Journal of mathematics and computer science 11 (2014) 191-208

Contents list available at JMCS

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com

CCGDC: A new crossover operator for genetic data clustering

Gholam Hasan Mohebpour, Arash Ghorbannia Delavar

Department of Computer Science, Payame Noor University, Tehran, Iran

Department of Computer Science, Payame Noor University, Tehran, Iran

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

Article history:
Received April 2014
Accepted May 2014
Available online June 2014

Abstract

Genetic algorithm is an evolutionary algorithm and has been used to solve many problems such as data clustering. Most of genetic data clustering algorithms just have introduced new fitness function to improve the accuracy of algorithm in evaluation of generated chromosomes. Crossover operator is the backbone of the genetic algorithm and should create better offspring and increase the fitness of population with maintaining the genetic diversity. A good crossover should result in feasible offspring chromosomes when we crossover feasible parent chromosomes. In this paper we introduce a new crossover operator for genetic data clustering. Experimental results show that clustered crossover for genetic data clustering (CCGDC) creates better offspring and increases the fitness of population and also will not produce illegal chromosome.

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

1. Introduction

1.1. Data clustering

Let $X = \{x_1, x_2, \cdots, x_N\}$ be a set of N data points in m- dimensional data space R^m.Data clustering means partitioning these data points in to K groups $C = \{C_1, C_2, \dots, C_K\}$ as clusters 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 data point has the most similarity to its co-cluster data points and less similarity as possible to data points of other clusters according to a distance measure function $\mathrm{d}(\mathrm{x},\mathrm{y})$ like Euclidean distance.

1.2. K-means algorithm

K-means is one of the well-known algorithms for clustering. It is a center-based and unsupervised partitioning algorithm .K-means partitions the dataset into k mutually exclusive clusters, and treats each data point as an object having a specific location in data space. It finds a partition in which data points within each cluster are as close to each other as possible, and as far from data points in other clusters as possible. It selects k data points as cluster centers randomly and tries to minimize sum of squared error. At the next steps the mean of each cluster will be computed as cluster center. 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.

1.3. Genetic algorithms

Genetic algorithm is a search algorithm which is based on the biological evolution and originally developed by Holland [1] and later refined by De Jong [2], Goldberg [3], and many others. It is a search heuristic that mimics the process of natural evolution and principle of survival of the fittest laid by Charles Darwin.

In genetic algorithm we generate an initial population consist of a specific number of individuals and then our objective is to reach to a generation that has better fitness values than the last generations, as it happens in nature.
In other words, in nature, each species has to change its chromosome combination to survive in the living world. Genetic algorithm will mimic this nature rule and tries to generate better offspring. In Genetic algorithm each chromosome of the population will be evaluated and assigned a value derived from fitness function and then chromosomes with better fitness values will be more likely to be selected for producing new offspring. A competitive strategy was employed to improve the selection performance such as roulette wheel or tournament selection method. After that, crossover will be done on selected parents and finally mutation will be used on generated offspring. If stopping criteria didn't reached the whole steps will be repeated.Fig.1 shows the flowchart of genetic algorithm.

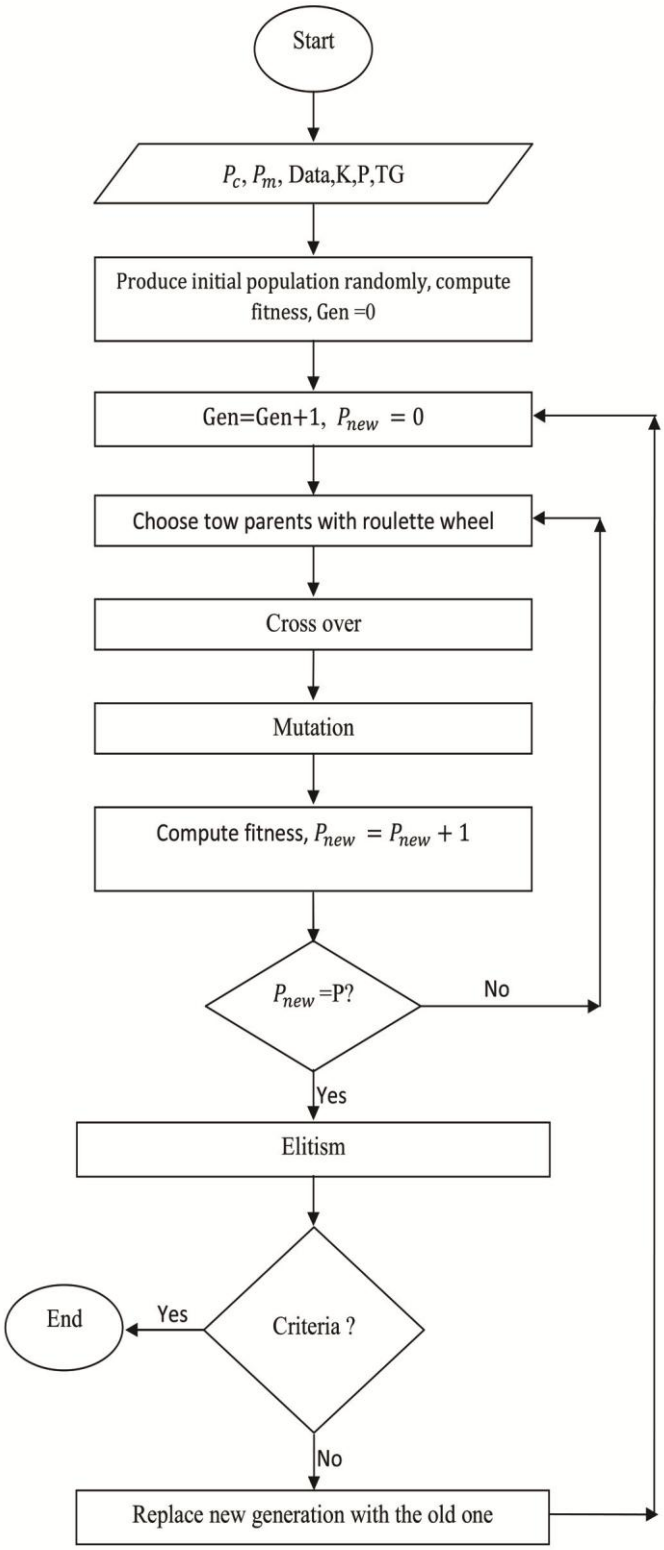

Fig.1. Flowchart of genetic algorithm

If we try to categorize structure of chromosomes in genetic algorithm, four aspects that are important and have more effects on the structure of chromosomes and depend to the problem that we want to solve are:

- **Length**
  Depending on the problem, we can have chromosome structures with Fixed-length or Variable-length. For instance in Traveling Salesman Problem which number of cities will be specific from the beginning, we will have a Fixed-length chromosome structure.
- **Order**
  An ordered chromosome or Position-based chromosome is the one that the place of genes are important and each permutation of the same genes will be decoded as a different solution of the problem, we call the position of each gene, Locus. For example in Traveling Salesman Problem, the genes are ordered and any single permutation of n cities yields a different solution. We call the encodings with ordered chromosomes a permutation encoding. In permutation encoding, every gene in chromosome represents a position in a sequence.
- **Gene structure**
  For some problems, it is necessary to have genes with different alleles. We call these kinds of encodings direct value encoding and it can be used in problems where some more complicated values such as real numbers are used. Use of binary encoding for this type of problems would be difficult. In value encoding, every chromosome is a sequence of genes which they can be anything connected to the problem, such as real numbers, characters, strings or any objects.
- **Gene repetition**
  In some problems like Traveling Salesman Problem, which each gene shows one of the visited cities and each city will be visited exactly once, gene repetition is not allowed, but in problems like finding the roots of an equation which two or more roots may be equal, genes may have same values.

The above aspects not only effect on the structure of chromosome and problem encoding, but also they effect on genetic operators like crossover and mutation. For example ordered crossover operator developed by Davis [4] or cycle crossover operator proposed by Oliver et al. [5] are suitable for ordered chromosomes and permutation encoding.
There are varieties of crossover and mutation operators which differ from each other in described aspects. Although there are some famous and widely used crossover and mutation operators, you cannot find a crossover or mutation operator which is suitable for all kinds of encodings and chromosome structures.

In rest of this paper we will explain problem encoding for our genetic data clustering and introduce a new crossover operator which is suitable for genetic data clustering and compare it with ten famous crossover operators.

2. Related work

Since 1975, several attempts have been done to improve efficiency of genetic algorithm. These attempts have been done in different aspects of genetic algorithm such as initial population, fitness function, crossover and mutation operator .Crossover operator should create a better offspring and increase the fitness of population and prevent from inheriting just good genes to maintain the genetic diversity.

Wu and Chow [6] compared the one-point, two-point, three-point, and four-point crossover operators and showed that two-point, three-point, and four-point crossover operators are better than the one-point crossover.
Jenkins [7] argues in favor of multi-point crossover operator in term of fast progress becomes very slow in case single-point crossover is used. Using one-point crossover,
Dejong and Spears [8] introduced the relationship between crossover operators and population size. They state that two-point crossover is performs better in the problems in which the population is large, but uniform crossover is better for the small size populations.
Syswerda [9] showed that the uniform crossover operator is more efficient when compared with two-point crossover.
Erbatur and Hasanc¸ebi [10, 11] suggested combining two crossover operators in their study about the effects of crossover operators on the behavior of GA.
Mustafa Kaya [12] has introduced sequential and random mixed crossover operators and has compared them with other crossover operators on RC beam and the space truss problems
Hong He and Yonghong Tan [13] have used a parallel crossover for automatic clustering of data without having number of clusters as input parameter. Their parallel crossover uses one point crossover and exchanges genes in length equal to smaller individual length.
Dongxia Chang and et al. [14] introduce a genetic clustering algorithm using a message-based similarity measure for automatic data clustering but they also use one point crossover.
Amin Aalaei et al. [15] have used a four point crossover operator for their matrix based chromosome structure to select a sub matrix from each parent chromosome and exchange it.
Jose A. Castellanos-Garzon and Fernando Diaz [16] have proposed a new hierarchical clustering method using genetic algorithms for the analysis of gene expression data. They have used a crossover operator which works on parent's dendrogram to obtain a child dendrogram.
After reviewing some known crossover operators, we find that they neglect the fact that when the algorithm converges to a solution, most of genes of individuals will be same and they produce illegal offspring and also to maintain population diversity they decrease population fitness and some of them cannot be used when we have a chromosome with few number of genes.

3. Program encoding

3.1. Chromosome representation

In our genetic data clustering problem, 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 dataset. The chromosome structure of our genetic data clustering problem is not ordered which means that the place of genes is not important and any permutation of genes produces the same chromosome.

This structure will not produce any feasible chromosome but illegal chromosomes may 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 crossover operator we use the simplest fitness function for genetic data clustering. Objective function of k-means is defined as follow:

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

Where c_i is the center of i th 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:

$$F_i = \frac{\mathbf{1}}{\boldsymbol{E}} \quad (2)$$

3.3. Selection

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

The probability Pi for each individual is defined by Equation () and F_i is the fitness of i th individual.

$$p_i = \frac{F_i}{\sum_{j=1}^{pop\ size} F_j} \quad (3)$$

In roulette wheel selection, the individuals are mapped to sectors 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. 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.

4. Crossover operators

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.

Crossover is the backbone of the genetic algorithm and applied to mating pool with hope that it creates a better offspring and increases the fitness of population. Crossover prevents from inheriting just good genes to maintain the genetic diversity. A good crossover should result in feasible offspring chromosomes when we crossover feasible parent chromosomes.

