Empirical Comparison of Fast Clustering Algorithms for Large Data Sets

Chih-Ping Wei, Yen-Hsien Lee and Che-Ming Hsu
Department of Information Management
National Sun Yat-Sen University
Kaohsiung, Taiwan, R.O.C.
e-mail: cwei@mis.nsysu.edu.tw

Abstract

Several fast algorithms for clustering very large data sets have been proposed in the literature. CLARA is a combination of a sampling procedure and the classical PAM algorithm, while CLARANS adopts a serial randomized search strategy to find the optimal set of medoids. GAC-R^3 and GAC-RAR_w exploit genetic search heuristics for solving clustering problems. In this research, we conducted an empirical comparison of these four clustering algorithms over a wide range of data characteristics. According to the experimental results, CLARANS outperforms its counterparts both in clustering quality and execution time when the number of clusters increases, clusters are more closely related, more asymmetric clusters are present, or more random objects exist in the data set. With a specific number of clusters, CLARA can efficiently achieve satisfactory clustering quality when the data size is larger, whereas GAC-R^3 and GAC-RAR_w can achieve satisfactory clustering quality and efficiency when the data size is small, the number of clusters is small, and clusters are more distinct or symmetric.

1. Introduction

With the rapid growth in size and number of available databases, mining knowledge, regularities or high-level information from data becomes essential to support decision-making and predict future behavior [2, 3, 4, 6]. Data mining techniques can be classified into the following categories: classification, clustering, association rules, sequential patterns, time-series patterns, link analysis and text mining [2, 4, 15]. Due to its undirected nature, clustering is often the best technique to adopt first when a large, complex data set with many variables and many internal structures are encountered.

Clustering is a process whereby a set of objects is divided into several clusters in which each of the members is in some way similar and is different from the members of other clusters [9, 11, 12, 14, 16]. The most distinct characteristics of clustering analysis is that it often encounters very large data sets, containing millions of objects described by tens or even hundreds of attributes of various types (e.g., interval-scaled, binary, ordinal, categorical, etc.). This requires that a clustering algorithm be scalable and capable of handling different attribute types. However, most classical clustering algorithms either can handle various attribute types but are not efficient when clustering large data sets (e.g., the PAM algorithm [9, 12]) or can handle large data sets efficiently but are limited to interval-scaled attributes (e.g., the k-means algorithm [1, 2, 8, 14]).

Several fast clustering algorithms have been proposed in the literature, including CLARA [9], CLARANS [12], and genetic algorithm based clustering methods (called GAC-R^3 and GAC-RAR_w in this research) [5]. CLARA is a combination of a sampling approach and the PAM algorithm. Instead of finding medoids, each of which is the most centrally located object in a cluster, for the entire data set, CLARA draws a sample from the data set and uses the PAM algorithm to select an optimal set of medoids from the sample. To alleviate sampling bias, CLARA repeats the sampling and clustering process multiple times and, subsequently, selects the best set of medoids as the final clustering. On the other hand, CLARANS views the process of finding optimal medoids as searching through a certain graph, in which each node represents a set of medoids. Two nodes are neighbors if their sets differ by only one object. Instead of using an exhaustive search strategy, CLARANS adopts serial randomized search. That is, starting from an arbitrary node in the graph, CLARANS randomly checks one of its neighbors. If the neighbor results in a better clustering, CLARANS proceeds to this neighbor; otherwise, CLARANS randomly checks another neighbor until a better neighbor is found or a pre-determined maximal number of neighbors is reached. To avoid being trapped in a suboptimal solution, CLARANS repeatedly starts from different initial nodes and selects the best node as the final clustering. Finally, GAC-R^3 and GAC-RAR_w employ a genetic algorithm to search for the optimal set of medoids. The genetic algorithm adopts a probabilistic, parallel-randomized-search strategy similar to biological

evolution. It starts with a random fixed-sized population of chromosomes, each of which encodes a possible set of medoids. Iteratively, a new generation of the same size is generated by randomly applying genetic operators (including reproduction, crossover and mutation) on probabilistically selected parent chromosomes based on their fitness as a solution. The process terminates as soon as there is no further improvement over generations or after a pre-defined maximal number of generations has been reached. The fittest chromosome of the last generation or among all generations is then selected as the final clustering.

