being found by a classic genetic algorithm, we would presumably need to make some changes in the encoding: The proposed algorithm looks simultaneously for these goals and it doesn't require any alternations in the coding of solutions. In the experiments, we set population size to be 100 and the number of generations to be 30. The termination condition is met when we reach the predefined number of generations. The crossover and mutation operators were implemented as mentioned earlier. Maximum number of clusters set to be 6 to avoid too much calculation. Table 1 shows these parameters along with some other related parameters.

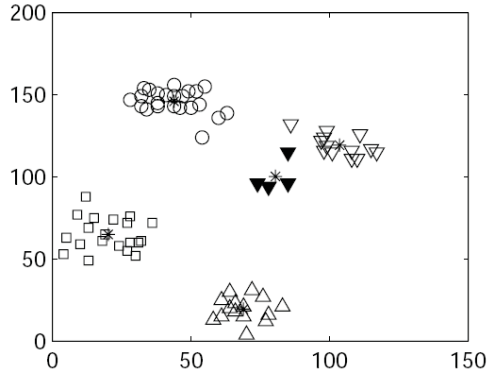
By considering a different structure for the genetic algorithm we managed to find the suitable centroids and the appropriate number of clusters, in a reasonable time. Fig. 6 shows the experimental results of running the proposed algorithm on datasets 2.



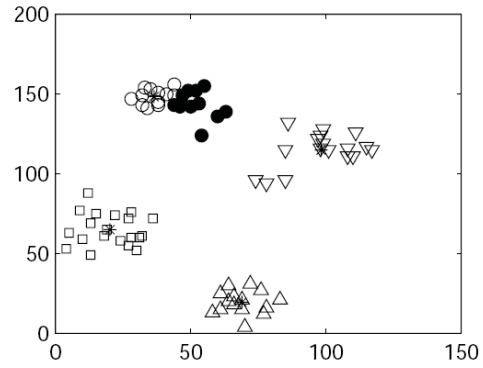
a : (Ruspini) raw data



b: Clustering with k=4.



c: Clustering with k=5 (Random centers1).



d : Clustering with k=5 (Random centers2)

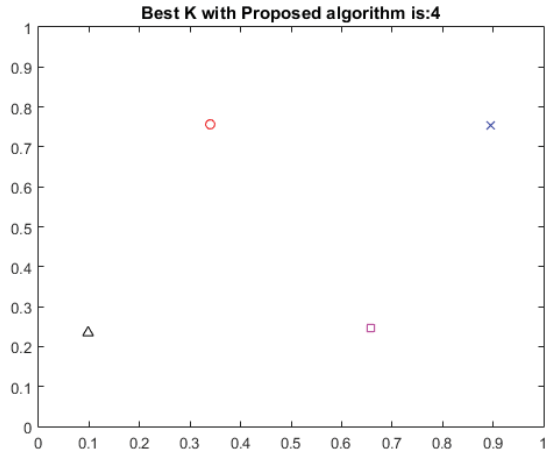
Fig. 5. The Ruspini dataset [8].

Table 1. Configuration parameters for the proposed algorithm.

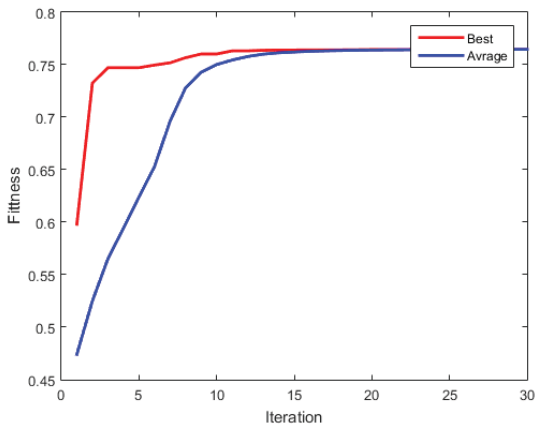
Parameter	**Value**
Population size	100
Sub-populations size	Randomly
Initial crossover probability	0.8
Initial mutation probability	0.1
Probability of changing Nr. of genes	*f(iteration)*
Probability of changing genes values	
Maximum Nr. of clusters (Kmax)	6
Number of generations	30

Fig. 6-a Shows how the proposed method managed to determine the clusters centroids truly. Fig. 6-b shows how the proposed algorithm could converge after only 15 iterations. Fig. 6-c shows (average) number of clusters of the fittest chromosome in every generation. These information were obtained from averaging results of ten independent runs of the algorithm during the first 30 generations. In early iterations, some points of the search space with k=3, compared to the others gain higher fitness (Fig. 6-c) but mutation and transformation of information between subpopulations could eventually lead the process to converge toward the better solution k=4.

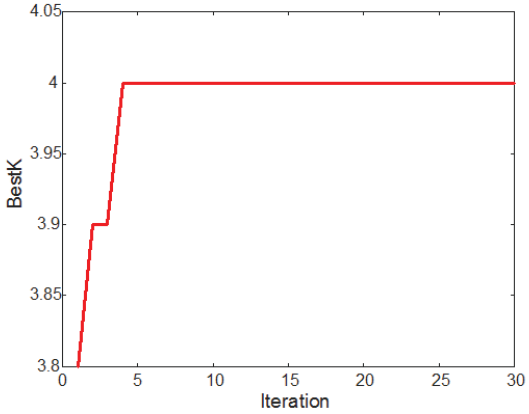
Table 2 shows the clusters centroids and number of clusters found by the proposed method. In fact it shows how the proposed algorithm manages to find clusters centers and number of clusters correctly.



