A Genetic Algorithm Based Clustering Approach with Tabu Operation and *K*-Means Operation

Yongguo Liu,[1,2,3] Hua Yan[1] and Kefei Chen[3]

[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731; [2] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100191; and [3]Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, P. R. China

ABSTRACT

Clustering analysis is a nonconvex problem that possesses many locally optimal values, with the result that its solution often falls into these traps. In this article, a hybrid genetic clustering algorithm called Genetic algorithm with Tabu operation and *K*-means operation based Clustering (GTK-Clustering) is developed to deal with the clustering problem. With the cooperation of tabu operation and *k*-means operation, the GTK-Clustering algorithm can keep a balance between population diversity and the speed of convergence. On the one hand, the tabu operation prevents the population from being dominated by several fitter individuals and maintains a high level of population diversity, and on the other hand, the *k*-means operation improves the distribution of objects and enhances the speed of convergence of the clustering algorithm. Its superiority over the *k*-means algorithm and another genetic clustering approach is demonstrated for artificial and real life data sets.

KEYWORDS: clustering, genetic algorithm, tabu search, *k*-means algorithm

1. INTRODUCTION

The clustering problem is a fundamental problem that frequently arises in a great variety of fields such as pattern recognition, machine learning, and statistics.

Correspondence: Dr. Yongguo Liu. School of Computer Science and Engineering, University of Electronic Science and Technology of China. No. 2006, Xiyuan Ave., West Hi-tech Zone, Chengdu 611731, P. R. China; E-mail: liuyg@uestc.edu.cn

Clustering analysis is a formal study of algorithms and methods for grouping or classifying objects without category labels. In clustering analysis, objects are generally denoted by points in m-dimensional Euclidean space. Our aim is to divide these objects into different clusters such that a specified similarity measure is optimized. In general, the clustering problem is a nonconvex problem that possesses many locally optimal values, resulting that its solution often falls into these traps. Many clustering approaches have been reported which can be classified into two categories: hierarchical and partitional (Omran et al., 2007; Pedrycz, 2005). Among them, k-means algorithm, a typical iterative hill-climbing method, is very important (MacQueen, 1967). However, the major drawbacks of the k-means algorithm are that it often gets stuck at local minima and its result is largely dependent on the choice of the initial cluster centers (Selim & Ismail, 1984). In order to overcome the shortcomings of the k-means algorithm, researchers developed some improved clustering techniques (Bagirov, 2008; Lu et al., 2008; Redmond & Heneghan, 2007). For example, Bifulco et al. (2008) designed a global optimization method based on controlled random search to improve the k-means algorithm. Recently, the metaheuristic algorithms such as simulated annealing, particle swarm optimization, and tabu search are applied to the clustering problem (Bandyopadhyay et al., 2001; Jarboui et al., 2007; Sung & Jin, 2000). In order to efficiently use the metaheuristic algorithms in clustering analysis, researchers combined the metaheuristic algorithms with local descent approaches (Güngör & Ünler, 2007, 2008; Yang et al., 2009). For example, Kao et al. (2008) proposed a hybrid clustering technique called K-NM-PSO which combines the k-means algorithm, Nelder-Mead simplex search, and particle swarm optimization to solve the clustering problem. Since the clustering problem may be viewed as searching for a number of clusters in the feature space such that a given measure metric is optimized, the application of genetic algorithms (GA) to the clustering problem seems natural and appropriate (Hall et al., 1999; Laszlo & Mukherjee, 2006; Murthy & Chowdhury, 1996). In the genetic clustering method, the designer encodes the clustering partition or the cluster centers as a chromosome. Then genetic operations such as selection, mutation, and crossover are applied to explore the search space. When the specified terminal condition is satisfied, the best known individual is defined as the final result. Hall et al. (1999) used the binary gray encoding to represent the cluster centers and adopted 2-fold tournament selection, two-point crossover, and standard mutation to deal with the clustering problem. Laszlo and Mukherjee (2006) presented a genetic algorithm for

evolving the cluster centers in the *k*-means algorithm. The set of the cluster centers is represented using a hyper-quadtree constructed on the data. Murthy and Chowdhury (1996) proposed a genetic clustering method which adopts the string-of-group-numbers encoding to solve the clustering problem.

There are two important issues in GA: selection pressure and population diversity (Whitley, 1989). Selection pressure guides GA to exploit information behind the fitter individuals and to create better offspring iteratively, while population diversity helps GA to explore the unvisited search space and to find better individuals. On one hand, strong selection pressure accelerates the speed of convergence but potentially leads to the premature phenomenon because of the loss of population diversity. On the other hand, high population diversity possibly results in better outputs but often slows down the speed of convergence due to the lack of selection pressure. So, a good GA should be able to maintain a balance between these two aspects (Ting et al., 2003). In some improved genetic algorithms, tabu search (TS) (Glover & Laguna, 1997), another metaheuristic technique which applies memory to record and guide the search trajectory, has been adopted to enhance the performance of GA by combining GA and TS. An approach of concatenating GA and TS is applied to the placement problem of analog LSI chip designs, which switches the search to TS when GA stops improving the solution (Handa and Kuga, 1995). To solve the hub location problem, another combination is reported to regard TS as a local operator of GA, which selects the best individual from the population and exploits its neighborhood for better individuals (Abdinnour-Helm, 1998). To keep a high level of population diversity, Ting et al. (2003) integrated TS into GA to prevent two individuals of the same ancestor from mating each other so as to deal with the traveling salesman problem. In fact, the key problem of premature convergence lies in the loss of population diversity. Under strong selection pressure, several fitter individuals dominate the population and copy themselves into the child population. As a result, population diversity greatly decreases. In this paper, our work is to prohibit these individuals from cloning themselves and dominating the population. But the fitter individuals should be encouraged to guide the evolutionary process.

In this article, a hybrid genetic clustering algorithm called GTK-Clustering is proposed to find the clustering result. By integrating the tabu operation and the *k*-means operation, the presented method can deal with above-mentioned problems, i.e., enhancing the speed of convergence and maintaining population diversity. Here,

the tabu operation is used to prevent the population from being dominated by several fitter individuals and to maintain a high level of population diversity. On the other hand, the *k*-means operation is used to fine-tune the distribution of objects and to enhance the speed of convergence of the clustering algorithm. By computer simulations, it is shown that the GTK-Clustering method is superior to the *k*-means algorithm and another genetic clustering approach.

The remaining part of this paper is organized as follows. In Section 2, the statement of the clustering problem under consideration is given. In Section 3, the GTK-Clustering algorithm and its components are extensively described. In Section 4, the contribution of each component is shown in detail. Performance comparison between the GTK-Clustering approach and two other algorithms is conducted on three artificial and three real life data sets. Finally, some conclusions are drawn in Section 5.

2. STATEMENT OF THE CLUSTERING PROBLEM

In this paper, the similarity measure is defined as the distance between objects within a cluster and their cluster center, and our aim is to minimize this metric. We consider the clustering problem stated as follows: Given N objects in R^m, allocate each object to one of K clusters such that the sum of squared Euclidean distances between each object and the center of its belonging cluster for every such allocated object is minimized. This problem can be mathematically described as

$$\min_{W,C} F(W,C) = \sum_{i=1}^{N}\sum_{j=1}^{K} w_{ij} \| \mathbf{x}_i - \mathbf{c}_j \|^2 , \tag{1}$$

where $\sum_{j=1}^{K} w_{ij} = 1$, $i = 1,\ldots,N$. If object $\mathbf{x}_i$ is allocated to cluster C_j, then w_{ij} is equal to 1, otherwise w_{ij} is equal to 0. In Eq. (1), N denotes the number of objects, K denotes the number of clusters, $C = \{C_1,\ldots,C_K\}$ denotes the set of K clusters, and $W = [w_{ij}]_{N\times K}$ denotes the 0–1 partition matrix. Cluster center $\mathbf{c}_j$ is calculated as

$$\mathbf{c}_j = \frac{1}{n_j}\sum_{i=1}^{N} w_{ij}\mathbf{x}_i , \tag{2}$$

where n_j denotes the number of objects belonging to cluster C_j.