The search strategies employed by the above-mentioned clustering algorithms are fundamentally different. Therefore, their performance in terms of clustering quality and execution time for clustering data sets with different characteristics may vary. Thus, the purpose of this research was to conduct an empirical evaluation of the above-mentioned clustering algorithms. The evaluation included the effects of data size, number of clusters, degree of cluster distinctness, degree of cluster asymmetry, and level of data randomness on the clustering quality and execution time of the target clustering algorithms. The remainder of the paper is organized as follows. Section 2 reviews the target fast-clustering algorithms. Section 3 details the design of empirical experiments, including synthetic data set generation procedure, evaluation criteria, experimental procedure, and parameter tuning experiments. A set of experiments based on synthetic data sets with pre-specified data characteristics was conducted and the results are summarized in Section 4. Finally, research contributions as well as future directions will be presented in Section 5.

2. Review of target clustering algorithms

This section reviews the four fast clustering algorithms included in this study, including CLARA, CLARANS, GAC-R^3, and GAC-RAR_w. To facilitate subsequent discussion, the main symbols used through the paper and their definitions are summarized in Table 1.

Table 1: Summary of symbols and definitions

Symbols	Definitions
D	Data set to be clustered
n	Number of objects in D
O_i	Object i in D
k	Number of clusters
S	A sample of D
s	Size of S

2.1. CLARA algorithm

CLARA (Clustering LARge Applications) relies on the sampling approach to handle large data sets [9]. Instead of finding medoids for the entire data set, CLARA draws a small sample from the data set and applies the PAM algorithm to generate an optimal set of medoids for the sample. The quality of resulting medoids is measured by the average dissimilarity between every object in the entire data set D and the medoid of its cluster, defined as the following cost function:

$$\mathrm{Cost}(M,D) = \frac{\sum_{i=1}^{n} \mathrm{dissimilarity}(O_i, \mathrm{rep}(M, O_i))}{n}$$

where M is a set of selected medoids, dissimilarity(O_i, O_j) is the dissimilarity between objects O_i and O_j, and rep(M, O_i) returns a medoid in M which is closest to O_i.

To alleviate sampling bias, CLARA repeats the sampling and clustering process a pre-defined number of times and subsequently selects as the final clustering result the set of medoids with the minimal cost. Assume q to be the number of samplings. The CLARA algorithm is detailed in Figure 1.

```
Set mincost to a large number;
Repeat q times
    Create S by drawing s objects randomly from D;
    Generate the set of medoids M from S by applying
    the PAM algorithm;
    If Cost(M, D) <mincost
    Then
        mincost = Cost(M, D);
        bestset = C;
    End-if;
End-repeat;
Return bestset;
```

Figure 1: CLARA algorithm

Since CLARA adopts a sampling approach, the quality of its clustering results depends greatly on the size of the sample. When the sample size is small, CLARA's efficiency in clustering large data sets comes at the cost of clustering quality.

2.2. CLARANS algorithm

CLARANS (Clustering Large Applications based on RANdomized Search) views the process of finding k medoids as searching in a graph [12]. In this graph, a node is represented by a set of k objects $\{O_1, \ldots, O_k\}$, indicating that $O_1, \ldots, O_k$ are the selected medoids. Two nodes are neighbors (i.e., connected by an arc) if their sets differ by

only one object. It is evident that each node in the graph has $k(n-k)$ neighbors. Since a node represents a collection of k medoids, each node corresponds to a possible clustering whose quality is measured by the cost function defined in (1).

