# AN EFFICIENT HYBRID METHOD FOR CLUSTERING PROBLEMS

H. Panahi and R. Tavakkoli-Moghaddam
*Department of Industrial Engineering, Faculty of Engineering, University of Tehran*
*P.O. Box: 11365-4563, Tehran, Iran*

Keywords: Clustering problems, Ant colony optimization, Genetic algorithms, Paired comparison test.

Abstract: This paper presents a hybrid efficient method based on ant colony optimization (ACO) and genetic algorithms (GA) for clustering problems. This proposed method assumes that agents of ACO has life cycle which is variable and changes by a special function. We also apply three local searches on the basis of heuristic rules for the given clustering problem. This proposed method is implemented and tested on two real datasets. Further, its performance is compared with other well-known meta-heuristics, such as ACO, GA, simulated annealing (SA), and tabu search (TS). At last, paired comparison *t*-test is also applied to proof the efficiency of our proposed method. The associated output gives very encouraging results; however, the proposed method needs longer time to proceed.

## 1 INTRODUCTION

Clustering is implemented to group objects into clusters so that the objects with similar attributes bring together into a batch. Some applications of cluster analysis are qualitative interpretation and data compression, process monitoring, local model development, toxicity testing, finding structure-activity relations, classification of coals, pattern recognition, and optimization.

The objective functions of clustering problems are usually nonlinear and non-convex so some clustering algorithms may fall into local optimum. Moreover, they possess exponential complexity in terms of number of clusters and become an NP-hard problem when the number of clusters exceeds (Shelokar et al., 2004). Several algorithms have been proposed to solve clustering problems (Banfield and Raftery, 1993; Jiang et al., 1997). Meta-heuristic algorithms, such as ant colony optimization (ACO) (Shelokar et al., 2004), tabu search (TS) (Al-Sultan, 1995), genetic algorithms (GA) (Jiang et al., 1997; Murthy and Chowdhury, 1996), simulated annealing (SA) (Selim and Al-Sultan, 1991; Sun et al., 1994), and particle swarm optimization (PSO) (Paterlini and Krink, 2006) have been used for clustering problems.

In this paper, a new method by hybridizing two algorithms, namely GA and ACO, is represented and proposed for clustering problems. In Sections 2, ACO algorithm and local searches are described. In Section 3 the proposed GA-ACO method is explained. Section 4 illustrates the computational results to evaluate the performance of the proposed method on two real datasets. Finally, the remarking conclusion is derived.

## 2 ACO CLUSTERING METHOD

Ant colony optimization (ACO) was first introduced by Dorigo et al. (1996) to solve discrete optimization problems. This algorithm simulates the way that real ants find the shortest way between a food source and their nest. Ants communicate with each other by pheromone and exchange information about selecting the best path. The path chosen by more ants gains more chance to be chosen by other ants and if they choose that path, they increase the probability of that path to be chosen by next ants. The idea of ACO algorithm applied in this paper is inspired from Shelokar et al. (2004).

Each agent represents a feasible solution. After assigning such objects to clusters each agent gets a fitness due to Equation (2). Information about each agent is saved in to a matrix called pheromone trail matrix. Indeed this matrix is built due to the value of assigning objects to clusters by agents. In other

Panahi H. and Tavakkoli-Moghaddam R. (2008).
AN EFFICIENT HYBRID METHOD FOR CLUSTERING PROBLEMS.
In *Proceedings of the Tenth International Conference on Enterprise Information Systems - AIDSS*, pages 288-294
DOI: 10.5220/0001703002880294
Copyright © SciTePress

words, if assigning an object to a cluster makes a good reduction in the objective function, it affects the pheromone trail matrix in a way that other agents will assign such object to such cluster with a high probability but not certainly. Stopping criterion of the proposed method is a number of iterations. Following, we describe our proposed method.

Assume that we want to cluster *N* objects to a predefined number of clusters, *K*. For this purpose, we applied *R* agents. The proposed string representation is depicted in Fig. 1. This figure is a solution string for a problem with ten objects and four clusters. The number of assigned clusters is shown in its related cell.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 3 | 1 | 2 | 4 | 1 | 2 | 3 | 1 |

Figure 1: String representation.