3. THE GTK-CLUSTERING ALGORITHM

Based on the evolutionary structure of GA, the GTK-Clustering algorithm gathers the global optimization property of GA, the memory structure of TS, and the local search capability of the *k*-means algorithm together. It maintains strong selection pressure and high population diversity. Figure 1 gives the description of the GTK-Clustering algorithm.

```
Begin
   initialize parameters
   establish the initial population for clustering
   while (not termination-condition) do
      k-means operation
      selection operation
      tabu operation
      crossover operation
      mutation operation
 end
end
```

Fig. 1: General description of the GTK-Clustering algorithm

From Figure 1, it is seen that if the *k*-means operation and the tabu operation are removed, then the GTK-Clustering algorithm degenerates into a standard genetic algorithm for clustering. Here, the *k*-means operation modulates the distribution of objects among different clusters, improves the similarity between objects and their cluster centroids, and increases the speed of convergence of the clustering algorithm. In addition, evolutionary operations (selection, crossover, and mutation) are performed to find the optimal solution. In the GTK-Clustering algorithm, the tabu operation follows the selection operation.

Based on the principle of survival of the fittest, organisms with high fitness, in nature, are in general more successful at survival and reproduction. GA imitates this rule and selects some individuals to reproduce so that individuals with high fitness are more likely to survive to next population. In this way, super-fit individuals may be chosen more than once, which causes the loss of population diversity. Premature convergence of GA can be blamed on (1) loss of critical alleles due to selection, (2) schemata disruption due to crossover, and (3) parameter settings such as mutation

probability, crossover probability, and population size (Potts et al., 1994). So, we should not only observe the evolutionary principle but also avoid repeatedly choosing several fitter individuals. However, if we only prevent the fitter ones from being repeatedly copied, population diversity is kept at the cost of sacrificing the speed of convergence.

In this article, we utilize the local search capability of the *k*-means operation to improve the relationship between objects and their cluster centers so as to enhance the speed of convergence. In order to avoid the greedy characteristics of the *k*-means operation, we employ the global search property of GA to explore the solution space. In order to harmonize the tradeoff between population diversity and the speed of convergence, we integrate the tabu operation into the evolution process to supervise the reproduction procedure. The tabu operation with memory structure judges whether an individual can survive to next operation. In this operation, we adopt the probability threshold to create a new individual which is similar to but different from the current tabu one. As a result, the repeated selection of the fitter individuals disappears and population diversity is maintained. In the following, each component of the GTK-Clustering algorithm is discussed in detail.

3.1 Individual Representation

In this article, the chromosome length is set to be equal to the number of objects, which is suitable for the crossover operation and the comparison between two genetic clustering algorithms. The value of the i^{th} gene of the chromosome denotes the cluster number assigned to the i^{th} object, where $i = 1,\ldots,N$. For instance, a clustering partition, $(\mathbf{x}_1,\mathbf{x}_3)$ $(\mathbf{x}_2,\mathbf{x}_5,\mathbf{x}_8)$ $(\mathbf{x}_4,\mathbf{x}_6,\mathbf{x}_7)$, is represented by the chromosome 1 2 1 3 2 3 3 2.

3.2 *K*-means Operation

The *k*-means operation is used to modulate the distribution of objects among different clusters and to update cluster centers. It is described as follows: Given the i^{th} individual $X_i = x_{i1},\ldots,x_{ij},\ldots x_{iN}$, $i = 1,\ldots,P$, where P is the population size and N is the number of objects, reassign object $\mathbf{x}_j$ to cluster C_k^i, $k = 1,\ldots,K$, iff

$$\| \mathbf{x}_j - \mathbf{c}_k^i \|^2 < \| \mathbf{x}_j - \mathbf{c}_l^i \|^2,\ l = 1,\ldots,K, \text{ and } k \neq l. \tag{3}$$

If object $\mathbf{x}_j$ is reassigned to cluster C_k^i, then cluster center $\mathbf{c}_k^i$ is updated as

$$\mathbf{c}_k^i = \frac{1}{n_k^i} \sum_{j=1}^{N} w_{jk}^i \mathbf{x}_j , \tag{4}$$

where n_k^i denotes the number of objects belonging to cluster C_k^i.

3.3 Selection Operation

The popular proportional selection strategy is adopted in this article. Since the current problem is to minimize the objective function, the probability of selecting an individual is inversely proportional to its objective function value. That is, the individual with the minimum objective function value is the best. In addition, the elitist model of selection is used to carry the best individual from previous population to next population, which assures the evolution process to converge to the optimal result and keep strong selection pressure (Bhandari et al., 1996).

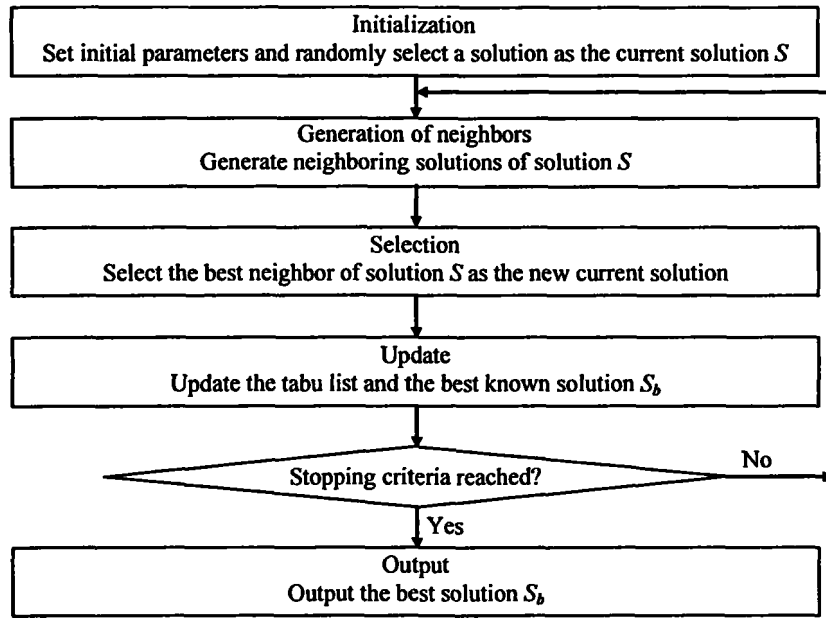


Fig. 2: General flow chard to tabu search

3.4 Tabu Operation

We give the general description of tabu search, which may be helpful in understanding the tabu operation. The general flow chart of tabu search is shown as Figure 2.

The basic elements of tabu search are described in the following.

- **Configuration** denotes an assignment of values to variables. That is, it is a solution to the optimization problem to be solved.
- **Move** denotes a specific procedure for getting a trial solution that is feasible to the optimization problem and related to the current configuration.
- **Neighborhood** denotes the set of all neighbors, which are the "adjacent solutions" that can be reached from the current configuration. It also includes neighbors that do not satisfy the feasible conditions defined.
- **Candidate subset** denotes a subset of the neighborhood. It is to be examined instead of the whole neighborhood, especially for huge problems where the neighborhoods include many elements.
- **Tabu restrictions** are constraints that prevent the chosen moves to be reversed or repeated, which play a memory role for the search by making the forbidden moves as tabu. The tabu moves are stored in the tabu list.
- **Aspiration criteria** denote rules that determine when the tabu restrictions can be overridden, thus removing a tabu classification otherwise applied to a move. If a certain move is forbidden by the tabu restrictions then the aspiration criteria, when satisfied, can make this move allowable.

