Journal of mathematics and computer science 12 (2014), 282-294

Contents list available at JMCS
Journal of Mathematics and Computer Science
Journal Homepage: www.tjmcs.com

Some new mutation operators for genetic data clustering

Gholam Hasan Mohebpour, Arash Ghorbannia Delavar

Department of Computer Science, Payame Noor University, PO BOX 19395-3697, Tehran, Iran

mohebpoor_gh_h@yahoo.com, a_ghorbannia@pnu.ac.ir

Article history:
Received July 2014
Accepted August 2014
Available online August 2014

Abstract

Genetic algorithm is one of evolutionary algorithms which have been used widely to solve many problems such as data clustering. There are lots of genetic data clustering algorithms which have worked on fitness function to improve the accuracy of algorithm in evaluation of generated chromosomes and have used simple and all purpose crossover and mutation operators such as one point crossover and random change mutation. Mutation process randomly modifies the gene values at selected locations to increase genetic diversity, by forcing the algorithm to search areas other than the current area. Simple non heuristic mutations such as random change mutation increase genetic diversity but they also increase execution time and decrease fitness of population. In this paper we introduce some new heuristic mutation operators for genetic data clustering. Experimental results show that all of proposed mutation operators creates better offspring than random change mutation and increases the fitness of population.

Keywords: Data mining, data clustering, genetic algorithm, mutation operator, partitioning

1. Introduction

1.1. Data clustering

Suppose $X = \{x_1, x_2, \cdots, x_N\}$ is a set of N instances in m- dimensional data space R^m.Data clustering means partitioning these instances to K clusters $C = \{C_1, C_2, \dots, C_K\}$ where $C_i \neq \emptyset\ i = 1,2,\dots,k$, $C_i \cap C_j = \emptyset \quad i \neq j$ and $\bigcup_{i=1}^{K} C_i = X$ which means each instance is as similar as possible to its co-cluster instances according to a distance measure function $\mathrm{d}(\mathrm{x},\mathrm{y})$ like Euclidean distance.

1.2. K-means algorithm

Although K-means is simple and can be used for a wide variety of data types, it is quite sensitive to initial positions of cluster centers and traps in local optima.

There are several different data clustering methods, but k-means is a typical clustering algorithm which is widely used for clustering data sets and is one of the simplest, non-supervised algorithms and also it doesn't need any prior knowledge about the data distribution. The K-means algorithm starts by initializing the *k* cluster centers randomly. The input instances are then allocated to one of the existing clusters according to the square of the Euclidean distance from the cluster centers, choosing the closest. For each cluster we introduce the mean of instances as new cluster center.

$$Z_j = \frac{\sum_{i=1}^{n_j} . \mathrm{xi}}{n_j} \tag{1}$$

And $Z = \{Z_1, Z_2, \dots, Z_K\}$ is the set of cluster centers. The process of reassigning the data points and the updating of the cluster centers will be repeated until no more change in the cluster centers and no more reassigning of data points.

1.3. Genetic algorithms

Genetic algorithm is a search and optimization technique based on the principles of genetics and natural selection proposed by Holland [1] and later refined by De Jong [2], Goldberg [3], and many others. It is a heuristic algorithm which mimics the principle of survival of the fittest laid by Charles Darwin. Generally, an initial population of chromosomes is generated randomly and while stopping criteria didn’t reached, fitness value of each individual will be computed and selection performance such as roulette wheel or tournament selection method will be used to select chromosomes with better fitness values for producing new offspring. Crossover selected parents will be done on and finally mutation will be done on generated offspring. This process will be repeated for some generations and the best individual will be introduced as final result.Fig.1 shows the flowchart of genetic data clustering algorithm based on K-means.