As we mentioned in introduction section ,there are varieties of crossover and mutation operators which differ from each other in aspects like having fixed or variable length, being ordered or not and gene repetition .You cannot find a crossover or mutation operator which is suitable for all kinds of encodings and chromosome structures. In center-based genetic data clustering algorithm with predefined number of clusters, we need a Fixed-length, in ordered, identical gene structure without gene repetition chromosome structure and so we need crossover operators which are usable for this problem encoding. Now we explain ten famous and widely used crossover operators that can be used in our genetic data clustering problem.

4.1. One Point Crossover[17]

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.Fig.2 (a) shows a one point crossover.

4.2. Two Point Crossover[1][18]

In two point crossover two cutoff points between 1 and length of chromosome ($0<C_1<C_2<L$) are randomly selected to divide each parent chromosome in to three parts then first part of the first parent, second part of the second parent and third part of the first parent generate the first offspring and first part of the second parent, second part of the first parent and third part of second parent generate the second offspring. Fig.2 (b) shows a two point crossover.

4.3. Three Point Crossover[19]

In three point crossover three cutoff points between 1 and length of chromosome ($0<C_1<C_2<C_3<L$) are randomly selected to divide each parent chromosome in to five parts then odd parts of the first parent and even parts of second parent generate the first offspring and odd parts of the second parent and even parts of first parent generate the second offspring. Fig.2 (c) shows a three point crossover.

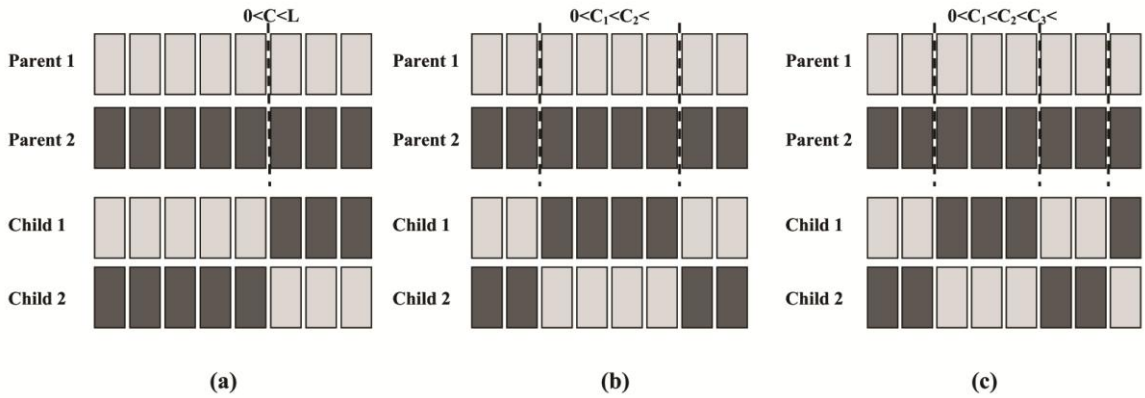

Fig.2. (a) One point crossover (b) Two point crossover (c) Three point crossover

4.4. Multi Point Crossover[19]

In multi point crossover N cutoff points between 1 and length of chromosome ($0< C_1<\ldots<C_n <L$) are randomly selected to divide each parent chromosome in to N+2 parts then odd parts of the first parent and even parts of the second parent generate the first offspring and odd parts of the second parent and even parts of the first parent generate the second offspring.(N will be selected between 0 and L randomly).

4.5. Uniform Crossover[20]

In uniform crossover we will form a binary mask with the same length with chromosomes of parents. Then randomly generate a binary digit for this mask. If the code of the nth site of the mask is 1, we will copy nth gene of the first parent to the first offspring and copy nth gene of the second parent to the second offspring. If the code of the nth site of the mask is 0, we will copy nth gene of the first parent to the second offspring and copy nth gene of the second parent to the first offspring. Fig.3 (a) shows a Uniform point crossover.

4.6. Variable to variable Crossover[21]

In variable to variable crossover odd genes of the first parent and even genes of the second parent generate the first offspring and odd genes of the second parent and even genes of the first parent generate the second offspring. Fig.3 (b) shows a variable to variable crossover.

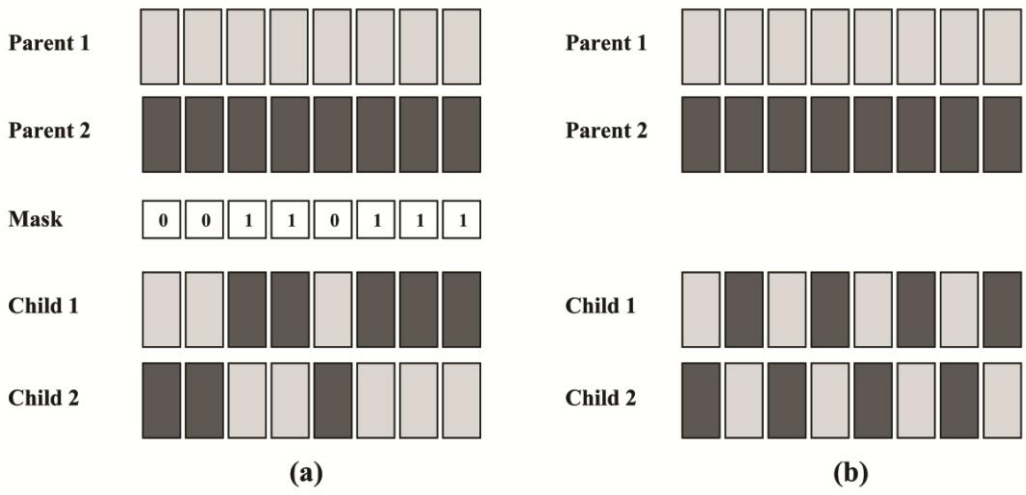

Fig.3. (a) Uniform crossover (b) Variable to variable Crossover

4.7. Mixed Crossover[21]

The mixed crossover uses a combination of one point, two point and three point crossovers .It will randomly use one of these crossovers to generate offspring.

4.8. Random Mixed Crossover[19]

The Random mixed crossover randomly uses one of one point, two point, three point, multi point, uniform, variable to variable or mixed crossovers to generate offspring.