In this paper, the tabu operation is adopted to avoid repeatedly copying the fitter individuals. If an individual does not violate the tabu restriction or is good enough to satisfy the aspiration criterion, then it is accepted, otherwise it is tabooed and a new individual is created by the probability threshold. The aim of the tabu operation is to decrease the loss of population diversity from the selection operation. For example, the probability of choosing an individual is proportional to its fitness in the roulette wheel selection. In this way, some fitter individuals may be chosen more than once. As a result, these super-fit individuals dominate the mating pool, which causes the loss of population diversity and weakens the exploration capability of the crossover operation. As the increase of selection pressure, variability decreases in the population, and consequently, there are fewer differences that the crossover

operation can explore by recombination (Potts et al., 1994). In the tabu operation, the probability threshold is adopted to create a new individual similar to but different from the current tabu individual. The fitter individuals are still selected to reproduce their children but they are modified and more different from each other than before. The tabu operation is described as follows:

Step 1: Given the current population, compute the objective function value of each individual. All individuals are ordered by their objective function values in an ascending order. Here, the ordered individuals are denoted as $X_1,\ldots,X_P$ and their objective function values are denoted as $f(X_1),\ldots,f(X_p)$, respectively, where P denotes the population size. It is found that the larger the population size, the better, because more individuals can be generated. However, this is done at the expense of more computational efforts. Therefore, deciding the proper population size is the process of exploring a balance between quality and cost. In this article, the population size $P=60$ can reach our goal. Set $i=1$.

Step 2: Considering individual X_1, let new individual $X_t^{'}=X_1$. If individual X_1 is not tabooed, then add individual X_1 into the tabu list and set the counter of the tabu list $t=t+1$. If $t>T$, then remove the first item of the tabu list, set $t=t-1$, and proceed to Step 7, where T denotes the size of the tabu list.

Step 3: If individual X_i is tabooed, proceed to Step 4, otherwise proceed to Step 5.

Step 4: If $f(X_t)<f_b$, that is, individual X_i satisfies the aspiration criterion, then let new individual $X_t^{'}=X_t$ and proceed to Step 7, otherwise proceed to Step 6, where f_b denotes the objective function value of the best known individual X_b.

Step 5: If $X_t\neq X_1$, then let new individual $X_i^{'}=X_i$ and proceed to Step 7, otherwise proceed to Step 6.

Step 6: Given individual X_i, neighboring individuals $X_1^t,\ldots,X_{Q_t}^t$ are established by the probability threshold, where Q_t denotes the number of neighboring individuals. Considering neighboring individual X_r^t, $r=1,\ldots,Q_t$, for $j=1,\ldots,N$, draw a random number $q_j\sim u(0,1)$, where $u(0,1)$ denotes the uniform probability distribution in the interval $[0,1]$. If $q_j<P_{pt}$, then $x_{rj}^t=x_{ij}$, otherwise draw randomly

an integer $\hat{l}$ from the following set $\{\hat{l} : \hat{l} = 1, \ldots, K, \hat{l} \neq x_{ij}\}$ and let $x_{rj}^{i} = \hat{l}$. As a result, neighboring individual X_r^t different from but similar to individual X_t is created. Note that neighboring individual X_r^t must be different from individual X_t. That is, we must continue this process until producing a feasible neighboring individual. Here, the probability threshold P_{pt} is used to tune the local search capability of the tabu operation by exploiting the neighboring space of individual X_t. After establishing the neighboring individuals of individual X_t, we define the best neighboring individual as new individual $X_t^{'}$.

Step 7: Add individual $X_t^{'}$ into the new population and let $i = i + 1$. If $i \leq P$, return to Step 3, otherwise proceed to Step 8.

Step 8: If $f(X_1) < f_b$, then $f_b = f(X_1)$, $X_b = X_1$, and stop.

It is seen that the tabu operation not only prevents the fitter individuals from cloning themselves so as to maintain population diversity but also keeps the best one in the mating pool so as to guide the evolutionary process.

3.5 Crossover Operation

Crossover is a probabilistic process that exchanges information between two parents for generating two children. In this paper, the single-point crossover with a fixed crossover probability p_c is adopted. A random integer called the crossover point is generated in the range $[1, N-1]$. The portions of the chromosomes lying to the right of the crossover point are exchanged to produce two offspring.

3.6 Mutation Operation

Mutation is also a probabilistic process that takes a single parent and modifies some genes with their alleles in a localized manner. As a result, bits of the chromosomes are chosen with probability p_m in the mutation phase and each chosen bit is changed from 1 to K. Schaffer et al. (1989) conducted extensive researches and gave the best parameter settings for genetic algorithms. According to the recommendation, the mutation probability and the crossover probability are chosen to be 0.01 and 0.8, respectively.

4. EXPERIMENTAL RESULTS

4.1 Performance Evaluation

In order to evaluate the performance of the GTK-Clustering algorithm, we use *Crude Oil data* to analyze the contribution made by each component. This data set has 56 data points, five features, and three classes. Hence the number of clusters is three (Johnson and Wichern, 1982). In addition, these experiments are used to determine the parameter settings. Each experiment includes 20 independent trials. Each trial includes a maximum of 1000 generations. The algorithm with the proper parameter setting often terminates before the specified number of generations.

- *Probability threshold*

  In the tabu operation, the probability threshold is used to moderate the shake-up on the current tabu individual and to create neighboring individuals. The higher the probability threshold, the less shake-up is allowed and, consequently, the more similar neighboring individuals to the current tabu one, and vice versa. However, with the increase of the probability threshold, it is more and more difficult to establish a feasible neighboring individual. Figure 3 shows the effect of the probability threshold on the objective function. It is seen that a high probability threshold can help the algorithm to attain the correct result. In Figure 4, the average and standard deviation values of generations where the best function value is obtained are shown for different probability thresholds. Table 1 gives the average value F_a and the standard deviation value F_{sd} of the clustering results for different probability thresholds.

From Table 1, it is found that the best result cannot be reliably obtained until P_{pt} is equal to 0.95. Under this condition, the number of generations is still large. In this article, the probability threshold P_{pt} is set to be 0.99. On one hand, it provides the

TABLE 1

Experimental results for different probability thresholds

P_{pt}	0.80	0.85	0.90	0.95	0.97	0.98	0.99	0.995	0.999
F_a	2524.9	2309.3	1872.4	1647.2	1647.2	1647.2	1647.2	1647.2	1647.2
F_{sd}	144.7556	210.4745	251.7575	0.0	0.0	0.0	0.0	0.0	0.0

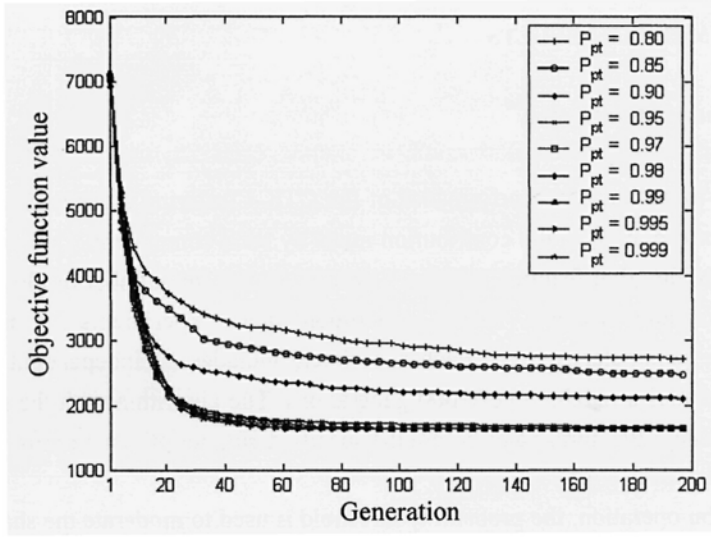


Fig. 3: Comparison of different probability

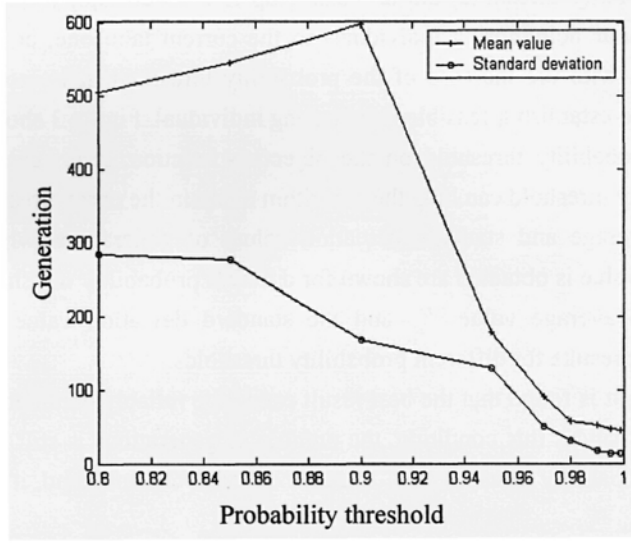


Fig. 4: Results of generations for different probability

high quality solution, and on the other hand, it does not need much computational resource.