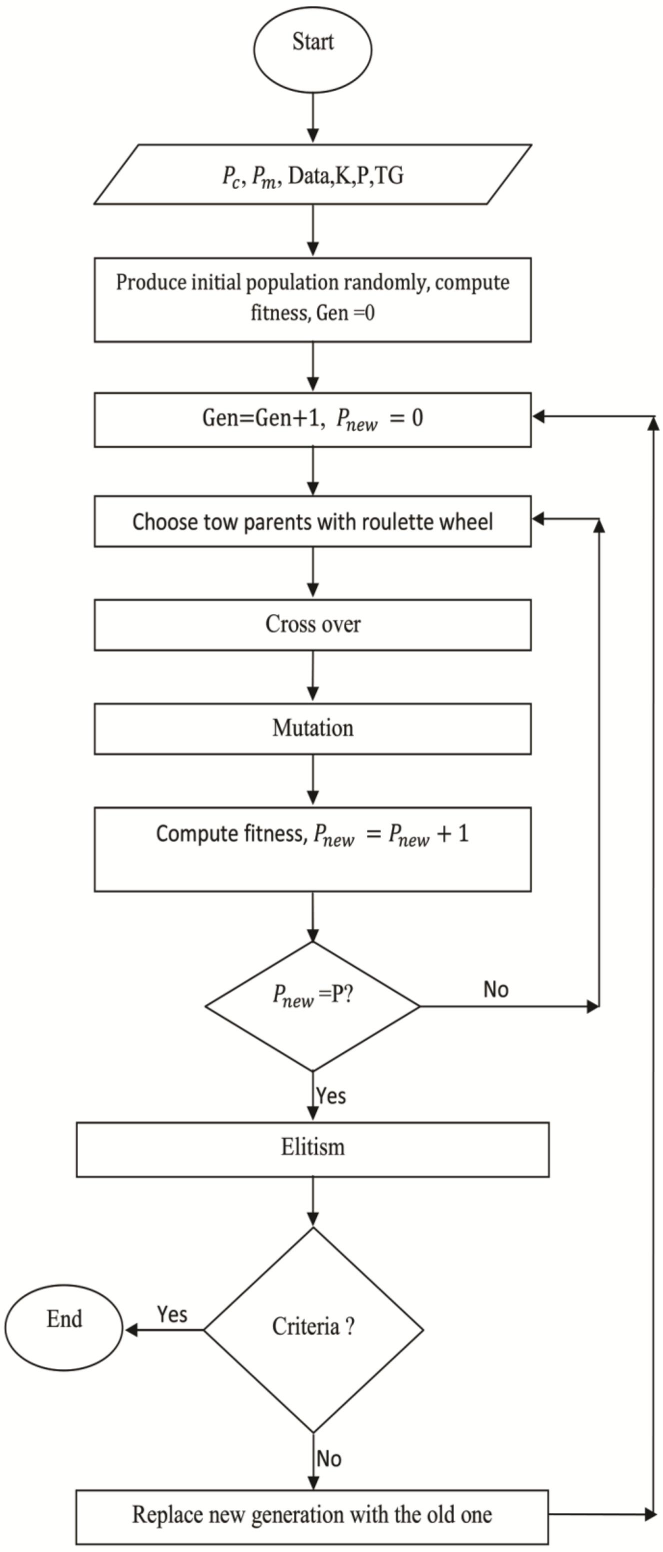
Start
P_c, P_m, Data,K,P,TG
Produce initial population randomly, compute fitness, Gen =0
Gen=Gen+1, P_{new} = 0
Choose tow parents with roulette wheel
Cross over
Mutation
Compute fitness, $P_{new} = P_{new} + 1$
P_{new}=P?
No
Yes
Elitism
Criteria ?
Yes
End
No
Replace new generation with the old one

Fig.1.Flowchart of genetic algorithm

2. Related work

There are several genetic data clustering algorithms but most of them have used random change mutation operator which is not heuristic.
Yongguo Liu et al. [4] have introduced Division–absorption mutation for their proposed automatic clustering algorithm. It consists of two sub-operations: division operation and absorption operation. Division operation computes a probability for each cluster and divides the sparser clusters and Absorption operation finds two nearest clusters and merges them. This algorithm uses Davies–Bouldin (DB) index to compute Division probability and distance between cluster centers to compute absorption probability so it will have problem with arbitrary shape clusters.
Hong He and YonghongTan [5] have used a Two-stage mutation. In the first stage the algorithm emphasizes on detecting the number of clusters and in the second stage it will emphasize on finding correct cluster centers. L.E. Agustin-Blas et al. [6] have also used split and merge mutation operator with a different chromosome representation.
Dongxia Chang et al. [7] have introduced six new mutation operators in their message-based similarity measure genetic clustering algorithm. These mutation operators add, delete or change a gene or split or merge two clusters randomly but they are not heuristic operators.
Amin Aalaei et al. [8] have used random change mutation operator for their matrix based chromosome structure to exchange binary value of tow genes with each other.
Jose A. Castellanos-Garzon and Fernando Diaz [9] have proposed a new hierarchical clustering method using genetic algorithms for the analysis of gene expression data. They have used a mutation operator which works dendrogram. The individuals (chromosomes) represent dendrograms on a given data set, encoded as an ordered set of clusterings, where each clustering is an ordered set of clusters.