4.9. Sequential Crossover[19]

In Sequential Crossover, one point, two point, three point, multi point, uniform, variable to variable, mixed and random mixed crossovers will be used sequentially on the parent chromosomes to generate offspring.

4.10. Region Crossover

In region crossover all genes of the first parent will be copied in to the first offspring and genes of the second parent will copied to the second offspring .after that two cutoff points between 1 and length of chromosome ($0<C_1<C_2<L$) are randomly selected to divide each parent chromosome in to three parts then the middle parts of two offspring will be swapped.

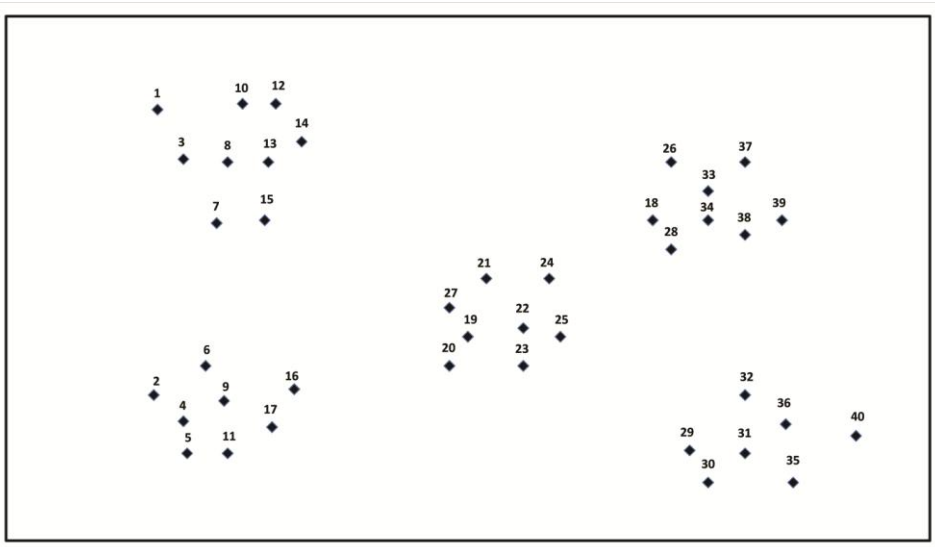

Fig.4.A Hypothetical dataset with index of each data point in dataset

Fig.4 shows a hypothetical dataset consist of five clusters with forty data points. The numbers show the indices of data points in dataset and won't change during the execution. Any non repetitive combination of these indices with a length equal with five can be a solution to our center-based clustering algorithm and each index shows the center data point of a cluster. For example {8, 17, 22, 31, 33}, {4, 8, 18, 22, 31} are two solutions. If our selection mechanism choose {8, 17, 22, 31, 33}, {4, 8, 18, 22, 31} as parents to be copied into mating pool, Fig.5 (a) to (f) shows the offspring generated by six of explained crossover operators. The repetition of genes that generate illegal chromosomes, have been showed with dotted rectangle.

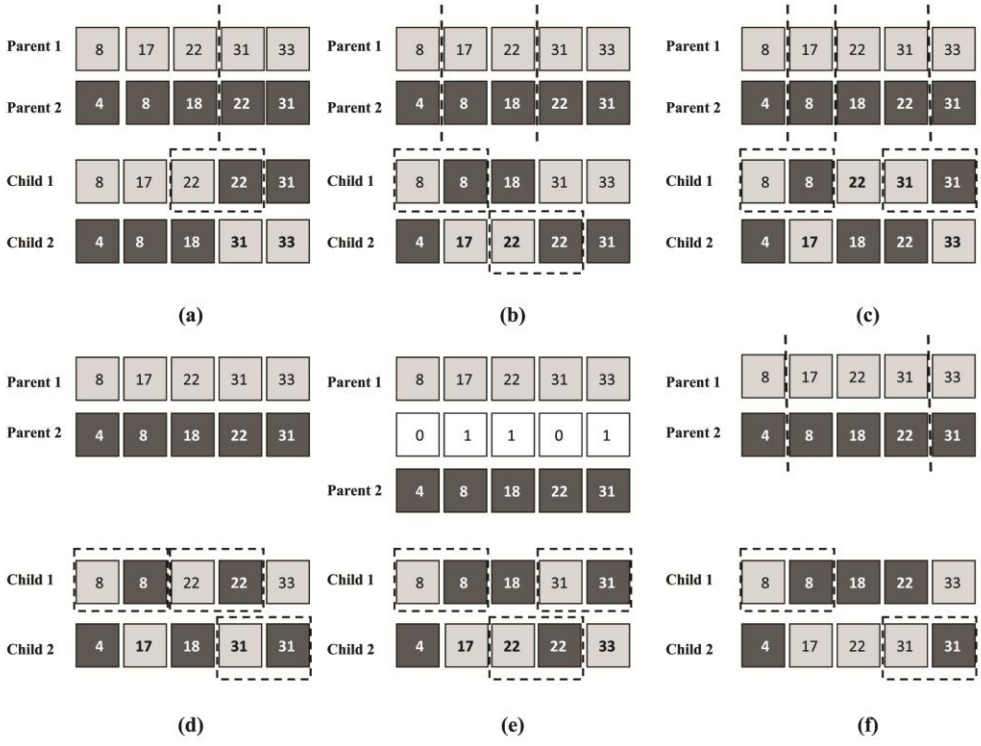

Fig.5 .Example of illegal offspring (a) one point crossover (b) two point crossover (c) Three point crossover (d) Variable to variable Crossover (e) Uniform crossover (f) Region Crossover

5. Proposed Crossover operator

Fig.4 shows a hypothetical dataset consist of five clusters with forty data points. Suppose that P1= {8,17,22,31,33},P2={4,8,18,22,31} have been chosen with selection mechanism as parents. We will introduce the proposed crossover with an example. At first Combine all genes of both parents and make a set {8,17,22,31,33}+{4,8,18,22,31}={ 8,17,22,31,33,4,8,18,22,31} then remove all duplicated genes from the set with their duplications. The duplicated or common genes are {8, 22, 31} .Then make a set with unique genes {17, 33, 4, 18}.After that sort the unique genes {4, 17, 18, 33}.Determine the number of duplicated or common genes and name it R.In our example R=3.Copy the common genes to offspring {8, 22, 31}.Cluster unique genes in K=L- R clusters where K>1.In our example k=2 (5-3=2) and as Fig. 6 shows, two clusters are {4, 17} and {18, 33}.If clustering algorithm hasn't detected any empty clusters, copy a gene from each cluster to offspring randomly .All of possible generated chromosomes have been shown in Fig.7. If clustering algorithm detected any empty clusters, we should randomly copy genes which haven't copied to offspring, instead of empty clusters.

