Data Clustering Using Hybrid Genetic Algorithm with k-Means and k-Medoids Algorithms

Md. Touhidul Islam

Id: 2015-2-60-049

Pappu Kumar Basak

Id: 2015-2-60-051

Priom Bhowmik

Id: 2015-2-60-064

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering
East West University

Dhaka-1212, Bangladesh

September, 2019

Declaration

I, hereby, declare that the work presented in this thesis is the outcome of the investigation performed by me under the supervision of name of your supervisor, Professor, Department of Computer Science and engineering, East West University. I also declare that no part of this thesis/project has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

.....

(Musharrat Khan)

Supervisor

Signature

.....

(Md. Touhidul Islam)

2015-2-60-049

Signature

.....

(Pappu Kumar Basak)

2015-2-60-051

Signature

.....

(Priom Bhowmik)

2015-2-60-064

Abstract

Clustering methods separate a set of data points into groups or clusters, where data points of each cluster have the similar properties and are dissimilar from those of other clusters. In general k-means and k-medoids methods are used for data clustering. These clustering methods are heuristic and may stuck in a local optimum. To avoid this problem, we propose a hybrid Genetic Algorithm (HGA) for data clustering. For this purpose, we propose a genetic encoding of the clustering problem, where data points are separated into k clusters. The cluster centers of the generated clusters are determined using the techniques of both k-means and k-medoids methods. The fitness of the clustering is calculated using the sum of Euclidian distances of each data point from its cluster center. We experiment with Iris, Seeds, and Ionosphere datasets. Experimental results show that the proposed HGA generates 2.67% to 28.68% higher clustering accuracies than the clustering accuracies previously reported in the literature.

Acknowledgment

As it is true for everyone, we have also arrived at this point of achieving a goal in our life through various interactions with and help from other people. However, written words are often elusive and harbor diverse interpretations even in one's mother language. Therefore, we would not like to make efforts to find the best words to express my thankfulness other than simply listing those people who have contributed to this thesis itself in an essential way. This work was carried out in the Department of Computer Science and Engineering at East West University, Bangladesh.

First of all, we would like to express my deepest gratitude to the Almighty for His blessings on us. Next, our special thanks go to our supervisor, **Musharrat Khan**, who gave us this opportunity, initiated us into the field of Data Mining, and without whom this work would not have been possible. His encouragements, visionaries and thoughtful comments and suggestions, unforgettable support at every stage of our B.Sc. study were simply appreciating and essential. His ability to muddle us enough to finally answer our own question correctly is something valuable what we have learned and we would try to emulate if ever we get the opportunity.

There are numerous other people too who have shown me their constant support and friendship in various ways, directly or indirectly related to our academic life. We will remember them in our heart and hope to find a more appropriate place to acknowledge them in the future.

Md. Touhidul Islam

September, 2019

Pappu Kumar Basak

September, 2019

Priom Bhowmik

September, 2019

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Data mining is a powerful concept with great potential to predict future trends and behavior. It refers to the extraction of hidden knowledge from large datasets using techniques like statistical analysis, machine learning, clustering, neural networks and genetic algorithms. Hybrid algorithms for data mining are a logical combination of multiple pre-existing techniques to enhance performance and provide better results. The hybrid algorithm that we proposed uses the concept of genetic algorithm and data clustering technique to classify the data samples.

1.1 Clustering

Clustering is a widely used data mining technique, which separates a set of data points into groups or clusters. Data points in a cluster are similar to each other and dissimilar from those in other clusters. The similarities and dissimilarities of data points are assessed based on the attributes of the data points. Clustering can be viewed as an unsupervised classification of data points, where the number of data categories and the category of each data point are unknown [1]. Clustering is also used for outlier detection [1]. Clustering has many applications in the field of biology, security, business intelligence, image pattern recognition, web search, etc. [1]. Distance-based clustering methods like k-means and k-medoids methods are widely used for data clustering [1], [2].

1.2 Types of Clustering

There are many clustering algorithms in the literature. It is difficult to provide a crisp categorization of clustering methods because these categories may overlap so that a method may have features from several categories. [1] Between the major fundamental clustering methods two of them that we used in our work are described here.

1.2.1 Centroid Based Clustering

In centroid-based clustering, clusters are represented by a central vector. This centroid can be the member of data set but, it is not always guaranteed that it will always be the member of the data set. The centroid of a cluster is the center point of a cluster. The centroid can be defined in various ways such as by the mean or medoid of the data points assigned to the cluster. The cluster are defined based on which data points are more similar to the centroid compared to others centroid. It randomly selects k of the objects where each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the Euclidean distance between the object and the cluster mean. For each cluster, it computes the new mean using the objects assigned to the cluster in the previous iteration. All the objects are then reassigned using the updated means as the new cluster centers. The iterations continue until the assignment is stable, that is, the clusters formed in the current round are the same as those formed in the previous round. K-mean is a good example of centroid based clustering.

1.2.2 Representative based clustering

In representative object-based technique clusters are represented by an actual data point instead of taking the mean value of the objects in a cluster. For each cluster there will be one actual representative. The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object and its corresponding representative object. The initial representative objects are chosen arbitrarily. If replacing a representative object by a non-representative object improve the clustering quality then the

non-representative object will be the new representative. All the possible replacements will be tried. The iterative process of replacing representative objects by other objects continues until the quality of the resulting clustering cannot be improved by any replacement. K-medoid is a good example of representative Object-Based Technique.