All the above mutation operators have been proposed for variable length chromosomes and we need mutation operators with fix length chromosome for clustering with predefined number of clusters.

3. Problem encoding

3.1 Chromosome representation

For any GA, a chromosome representation and program encoding will be used to describe each individual in the population of interest. The representation method determines how the problem is structured in the algorithm and the genetic operators that are used. Each chromosome is made up of a sequence of genes. In genetic algorithm clustering, each chromosome is a representation of one of possible solutions for clustering of a data set.

In our method, an integer-valued problem-specific chromosome representation is used. Each chromosome has a fixed length of $K * \log_2 N$ where K in the number of clusters and N is the number of data points in dataset. So we have k genes in each chromosome. Each gene is made up of index of center data point of a cluster. In the chromosome structure of proposed algorithm is not ordered which means the place of genes is not important and any permutation of genes produces the same chromosome.

This structure will not produce any infeasible chromosome but illegal chromosomes can be produced. In proposed structure repetition of genes produces illegal chromosomes because a data point cannot be center point of more than one cluster. So to be able to detect production of illegal chromosomes during crossover and mutation, we sort the genes in ascending order. The advantages of this chromosome structure are small length, fast detection of illegal chromosomes, fast detection of repetitive chromosomes and faster mutation and crossover operations.

3.2. Evaluation and fitness function

The fitness function has an important effect on success of a genetic algorithm. In this paper which we want to examine efficiency of a mutation operator we use the simplest fitness function for genetic data clustering. Objective function of k-means is defined as follow:

$$E = \sum_{i=1}^{k} \sum_{x \in C_i} |x - c_i|^2 \tag{2}$$

c_i is the center instance of ith cluster and E is the sum of the squared error of all instances in dataset. This Objective function tries to produce k clusters so that the instances in the same cluster are as compact as possible while the instances in different clusters are as separated as possible. The fitness function that we use is defined as bellow:

$$\mathrm{F_i} = \frac{\mathbf{1}}{\boldsymbol{E}} \tag{3}$$

3.3. Selection

Parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. Imagine a roulette wheel where are placed all chromosomes in the population, every has its place big accordingly to its fitness function

In rank-based wheel selection the individuals are sorted in a list based on their fitness value. The position of the individuals within the list is called rank of the individual so all individuals have the same chance to be selected. Rank-based wheel selection is used when we have individuals with big fitness values and want to prevent form effect of them on selection

A common selection approach assigns a probability of selection $\mathrm{P_i}$ to each individual i based on its fitness value. The probability Pi for each individual is defined by Equation (4)

$$\mathrm{p_i} = \frac{\mathrm{F_i}}{\sum_{\mathrm{j=1}}^{\mathrm{pop\ size}} \mathrm{F_j}} \tag{4}$$

$\mathrm{F_i}$ Equals the fitness of individual i. The use of roulette wheel selection limits the genetic algorithm to maximization, because the evaluation function must map the solutions to a fully ordered set of values on R^+. Extensions, such as windowing and scaling, have been proposed to allow for minimization and negativity.

In roulette wheel selection, the individuals are mapped to sector of a circle which has a circumference equal with one, such that each individual's sector is equally sized to its fitness. A random number is generated and the individual whose segment spans the random number is selected. The process repeats until the desired number of individuals is obtained (called mating population). This technique is analogous to a roulette wheel with each slice proportionally sized to the fitness.

3.4 Crossover

Crossover is a process that exchanges information between two parent chromosomes for generating offspring chromosomes and occurs with a user specified probability, called the crossover probability P_c.
In this paper we have used One Point Crossover [10] with crossover probability P_c=1 to make our experimental conditions as same as possible. In one point crossover a cutoff point between 1 and length of chromosome (0<C<L) is randomly selected to divide each parent chromosome in to two parts then left part of the first parent and right part of the second parent generate the first offspring and right part of the first parent and left part of the second parent generate the second offspring.

3.5. Stopping criteria

The only criterion that we have chosen is number of generations. As we reached predefined number of generations we will stop the algorithm and introduce the best individual as clustering result.