- Tabu list

  The tabu list, an explicit memory structure of tabu search, is used to record the evolutionary history of the GTK-Clustering algorithm and to prevent it from

reversing the tabu moves under some conditions. The size of the tabu list determines how much memory the algorithm owns. The larger the size of the tabu list, the stronger the search capability of the algorithm, and vice versa. As the increase of the size of the tabu list, the capability of the global search is strengthened and the capability of the local search is restricted because many promising individuals are tabooed. If the size of the tabu list is too large or too small, the performance of the proposed method decreases.

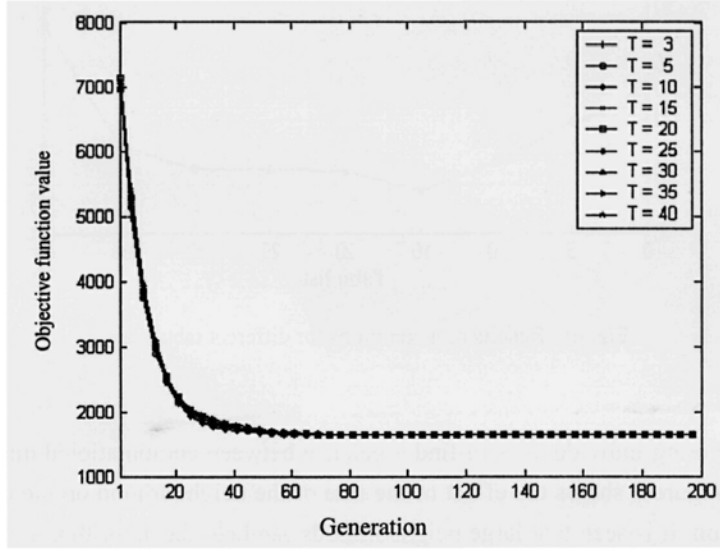


Fig. 5: Comparison of different tabu lists

As shown in Figure 5, the best result can be obtained in different cases. But the average and standard deviation values of generations vary with the size of the tabu list as shown in Figure 6. It is found that too large or too small tabu lists cause the algorithm to take more generations to find the best result. When $T = 15$, the best performance is obtained. Therefore, the size of the tabu list is chosen to be 15.

- *Neighborhood*

The size of the neighborhood denotes the number of neighboring individuals in this paper. It is found that the larger the size of the neighborhood the better, because the algorithm has more choices. However, this is done at the expense of more computational efforts. So, deciding on the proper number of

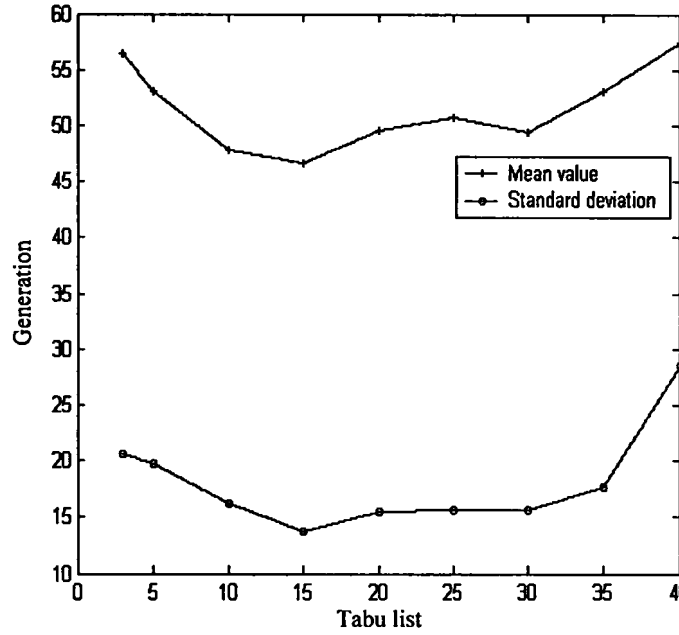


Fig. 6: Results of generations for different tabu lists

neighboring individuals is to find a balance between computational quality and cost. Figure 7 shows the effect of the size of the neighborhood on the objective function. It is seen that large neighborhoods can help the algorithm to attain the best result. In Figure 8, the average and standard deviation values of generations where the best function value is obtained are given. Table 2 shows the average value F_a and the standard deviation value F_{sd} of the clustering results for different sizes of the neighborhood.

TABLE 2

Experimental results for different sizes of the neighborhood

Q_t	5	10	20	40	60	80	100	120	140
F_a	1712.5	1654.7	1647.2	1647.2	1647.2	1647.2	1647.2	1647.2	1647.2
F_{sd}	103.7819	15.7065	0.0	0.0	0.0	0.0	0.0	0.0	0.0

It is found that the best result cannot be reliably obtained until Q_t is equal to 20. In this case, the number of generations is still large. In this article, the number of neighboring individuals is set to be 60. Under this condition, the balance between computational quality and cost can be maintained.

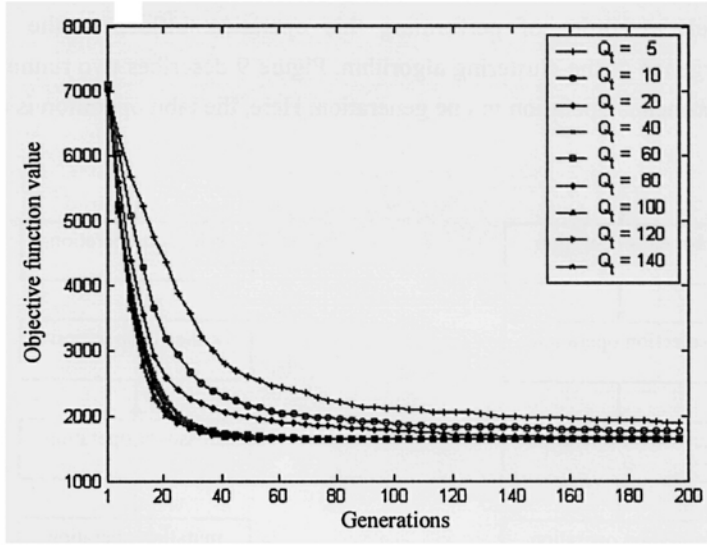


Fig. 7: Comparison of different neighborhoods

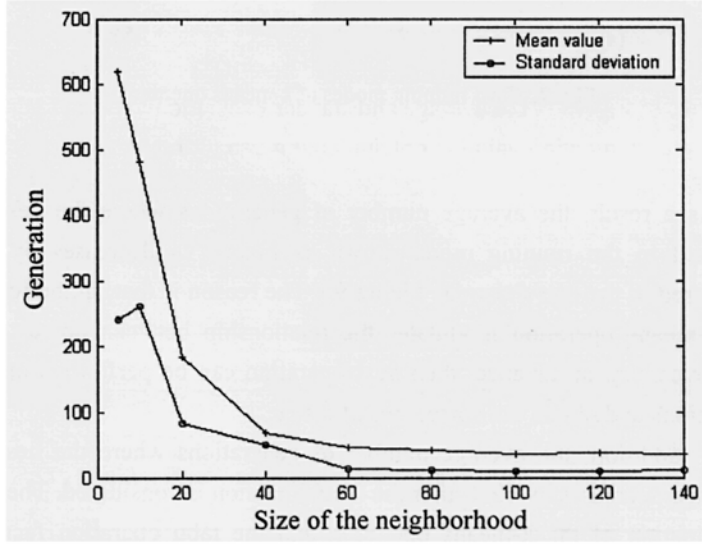


Fig. 8: Results of generations for different neighborhoods

- *K-means operation*

With simple and high-speed merits, the *k*-means operation is used to fine-tune the distribution of objects, to improve the similarity between objects and their cluster centroids, and to increase the speed of convergence of the clustering algorithm. Here, we discuss where it is better to perform the *k*-means operation. The relative order of performing this operation influences the speed of convergence of the clustering algorithm. Figure 9 describes two running modes of the *k*-means operation in one generation. Here, the tabu operation is not used.