1.3 Literature Review

In [4], a Genetic Algorithm (GA) is proposed to select cluster centroids for k-means clustering. A chromosome is a string of real numbers, where each real number represents a cluster centroid. After creating clusters based on the centroids in the chromosome, the new centroids are determined from the clusters and the centroids of the chromosome are then replaced by the newly determined centroids. The sum of Euclidian distances from the centroid to each data point is used as the fitness function. As an experimental result, the sum of Euclidian distances is reported. In our proposed HGA, rather than determining the cluster centroids in the GA process, we separate the data points into k clusters and improve the clustering quality in the GA process. In our work, we have reported the clustering accuracy. So, the results of [4] could not be compared with our results. In [5], another GA is proposed for k-means clustering, where data points from the data set are used as the cluster centroids. The chromosome is a binary string, which contains k number of 1s and remaining 0s. The data points corresponding to the 1s are the cluster centroids. Our proposed HGA differs from the GA of [5] in a similar manner discussed during the discussion of [4]. In [5], experiments are done with artificial datasets and, thus, could not be compared with our works. In [2], the authors clustered Iris dataset [6] using classical k-means and k-medoids methods and reported their clustering accuracies. In our work, we compared these clustering accuracies with clustering accuracies produced by our HGA. In [7], the authors used k-means method for clustering along with mutual information-based unsupervised feature transformation. Clustering are done for several datasets from [6] and their clustering accuracies are reported. In our work, we compared clustering accuracy of Seeds dataset reported in this paper with clustering accuracies produced by our HGA. In [8], the authors reduced dimensions of the high dimensional datasets using principal component analysis (PCA) based method and then

used constraint-partitioning k-means method for clustering. Experiments are done with Ionosphere and Parkinson's datasets from [6] and their clustering accuracies are reported. In our work, we compared clustering accuracy of Ionosphere dataset reported in this paper with clustering accuracies produced by our HGA.

1.4 **Proposed Methodology**

Distance-based clustering methods like k-means and k-medoids methods are widely used for data clustering. In these methods, k clusters of n data points are created, where k <= n. Both k-means and k-medoids methods are heuristic methods and may stuck at a local optimal clustering [1]. Clustering is combinations of data points in different clusters and, thus, is a combinatorial optimization problem. In practice it has been found that metaheuristic algorithms like Genetic Algorithms (GAs) are better choice for combinatorial optimization [3]. To overcome the problem of k-means and k-medoids methods of being stuck at a local optimal clustering, GA-based clustering methods may be more prospective. We have found only a few works that combines k-means method with GA for data clustering [4], [5]. So, there is ample scope of exploring GA based clustering method to improve the clustering accuracy. In this work, we propose a hybrid Genetic Algorithm (HGA), where the genetic encoding of the clustering problem separates n data points into k clusters (k <= n). The cluster centers of the generated clusters are determined using the techniques used in k-means and k-medoids methods of data clustering. The fitness of the clustering is then calculated as the sum of Euclidian distances of each data point from its corresponding cluster centers. We have experimented with Iris, Seeds, and Ionosphere datasets from UCI Machine Learning Repository [6]. Experimental result show that our proposed HGA generated 2.67% to 28.68% higher clustering accuracies than the previously reported clustering accuracies in the literature.

Chapter 2

Genetic Algorithm

A genetic algorithm is a heuristic search method used in artificial intelligence and computing. This algorithm follows the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found. Five phases are considered in a genetic algorithm.

- Initial population
- Fitness function
- Selection
- Crossover
- Mutation

2.1 Initial Population

The process begins with a set of individuals which is called a Population. Each individual is a solution to the problem we want to solve. An individual is characterized by a set of parameters known as Genes. Genes are joined into a string to form a Chromosome. In a genetic algorithm, the set of genes of an individual is represented using a string.

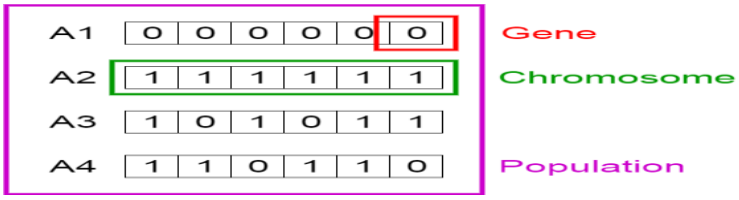


Fig 2.1 Initial Population

2.2 **Fitness Function**

The fitness function determines how fit an individual is. It gives a fitness score to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

2.3 **Selection**

The idea of selection phase is to select the fittest individuals. Two pairs of individuals are selected based on their fitness scores. Individuals with high fitness have more chance to be selected for reproduction.

2.4 **Crossover**

Crossover is the most significant phase in a genetic algorithm. For each pair of parents to be mated, a crossover point is chosen at random from within the genes. For example, consider the crossover point to be 3 as shown in figure 2.2.

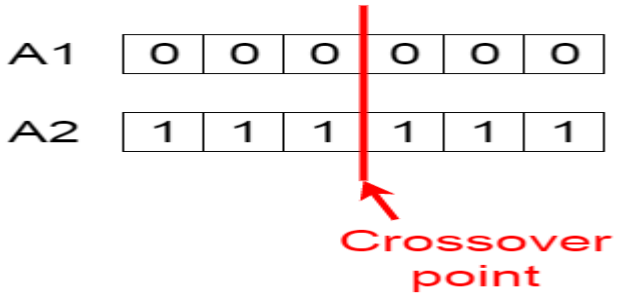


Fig 2.2 Crossover Point

Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached.

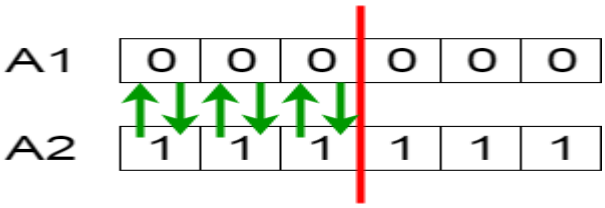


Fig 2.2 Crossover

The new offspring are added to the population.

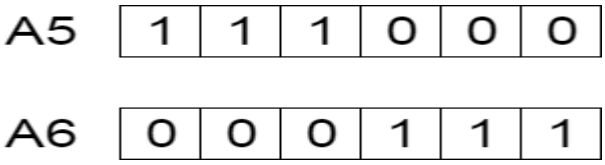


Fig 2.3 Offspring

2.5 **Mutation**

Mutation alters one or more gene values in a chromosome from its initial state. After the new offspring formed, some of the bits in the bit string are flipped shown in figure 2.4.

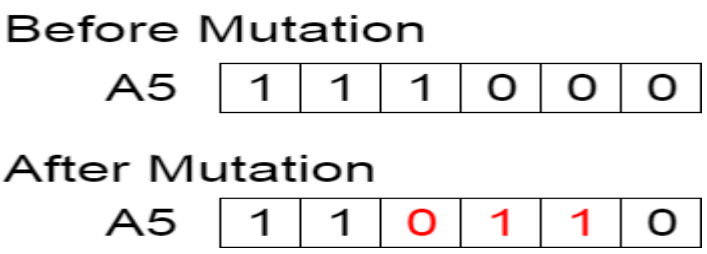


Fig 2.4 Mutation

Chapter 3

K-mean clustering

Clustering is one of the most common exploratory data analysis technique used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. One of the most popular clustering method is k-means clustering.