3.6. Random Change Mutation

In Random Change Mutation we choose a gene, and change it with a data point in data space which has been selected. In data clustering it means that we change one of cluster centers with a data point in data space. Both operations of choosing a gene and changing it with a data point in data space will be done randomly.

4. Proposed mutations

Mutation process randomly modifies the gene values at selected locations to increase genetic diversity, by forcing the algorithm to search areas other than the current focus and occurs with a user specified probability, called the mutation probability P_m. Genetic diversity helps not to converge to local optima solutions.
Swap mutation operator which changes the position of two genes randomly, inversion mutation which reverses the order of one or more genes or insertion mutation which cuts some genes form chromosome and paste them in another place can just be used in ordered chromosomes and are not useful in our clustering problem.

4.1. Same Cluster Random Change Mutation

In Same Cluster Random Change Mutation we choose a gene, and change it with a data point in data space which is in the same cluster with the chosen gene. Both operations of choosing a gene and changing it with a data point in the same cluster will be done randomly.

4.2. Maximum Neighbor Random Change Mutation

In Maximum Neighbor Random Change Mutation we choose a gene randomly, and change it with a data point in data space according to number of neighbors and roulette wheel mechanism. Number of neighbors will be computed in a radius which can be user defined or program specific.

At the beginning of genetic algorithm we will compute the distance of each data point and all other data points for just one time and will use it during the whole algorithm. If the radius is not determined with the user, we find the distance of each data point and nearest neighbor of it and take the median of these distances as a radius to count the number of Neighbors of each data points in this radius. Then according to number of neighbors, we will choose one of data points with roulette wheel to change it with the selected gene.

4.3. Same Cluster Maximum Neighbor Random Change Mutation

In Same Cluster Maximum Neighbor Random Change Mutation we choose a gene randomly, and change it with a data point in the same cluster according to number of neighbors and roulette wheel mechanism. Computing the radius and number of neighbors has the same algorithm with Maximum Neighbor Random Change Mutation but we just examine the same cluster data points.

4.4. Nearest To Center Change Mutation

In Nearest To Center Change Mutation we choose a gene randomly, and change it with ith data point in the list of data points which have been sorted according to their distance to center of data space in ascending order. (i is a random number $1<i<N$).

4.5. Less population Change Mutation

In Less population Change Mutation we choose the gene which is the center of cluster with less population than other clusters and change it with another data point randomly.

4.6. Same cluster population Change Mutation

In Same cluster population Change Mutation we choose the gene which is the center of cluster with less population than other clusters and change it with a data point in the same cluster randomly.

In all of above proposed mutations if the chosen data point is one of the current genes of chromosome or leads to production of a chromosome which we have it currently in new generation, we will repeat the choose operation again.

5. Experimental results

We have implemented proposed genetic clustering algorithm and all explained mutation operators with java language in Netbeans IDE 7.1 environment and used weka.jar library of weka 3.6.6 software which contains necessary classes for implementing data clustering algorithms. To compare mutation operators we need to have the same parameters in each execution. So for each data set we have made 10 random initial populations with Population Size equal to 100. For every data set we will execute the algorithm 10 times with each initial population and record fitness of each generation elite, sum of fitness of each generation and the fitness of final result. Table 1 shows execution parameters.

Table 1.Execution parameters

Population Size	100
Number Of Generations	20
Pc	1
Pm	1
Reinsertion Type	complete
Selection Type	Roulette Wheel
Fitness Type	Squared Error
Crossover Type	One Point Crossover

Table 2 shows the datasets of our experimental results which consists of 9 famous real life and 5 artificial datasets [11] .Fig 2 shows these artificial datasets.

Table 2.Datasets

Dataset	instances	attributes	clusters
sonar	208	60	2
Ionosphere	351	34	2
iris	150	4	3
tae	151	5	3
wine	178	13	3
vehicle	846	18	4
glass	214	9	6
ecoli	336	7	8
vowel	990	10	11
DS 2	680	2	2
DS 3	600	2	3
DS 4	750	2	4
DS 5	1200	2	5
DS 6	1600	2	6
DS 7	1100	2	7