As mentioned before, agents find their way due to pheromone trail matrix, $\pi$, with the size of $N \times K$. This size does not depend on the number of agents so it means the information of agents is accumulated in such a matrix and all of the agents use the same matrix for assigning objects to clusters. $\pi(i, j)$ represents the concentration of assigning object *i* to cluster *j*. The value of elements of a pheromone trail matrix varies by two factors: (1) Fitness of agents and (2) evaporation rate. Evaporation rate is applied to lessen the effect of past information. It helps to discover solution space. If evaporation rate is set to a high value it means the information gathered from the past is diminished fast but the low value of evaporation rate implies that the information obtained past iterations are diminished slowly. Finally, three local search methods are designed and applied with a predefined probability in order to improve the solution quality.

## 2.1 Algorithm Details

At first iteration, agents assign objects to clusters randomly. The pheromone trail matrix is fed by equal values, the value is not important because we use a normalized pheromone trail matrix for decision making. At the end of each generation, the pheromone trail matrix is updated by elitist agents.

Agents assign objects to clusters by one of the following two procedures:

**Procedure (I):** Due to pheromone trail matrix, for a given object, they choose the cluster with the highest value. For example, if the agent decides to assign a cluster to object <u>*i*</u> by Procedure (I), then it finds the maximum value in the *i*-th line of the pheromone trail matrix, the associated cluster is assigned to object *i*. Agents choose procedure (I) by a predefined probability $q_0$; known as exploitation.

**Procedure (II):** The cluster is chosen by a stochastic probability. This procedure is chosen by a probability of $(1\text{-}q_0)$ and known as biased exploration.

Briefly, each agent generates a random number for each object. If the random number is less than or equal to $q_0$, then it applies Procedure (I) else it uses Procedure (II). Suppose that for object *i*, Procedure (II) is taken. At first step, we must make a probabilistic distribution as follows:

$$p_{ij} = \frac{\pi_{ij}}{\sum_{k=1}^{k} \pi_{ik}}, \quad j = 1,\ldots,k \tag{1}$$

By the use of Equation (1), a probabilistic distribution is made and it is obvious that $P_{iK}$ =1. By this way, *K* intervals are made with $P_{ij}$'s. Then another random number is generated. Depending on which interval contains the random number, the associated cluster is assigned to object *i*.

After allocating all objects to clusters for each agent, agents should be evaluated. In this paper, the objective function is set to be the sum of squared Euclidean distance between each object and its associated cluster center. We assume that each object has *A* attributes. Assume that $m_{i\gamma}$ represents the average of the $\gamma$-th attribute of the *i*-th cluster and $w_{ij}$ is an integer variable where if object *i* is assigned to cluster *j* is equal to one else is set to zero and $x_{i\gamma}$ is the value of the $\gamma$-th attribute of the *i*-th object. The objective function is computed by Equation (2). Moreover, $w_{ij}$ must satisfy Equations (3) and (4) and $m_{i\gamma}$ is computed by Equation (5).

$$\min\ F = \sum_{j=1}^{K} \sum_{i=1}^{N} \sum_{\gamma=1}^{A} w_{ij} \times \left\| x_{i\gamma} - m_{j\gamma} \right\|^2 \tag{2}$$

$$\sum_{j=1}^{K} w_{ij} = 1, \qquad i = 1,\ldots,N \tag{3}$$

$$\sum_{i=1}^{N} w_{ij} \geq 1, \qquad j = 1,\ldots,K \tag{4}$$

$$m_{j\gamma} = \frac{\sum\limits_{i=1}^{N} w_{ij} x_{i\gamma}}{\sum\limits_{i=1}^{N} w_{ij}}, \quad j = 1,\ldots,K \quad v = 1,\ldots,A \tag{5}$$