3.1 K-means clustering algorithm

Clustering is one of the most common exploratory data analysis technique used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. In other words, its try to find homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as Euclidean-based distance or correlation-based distance.

The way k-means algorithm works is as follows:

1) Randomly choose k data points from the dataset as the initial cluster centroids

2) Repeat

3) Assign/reassign each data point to a cluster to which the distance from the data point to the cluster centroid is smaller

4) For each cluster, calculate the mean value of the data points and update the cluster centroid using the mean value

5) Until there is no reassignment of data points is possible

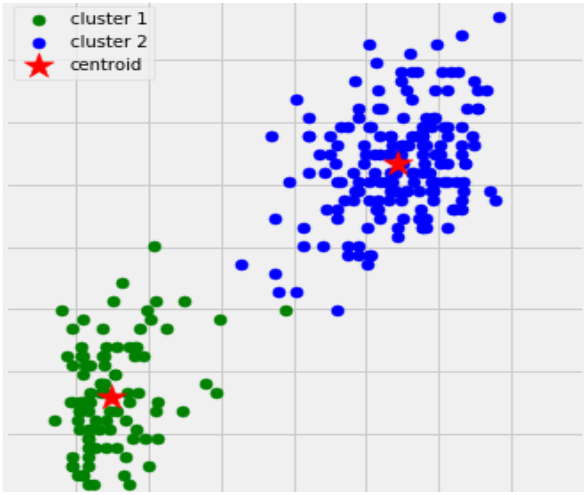


Fig 3.1 K-means clustering

k-Means is relatively an efficient method. However, we need to specify the number of clusters, in advance and the final results are sensitive to initialization and often terminates at a local optimum. Unfortunately, there is no global theoretical method to find the optimal number of clusters. A practical approach is to compare the outcomes of multiple runs with different k and choose the best one based on a predefined criterion. In general, a large k probably decreases the error but increases the risk of overfitting.

The approach k-means follows to solve the problem is called Expectation-Maximization. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster. Below is a breakdown of how we can solve it mathematically.

The objective function is:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|x^i - \mu_k\|^2 \qquad (1)$$

Where, $w_{ik}=1$ for data point x^i if it belongs to cluster k; otherwise, $w_{ik} = 0$. In addition, μ_k is the centroid of x^i's cluster. It is a minimization problem of two parts. We first minimize J w.r.t. w_{ik} and treat μ_k fixed. Then we minimize J w.r.t. μk and treat wik fixed. Technically speaking, we differentiate J w.r.t. wik first and update cluster assignments (E-step). Then, we

differentiate J w.r.t. μ_k and recomputed the centroids after the cluster assignments from previous step (M-step).

Therefore, E-step is:

$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^{m} \sum_{k=1}^{K} \|x^i - \mu_k\|^2$$

$$\Rightarrow w_{ik} = \begin{cases} 1 & \text{if } k = argmin_j \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

In other words, assign the data point x^i to the closest cluster judged by its sum of squared distance from Cluster's centroid.

And M-step is:

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{i=1}^{m} w_{ik}(x^i - \mu_k) = 0$$

$$\Rightarrow \mu_k = \frac{\sum_{i=1}^{m} w_{ik} x^i}{\sum_{i=1}^{m} w_{ik}} \quad (3)$$

Which translates to recomputing the centroid of each cluster to reflect the new assignments.

Few things to note here:

1. Since clustering algorithms including k-means use distance-based measurements to determine the similarity between data points, it is recommended to standardize the data to have a mean of zero and a standard deviation of one since usually the features in any dataset would have different units of measurements.

2. Given k-means iterative nature and the random initialization of centroids at the start of the algorithm, different initializations may lead to different clusters since k-means algorithm may stuck in a local optimum and may not converge to global optimum. Therefore, it's recommended to run the algorithm using different initializations of centroids and pick the results of the run that that yielded the lower sum of squared distance.

Assignment of examples is not changing is the same thing as no change in within-cluster variation:

$$\frac{1}{m_k}\sum_{i=1}^{m_k}\left\|x^i-\mu_{c^k}\right\|^2 \tag{4}$$

Unlike supervised learning, clustering is considered an unsupervised learning method since it is don't have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance. It only want to investigate the structure of the data by grouping the data points into distinct subgroups.

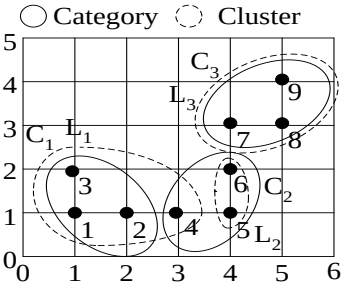
Chapter 4

Proposed Hybrid-GA

A genetic algorithm is a heuristic search method used in artificial intelligence and computing. It is used for finding optimized solutions to search problems based on the theory of natural selection and evolutionary biology. Genetic algorithms are excellent for searching through large and complex data sets. They are considered capable of finding reasonable solutions to complex issues as they are highly capable of solving unconstrained and constrained optimization issues. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

4.1 Genetic Encoding of Clustering Problem

Genetic encoding of a problem is the most difficult task in solving the problem using Genetic Algorithms (GAs). In the GAs of [4], [5], the chromosome encodes the centroids of k-means based clustering. Here, we propose a new genetic encoding for clustering problem, where the chromosome represents the separation of data points into k clusters.



(a) Example data categories and clusters.

Data Points

Clusters	1	2	3	4	5	6	7	8	9
C_1	1	1	1	1	0	0	0	0	0
C_2	0	0	0	0	1	1	0	0	0
C_3	0	0	0	0	0	0	1	1	1

(b) Chromosome representation of three clusters from (a).