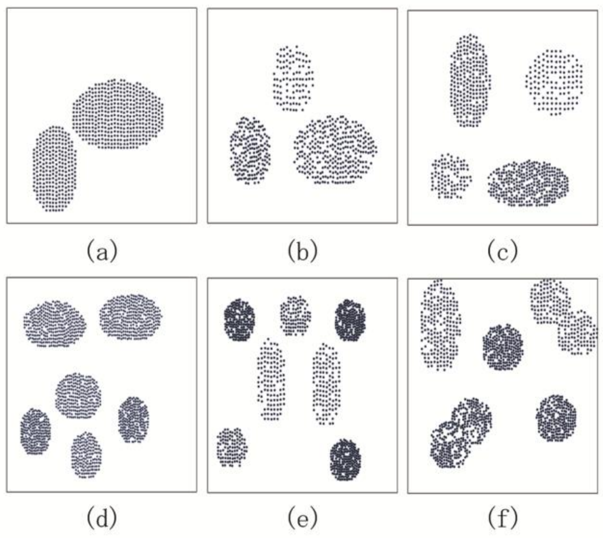

Fig.2.Artificial datasets (a) DS 2 (b) DS 3 (c) DS 4 (d) DS 5 (e) DS 6 (f) DS 7

After 100 executions of proposed genetic data clustering on each dataset with one of mutation operators, Table 3 shows the average of Fitness of final result of executions, Table 4 shows the average of fitness of best elite of generations and Table 5 shows the average of sum of fitness of generations for artificial datasets. Fig .3 also shows Comparison of mutation operators in average of fitness of final result for some artificial datasets.

Table 3.Fitness value of final elite

Mutation	DS 2	DS 3	DS 4	DS 5	DS 6	DS 7
Less Population Change	0.000778	0.001727	0.001075	0.000769	0.000619	0.000936
Same Cluster Population Change	0.000818	0.001757	0.001092	0.000733	0.000563	0.000884
Nearest To Center Change	**0.000949**	**0.001981**	**0.001372**	**0.000894**	**0.000683**	**0.001072**
Random Change	0.000817	0.001637	0.001031	0.000662	0.000517	0.000812
Maximum Neighbor Random Change	0.000814	0.001580	0.000661	0.000559	0.000496	0.000638
Same Cluster Maximum Neighbor Random Change	0.000856	0.001710	0.001070	0.000694	0.000597	0.000877
Same Cluster Random Change	0.000851	0.001746	0.001185	0.000770	0.000580	0.000949

Table 4.Average of fitness of best elite of generations for artificial data sets

Mutation	DS 2	DS 3	DS 4	DS 5	DS 6	DS 7
Less Population Change	0.000782	0.001701	0.001071	0.000743	0.000595	0.000922
Same Cluster Population Change	0.000819	0.001740	0.001102	0.000720	0.000549	0.000863
Nearest To Center Change	**0.000947**	**0.001943**	**0.001321**	**0.000839**	**0.000638**	**0.000987**
Random Change	0.000809	0.001617	0.001038	0.000654	0.000511	0.000806
Maximum Neighbor Random Change	0.000813	0.001569	0.000781	0.000592	0.000498	0.000706
Same Cluster Maximum Neighbor Random Change	0.000854	0.001701	0.001073	0.000695	0.000573	0.000867
Same Cluster Random Change	0.000849	0.001725	0.001159	0.000738	0.000564	0.000904

Table 5.Average of sum of fitness of generations for artificial data sets

Mutation	DS 2	DS 3	DS 4	DS 5	DS 6	DS 7
Less Population Change	0.064115	0.144946	0.088200	0.061810	0.051312	0.075828
Same Cluster Population Change	0.072587	0.155481	0.097389	0.062610	0.048038	0.074528
Nearest To Center Change	**0.083092**	**0.181120**	**0.120057**	**0.074200**	**0.056783**	**0.085600**
Random Change	0.061170	0.123517	0.079986	0.050516	0.039922	0.061039
Maximum Neighbor Random Change	0.061462	0.122417	0.061277	0.048523	0.038299	0.055702
Same Cluster Maximum Neighbor Random Change	0.072374	0.147012	0.093950	0.060413	0.049426	0.076052
Same Cluster Random Change	0.071299	0.147258	0.098780	0.063062	0.048753	0.076482

Table 6 shows the average of Fitness of final result of executions. Table 7 shows the average of fitness of best elite of generations and Table 8 shows average of sum of fitness of generations for real life datasets. Fig .4 also shows Comparison of mutation operators in average of fitness of final result for some real life datasets.