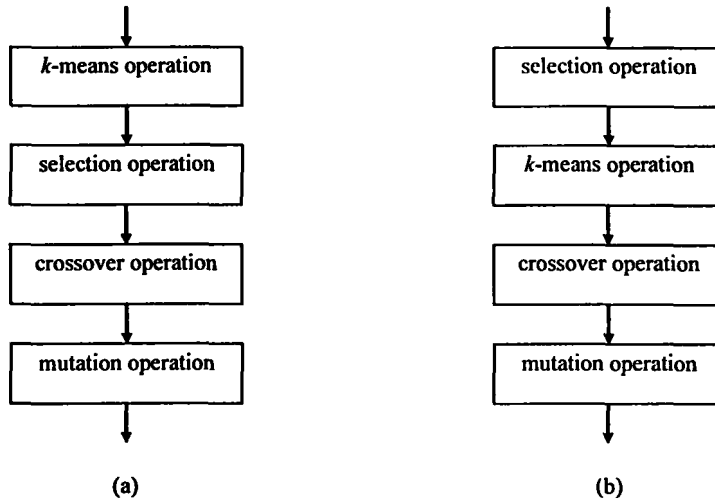


Fig. 9: Two running modes of *k*-means operation

As a result, the average number of generations where the best result is obtained in the running mode shown as Figure 9a decreases by over 8% compared to the one shown as Figure 9b. The reason is that, in the former case, the k-means operation modulates the relationship between objects and their cluster centers in advance, then next operation can be performed under better conditions and obtain the correct result faster.

In addition, the average number of generations where the best result is obtained decreases by 12% after the tabu operation is considered. Therefore, the combination of the k-means operation and the tabu operation facilitates the GTK-Clustering algorithm to output the correct result.

- Population diversity

  It is well known that the population diversity problem is vital to GA. In this paper, this problem is still important because the basic architecture of the GTK-Clustering algorithm is based on GA. But it is difficult to evaluate the difference between two clustering solutions. For example, two solutions, $X_1 = 122132131$ and $X_2 = 233213212$, seem to be different. In fact, they represent the same partition. The clustering partition of solution X_1 is denoted as $(\mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_7, \mathbf{x}_9)$ $(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_6)$ $(\mathbf{x}_5, \mathbf{x}_8)$, and the clustering partition of solution X_2 is denoted as $(\mathbf{x}_5, \mathbf{x}_8)$ $(\mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_7, \mathbf{x}_9)$ $(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_6)$. Since solutions X_1 and X_2 represent the same clustering partition, they are equal in fact. In clustering analysis, objects are often randomly allocated among different clusters at the initialization stage. Two different solutions may represent the same partition. Then the difference between two solutions should be measured by the clustering partition. In the genetic clustering method, the chromosome shows the relationship among different objects, i.e., whether two objects belong to the same cluster. As objects in clustering analysis are unlabeled, it is necessary to label a clustering partition so as to estimate the difference among individuals. In this article, we view the best individual in each generation as the reference partition and label it. Then the difference between each individual and the labeled one is calculated by estimating the number of objects belonging to different clusters. As the decrease of the difference between individuals and the labeled one, the population diversity decreases. The following example may be helpful in understanding how to calculate the value of population diversity. Given a population with five individuals, suppose individual X_1^t as the best individual X_b^t in the t^{th} generation, then its clustering partition is labeled. The population diversity is calculated as follows:

  Step 1: Set up the clustering partition for each individual. According to the given population as shown in Figure 10a, the objects belonging to the same cluster are grouped together, then the clustering partitions are shown as Figure 10b. Take individual X_1^t for instance, objects $\mathbf{x}_1$, $\mathbf{x}_4$, $\mathbf{x}_7$, and $\mathbf{x}_9$ belonging to cluster C_{11}^t, objects $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_6$ belonging to cluster C_{12}^t, and objects $\mathbf{x}_5$ and $\mathbf{x}_8$ belonging to cluster C_{13}^t are grouped into clusters C_{11}^t, C_{12}^t, and C_{13}^t, respectively.

Step 2: Build the $P \times N$ matrix $\mathbf{D}^t$ representing the difference between individuals X_i^t and X_h^t, $i = 1,\ldots,P$. If

$$\| \mathbf{c}_{ik}^t - \mathbf{c}_{bp}^t \|^2 < \| \mathbf{c}_{ik}^t - \mathbf{c}_{bq}^t \|^2, \tag{5}$$

then clusters C_{bp}^t is associated with cluster C_{ik}^t and prevented from being associated with any other cluster of individual X_i^t, where $k, p, q = 1,\ldots,K$, and $p \neq q$. In Eq. (5), $\mathbf{c}_{ik}^t$ denotes the cluster center of cluster C_{ik}^t of individual X_i^t, $\mathbf{c}_{bp}^t$ and $\mathbf{c}_{bq}^t$ denote cluster centers of clusters C_{hp}^t and C_{bq}^t of individual X_h^t, respectively. If object $\mathbf{x}_j$ belonging to cluster C_{ik}^t also belongs to cluster C_{bp}^t, $d_{ij}^t = 0$, otherwise $d_{ij}^t = 1$, where $j = 1,\ldots,N$. Since individual X_1^t represents the reference partition, set $d_{1j}^t = 0$. As a result, matrix $\mathbf{D}^t = [d_{ij}^t]_{P \times N}$ is set up as shown in Figure 10c.

Step 3: Calculate the population diversity

$$d(t) = \frac{1}{P * N} \sum_{i=1}^{P} \sum_{j=1}^{N} d_{ij}^t, \tag{6}$$

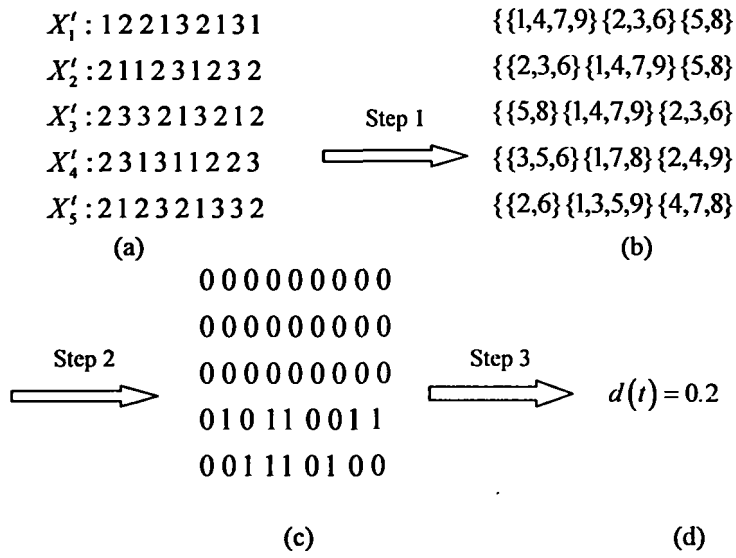
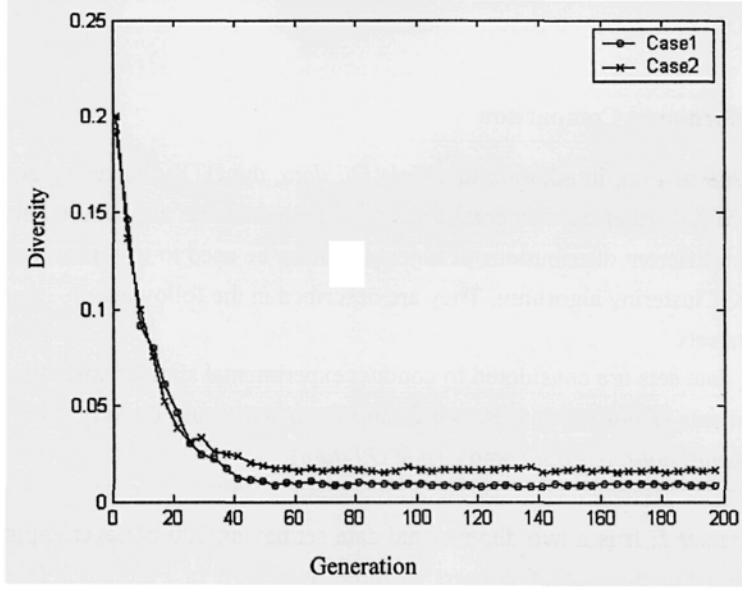