Fig 4.1 Chromosome structure

Fig. 4.1(a) shows an example dataset with nine data points categorized into three ground truth data categories, namely category L1 consisting of data points 1, 2, 3; category L2 consisting of data points 4, 5, 6; and category L3 consisting of data points 7, 8, 9. The dataset has two attributes x-coordinate value and y-coordinate value. Consider that a clustering method creates three clusters, namely cluster C1 consisting of data points 1, 2, 3, 4; cluster C2 consisting of data points 5, 6; and cluster C3 consisting of data points 7, 8, 9. Only the data point 4 is incorrectly clustered. Fig. 4.1(b) shows the chromosome structure of the clusters in Fig. 4.1(a). The chromosome is a k × n binary array. The k rows represent the k clusters and the n columns represent the n data points of the dataset. Data points 1, 2, 3, 4 are in the cluster C1. So, in the first row corresponding to the cluster C1, cells corresponding to the data points 1, 2, 3, 4 are set to 1 and other cells are set to 0. Similarly, other two rows are set to 0s and 1s. In this chromosome structure, every column has only one 1s and other 0s, which means that a data point is in exactly one cluster.

4.2 **Fitness Function**

For determining fitness or quality of the clustering represented by a chromosome, we use the clustering quality measures used in k-means and k-medoids methods. For both k-means and k-medoids based HGA, we use Euclidean distance [1] as the distance measure. For k-means based method, for a cluster represented by the chromosome, we determine the centroid of the cluster as the mean of the data points of the cluster. For example, the centroid of the cluster C1 is ((1 + 2 + 1 + 3) = 4; (1 + 1 + 2 + 1) = 4) = (1.75, 1.25). The Euclidean distances between the centroid (1.75, 1.25) and the data points 1, 2, 3, and 4 are 0.79, 0.35, 1.06, and 1.27, respectively. The sum of these Euclidean distances is 0.79 + 0.35 + 1.06 + 1.27 = 3.47. Similarly, the centroid of the cluster C2 is (4, 1:5). The Euclidean distances between the centroid (4, 1:5) and the data points 5 and 6 are 0.5 and 0.5, respectively. The sum of these Euclidean distances is 0.5 + 0.5 = 1.00. The centroid of the cluster C3 is (4.67, 3.33). The Euclidean distances between the centroid (4.67, 3.33) and the data points 7, 8, and 9 are 0.75, 0.47, and 0.75, respectively. The sum of these Euclidean distances is 0.75 + 0.47 + 0.75 = 1.97. Finally, the total sum of these three sums of Euclidean distances is 3.47+1.00+1.97 = 6.44. This sum of Euclidean distances is used as the fitness of the chromosome. The goal of the proposed HGA is to minimize the fitness to improve the clustering quality.

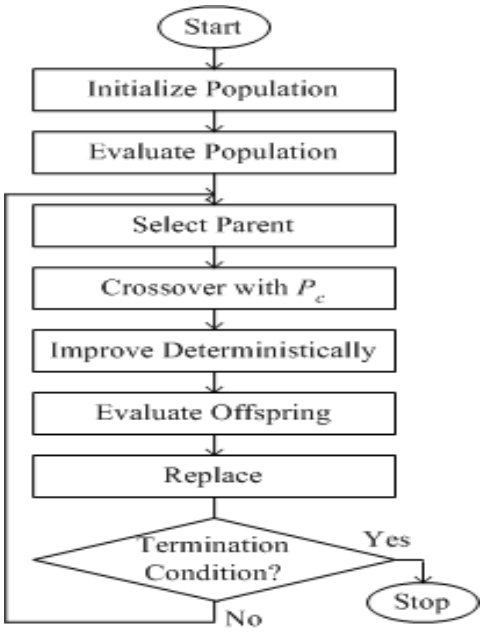


Fig 4.2 The Hybrid Genetic Algorithm.

For k-medoids based method, for a cluster represented by the chromosome, we determine the medoid of the cluster as a data point of the cluster from which the sum of the Euclidean distances from that data point to the remaining data points of the cluster is minimum. For example, in cluster C1, the sum of Euclidean distances from the data point 1 to the other three data points is 4; the sum of Euclidean distances from the data point 2 to the other three data points is 3.41; the sum of Euclidean distances from the data point 3 to the other three data points is 4.65; and the sum of Euclidean distances from the data point 4 to the other three data points is 5.24. Thus, the data point 2 is the medoid of the cluster C1, since data point 2 produces the minimum sum of Euclidean distances from the data point 2 to the remaining three data points of cluster C1. Similarly, in cluster C2, the Euclidean distance between data points 5 and 6 is 1 and either data point 5 or 6 can be considered as the medoid of cluster C2. We arbitrarily take the data point 5 as the medoid of cluster C2. In cluster C3, the sum of Euclidean distances from the data point 7 to the other two data points is 2.41; the sum of Euclidean distances from the data point 8 to the other two data points is 2; and the sum of Euclidean distances from the data point 9 to the other two data points is 2.41. Thus, the data point 8 is the medoid of the cluster C3, since the data point 8 produces the minimum sum of Euclidean distances from data point 8 to the remaining two data points of cluster C3. After determining the three medoid

for the three clusters, the total sum of Euclidean distances is 3:41+1:00+2:00 = 6:41. This sum of Euclidean distances is used as the fitness of the chromosome. The goal of the proposed HGA is to minimize the fitness to improve the clustering quality.

4.3 The Hybrid Genetic Algorithm

We have used a steady-state Genetic Algorithm [9] and made it hybrid using deterministic improvement of the clustering quality. The proposed hybrid Genetic Algorithm (HGA) is shown in Fig. 4.2 and each step is discussed below. The chromosome length is denoted by *ChLen*, which is equal to the number of data points n in the dataset. The chromosome width is the number of clusters k. The chromosome is the k × *ChLen* binary array. The population size is designated by *PopSize*. We have experimented with different population sizes, and finally selected a *PopSize* that generates a better clustering. The chromosomes of the initial population are randomly initialized. The fitness of all chromosomes are calculated using the techniques discussed in sub-section 4.3. Then we determine the chromosome with the minimum fitness, which is the desired solution. We randomly select two parents and perform a one-point crossover [9] with probability PC. For deterministic improvement in the k-means based method, consider the data point 4 in Fig. 4.1. It is in the cluster C1. The Euclidean distance from the data point 4 to the three centroids of clusters C1, C2, and C3 are 1.27, 1.12, and 2.87, respectively. Therefore, it is more appropriate that the data point 4 will belong to the cluster C2 and, thus, the data point 4 is reassigned to the cluster C2. Using this approach, we perform possible reassignment of all data points of the two offspring. For deterministic improvement in the k-medoids based method, consider the data point 4 in Fig. 4.1. It is in the cluster C1. The Euclidean distance from the data point 4 to the three medoids of clusters C1, C2, and C3, that is the data points 2, 5, and 8 are 1, 1, and 2.83, respectively. Therefore, the data point 4 may belong to either cluster C1 or cluster C2. The data point 4 can be arbitrarily reassigned to the cluster C2. Using this approach, we perform possible reassignment of all data points of the two offspring. We determine the fitness values of *offspring1* and *offspring2* and represent them by *fit1* and *fit2*, respectively. From the population, we determine the maximum fitness *MaxFit* and the corresponding chromosome *ChromMax*. If *fit1* < *MaxFit*, then the *ChromMax* chromosome is replaced by *offspring1*. After possible replacement, if any, we again determine the maximum fitness *MaxFit* and the corresponding chromosome *ChromMax*. If fit2 < MaxFit, then the *ChromMax* chromosome is replaced by *offspring2*. After possible replacement, if any,