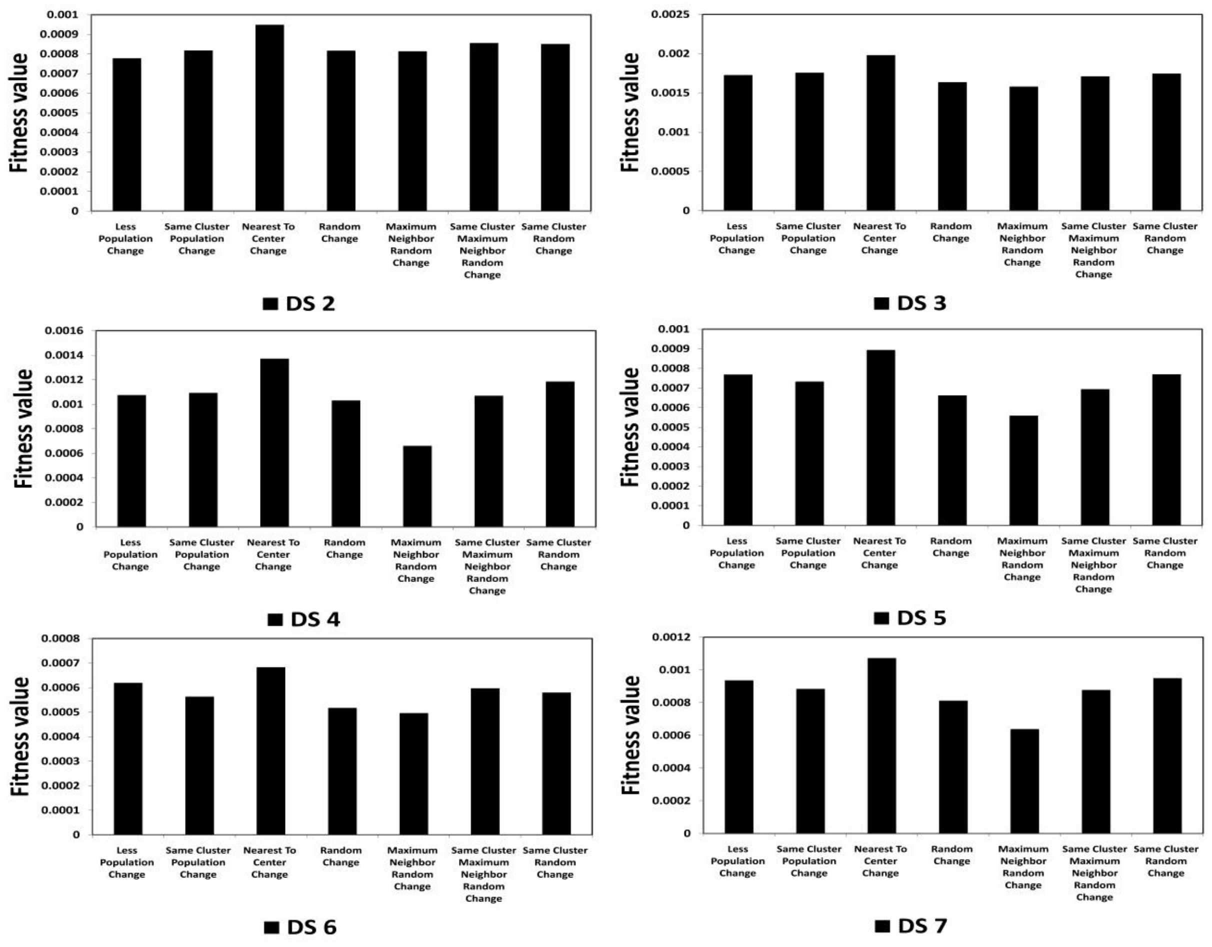

Fig.3. Comparison of mutation operators in average of fitness of final result for artificial datasets.