## 2.2 Local Search

Usually local searches are performed on a few percent of the population. After calculating fitness of each agent, agents are sorted due their fitness value. Three different local searches are designed in this paper. Before describing our proposed local searches, we need to define a matrix; *eval_cluster*. This is a matrix of $R \times K \times N$. When we assign cluster *j* to object *i* it means we have added a value to our objective function which is equal to $\sum_{\gamma=1}^{A} w_{ij} \times \left\| x_{i\gamma} - m_{j\gamma} \right\|^2$ ; however, we do not know that if this assignment is the best because we have *k-1* options else. By this way, we make a matrix for each agent in the following manner:

$$\begin{bmatrix} \sum_{\gamma=1}^{A} w_{11} \times \left\| x_{1\gamma} - m_{1\gamma} \right\|^2 & . & . & \sum_{\gamma=1}^{A} w_{1K} \times \left\| x_{1\gamma} - m_{K\gamma} \right\|^2 \\ . & . & . & . \\ . & . & . & . \\ \sum_{\gamma=1}^{A} w_{N1} \times \left\| x_{N\gamma} - m_{1\gamma} \right\|^2 & . & . & \sum_{\gamma=1}^{A} w_{NK} \times \left\| x_{N\gamma} - m_{K\gamma} \right\|^2 \end{bmatrix}$$

In other words, the element in the *i*-th row and the *j*-th column is the value that is added to the objective function if object *j* is assigned to cluster *i*. In the next section, we describe how this matrix is applied to optimize clustering problems and we explain each local search.

### 2.2.1 Local Search A_1

This local search is designed for agent with rank one. After sorting agents in ascending trend due to fitness values, first agent is chosen for the local search "A_1". A random vector is generated with the length of *N*. If any of vector values is less than or equal to a predefined threshold, which is set to 0.02, the local search will be applied. Assume that agent with rank one assigned object *i* to cluster *j*. Suppose that the associated random vector of object *i* is suitable for the local search. As the first step, we find the minimum value of Line 1 of the *eval_cluster* matrix.

$$opt\_cluster = \arg\{\min(eval\_cluster(1,i,:))\} \qquad (6)$$

If *opt_cluster* is equal to *j*, then we take no action; however, if *opt_cluster* is anything except *j*, we assign object *i* to cluster *opt_cluster*. Then we calculate the fitness of the agent, if any improvement is made, we accept the new assignment; however, if the objective function is worse than before, we do not accept the new assignment.

### 2.2.2 Local Search A_2

This local search works the same as local search "A"; however, the threshold probability for this local search is smaller. The reason of such parameter tuning is computational time. This local search performs on agents with rank 2,..,*L* agents. These *L* best agents also update the pheromone trail matrix as will be described later. *L* is a parameter which must be the threshold probability value for this set to 0.01.

### 2.2.3 Local search B

In this local search, we intend to explore each cluster in order to find distant objects and attempt to assign them to another cluster. This local search is done on agents with rank one and two with a probability threshold of 0.03.

Again, we consider the *eval_cluster* matrix. As the first step, we try to sort values in each column. Indeed, we sort objects in each cluster due to their distance to the cluster center. Some values are equal to zero because each object is assigned to just one cluster. Assume that object *i* is chosen. Then again we utilize *opt_cluster*. If *opt_cluster* is not equal to the assigned cluster, we assign object *i* to *opt_cluster* and calculate the fitness of the agent. If the fitness of the agent is improved, we accept the new assignment; else we do not accept it.

### 2.2.4 Local Search C

This local search builds a matrix *list* of size $L \times N$. The matrix *list* is build as follows:

$$list(i,j) = \arg\{\min(eval\_cluster(i,:,j))\}$$
where *i=1,…L* and *j=1,…,N.* (7)

For each of the best *L* agents, we generate a random vector of size $1 \times N$. For each object, if the associated random number is equal or less than a threshold probability, which is set to 0.05, then we assign the object to a cluster randomly else we assign the object to the associated *list* matrix element. At last we calculate the objective function of the agents and verify if any improvements happened or not, if so changes are accepted else changes are not accepted. The proposed local searches are more likely to heuristic algorithms and their concepts and basis are on heuristic rules that we establish it.

## 2.3 Pheromone Matrix Update

At the end of each iteration, the pheromone matrix should be updated. This updating is based on the *L* best agents. This updating helps agents to correct their way. The pheromone trail matrix at the *t*-th iteration is updated by Equation (8).

$$\pi_{ij}(t+1)=(1-\rho)\pi_{ij}(t)+\sum_{l=1}^{L}\Delta\pi_{ij}^{l} \quad i=1,..,N \quad j=1,..,K \qquad (8)$$

Where $\rho$ is the steadfastness of trail and therefore $(1-\rho)$ is the evaporation rate. Higher value of $\rho$ means that the information achieved in the past is forgotten faster and $\Delta\pi_{ij}^{l}$ is computed as follows:

$$\Delta\pi_{ij}^{l}=\begin{cases} 1/F_l & \text{if cluster } j \text{ is assigned to object } i,l \\ 0 & \text{elsewhere.} \end{cases} \qquad (9)$$

Three main steps at iteration are as follows: (I) Generation of new *R* solutions via the last updated pheromone trail matrix; (II) Executing local search procedures; (III) Updating pheromone trail matrix.