we determine the minimum fitness and the corresponding chromosome as the current solution.

If minimum fitness value does not improve for consecutive stagnation period of $2 \times PopSize$ generations, then we terminate the HGA.

In general k-means and k-medoids methods are used for data clustering. These clustering methods are heuristic and may stuck in a local optima. To avoid this problem, here used hybrid Genetic Algorithm (HGA) for data clustering. For this purpose, here used a genetic encoding of the clustering problem, where data points are separated into k clusters. The cluster centers of the generated clusters are determined using the techniques of both k-means and k-medoids methods.

Chapter 5

K-medoid clustering

Clustering is one of the most common exploratory data analysis technique used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. One of the most popular clustering method is k-medoids clustering.

5.1 K-medoids clustering algorithm

Partitioning Around Medoids or the K-medoids algorithm is a partitional clustering algorithm which is slightly modified from the K-means algorithm. They both attempt to minimize the squared-error but the k-medoids algorithm is more robust to noise than K-means algorithm. In K-means algorithm, they choose means as the centroids but in the K-medoids, data points are chosen to be the medoids. A medoid can be defined as that object of a cluster, whose average dissimilarity to all the objects in the cluster is minimal.

The difference between k-means and k-medoids is analogous to the difference between mean and median; where mean indicates the average value of all data items collected, while median indicates the value around that which all data items are evenly distributed around it. The basic idea of this algorithm is to first compute the K representative objects which are called as medoids. After finding the set of medoids, each object of the data set is assigned to the nearest medoid. That is, object *i* is put into cluster v_i, when medoid mv_i is nearer than any other medoid m_w

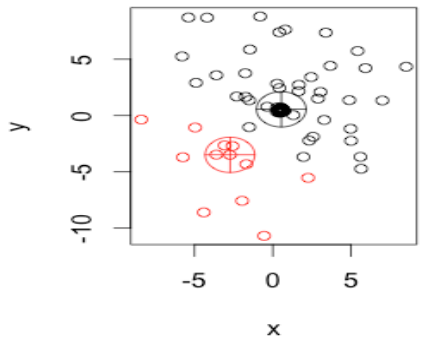


Fig 5.1 k-medoids clustering

There are many possible k-medoids clustering methods [1]. The most simple k-medoids clustering method works as follows:

1) Randomly choose k data points from the dataset as the initial cluster medoids

2) Repeat

3) Assign/reassign data points other than the selected cluster medoids to a cluster to which the distance from the data point to the cluster medoid is smaller

4) For each cluster, find out the data point for which the sum of the distances of other data points of the cluster is minimum and update the cluster medoid using that data point

5) Until there is no reassignment of data points is possible

Unlike supervised learning, clustering is considered an unsupervised learning method since it is don't have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance. It only want to investigate the structure of the data by grouping the data points into distinct subgroups.

Chapter 6

Experimental Results

The proposed algorithms are implemented using C++ language and run on a personal computer with Intel Core i3- 4030U 1.90GHz processor, 8GB RAM, and Windows 10 Pro (64-bit) operating system. We have experimented with three datasets, namely Iris, Seeds, and Ionosphere datasets, from UCI Machine Learning Repository. In this section performance of all the considered approaches have been evaluated and analyzed using various performance measurements including accuracy and accuracy improvement by compare with previous accuracy.

We have calculated the clustering accuracy (ACC) using the formula in (1).

$$ACC = \frac{No.\ of\ data\ points\ cluster\ correctly}{Number\ of\ data\ points\ in\ the\ dataset} \times 100\% \ldots\ldots\ldots (1)$$

We have calculated the clustering accuracy improvement (IMPV) by our HGA using the formula in (2).

$$IMPV = \frac{Our\ accuracy - Previous\ accuracy}{Previous\ accuracy} \times 100\% \quad \ldots\ldots\ldots (2)$$

6.1 For Iris dataset

The Iris dataset has 4 real attributes. It has 150 data points with three data categories, namely Setosa, Versicolor and Virginica. Each category has 50 data points. For our experiment, we take the number of clusters k equal to the number of data categories mentioned in the dataset. Hence, for Iris dataset, k = 3 is used. The confusion matrix generated by our HGA with k-means based method for Population size = 900 and PC = 0.75 is given in Table 6.1. The clustering accuracy is (50 + 49 + 41) /150 × 100% = 93.33%.

TABLE 6.1: CONFUSION MATRIX GENERATED BY OUR HGA WITH k-MEANS BASED METHOD FOR IRIS DATASET

Data Category	**Cluster ID**		
	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	49	1
Virginica	0	9	41

TABLE 6.2: CONFUSION MATRIX GENERATED BY K. G. SONI and A. PATEL WITH k-MEANS METHOD FOR IRIS DATASET [2]

Data Category	**Cluster ID**		
	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	47	3
Virginica	0	14	36

The confusion matrix generated by our HGA with k-medoids based method for population size = 750 and PC = 0.80 is given in Table 6.3. The clustering accuracy is (50+44+50) /150 ×100% = 96.00%. Clustering accuracy comparison with previous work in [2] is shown in Table 6.3.