Table 6.Fitness value of final elite

Mutation	sonar	Ionosphere	iris	tae	wine	vehicle	glass	ecoli	vowel
Less Population Change	0.003545	0.001127	0.008462	0.000463	0.000039	0.000014	0.003331	0.013098	0.000474
Same Cluster Population Change	0.003542	0.001092	0.008602	0.000450	0.000039	0.000014	0.003251	0.012179	0.000461
Nearest To Center Change	**0.003793**	**0.001245**	**0.009518**	**0.000478**	**0.000042**	**0.000014**	**0.003441**	**0.013709**	**0.000501**
Random Change	0.003436	0.001093	0.008176	0.000461	0.000037	0.000014	0.003209	0.012273	0.000474
Maximum Neighbor Random Change	0.003573	0.001132	0.008378	0.000456	0.000038	0.000013	0.003047	0.012868	0.000480
Same Cluster Maximum Neighbor Random Change	0.003573	0.001161	0.009027	0.000445	0.000041	0.000014	0.003299	0.012967	0.000466
Same Cluster Random Change	0.003442	0.001074	0.008709	0.000444	0.000040	0.000014	0.003265	0.012190	0.000443

Table 7.Average of fitness of best elite of generations for real life data sets

Mutation	sonar	Ionosphere	iris	tae	wine	vehicle	glass	ecoli	vowel
Less Population Change	0.003538	0.001124	0.008439	0.000461	0.000038	0.0000140	0.003309	0.012924	0.000473
Same Cluster Population Change	0.003530	0.001092	0.008491	0.000452	0.000039	0.0000136	0.003238	0.012291	0.000463
Nearest To Center Change	**0.003782**	**0.001239**	**0.009491**	**0.000477**	**0.000443**	**0.0000145**	**0.003390**	**0.013431**	**0.000493**
Random Change	0.003449	0.001100	0.008161	0.000458	0.000037	0.0000137	0.003198	0.012306	0.000473
Maximum Neighbor Random Change	0.003588	0.001130	0.008421	0.000460	0.000038	0.0000130	0.003115	0.012725	0.000477
Same Cluster Maximum Neighbor Random Change	0.003569	0.001161	0.008964	0.000448	0.000040	0.0000139	0.003302	0.012839	0.000468
Same Cluster Random Change	0.003452	0.001080	0.008621	0.000039	0.000042	0.0000136	0.003268	0.012243	0.000443

Table 8.Average of sum of fitness of generations for real life data sets

Mutation	sonar	Ionosphere	Iris	tae	wine	vehicle	glass	ecoli	vowel
Less Population Change	0.325497	0.114004	0.100158	0.039446	0.003127	0.001179	0.292188	1.212852	0.045078
Same Cluster Population Change	0.323180	0.094399	0.727181	0.038622	0.003349	0.001111	0.280782	1.131965	0.044142
Nearest To Center Change	**0.353476**	**0.285848**	**0.884550**	**0.043544**	**0.003895**	**0.001213**	**0.295614**	**1.259608**	**0.047117**
Random Change	0.309992	0.091235	0.603596	0.038158	0.002994	0.001098	0.283310	1.106031	0.044361
Maximum Neighbor Random Change	0.327014	0.099970	0.617972	0.038701	0.002696	0.000975	0.270380	1.141883	0.044721
Same Cluster Maximum Neighbor Random Change	0.325757	0.102010	0.775866	0.038410	0.003556	0.001172	0.289711	1.195008	0.044360
Same Cluster Random Change	0.309962	0.087706	0.724886	0.037655	0.003158	0.001126	0.278136	1.112881	0.042142