Fig. 10: The calculation process of population diversity

where $d(t)$ denotes the population diversity in the t^{th} generation. As the increase of the number of the fitter individuals, the population diversity reduces. If the population is dominated by the fittest one, i.e., all clustering results converge to the best, then the population diversity will decrease to zero. The calculation process of population diversity is illustrated in Figure 10.

It is known that population diversity varies with the number of generations. Figure 11 shows the results of population diversity in two cases. In the first case, only the *k*-means operation is integrated into the GTK-Clustering algorithm, while both the *k*-means operation and the tabu operation are integrated into the GTK-Clustering algorithms in the second case. It is found that, after eighty generations, the average value of population diversity increases by over 87% in the second case compared to that in the first case. Figure 12 shows the growth of population diversity in the process of generations when both the *k*-means operation and the tabu operation are considered.



Case 1: GTK-Clustering with *k*-means operation only;
Case 2: GTK-Clustering with *k*-means operation and tabu operation

Fig. 11: Comparison of population diversity in two cases

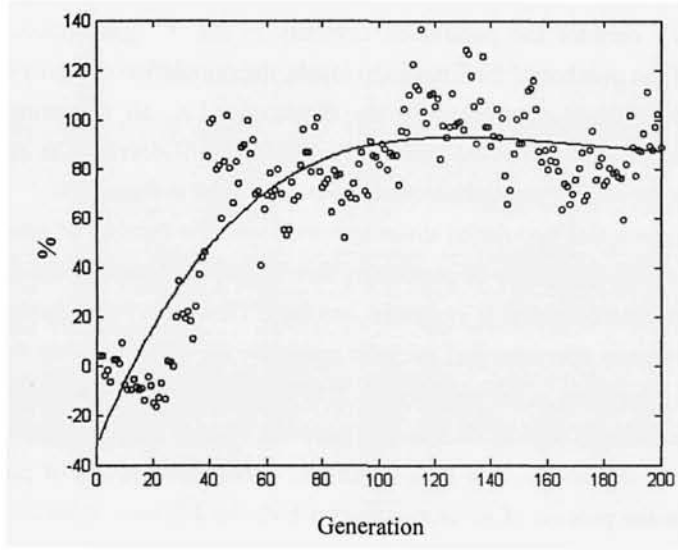


Fig. 12: Growth of population diversity

4.2. Performance Comparison

In this section, in addition to *Crude Oil data*, the GTK-Clustering algorithm is applied to five other experimental data sets. These data sets are chosen because they represent different distributions of objects and can be used to test the adaptability of the GTK-Clustering algorithm. They are described in the following.

- Data sets

  Six data sets are considered to conduct experimental simulations, three artificial data sets (*Data set 1*, *Data set 2*, and *Data set 3*) and three real life data sets (*Ruspini data*, *Iris data*, and *Crude Oil data*).

  Data set 1: It is a two dimensional data set having 200 nonoverlapping objects where the number of clusters is three as shown in Figure 13 (Kaufman & Rousseeuw, 1990).

  Data set 2: It is a two dimensional data set having 250 overlapping objects where the number of clusters is five as shown in Figure 14.

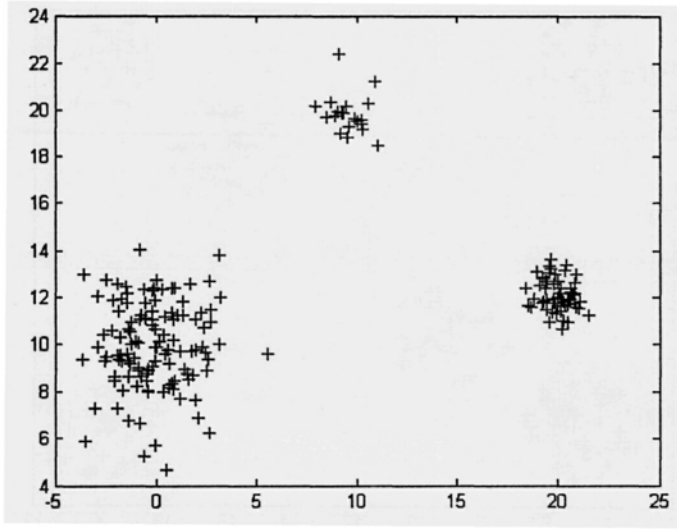


Fig. 13: Data set 1

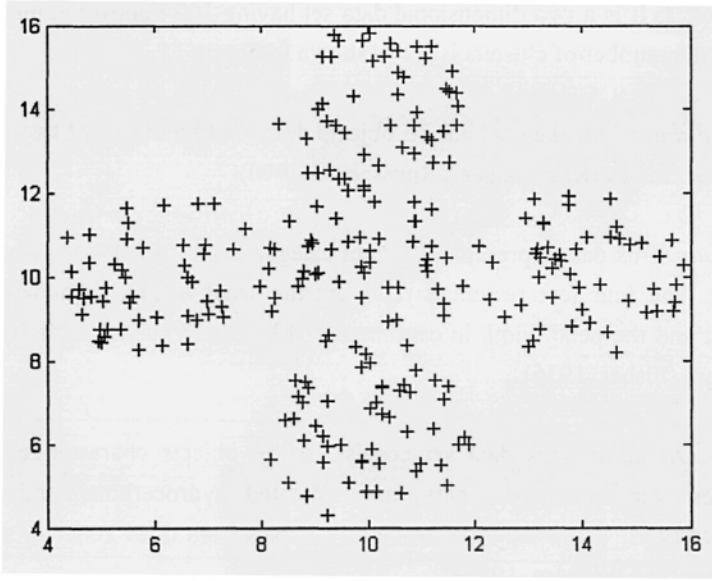
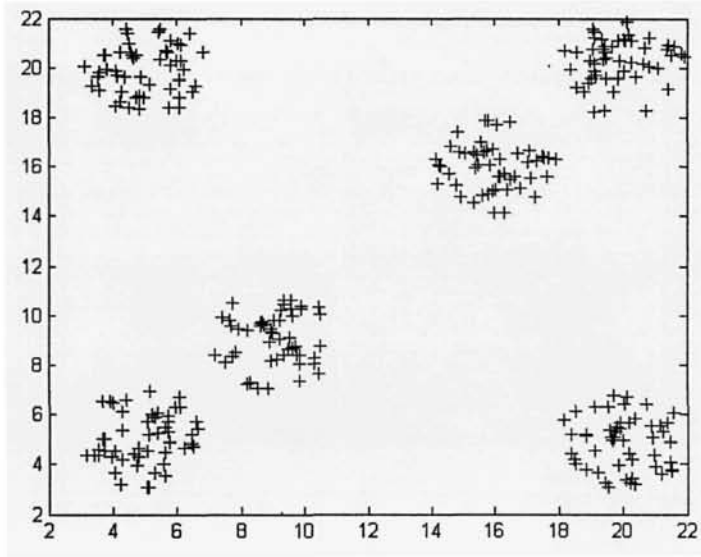


Fig. 14: Data set 2



Data set 3: It is a two dimensional data set having 300 nonoverlapping objects where the number of clusters is six as shown in Figure 15.