TABLE 6.3: CONFUSION MATRIX GENERATED BY OUR HGA WITH k-MEDOIDS BASED METHOD FOR IRIS DATASET

Data Category	**Cluster ID**		
	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	44	6
Virginica	0	0	50

TABLE 6.4: CONFUSION MATRIX GENERATED BY K. G. SONI AND A. PATEL WITH k-MEDOIDS BASED METHOD FOR IRIS DATASET [2]

Data Category	**Cluster ID**		
	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	41	9
Virginica	0	3	47

From Table 6.5 we see that for clustering Iris dataset our HGA with k-means and k-medoids based methods improves clustering accuracies by 5.22% and 4.35% than the clustering accuracies reported in [2] for classical k- means and k-medoids, respectively.

TABLE 6.5: CLUSTERING ACCURACY COMPARISON OF OUR HGA FOR IRIS DATASET WITH PREVIOUS WORK

ACC in [2]		Result of our HGA			
K-means	K-medoids	K-means based		K-medoids based	
		ACC	IMPV	ACC	IMPV
88.70%	92.00%	93.33%	5.22%	96.00%	4.35%

6.2 For Seeds dataset

The Seeds dataset has 7 real attributes and 210 data points. It has three data categories, namely Kama, Rosa, and Canadian. Each category has 70 data points. For our experiment, we take the number of clusters k equal to the number of data categories mentioned in the dataset. Hence, for Seeds dataset, k = 3 is used. The confusion matrix generated by our HGA with k-means based method for population size = 1000 and PC = 0.85 is given in Table 6.6. The clustering accuracy is (58 + 68 + 69) / 210 × 100% = 92.86%.

TABLE 6.6: CONFUSION MATRIX GENERATED BY OUR HGA WITH k-MEANS BASED METHOD FOR SEEDS DATASET

Data Category	**Cluster ID**		
	Kama	**Rosa**	**Canadian**
Kama	58	12	16
Rosa	2	68	1
Canadian	0	1	69

The confusion matrix generated by our HGA with k-medoids based method for population size = 1500 and PC = 0.90 is given in Table 6.7. The clustering accuracy is (53+69+70)/210 × 100% = 91.43%.

TABLE 6.7: CONFUSION MATRIX GENERATED BY OUR HGA WITH k-MEDOIDS BASED METHOD FOR SEEDS DATASET

Data Category	**Cluster ID**		
	Kama	**Rosa**	**Canadian**
Kama	53	1	16
Rosa	0	69	1
Canadian	0	0	70

Clustering accuracy comparison with previous work in [7] is shown in Table 6.8. From Table 6.8 we see that for clustering Seeds dataset our HGA with k-means based and k-medoids based methods generate 4.23% and 2.67% higher accuracies, respectively, than the accuracy produced in [7] using variants of k-means method.

TABLE 6.8: CLUSTERING ACCURACY COMPARISON OF OUR HGA FOR SEEDS DATASET WITH PREVIOUS WORK

ACC in [7] (variant of k-means)	Result of our HGA			
	K-means based		K-medoids based	
	ACC	IMPV	ACC	IMPV
89.05%	92.86%	4.23%	91.43%	2.67%

6.3 For Ionosphere dataset

The Ionosphere dataset has 34 integer and real attributes and 351 data points. It has two data categories, namely Good and Bad with 225 and 126 data points, respectively. For our experiment, we take the number of clusters k equal to the number of data categories mentioned in the dataset. Hence, for Ionosphere dataset, k = 2 is used. The confusion matrix generated by our HGA with k-means based method for Population Size = 1200 and PC = 0.90 is given in Table 6.9. The clustering accuracy is (204+119) / 351 × 100% = 92.02%.

TABLE 6.9: CONFUSION MATRIX GENERATED BY OUR HGA WITH k-MEANS BASED METHOD FOR IONOSPHERE DATASET

Data Category	**Cluster ID**	
	Good	**Bad**
Good	204	21
Bad	7	119

The confusion matrix generated by our HGA with k-medoids based method for population Size = 1500 and PC = 0.90 is given in Table 6.10. The clustering accuracy is (199 + 122) / 351×100% = 91.45%. Clustering accuracy comparison with previous work in [8] is shown in Table 6.10.

TABLE 6.10: CONFUSION MATRIX GENERATED BY OUR HGA WITH k-MEDOIDS BASED METHOD FOR IONOSPHERE DATASET

Data Category	**Cluster ID**	
	Good	**Bad**
Good	199	26
Bad	4	122

From Table 6.11 we see that for clustering Ionosphere dataset our HGA with k-means based and k-medoids based methods generate 28.68% and 27.88% higher accuracies, respectively, than the accuracy produced in [8] using variant of k-means.

TABLE 6.11: CLUSTERING ACCURACY COMPARISON OF OUR HGA FOR IONOSPHERE DATASET WITH PREVIOUS WORK

ACC in [8] (variant of k-means)	Result of our HGA			
	K-means based		K-medoids based	
	ACC	IMPV	ACC	IMPV
71.51%	92.02%	28.68%	91.45%	27.88%

We have experimented with Iris, Seeds, and Ionosphere datasets from UCI Machine Learning Repository [6]. Experimental result show that our proposed HGA generated 2.67% to 28.68% higher clustering accuracies than the previously reported clustering accuracies in the literature.

Chapter 7

Conclusion and Future Works

One of the useful and important data mining techniques is data clustering. Usually k-means and k-medoids methods are used for data clustering [2]. Data clustering can be considered as a combinatorial optimization problem and metaheuristic algorithm like Genetic Algorithm (GA) may be a good choice for data clustering. We have found only a few works reported in the literature that used GA with k-means algorithm for data clustering [4], [5].

7.1 Conclusion

In the present work, we propose a hybrid Genetic Algorithm (HGA) for data clustering with both k-means based and k-medoids based methods for calculating clustering quality. For this purpose, we use a new genetic encoding of the clustering problem. The chromosome encodes separation of all n data points into k clusters, where k _ n. In the proposed HGA process the clustering accuracies are improved than the clustering accuracies produced by classical k-means method and its variants. We experiment with three datasets from UCI Machine Learning Repository [6], namely Iris, Seeds, and Ionosphere datasets. Experimental results of these datasets show that our proposed HGA with both k-means based and k-medoids based methods generate 2.67% to 28.68% higher clustering accuracies than the previously reported clustering accuracies reported in the literature**.**

7.2 Future Works

As our future work we are planning to experiment with more datasets to compare the performance of our proposed HGA for data clustering. We are also planning to use other distance measure like Manhattan distance [1] to investigate the performance of the proposed HGA for data clustering.