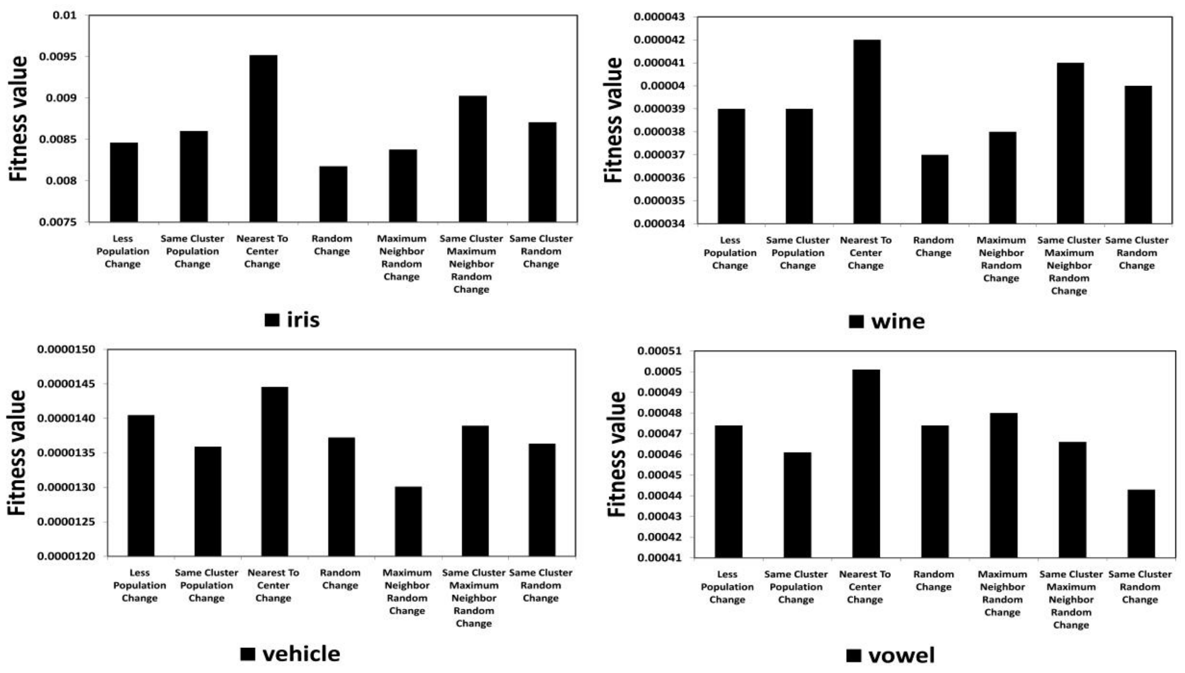

Fig.4. Comparison of mutation operators in average of fitness of final result for real life datasets.

6. Conclusions

In this paper we introduced 6 heuristic mutation operators for genetic data clustering which all of them averagely increase the fitness of final result, fitness of generation elite and sum of fitness of generations more than Random Change Mutation. According to experimental results, the best one of those is Nearest To Center Change Mutation .The second one is Less Population Change Mutation and the third one is Same Cluster Maximum Neighbor Random Change Mutation. We have used introduced heuristic mutation operators just in genetic data clustering and we can examine the usability and efficiency of proposed mutation operators for other problems as future work.

References

[1] John H. Holland, Adaptation in Natural and Artificial Systems, the University of Michigan Press, 1975.
[2] De Jong, Kenneth A. (1975). An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Doctoral thesis, Dept. Computer and Communication Sciences, University of Michigan, Ann Arbor.
[3] D. E. Goldberg, Genetic Algorithm in Search, Optimization and Machine Learning,Addison –Wesley, New York ,1989.
[4] Yongguo Liu , Xindong Wu, Yidong Shen , Automatic clustering using genetic algorithms, Applied Mathematics and Computation 218 (2011) 1267–1279.
[5] Hong He, YonghongTan, A two-stage genetic algorithm for automatic clustering, Neurocomputing 81 (2012) 49–59.
[6] L.E. Agustin-Blas , S. Salcedo-Sanz , S. Jimenez-Fernandez , L. Carro-Calvo , J. Del Ser ,J.A. Portilla-Figueras, A new grouping genetic algorithm for clustering problems, Expert Systems with Applications 39 (2012) 9695–9703.
[7] Dongxia Chang , Yao Zhao , Changwen Zheng , Xianda Zhang , A genetic clustering algorithm using a message-based similarity measure, Expert Systems with Applications 39 (2012) 2194–2202.

[8] Amin Aalaei, Hamed Fazlollahtabar, Iraj Mahdavi, Nezam Mahdavi-Amiri, Mohammad Hassan Yahyanejad, A genetic algorithm for a creativity matrix cubic space clustering: A case study in Mazandaran Gas Company, Applied Soft Computing 13 (2013) 1661–1673.

[9] Jose A. Castellanos-Garzon , Fernando Diaz, An evolutionary computational model applied to cluster analysis of DNA microarray data, Expert Systems with Applications 40 (2013) 2575–2591.

[10] Riccardo Poli , W. B. Langdon,Genetic programming with one-point crossover,In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing. Springer-Verlag London, 23-27 June 1997.

[11] http://repository.seasr.org/Datasets/UCI/arff/

[12] R. Maghsoudi , A. Ghorbannia Delavar, S. Hoseyny, R. Asgari, Y. Heidari, Representing the New Model for Improving K-Means Clustering Algorithm based on Genetic Algorithm, The Journal of Mathematics and Computer Science Vol .2 No.2 (2011) 329-336.