The algorithm performs the above steps till a predefined number of iterations happen.

## 3 HYBRID GA- ACO METHOD

The main idea of applying a hybrid method is to benefit from two efficient algorithms, namely GA and ACO. In the traditional ACO algorithm, all the agents have one life and there is no concept of reproduction of agents and one agent just corrects its way by the means of the pheromone trail matrix. Although diversity of the path of an agent is made by the means of a threshold parameter, $q_0$; however, we believe that we can search the solution space more effectively by the means of the reproduction operators of GA, such as crossover and mutation.

This usage of GA operators must be done carefully so that the information is gathered by the agents through iterations will not be lost. This means that we must implement GA due to the ACO's objective function. In this way, we must define a life for each agent. This can happen in two ways: (1) define a constant life cycle for each agent and (2) define the life cycle due to a predefined function. In this paper, we choose the second way. The reason of such setting is that at the last generations most of the agents are the same and agents will not change their path because of high values of the accumulated pheromone trail matrix. But, if we apply a method i.e. genetic algorithm in order to explore the solution space then we may obtain slight improvements by the means of the objective function at the last generation that is desirable. The life function can be interpreted as cooling temperature in SA. The proposed hybrid method can be divided to three sections: (1) ACO algorithm; (2) implementing GA in ACO; and (3) resuming ACO.

### 3.1 ACO Phase

The proposed hybrid method starts with ACO as defined in Section 2. ACO continues till lives of the agents are reached. The life function should have high values in early generation and small values in latter generations. We implement several functions and analyze their behaviour, and we propose a function illustrated in Equation (10).

$$T_i=T_0e^{-Ai^2}, \quad A=\left(\frac{1}{N^2}\right)\times\ln\left(\frac{T_0}{T_N}\right) \qquad (10)$$

### 3.2 Implementing GA in ACO

#### 3.2.1 Ranking Agents

Agents are sorted due to their calculated objective functions in ACO. This fitness is handy in GA's selection phase and finds its elitist parents. Indeed agents of ACO are taken as chromosomes of GA.

#### 3.2.2 Selection

Selection is based on universal sampling method.

#### 3.2.3 Recombination

We implement a uniform order-based crossover for the given clustering problem. Suppose that we have *K* clusters and *N* objects. So *K* clusters are assigned where $(K \prec N)$ cannot be done by the original uniform order-based crossover. Then, we modify this crossover for the clustering problem. In this crossover, two parents (say $P_1$ and $P_2$) are randomly selected and a random binary template is generated, as shown in Fig. 2. Some of the genes for offspring $C_1$ are filled by taking the genes from parent $P_1$ where there is a "1" in the template. At this point, we have $C_1$ partially filled; however, it has some "gaps". Genes of parent $C_1$ in the positions corresponding to zeros in the template are taken and filled by generating a random number between 1,…,*k*. $C_2$ are filled in the same way.

#### 3.2.4 Replacement

An elitist method is chosen in our algorithm in order to provide convergence.

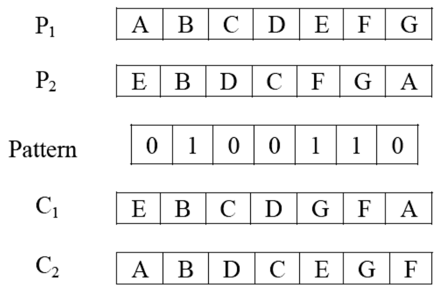

Figure 2: Modified uniform order-based crossover.

## 3.3 Resuming ACO

After performing GA, ACO must be resumed. Chromosomes inherited to the next generation must be treated as agents of ACO. At this point, we reach to a decision making point about continuing with current pheromone trail matrix or with a new one. We executed the algorithm for both situations and observed that resetting the matrix gives better answers than resuming with older one.