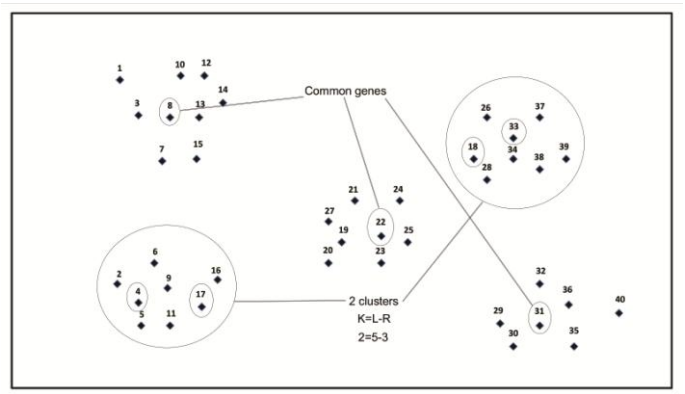

Fig.6.Example of clustered crossover

The steps of clustered crossover are as follow:

Step 1. Combine all genes of both parents and make a set.

Step 2. Remove all duplicated genes from the set with their duplications and make a set with them.

Step 3. Sort the remained genes.

Step 4. Determine the number of duplicated or common genes and name it R.

Step 5. Copy the duplicated genes to offspring.

Step 6. IF there is more than one cluster THEN cluster unique genes in L- R clusters and go to Step 7 ELSE select a non duplicated gene randomly and go to Step 8.

Step 7. FOR EACH cluster DO

Step 7.1. IF it is an empty cluster THEN go to Step 7.3 ELSE go to Step 7.2.
Step 7.2. Copy a gene from cluster to offspring randomly and go to Step 7.
Step 7.3. Copy genes that have not copied in previous steps, instead of empty cluster randomly and go to Step 7.

Step 8. Sort the genes of offspring and finish.

We can use any clustering algorithm for clustering non repeated genes. We have used k-means algorithm. As you know the order of k-means algorithm is O (KNI) where K is the number of clusters, N is the number of data points in dataset and I is the number of iterations. In the worst case which we don't have any common gene between parents, we totally have 2K data points and we want to cluster them in k clusters. So the order of clustered crossover in the worst case is equal to $O(2K^2I)$ .The best case which is undefined and impossible happens when all the genes of parents are equal .In the best case the order of clustered crossover is equal to O(1).

In problems with chromosome structures which repetition of genes generate illegal chromosomes all the available crossover operators may generate illegal chromosomes.
In cases that we have parent chromosomes with lots of common genes and datasets that have few clusters, we cannot use multi point crossover with lots of cutoff points because the probability of generating illegal offspring will be increased. Even using one point, two point or three point crossover may generate illegal offspring when we have datasets with less than 5 clusters.

In center-based genetic clustering algorithms with a suitable fitness function, when the algorithm converges, ideally most of genes of different chromosomes of generation should have same values. For example according to Fig.4 chromosomes like {8,17,22,31,33} and {4,8,18,22,31} seems to have chosen cluster centers of three clusters correctly {8 ,31, 33}, but none of available crossover operators are heuristic to pay attention to common genes , especially in center-based genetic clustering algorithms that we encoded which repetition of genes generates illegal chromosomes. So this weakness of available crossover operators put off convergence.

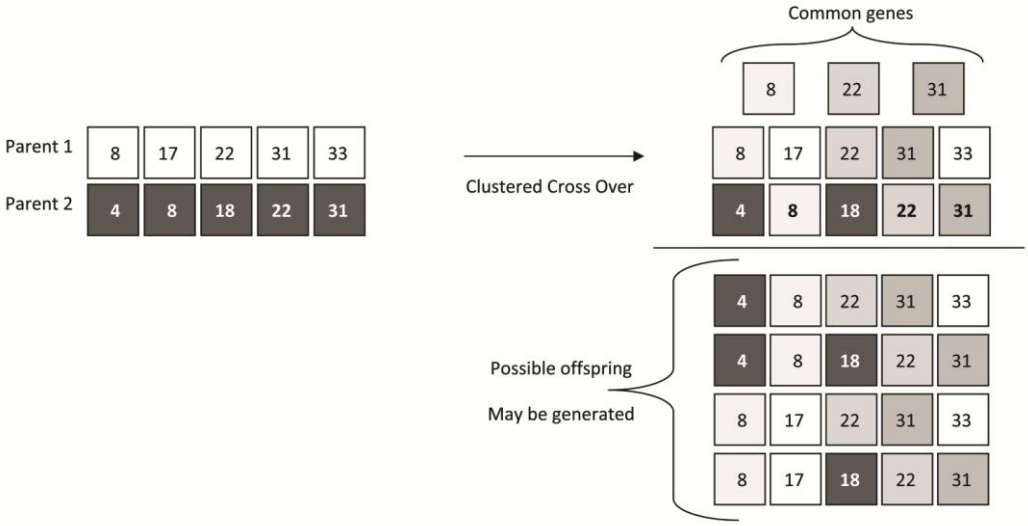

Fig.7.Possible offspring may be generated with clustered crossover

6. Experimental results

We have implemented proposed genetic clustering algorithm and all explained crossover operators with java language in Netbeens 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 crossover operators we need to have the same parameters in each execution and just change crossover operator. So for each dataset we have made 10 random initial populations with Population Size equal to 100 and prevent from mutation and elitism operators by setting mutation probability P_m=0 and elitism

probability P_e=0. For every dataset we will execute the algorithm 10 times with each initial population and record fitness of elite, sum of fitness of each generation, fitness of final result and number of illegal generated offspring. Table 1 shows execution parameters.