Bibliography

[1] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed. Morgan Kaufmann Publishers, 2012.

[2] K. G. Soni and A. Patel, "Comparative analysis of k-means and kmedoids algorithm on iris data," International Journal of Computational Intelligence Research, vol. 13, no. 5, pp. 899 – 906, 2017.

[3] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed. Springer, 1996.

[4] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," Pattern Recognition, vol. 33, pp. 1455 – 1465, 2000.

[5] H.-J. Lin, F.-W. Yang, and Y.-T. Kao, "An efficient GA-based clustering technique," Tamkang Journal of Science and Engineering, vol. 8, no. 2, pp. 113 – 122, 2005.

[6] UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/index.php, (Last accessed on 10 June 2019).

[7] M. Wei, T. W. S. Chow, and R. H. M. Chan, "Clustering heterogeneous data with k-means by mutual information-based unsupervised feature transformation," Entropy, vol. 17, pp. 1535–1548, 2015.

[8] A. George, "Efficient high dimension data clustering using constraintpartitioning k-means algorithm," International Arab Journal of Information Technology, vol. 10, no. 5, pp. 467 – 476, 2013.

[9] P. Mazumder and E. M. Rudnick, Genetic Algorithms for VLSI Design, Layout & Test Automation. Pearson Education Asia, 2002.

Appendix A

List of Acronyms

HGA	Hybrid Genetic Algorithm
GA	Genetic Algorithm
PCA	Principal component analysis
P_C	Probability count
ACC	Accuracy
IMPV	Improvement