6-a. The proposed centroids by the proposed method



6-b. Fitness of fittest chromosome and average fitness of population in every generation.



6-c. Nr. of clusters in the chromosome with the best fitness in every iteration (Average of 10 run)

Fig. 6. The proposed method output for dataset2.

Table 2. The clusters centroids and the number of clusters found by the proposed algorithm.

	Nr. of clusters	**The clusters centroids obtained by the proposed method**			
Dataset 1	3	(.79 , .79)	(.61 , .17)	(.18 , .60)	---
Dataset 2	4	(.30 , .77)	(.10 , .24)	(.62 , .25)	(.84 , .76)
Dataset 3	4	(.79 , .50)	(.18 , .49)	(.20 , .79)	(.19 , .20)
Ruspini	4	(65.22 , 18.95)	(16.36 , 69.26)	(148.37 , 40.18)	(115.70 , 101.94)

Considering a different structure for the population and chromosome patterns could let the procedure not require a separate run for every value of k, this means reducing runtime. It is obvious that we would get better results when the objective function could establish a better trade-off between minimize the total within cluster variation (TWCV) and maximize the variance between clusters. Although the fitness function we adopted from Llet´ et al [8] can also work efficiency for high dimension datasets, but combining it with other measures can improve it to work efficiently for overlapping data as well. In the testing phase, each dataset was used 20 times and then we calculated the average running time. The obtained results are shown in Table 3.

Table 3. Performance evaluation

	Proposed algorithm	**K-means algorithm**	**GA+KM (constant k)**
Dataset 1	*2.60*	*0.01*	*18.51*
Dataset 2	*1.03*	*0.013*	*15.32*
Dataset 3	*2.61*	*0.02*	*11.12*
Ruspini	*0.87*	*0.02*	*8.84*

If we try to use the classic genetic algorithm for solving problems which involve several search spaces, then for each search space we need to apply a different encoding and run the algorithm from the beginning; while the proposed approach doesn't require any change in the encoding and it performs the corresponding operations in parallel.

Since the proposed model for population consists of some subpopulations with different structures and we use improved genetic operators for recombining them, it is expected that offsprings will inherit appropriate information from their parent, leading to find an appropriate solution for the concerned problem. This cooperation and information exchange among subpopulations could also reduce the required processing time. In the implemented algorithm, despite presence of individuals from different search spaces, because of using the modified mutation operator, during consecutive generation, the best search space dominates and most of individuals in the last generation belong to the best solution space. When we want to apply our idea in solving optimization problems involving solutions from different search spaces, if we fix the number of individuals in subpopulations and allocate a certain proportion of the population to every subpopulation in successive generations, the algorithm can converge to the best solutions of different spaces.

5 Conclusion and future work

In this research a different structure for the genetic algorithm was introduced. This method can be applied to problems that several groups of different solutions are evaluated according to the same criteria. It shows that a heterogeneous population in genetic algorithm could accelerate the search process and improve the final results. In a further research, the fitness function can be improved by considering other criteria such as entropy for overlapping data or by considering a constant proportion for each subpopulation, apply it in solving problems that require to find best solutions from several search spaces.

References

1. A. Amirkhanyan, F. Cheng, C. Meinel. Real-time clustering of massive geodata for online maps to improve visual analysis. *11th International Conference on Innovations in Information Technology (IIT)*. 308-313. Dubai. (2015).
2. K. Krishna, M. N. Murty. Genetic K-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics,*, **29**(3): 433-439. (1999).
3. B. Al-Shboul, S. H. Myaeng. Initializing KMeans using genetic algorithms. *The International Journal of Computer, Electrical, Automation, Control and Information Engineering*, **3**(6): 1481-1485. (2009).
4. K. B. Sawant. Efficient Determination of Clusters in K-Mean Algorithm Using Neighborhood Distance. *The International Journal of Emerging Engineering Research and Technology* **3**(1): 22-27. (2015).
5. A. G. Karegowda, V. T. Shama, M. A. Jayaram, A. S. Manjunath. Improving Performance of K-Means Clustering by Initializing Cluster Centers Using Genetic Algorithm and Entropy Based Fuzzy Clustering for Categorization of Diabetic Patients. , *In Proceedings of International Conference on Advances in Computing*, 899-904. MSRIT, Bangalore: Springer India. (2013).
6. Y. Lu, S. Lu, F. Fotouhi, Y. Deng, S. J. Brown. Incremental Genetic K-Means Algorithm And Its Application In Gene Expression Data Analysis. *BMC Bioinformatics* **5**(1):172. (2004).
7. G. Hamerly, C. Elkan. Learning the k in k-means. *Advances in Neural Information Processing Systems* 17:281–288. (2004).
8. R. Llet´, M.C. Ortiz, L.A. Sarabia, M.S. Sanchez. Selecting variables for k-means cluster analysis by using a genetic algorithm that optimises the silhouettes. *Analytica Chimica Acta* **515**(1): 87-100. (2004).
9. R. Li, X. Chang. A Modified Genetic Algorithm with Multiple Subpopulations and Dynamic Parameters Applied in CVaR Model. *International Conference on Computational Intelligence for Modelling Control and Automation*, 151. Washington, DC: IEEE Computer Society. (2006).
10. V. Chittu, N. Sumathi. A Modified Genetic Algorithm Initializing K-Means Clustering. *Global Journal of Computer Science and Technology* **11**(2): 54-62. (2011).
11. E. H. Ruspini. Numerical methods for fuzzy clustering. *Information Sciences* **2**(3): 319–350. (1970).