## 3.4 Parameters Tuning

Several simulations were performed to tune parameters of hybrid method. After several simulations the best parameters for the given clustering problem are gathered in Table 1.

## 4 RESULTS AND DISCUSSION

### 4.1 Introducing Datasets

Our proposed method is implemented on two well-known datasets, namely *iris* and *wine* in order to compare with the previous proposed algorithms. The data are obtained from the UCI repository of machine learning databases (Newman et al., 1998). All algorithms are executed in MATLAB and all experiments are performed on a PIV 2.66MHz and 512 MB RAM.

To evaluate the performance of our hybrid method, we compare it with several typical stochastic algorithms including ACO and SA (Selim and Al-Sultan, 1991), GA (Murthy and Chowdhury, 1996), and TS (Al-Sultan, 1995). The results from ACO, GA, TS, and SA are taken from Shelokar et al. (2004). The outputs for the *iris* dataset are given in Table 2. The objective function curve for the best solution of our hybrid method is shown in Fig. (3).

The results obtained for the clustering problem from the *wine* dataset are given in Table 3. The objective function curve for the best solution of our hybrid method during iterations is shown in Fig. (4).

Table 1: Parameters of the proposed hybrid method.

| *R* (ants) | $q_0$ | local search probability | Evaporation rate ($\rho$) | Iterations |
|---|---|---|---|---|
| 50 | 0.98 | 0.01-0.05 | 0.07 | 300 |

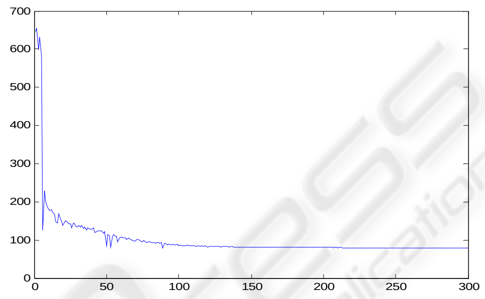

Figure 3: Illustration of the best solution for the *iris* dataset.

Table 2: Results of *iris* dataset.

| Algorithm | Function value | | | Time (sec) |
|---|---|---|---|---|
| | F best | F average | F worst | |
| Hybrid GA-ACO | 78.9408 | 81.9469 | 91.2163 | 83.60 |
| ACO | 97.1007 | 97.1715 | 97.8084 | 33.72 |
| GA | 113.9865 | 125.1970 | 139.7782 | 105.53 |
| TS | 97.3659 | 97.8680 | 98.5694 | 72.86 |
| SA | 97.1007 | 97.1364 | 97.2638 | 95.92 |

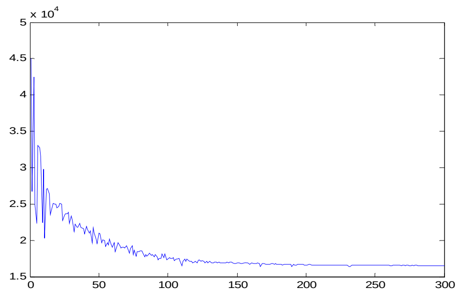

Figure 4: Illustration of the best solution for the wine dataset.

Table 3: Results of the wine dataset.

| Algorithm | Function value | | | Time (sec) |
|---|---|---|---|---|
| | F best | F average | F worst | |
| Hybrid GA-ACO | 16387.7595 | 16438.1623 | 16530.5338 | 216.01 |
| ACO | 16530.5338 | 16530.5338 | 16530.5338 | 68.29 |
| GA | 16530.5338 | 16530.5338 | 16530.5338 | 226.68 |
| TS | 16530.5338 | 16785.4592 | 16837.5356 | 161.45 |
| SA | 16530.5338 | 16530.5338 | 16530.5338 | 57.28 |

## 4.2 Paired Comparison Results

As the ACO solution is better than SA, GA and TS solutions, so our hybrid GA-ACO method is compared with ACO. Considering the data in Tables 2 and 3, two paired comparison designs are carried out on 10 pairs of the best solutions of ACO and hybrid GA-ACO for each dataset to see whether if any improvements achieved. The related results of the paired comparison test are illustrated in Table 4. Design of the comparison test is as follows:

$X_i$: The *i*-th best solution of the hybrid GA-ACO method.