Ruspini data: This data set has 75 objects described by means of two attributes and four classes (Kaufman and Rousseeuw, 1990).

Iris data: This data represents different categories of irises having four feature values. The four feature values represent the sepal length, sepal width, petal length, and the petal width in centimeters. It has three classes with 50 samples per class (Fisher, 1936).

Crude Oil data: This data set consists of 56 objects characterized by five features: vanadium, iron, beryllium, saturated hydrocarbons, and aromatic hydrocarbons. There are three crude-oil samples from three zones of sandstone (Johnson and Wichern, 1982).

- Comparative performance

  In computer simulations, experimental results of the *k*-means algorithm, the genetic clustering algorithm proposed by Murthy and Chowdhury (1996) called

GA-Clustering in this paper, and the GTK-Clustering algorithm are obtained after 1000 generations. Each experiment includes 20 independent trials. In all experiments, the *k*-means algorithm terminates much before the specified number of iterations. In order to make a fair comparison, we add the run number of the *k*-means algorithm up to 500.

The best, worst, average, and standard deviation values of the clustering results (F_b, F_w, F_a, and F_{sd}) obtained by experimental methods for *Data set 1* are shown as Table 3. The minimum value is 827.07819500, which is found by the *k*-means algorithm and the GTK-Clustering algorithm. The *k*-means algorithm attains this value in 328 of all runs. In other cases, it gets stuck at suboptimal values. The GTK-Clustering algorithm achieves the minimum value in all trials. Noticeably, the GA-Clustering algorithm fails to find this value even once and its best value obtained is far worse than the minimum one.

TABLE 3

Experimental results for *Data set 1*

	k-means	GA-Clustering	GTK-Clustering
F_b	827.07819500	9292.19170588	827.07819500
F_w	3107.01572127	12772.15604451	827.07819500
F_a	1605.63549463	10465.38301918	827.07819500
F_{sd}	1076.23166737	917.88843250	0.0

TABLE 4

Experimental results for *Data set 2*

	k-means	GA-Clustering	GTK-Clustering
F_b	488.02127209	2532.74791118	488.02127209
F_w	489.72002017	2837.88808552	488.02127209
F_a	488.76498588	2693.07766089	488.02127209
F_{sd}	0.61285389	85.83906879	0.0

Table 4 shows the clustering results for *Data set 2*. It is seen that the GTK-Clustering algorithm is superior to the other two approaches. The minimum value is 488.02127209, which is found by the GTK-Clustering algorithm in each

TABLE 5

Experimental results for *Data set 3*

	k-means	GA-Clustering	GTK-Clustering
F_b	543.17158863	18942.54626552	543.17158863
F_w	11784.43513297	20898.25357788	543.17158863
F_a	1508.63097356	20076.80008430	543.17158863
F_{sd}	1456.25855407	492.08032040	0.0

trial. As objects are overlapping, the *k*-means algorithm attains the minimum value in 5 of 500 runs and the GA-Clustering algorithm cannot obtain the best result in all trials. In most runs, the results provided by the *k*-means algorithm are close to the minimum value. In addition, the GA-Clustering algorithm is still unable to output meaningful results.

Considering *Data set 3,* the clustering results are shown as Table 5. The minimum value is 543.17158863, which is achieved by the *k*-means algorithm and the GTK-Clustering algorithm. The *k*-means algorithm attains this value in 128 of all runs. In many cases, it falls into local minima. In addition, its standard deviation value is the highest. Here, the GTK-Clustering algorithm provides the best result in all runs.

In Table 6, the clustering results obtained by experimental methods for the *Ruspini data* are shown. The minimum value is 12881.05123615, which is found by the *k*-means algorithm and the GTK-Clustering algorithm. The GA-Clustering algorithm cannot find the minimum value. The *k*-means algorithm attains the best value in 285 of 500 trials and its standard deviation value is still the highest. In this experiment, the GTK-Clustering algorithm provides the best result in each trial.

The best, worst, average, and standard deviation values of the clustering results obtained by experimental methods for *Iris data* are shown as Table 7. It is seen that the GTK-Clustering algorithm is still superior to the other two algorithms. The minimum value is 78.94084143, which is found by the GTK-Clustering algorithm in all runs. The *k*-means algorithm attains the minimum value in 211 of all trials. For *Iris data*, the GA-Clustering algorithm is unable to provide meaningful results within 1000 generations.

TABLE 6

Experimental results for *Ruspini data*

	k-means	GA-Clustering	GTK-Clustering
F_b	12881.05123615	49333.47474747	12881.05123615
F_w	50737.37500000	96892.91345029	12881.05123615
F_a	28657.04460704	75795.98314009	12881.05123615
F_{sd}	18186.93860351	12080.62920003	0.0

TABLE 7

Experimental results for *Iris data*

	k-means	GA-Clustering	GTK-Clustering
F_b	78.94084143	252.34075132	78.94084143
F_w	145.27932204	365.63778259	78.94084143
F_a	93.07903807	298.38629892	78.94084143
F_{sd}	26.80459670	33.65897180	0.0

TABLE 8

Experimental results for *Crude Oil data*

	k-means	GA-Clustering	GTK-Clustering
F_b	1647.18926793	1722.42257008	1647.18926793
F_w	2634.75937897	2546.07127942	1647.18926793
F_a	1656.92787119	2105.33821197	1647.18926793
F_{sd}	59.85298654	226.56419016	0.0

For *Crude Oil data*, the best, worst, average, and standard deviation values of the clustering results obtained by experimental algorithms are shown as Table 8. The GTK-Clustering algorithm finds the best result in all runs. The *k*-means algorithm attains the minimum value 1647.18926793 in 72 of 500 trials and the GA-Clustering algorithm fails to provide the best result in all trials.

In face of experimental data sets, the GTK-Clustering algorithm provides the best results in all trials. It is surprising that the GA-Clustering algorithm is bound to be the worst in all cases. We also find that it can still obtain improved results if more generations are executed. In the following, we add the number of generations up to 10000 suggested by Murthy and Chowdhury (1996) so as to test the GA-Clustering algorithm and compare the speed of convergence of two genetic clustering approaches.

- Comparison of two genetic clustering algorithms

  Firstly, the time complexities of two algorithms are analyzed as follows. For the GA-Clustering algorithm, its time complexity is $O(GPNm)$, where G denotes the number of generations. Considering the GTK-Clustering algorithm, in each generation, the complexities of selection operation, tabu operation, crossover operation, mutation operation and *k*-means operation are $O(PNm)$, $O(Q_tPNm)$, $O(P)$, $O(PN)$, and $O(KPNm)$, respectively. In general, $Q_t >> K$, the time complexity is dominated by the tabu operation. Then the time complexity of the GTK-Clustering algorithm is equal to $O(G'Q_tPNm)$, where G' denotes the number of generations of the GTK-Clustering algorithm. If $G = G'$, then the proposed method needs more computational resource to obtain the best result. In experimental simulations, we let G be equal to G' in order to make a fair comparison and observe the evolutionary process. We find that, in all trials, the GTK-Clustering algorithm attains the best results much before the specified number of generations. In addition, the GA-Clustering algorithm is unable to provide meaningful results for all data sets. To compare the speed of convergence of two genetic clustering algorithms, we add the number of generations of the GA-Clustering algorithm up to 10000 suggested by Murthy and Chowdhury (1996). It is reported that the GA-Clustering algorithm can find the best result of *Crude Oil data* (Murthy and Chowdhury, 1996). We keep the number of generations of the GTK-Clustering algorithm constant because it can take much less than 1000 generations to find the best result.

  In Table 9, T_b denotes the number of trials in which the algorithm attains the best result, G_a and G_a' denote the average number of generations for which the best result is obtained by the GA-Clustering algorithm and the GTK-Clustering algorithm, respectively. We find that, with the increase of the number of generations, the GA-Clustering algorithm provides better results and F_a greatly decreases. It outputs the best results for *Ruspini data* and *Crude Oil data*

in some runs, but does not attain the best results for other data sets. In addition, we find that $G_a' << G_a$ and $G_a' Q_t << G_a$. As a result, the speed of convergence of the GTK-Clustering algorithm is much higher than that of the GA-Clustering algorithm.