Table 1.Execution parameters

Population Size	100
Number Of Generations	20
P_c	1
P_m	0
P_e	0
Reinsertion Type	Complete
Selection Type	Roulette Wheel
Fitness Type	Squared Error

Table 2 shows the datasets of our experimental results which consists of 9 famous real life datasets [22] and 5 artificial datasets. Fig .8 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

After 100 executions of proposed genetic data clustering on each dataset with one of crossovers, Table 3 shows the average of illegal generated offspring, Table 4 shows average of Fitness of final result of executions, Table 5 shows average of fitness of best elite of generations and Table 6 shows average of sum of fitness of generations for artificial datasets.

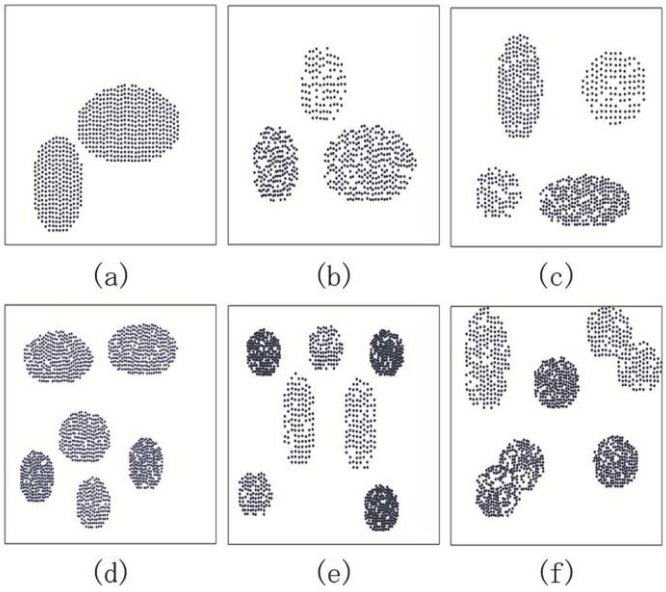

Fig .8.Artificial datasets (a) DS 2 (b) DS 3 (c) DS 4 (d) DS 5 (e) DS 6 (f) DS 7

Region Crossover, Two Point Crossover and Three Point Crossover are not usable for datasets with 2 clusters and Three Point Crossover is not also usable for datasets with 3 clusters, so in all result tables we don't have any value for these datasets.

Table 3.Average of illegal generated offspring in each execution for artificial datasets

Crossover	DS 2	DS 3	DS 4	DS 5	DS 6
Clustered	0.00	0.00	0.00	0.00	0.00
Mixed	542.21	215.25	3.53	4.15	4.43
Multi Point	2.18	2.06	3.62	3.93	4.01
One Point	1.38	2.85	3.50	4.20	4.70
Random Mixed	363.70	142.98	4.92	7.51	7.53
Region	-	2.17	3.86	5.31	5.29
Sequential	1273.29	1004.32	422.99	440.21	401.38
Three Point	-	-	3.08	3.67	4.08
Two Point	-	1.84	3.21	3.76	4.99
Uniform	2.21	3.82	6.79	10.32	12.50
Variable to variable	2.26	6.23	12.25	16.27	20.36

Table 4.Average of Fitness of final result of executions for artificial datasets

Crossover	DS 2	DS 3	DS 4	DS 5	DS 6
Clustered	**0.00084801**	**0.00168425**	**0.00114500**	**0.00072495**	**0.00056112**
Mixed	0.00084251	0.00166363	0.00107163	0.00067577	0.00052513
Multi Point	0.00083737	0.00167924	0.00106174	0.00067601	0.00052975
One Point	0.00083335	0.00165841	0.00108611	0.00067152	0.00052656
Random Mixed	0.00082711	0.00166024	0.00109162	0.00068310	0.00053573
Region	-	0.00166160	0.00108675	0.00068533	0.00051703
Sequential	0.00083570	0.00159192	0.00096490	0.00059990	0.00044618
Three Point	-	-	0.00102699	0.00067304	0.00051788

Two Point	-	0.00161057	0.00107561	0.00066429	0.00051719
Uniform	0.00084311	0.00167691	0.00109459	0.00067648	0.00053040
Variable to variable	0.00083917	0.00167202	0.00110183	0.00068983	0.00055876

Table 5.Average of fitness of best elite of generations for artificial datasets

Crossover	DS 2	DS 3	DS 4	DS 5	DS 6
Clustered	**0.00085**	**0.00168**	**0.00114**	**0.00073**	**0.00057**
Mixed	0.00084	0.00166	0.00107	0.00067	0.00052
Multi Point	0.00084	0.00167	0.00106	0.00067	0.00052
One Point	0.00083	0.00165	0.00108	0.00066	0.00052
Random Mixed	0.00083	0.00166	0.00108	0.00068	0.00053
Region	-	0.00166	0.00108	0.00068	0.00051
Sequential	0.00084	0.00159	0.00099	0.00061	0.00046
Three Point	-	-	0.00103	0.00066	0.00051
Two Point	-	0.00161	0.00107	0.00066	0.00051
Uniform	0.00084	0.00166	0.00109	0.00067	0.00053
Variable to variable	0.00084	0.00167	0.00110	0.00069	0.00056