$Y_i$: The *i*-th best solution of the ACO method.

$$D_i = X_i - Y_i \qquad i = 1,2,\ldots,n \Rightarrow \overline{D} = \frac{\sum_{i=1}^{n} D_i}{n} \tag{11}$$

$$S_D = \sqrt{\frac{\sum_{i=1}^{n}\left(D_i - \overline{D}\right)^2}{n-1}} \Rightarrow T = \frac{\overline{D}}{S_D \big/ \sqrt{n}} \tag{12}$$

$$\begin{vmatrix} H_0 : \mu_D = 0 \\ H_1 : \mu_D \neq 0 \end{vmatrix}$$

If $T \in [-t_{\alpha/2,n-1}, t_{\alpha/2,n-1}]$, $H_0$ is accepted.

Table 4: Results of Paired comparison test.

| | $\overline{D}$ | $S_D$ | $n$ | Test statistic | *Acceptance Interval* | Result |
|---|---|---|---|---|---|---|
| *iris* | -20.016 | 7.834 | 10 | -8.079 | [-2.262,2.262] | Reject $H_0$ |
| wine | -692.776 | 319.53 | 10 | -6.856 | [-2.262,2.262] | Reject $H_0$ |

By considering the test statistic values from Table 4, it is concluded that with $\alpha = 0.05$ ($\alpha$ : Level of significance), the hybrid GA-ACO method has a meaningful difference with the ACO algorithm in 10 independent runs.

It is worthy noting that the results obtained by the hybrid method are superior to that of the ACO, SA, GA, and TS methods. The associated results illustrate that our proposed hybrid approach can be considered as a viable and an efficient method to find optimal or near-optimal solutions for clustering problems in order to allocate *N* objects to *K* clusters.

## 5 CONCLUSIONS

In this paper, a hybrid GA-ACO method has been developed to solve clustering problems. To evaluate the performance of our hybrid GA-ACO method, the associated results are compared with other results obtained by other meta-heuristic algorithms, i.e. ACO, GA, SA, and TS by means of the paired test statistics. Our proposed hybrid method has been implemented and tested on several real datasets; preliminary computational experience is very encouraging in terms of the solution quality found and in all cases the best dominant solution is obtained by our proposed hybrid method and even the average and the worst solutions of this method is better than or equal to best solutions of the other algorithms. Although this method needs longer time to find solutions but the difference between the solutions of our hybrid GA-ACO method and other algorithms is totally encouraging and glamorous.

## REFERENCES

Al-Sultan, K.S., 1995. A Tabu search approach to the clustering problem, Pattern Recogn. 28 (9) 1443-1451.

Banfield, J., Raftery, A., 1993. Model-based Gaussian and non- Gaussian clustering, Biometrics, 49, 803-821.

Dorigo, M., Maniezzo, V. Colorni, A., 1996. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B* 26 (1) 29-41.

Jiang, J.-H., Wang, J.H., Chu, X., Yu, R.-Q., 1997. Clustering data using a modified integer genetic algorithm (IGA). Analytica Chimica Acta 354, 263-274.

Lin, H-J., Yang, F-W., Kao, Y-T., 2005. An efficient GA-based clustering technique, Tamkang Journal of Science and Engineering 8 (2) 113-122.

Murthy, C.A., Chowdhury, N., 1996. In search of optimal clusters using genetic algorithms. Pattern Recognition Letters. 17 (8) 825- 832.

Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J. 1998. UCI Repository of machine learning databases,

Department of Information and Computer Science, University of California, Irvine, CA, http://www.ics.uci.edu/~mlearn/MLRepository.html.
Paterlini, S., Krink, T., 2006. Differential evolution and particle swarm optimisation in partitional clustering, Computational Statistics and Data Analysis 50 (5) 1220-1247.
Selim, S.Z., Al-Sultan, K.S., 1991. A simulated annealing algorithm for the clustering problem. Pattern Recognition 24 (10) 1003-1008.
Shelokar, P.S., Jayaraman, V.K., Kulkarni, B.D., 2004. An ant colony approach for clustering, Analytica Chimica Acta 509, 187-195.
Sun, L.-X., Xie, Y.-L., Song, X.-H., Wang, J.-H. Yu, R.-Q., 1994. Cluster analysis by simulated annealing. Computers & Chemistry 18 (2) 103-108.