Appendix B

List of Notations

∑	Summation
μ	Mean
×	Multiplication

Appendix C

Code

```
using namespace std;
#include <bits/stdc++.h>
#define cluster_size 210
#define cromosom_size 3
#define total_cromosom 3000
#define iteration 3000
#define total_cluster (cromosom_size*(total_cromosom+2))
#define probability 1
#define r1 140
#define r2 190
#define dimension 7
int cntt=0;
int count_cluster[total_cluster][cluster_size];
double fitness[total_cromosom];
void initialize_val_cromosom(){
 for(int i=0;i<total_cromosom;i++){
      int val=0;
```

```
    while(val<cluster_size){
            int select=rand()%cromosom_size;
            count_cluster[i*cromosom_size+select][val]=1;
            val++;
          }
    }
}
double fitness_fun(int C){
   double WW,XX,YY,ZZ,QQ,MM,NN,fit=0.0;
   int CC=0;
   for(int i=0;i<cromosom_size;i++){
         WW=0.0,XX=0.0,YY=0.0,ZZ=0.0,QQ=0.0,MM=0.0,NN=0.0,CC=0;
      for(int j=0;j<cluster_size;j++){
          if(count_cluster[C*cromosom_size+i][j]==1){
             CC++;
             WW+=w[j];
             XX+=x[j];
             YY+=y[j];
             ZZ+=z[j];
             QQ+=qq[j];
             MM+=mm[j];
             NN+=nn[j];
          }
      }
```

```
        WW=WW/CC;
        XX=XX/CC;
        YY=YY/CC;
        ZZ=ZZ/CC;
        QQ=QQ/CC;
        MM=MM/CC;
        NN=NN/CC;
        for(int j=0;j<cluster_size;j++){
          if(count_cluster[C*cromosom_size+i][j]==1){
              fit+=sqrt(abs(WW-w[j])*abs(WW-w[j])+abs(XX-x[j])*abs(XX-x[j])+abs(YY-
y[j])*abs(YY-y[j])+abs(ZZ-z[j])*abs(ZZ-z[j])+abs(QQ-qq[j])*abs(QQ-qq[j])+abs(MM-
mm[j])*abs(MM-mm[j])+abs(NN-nn[j])*abs(NN-nn[j]));
           }
        }
    }
     return fit;
}
void crossover_fitness(int rep,int b){
    for(int i=0;i<cromosom_size;i++){
        for(int j=0;j<cluster_size;j++)
           count_cluster[rep*cromosom_size+i][j]=count_cluster[b*cromosom_size+i][j];
     }
 }
 void hybrid_GA(int C){
    double WW,XX,YY,ZZ,QQ,MM,NN,fit=0.0;
    int CC=0;
```

```
double dim[cromosom_size][dimension];
for(int i=0;i<cromosom_size;i++){
      WW=0.0,XX=0.0,YY=0.0,ZZ=0.0,QQ=0.0,MM=0.0,NN=0.0,CC=0;
   for(int j=0;j<cluster_size;j++){
       if(count_cluster[C*cromosom_size+i][j]==1){
           CC++;
           WW+=w[j];
           XX+=x[j];
           YY+=y[j];
           ZZ+=z[j];
           QQ+=qq[j];
           MM+=mm[j];
           NN+=nn[j];
       }
   }
   dim[i][0]=WW/CC;
   dim[i][1]=XX/CC;
   dim[i][2]=YY/CC;
   dim[i][3]=ZZ/CC;
   dim[i][4]=QQ/CC;
   dim[i][5]=MM/CC;
   dim[i][6]=NN/CC;
}
  for(int i=0;i<cromosom_size;i++){
    for(int j=0;j<cluster_size;j++){
```

```
            if(count_cluster[C*cromosom_size+i][j]==1){
                double ddd[0][3]; int pppp;
                ddd[0][0]=sqrt(abs(dim[0][0]-w[j])*abs(dim[0][0]-w[j])+abs(dim[0][1]-
x[j])*abs(dim[0][1]-x[j])+abs(dim[0][2]-y[j])*abs(dim[0][2]-y[j])+abs(dim[0][3]-z[j])*abs(dim[0][3]-
z[j])+abs(dim[0][4]-qq[j])*abs(dim[0][4]-qq[j])+abs(dim[0][5]-mm[j])*abs(dim[0][5]-
mm[j])+abs(dim[0][6]-nn[j])*abs(dim[0][6]-nn[j]));
                ddd[0][1]=sqrt(abs(dim[1][0]-w[j])*abs(dim[1][0]-w[j])+abs(dim[1][1]-
x[j])*abs(dim[1][1]-x[j])+abs(dim[1][2]-y[j])*abs(dim[1][2]-y[j])+abs(dim[1][3]-z[j])*abs(dim[1][3]-
z[j])+abs(dim[1][4]-qq[j])*abs(dim[1][4]-qq[j])+abs(dim[1][5]-mm[j])*abs(dim[1][5]-
mm[j])+abs(dim[1][6]-nn[j])*abs(dim[1][6]-nn[j]));
                ddd[0][2]=sqrt(abs(dim[2][0]-w[j])*abs(dim[2][0]-w[j])+abs(dim[2][1]-
x[j])*abs(dim[2][1]-x[j])+abs(dim[2][2]-y[j])*abs(dim[2][2]-y[j])+abs(dim[2][3]-z[j])*abs(dim[2][3]-
z[j])+abs(dim[2][4]-qq[j])*abs(dim[2][4]-qq[j])+abs(dim[2][5]-mm[j])*abs(dim[2][5]-
mm[j])+abs(dim[2][6]-nn[j])*abs(dim[2][6]-nn[j]));
                if(ddd[0][0]<=ddd[0][1]) pppp=0;
                else pppp=1;
                if(ddd[0][pppp]>=ddd[0][2])
                   pppp=2;
                count_cluster[C*cromosom_size+i][j]=0;
                count_cluster[C*cromosom_size+pppp][j]=1;
            }
        }
    }
}
void print(){
        int index;
        double cmp=1000000000000000.0;
        int j;
```

```
 for( j=0;j<total_cromosom;j++){
   if(fitness[j]<cmp){
       cmp=fitness[j];
       index=j;
   }
 }
  cout<<endl<<endl;
        cout<<"\t# Normal medoid based centroid (Seeds dataset) #\n"<<endl;
cout<<"\tFitness :"<<1425.57<<endl;
 cout<<"\tprobability :"<<0.9<<endl;
 cout<<"\tpopulation :"<<1500<<endl<<endl;
 cout<<setw(15)<<"Kama"<<setw(15)<<"Rosa"<<setw(15)<<"Canadian"<<endl;
 cout<<setw(15)<<1<<setw(15)<<69<<setw(15)<<0<<endl;
 cout<<setw(15)<<16<<setw(15)<<1<<setw(15)<<70<<endl;
 cout<<setw(15)<<53<<setw(15)<<0<<setw(15)<<0<<endl;
 int cn1=0,cn2=0,cn3=0,sum=0;
 for( j=index*cromosom_size;j<(index*cromosom_size+cromosom_size);j++){
       cn1=0,cn2=0,cn3=0;
        for(int t=0;t<cluster_size;t++){
           if(t>=0&&t<70){
              if(count_cluster[j][t]==1) cn1++;
           }
           else if(t>=70&&t<140){
               if(count_cluster[j][t]==1) cn2++;
           }
```

```
                else{
                    if(count_cluster[j][t]==1) cn3++;
                }
            }
        }
}
int main(){
    int cnt=0;
    initialize_val_cromosom();
    for(int i=0;i<total_cromosom;i++) fitness[i]=fitness_fun(i);
   for(;;){
        if((double)rand()/(double)RAND_MAX<=probability){
            bool s1,s2;
            double fit1,fit2;
            cnt++;
           while(1){
            int c1,c2;
             fit1=0,fit2=0;
             s1=0,s2=0;
```

```
int mark=0;
        for(;;){
            int a1=rand()%total_cromosom;
            int b1=rand()%total_cromosom;
            if((b1!=a1)&&(fitness[b1]!=fitness[a1])){
               c1=(fitness[a1]<fitness[b1])? a1:b1;
               break;
            }
         }

         for(;;){
             int b1=rand()%total_cromosom;
             int a1=rand()%total_cromosom;
             if((b1!=a1)&&(a1!=c1)&&(b1!=c1)&&(fitness[b1]!=fitness[a1])){
                  c2=(fitness[a1]<fitness[b1])? a1:b1;
                  break;
             }
         }

        int split;
        for(;;){

            split=rand()%cluster_size;
            if(split>=r1&&split<=r2)
```

```
                    break;
                }
              for(int i=0;i<cromosom_size;i++){
                for(int j=0;j<cluster_size;j++){
                  if(j<=split){

count_cluster[total_cromosom*cromosom_size+i][j]=count_cluster[c1*cromosom_size+i][j]z

count_cluster[(total_cromosom+1)*cromosom_size+i][j]=count_cluster[c2*cromosom_size+i][j];
                  }
                  else{

count_cluster[total_cromosom*cromosom_size+i][j]=count_cluster[c2*cromosom_size+i][j];

count_cluster[(total_cromosom+1)*cromosom_size+i][j]=count_cluster[c1*cromosom_size+i][j];
                  }
                }
              }

                fit1=fitness_fun(total_cromosom),fit2=fitness_fun(total_cromosom+1);
                for(int i=0;i<total_cromosom;i++){
                  if(fit1==fitness[i]) s1=1;
                  if(fit2==fitness[i]) s2=1;
                }

              if(s1==0||s2==0) break;
```

```
}
    int index;
    int flag1=0;
    int flag2=0;
    double large=-1.0;
     if(s1==0){
    hybrid_GA(total_cromosom);
    fit1=fitness_fun(total_cromosom);

    for(int i=0;i<total_cromosom;i++){
      if(fitness[i]>fit1&&fitness[i]>large){

            index=i;
            flag1=1;
            large=fitness[i];
       }

    }
    if(flag1==1){
        cntt++;

      crossover_fitness(index,total_cromosom);
      fitness[index]=fit1;
    }
```

```
        }
        if(s2==0){
          hybrid_GA(total_cromosom+1);
          fit2=fitness_fun(total_cromosom+1);
           large=-1.0;

          for(int i=0;i<total_cromosom;i++){
            if(fitness[i]>fit2&&fitness[i]>large){
                  index=i;
                  flag2=1;
                  large=fitness[i];

             }
          }
          if(flag2==1){
               cntt++;

            crossover_fitness(index,total_cromosom+1);
            fitness[index]=fit2;
          }
        }

      if (cnt>=iteration) break;
     }
}
```

```
        print();
  return 0;
}
```