Table 6.Average of sum of fitness of generations for artificial datasets

Crossover	DS 2	DS 3	DS 4	DS 5	DS 6
Clustered	**0.07382**	**0.14452**	**0.10054**	**0.06442**	**0.05021**
Mixed	0.06928	0.14130	0.09100	0.05723	0.04418
Multi Point	0.06984	0.14433	0.09014	0.05712	0.04402
One Point	0.07016	0.14186	0.09237	0.05658	0.04449
Random Mixed	0.06945	0.14284	0.09162	0.05827	0.04599
Region	-	0.14317	0.09231	0.05813	0.04441
Sequential	0.06028	0.11676	0.07696	0.04872	0.03627
Three Point	-	-	0.00851	0.00566	0.00436
Two Point	-	0.13767	0.00908	0.00564	0.00438
Uniform	0.06972	0.14420	0.09310	0.05833	0.04570
Variable to variable	0.06983	0.14237	0.09270	0.05802	0.04697

After 100 executions of proposed genetic data clustering on each real life dataset with one of crossovers, Table 7 shows the average of illegal generated offspring, Table 8 shows average of Fitness of final result of executions, Table 9 shows average of fitness of best elite of generations and Table 10 shows average of sum of fitness of generations for real life datasets.

Table 7.Average of illegal generated offspring in each execution for real life datasets

Crossover	sonar	Ionosphere	iris	tae	wine	vehicle	glass	ecoli	vowel
Clustered	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mixed	519.74	528.92	223.88	218.35	227.22	4.60	6.40	6.90	8.23
Multi Point	4.14	2.92	3.16	3.83	3.23	5.05	5.13	5.30	64.02
One Point	4.47	3.70	3.47	5.84	3.91	5.88	6.70	8.29	8.29
Random Mixed	346.55	350.05	147.28	146.99	153.23	6.88	10.62	15.56	22.52
Region	-	-	2.61	4.36	3.85	4.60	7.36	8.75	11.05
Sequential	1178.32	1233.31	1093.07	977.51	1025.57	432.95	433.83	384.92	369.73
Three Point	-	-	-	-	-	3.58	5.56	5.86	7.80
Two Point	-	-	2.53	2.37	3.16	4.74	6.91	6.97	8.63
Uniform	4.94	4.78	5.47	6.07	4.36	9.99	14.07	20.12	33.91
Variable to variable	6.19	4.74	7.22	12.36	7.91	14.28	23.19	33.01	44.27

Table 8.Average of Fitness of final result of executions for real life datasets

Crossover	Sonar	Ionosphere	iris	tae	wine	vehicle	glass	ecoli	vowel
Clustered	**3.46178E-03**	**1.11426E-03**	**8.50079E-03**	**4.70357E-04**	**3.77442E-05**	**1.39398E-05**	**3.28827E-03**	**1.22544E-02**	**4.66348E-04**
Mixed	3.44348E-03	1.08846E-03	8.44622E-03	4.61394E-04	3.72245E-05	1.37703E-05	3.18056E-03	1.21832E-02	4.58374E-04
Multi Point	3.43343E-03	1.09319E-03	8.47241E-03	4.58741E-04	3.75412E-05	1.37981E-05	3.17294E-03	1.21759E-02	4.61440E-04
One Point	3.43915E-03	1.07638E-03	8.41907E-03	4.56790E-04	3.74026E-05	1.37976E-05	3.20884E-03	1.21713E-02	4.59968E-04
Random Mixed	3.45492E-03	1.08221E-03	8.46690E-03	4.64791E-04	3.74060E-05	1.39277E-05	3.23321E-03	1.20635E-02	4.55068E-04
Region	**3.46178E-03**	**1.11426E-03**	8.42433E-03	4.63160E-04	3.76147E-05	1.37676E-05	3.19985E-03	1.16865E-02	4.56309E-04
Sequential	3.45506E-03	1.10726E-03	8.25742E-03	4.66323E-04	3.47172E-05	1.34510E-05	3.01580E-03	1.14654E-02	4.38889E-04
Three Point	-	-	-	-	-	1.37898E-05	3.17318E-03	1.20970E-02	4.60977E-04
Two Point	-	-	8.42637E-03	4.54271E-04	3.66757E-05	1.37638E-05	3.24468E-03	1.21709E-02	4.60893E-04
Uniform	3.43668E-03	1.08869E-03	8.49173E-03	4.51794E-04	3.76887E-05	1.38697E-05	3.18510E-03	1.21299E-02	4.54180E-04
Variable to variable	3.41493E-03	1.08983E-03	8.46372E-03	4.40359E-04	3.73028E-05	1.38589E-05	3.22982E-03	1.22375E-02	4.57757E-04

Table 9.Average of fitness of best elite of generations for artificial datasets

Crossover	sonar	Ionosphere	iris	tae	wine	vehicle	glass	ecoli	vowel
Clustered	**3.46215E-03**	**1.11426E-03**	**8.48134E-03**	**4.70357E-04**	**3.74480E-05**	**1.38692E-05**	**3.26546E-03**	**1.22807E-02**	**4.69179E-04**
Mixed	3.46178E-03	1.09248E-03	8.40545E-03	4.62500E-04	3.66855E-05	1.37766E-05	3.18341E-03	1.22074E-02	4.63073E-04
Multi Point	3.43466E-03	1.09815E-03	8.44695E-03	4.59853E-04	3.71451E-05	1.37675E-05	3.18183E-03	1.22015E-02	4.66557E-04
One Point	3.44399E-03	1.08034E-03	8.42109E-03	4.56782E-04	3.70485E-05	1.37606E-05	3.19743E-03	1.22281E-02	4.65668E-04
Random Mixed	3.45306E-03	1.08597E-03	8.41131E-03	4.62953E-04	3.69083E-05	1.37768E-05	3.24134E-03	1.21803E-02	4.61419E-04
Region	3.46178E-03	**1.11426E-03**	8.37041E-03	4.60134E-04	3.71253E-05	1.36860E-05	3.20055E-03	1.18944E-02	4.61574E-04
Sequential	3.45738E-03	1.10597E-03	8.25742E-03	**4.70357E-04**	3.47172E-05	1.35268E-05	3.08040E-03	1.17781E-02	4.49285E-04
Three Point	-	-	-	-	-	1.37681E-05	3.19158E-03	1.21422E-02	4.64935E-04
Two Point	-	-	8.42523E-03	4.56367E-04	3.64395E-05	1.37318E-05	3.22722E-03	1.21902E-02	4.65376E-04
Uniform	3.43855E-03	1.09162E-03	8.45137E-03	4.54065E-04	3.72689E-05	1.38459E-05	3.19442E-03	1.22374E-02	4.60478E-04
Variable to variable	3.42109E-03	1.09029E-03	8.44019E-03	4.42438E-04	3.69262E-05	1.38059E-05	3.22020E-03	1.22393E-02	4.60366E-04