TABLE 9

Comparison of two genetic clustering algorithms

Data set	GA-Clustering			GTK-Clustering			
	T_b	G_a	F_a	T_b	G'_a	Q_t	$G'_a Q_t$
Data set 1	0	-	3909.88	20	1.3	60	78
Data set 2	0	-	1919.29	20	21.1	60	1266
Data set 3	0	-	14168.88	20	6.6	60	396
Ruspini data	9	9748.6	24001.28	20	2.8	60	168
Iris data	0	-	136.51	20	8.0	60	480
Crude Oil data	14	9390.6	1701.64	20	6.8	60	408

5. CONCLUSIONS

The clustering problem is a fundamental problem that frequently arises in a great variety of fields. In general, the clustering problem is a nonconvex problem that possesses many locally optimal values, resulting that its solution often falls into these traps. In this paper, a hybrid genetic clustering algorithm called GTK-Clustering is proposed. With the cooperation of the tabu operation and the *k*-means operation, the GTK-Clustering algorithm achieves the harmony between population diversity and the speed of convergence. The tabu operation is used to prevent the population from being dominated by some fitter individuals and to keep a high level of population diversity. Moreover, the *k*-means operation is adopted to improve the distribution of objects and to increase the speed of convergence. By analyzing the components of the GTK-Clustering algorithm, we determine the proper parameter settings. The superiority of the proposed algorithm over the *k*-means algorithm and the GA-Clustering algorithm is demonstrated for experimental data sets. After analyzing two genetic clustering algorithms, we find that the speed of convergence of the proposed algorithm is much higher than that of the GA-Clustering algorithm.

In future, we have two research works to do. One is that the estimation of the number of clusters should be considered. In this paper, the number of clusters is known as input. But this parameter is unknown for the designer in many cases. Liu et al. (2004) proposed a tabu search based clustering algorithm consisting of two stages to detect unknown intrusions. The presented method can automatically set up clusters and detect intrusions by labeling normal and abnormal groups. Handl and Knowles (2005) developed the MOCK algorithm, a multiobjective clustering algorithm, to automatically detect the number of clusters. So, more attention should be paid to solving this problem by extending the principle of the GTK-Clustering algorithm to the case where the number of clusters is not known *a priori*. The other is that, in this article, we focus on the balance between population diversity and the speed of convergence. How to keep the balance in complicate problems, especially in real-world high-dimensional cases, will be the subject of future research.

ACKNOWLEDGMENTS

We would like to thank anonymous referees for their constructive comments and suggestions, which have been very helpful in improving the paper. We feel greatly indebted to Fred Glover, Patrick Siarry, and Rachid Chelouah for their valuable comments and suggestions on our works. In addition, we would like to thank Sanghamitra Bandyopadhyaya for offering *Data set 2* and *Data set 3*. This research was supported in part by the National Natural Science Foundation of China (Grant No. 60903074), the National High Technology Research and Development Program (863 Program) of China (Grant No. 2008AA01Z119), and the Youth Key Foundation of University of Electronic Science and Technology of China.

REFERENCES

Abdinnour-Helm, S. (1998). A hybrid heuristic for the uncapacitated hub location problem, *European Journal of Operational Research*, **106**, 489-499.

Bagirov, A.M. (2008). Modified global k-means algorithm for minimum sum-of-squares clustering problems, *Pattern Recognition*, **41**, 3192-3199.

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

Bhandari, D., Murthy, C.A., Pal, S.K. (1996). Genetic algorithm with elitist model and its convergence, *International Journal of Pattern Recognition and Artificial Intelligence*, **10**, 731-747.

Bifulco, I., Murino, L., Napolitano, F., Raiconi, G., Tagliaferri, R. (2008). Using global optimization to explore multiple solutions of clustering problems, In: *Proceedings of the 12th International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, Springer-Verlag, Zagreb, Croatia, pp. 724-731.

Fisher, R.A. (1936). The use of multiple measurements in taxonomic problem, *Annals of Eugenics*, 7, 179-188.

Glover, F., Laguna, M. (1997). *Tabu Search*, Boston, Kluwer.

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

Güngör, Z., Ünler, A. (2008). *K*-Harmonic means data clustering with tabu-search method, *Applied Mathematical Modelling*, **32**, 1115-1125.

Hall, L.O., Ozyurt, B., Bezdek, J.C. (1999). Clustering with a genetically optimized approach, *IEEE Transactions Evolutionary Computation*, **3**, 103-112.

Handa, K., Kuga, S. (1995). Pollycell placement for analog LSI chip designs by genetic algorithms and tabu search. In: *Proceeding of IEEE International Conference on Evolutionary Computation*, IEEE Press, Perth, pp. 716-721.

Handl, J., Knowles, J. (2005). Exploiting the trade-off – The benefits of multiple objectives in data clustering, *Lecture Notes in Computer Science*, **3410**, 547-560.

Jarboui, B., Cheikh, M., Siarry, P., Rebai, A. (2007). Combinatorial particle swarm optimization (CPSO) for partitional clustering problem, *Applied Mathematics and Computation*, **192**, 337-345.

Johnson, R.A., Wichern, D.W. (1982). *Applied multivariate statistical analysis*, New Jersey, Prentice-Hall.

Kao, Y.T., Zahara, E., Kao, I.W. (2008). A hybridized approach to data clustering, *Expert Systems with Applications*, **34**, 1754-1762.

Kaufman, L., Rousseeuw, P.J. (1990). *Finding groups in data – An introduction to cluster analysis*, New York, Wiley.

Laszlo, M., Mukherjee, S. (2006). A genetic algorithm using hyper-quadtrees for low-dimensional k-means clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**, 533-543.

Liu, Y.G., Liao, X.F., Li, X.M., Wu, Z.F. (2004). A tabu clustering algorithm for intrusion detection, *Intelligent Data Analysis*, **8**, 325-344.

Lu, J.F., Tang, J.B., Tang, Z.M., Yang, J.Y. (2008). Hierarchical initialization approach for K-Means clustering, *Pattern Recognition Letters*, **29**, 787-795.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations, In: *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, CA, pp. 281-297.

Murthy, C.A., Chowdhury, N. (1996). In search of optimal clusters using genetic algorithms, *Pattern Recognition Letters*, **17**, 825-832.

Omran, M.G.H., Engelbrecht, A.P., Salman, A. (2007). An overview of clustering methods. *Intelligent Data Analysis*, **11**, 583-605.

Pedrycz, W. (2005). *Knowledge-based Clustering*, New Jersey, Wiley.

Potts, J.C., Giddens, T.D., Yadav, S.B. (1994). The development and evaluation of an improved genetic algorithm based on migration and artificial selection, *IEEE Transactions on Systems, Man and Cybernetics*, **24**, 73-86.

Redmond, S.J., Heneghan, C. (2007). A method for initialising the K-means clustering algorithm using *kd*-trees, *Pattern Recognition Letters*, **28**, 965-973.

Schaffer, J.D., Caruana, R.A., Eshelman, L.J., Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization, In: Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufmann, Fairfax, Virginia, pp. 51-60.

Selim, S.Z., Ismail, M.A. (1984). K-means-type algorithm: generalized convergence theorem and characterization of local optimality, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 81-87.

Sung, C.S., Jin, H.W. (2000). A tabu-search-based heuristic for clustering, *Pattern Recognition*, **33**, 849-858.

Ting, C.K., Li, S.T., Lee, C. (2003). On the harmonious mating strategy through tabu search, *Information Sciences*, **156**, 189-214.

Whitley, D. (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In: *Proceedings of 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp. 116-121.

Yang, F.Q., Sun, T.L. Zhang, C.H. (2009). An efficient hybrid data clustering method based on K-harmonic means and Particle Swarm Optimization, *Expert Systems with Applications*, **36**, 9847-9852.