Table 10.Average of sum of fitness of generations for real life datasets

Crossover	sonar	Ionosphere	iris	tae	wine	vehicle	glass	ecoli	vowel
Clustered	**3.14565E+00**	**9.47127E-01**	**7.21828E+00**	**3.94914E-01**	**3.00118E-02**	**1.19574E-02**	**2.85054E+00**	**1.14176E+01**	**4.43803E-01**
Mixed	3.13190E+00	9.39208E-01	7.17379E+00	3.92357E-01	2.95199E-02	1.16593E-02	2.81073E+00	1.13286E+01	4.40774E-01
Multi Point	3.12041E+00	9.39643E-01	7.20950E+00	3.91995E-01	2.98675E-02	1.17057E-02	2.81932E+00	1.12770E+01	4.43712E-01
One Point	3.13115E+00	9.20178E-01	7.11157E+00	3.89313E-01	2.96480E-02	1.16678E-02	2.82389E+00	1.13720E+01	4.42896E-01
Random Mixed	3.13125E+00	9.31567E-01	7.12949E+00	3.93548E-01	2.92679E-02	1.17440E-02	2.84640E+00	1.13133E+01	4.40585E-01
Region	3.11721E+00	8.95216E-01	7.12095E+00	3.93526E-01	2.95540E-02	1.15751E-02	2.82734E+00	1.09158E+01	4.41624E-01
Sequential	3.11721E+00	8.95216E-01	5.89800E+00	3.77758E-01	2.37243E-02	1.09860E-02	2.67962E+00	1.04928E+01	4.27188E-01
Three Point	-	-	-	-	-	1.15231E-02	2.81915E+00	1.13231E+01	4.43459E-01
Two Point	-	-	7.04533E+00	3.78926E-01	2.79687E-02	1.16811E-02	2.84111E+00	1.13005E+01	4.41625E-01
Uniform	3.12874E+00	9.36603E-01	7.13047E+00	3.87296E-01	2.97955E-02	1.17875E-02	2.81906E+00	1.13314E+01	4.39849E-01
Variable to variable	3.11597E+00	9.38340E-01	7.01385E+00	3.86687E-01	2.92487E-02	1.15631E-02	2.74941E+00	1.13072E+01	4.37571E-01

7. Conclusions

According to our experimental results, clustered crossover for genetic data clustering (CCGDC) creates better offspring and increases the fitness of population with maintaining the genetic diversity and results in feasible offspring chromosomes. Unlike two point and three point crossovers, the number of chromosomes will not effect on efficiency and usability of proposed crossover .With a suitable fitness function, when the algorithm converges, ideally most of genes of different chromosomes of generation should have same values. Proposed crossover is heuristic enough to pay attention to these common genes.

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] L. Davis,Job-shop Scheduling with Genetic Algorithms". Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp. 136-140, 1985.

[5] I.M. Oliver, D. J. Smith and J.R.C. Holland. "A Study of Permutation Crossover Operators on the Travelling Salesman Problem". In J.J. Grefenstette (ed.). Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms. Lawrence Erlbaum Associates, Hilladale, NJ, 1987.

[6] S.J. Wu, P.T. Chow, Steady-state genetic algorithm for discrete optimization of trusses, Computers and Structures 56 (6) (1995) 979–991.

[7] W.M. Jenkins, On the application of natural algorithms to structural design optimization, Engineering Structure 19 (4) (1997) 302–308.

[8] K. Dejong, W.M. Spears, An analysis of the interacting roles of population sizes and crossover in genetic function optimization, in: H.P. Schwefel, R. Manner (Eds.), Proceedings of Parallel Problem Solving from Nature, Springer, Berlin, 1990, pp. 38–47.

[9] G. Syswerda, Uniform crossover in genetic algorithms, in: J.D. Schaffer, M.Kaufman (Eds.), Proceedings of the Third International Conference on Genetic Algorithms, 1989, pp. 2–9.

[10] O. Hasanc¸ ebi, F. Erbatur, Evaluation of crossover operators in genetic algorithms based optimum structural design, Computers and Structures 1999 78 (2000) 435–448.

[11] D. Zaharie, Influence of crossover on the behavior of differential evolution algorithms, Applied Soft Computing 9 (June (3)) (2009) 1126–1138.

[12] Mustafa Kaya, The effects of two new crossover operators on genetic algorithm performance, Applied Soft Computing 11 (2011) 881–890.
[13] Hong He , YonghongTan, A two-stage genetic algorithm for automatic clustering, Neurocomputing 81 (2012) 49–59.
[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] V. A. Cicirello and S. F. Smith,Modeling GA performance for control parameter optimization,In GECCO-2000: Proceedings of the Genetic and Evolutionary Computation Conference, pages 235–242.Morgan Kaufmann Publishers, 8-12 July 2000.
[19] M, Kaya, The effects of two new crossover operators on genetic algorithm performance, Applied Soft Computing, 11(1) (2011) 881-890.
[20] G. Syswerda, Uniform crossover in genetic algorithms, in: J.D. Schaffer, M.Kaufman (Eds.), Proceedings of the Third International Conference on Genetic Algorithms, 1989, pp. 2–9.
[21] O. Hasanc¸ ebi, F. Erbatur, Evaluation of crossover operators in genetic algorithms based optimum structural design, Computers and Structures 1999 78 (2000) 435–448.
[22] http://repository.seasr.org/Datasets/UCI/arff/
[23] 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.