Instead of using an exhaustive search strategy, CLARANS adopts serial randomized search. That is, CLARANS starts from an arbitrary node in the graph and randomly selects one of its neighbors. If the cost of the selected neighbor is less than that of the current node, CLARANS proceeds to this neighbor and continues the neighbor selection and comparison process. Otherwise, CLARANS randomly checks another neighbor until a better neighbor is found or the pre-determined maximal number of neighbors to check has been reached. In the latter case, the current node is declared to be a "local" minimum. To avoid being trapped in a suboptimal solution, CLARANS repeats the process of searching the local minimum from a different initial node for a pre-determined number of times. Subsequently, the node with the minimal cost is selected as the final clustering. Let maxneighbor be the maximal number of neighbors to examine and numlocal be the number of local minima obtained. The detailed CLARANS algorithm is depicted in Figure 2.

```
Set mincost to a large number;
For i = 1 to numlocal do
    Randomly select a node as the current node C in the
    graph;
    Initialize j to 1;
    Repeat
        Randomly select a neighbor N of C;
        If Cost(N, D) < Cost(C, D) Then
            Assign N as the current node C;
            Reset j to 1;
        Else increment j by 1;
        End-if;
    Until j > maxneighbor;
    If Cost(C, D) < mincost Then
        mincost = Cost(C, D);
        bestnode = C;
    End-if;
End-for;
Return bestnode;
```

Figure 2: CLARANS algorithm

The performance of CLARANS was shown to be greatly influenced by the maximal number of neighbors and the number of local minima obtained. If the two parameters were large, the optimal clustering was more likely to be achieved at the cost of efficiency. On the other hand, CLARAN could be very efficient if these two parameters were small. However, the optimal (or near optimal) clustering result could not be guaranteed.

2.3. Genetic algorithm based clustering methods

Estivill-Castro and Murray [5] proposed clustering methods using genetic algorithms [10] for solving the medoid-based formulation of clustering. In their implementation, a chromosome (a possible solution) was implemented as an array of k different objects, each of which denoted a selected medoid. Two crossover operators, namely Random Respectful Recombination (R^3) and Random Assorting Recombination (RAR_w), were proposed. Given two parent chromosomes M_1 and M_2 of size k, R^3 included in the offspring chromosome M_0 all objects in $M_1 \cap M_2$. The remaining $k - |M_1 \cap M_2|$ objects were randomly selected from $(M_1 \cup M_2) - (M_1 \cap M_2)$. On the other hand, RAR_w iteratively added objects into the offspring chromosome M_0. At each iteration, a random number ρ was drawn uniformly in [0, 1]. If $\rho \leq$ cut (a pre-defined threshold), an object was selected randomly and uniformly from $(M_1 \cap M_2) - M_0$ and included into M_0. If $\rho >$ cut, an object was selected randomly and uniformly from $(M_1 \cup M_2) - [(M_1 \cap M_2) \cup M_0]$ and included into M_0. If $(M_1 \cap M_2) - M_0$ or $(M_1 \cup M_2) - [(M_1 \cap M_2) \cup M_0]$ was empty before M_0 had k objects, M_0 was completed with random objects from $(M_1 \cup M_2) - M_0$. In this study, these two genetic algorithm based clustering methods are called GAC-R^3 and GAC-RAR_w depending on which crossover operator was employed.

Other details of GAC-R^3 and GAC-RAR_w are as follows. The mutation operator occasionally modified a chromosome M by randomly swapping an object in M (i.e., a selected medoid) with an object in $D - M$ (i.e., a non-medoid object). The reproduction operator simply replicated a chromosome in the parent generation into its offspring generation. The fitness function employed in this study was the inverse of the cost function defined previously. That is, Fitness(M, D) = 1/Cost(M, D). The termination criterion was defined as convergence on fitness of the fittest chromosomes over generations or the pre-defined maximal number of generations. The fittest chromosome among all generations was selected as the clustering result. Assuming f to be the population size, rr to be the reproduction rate, rc to be the crossover rate where $rr + rc = 1$, and rm to be the mutation rate, the details of GAC-R^3 and GAC-RAR_w algorithms are depicted in Figure 3.

Randomly create an initial population P of f chromosomes $\{M_1, M_2, \ldots, M_f\}$;
Repeat
  Repeat
    Generate a random number γ;
    If $\gamma > rc$ Then
      Select a chromosome M_i from P by roulette proportional to relative fitness;
      Insert M_i into the new generation;
    Else
      Select M_i and M_j from P by roulette proportional to relative fitness;
      Create an offspring by applying R^3 or RAR_w crossover operator on M_i and M_j;
      Insert the offspring into the new generation;
    End-if;
  Until the new generation has f chromosomes;
  For every object of each chromosome in the new generation do
    Generate a random number λ;
    If $\lambda \le rm$ Then
      Perform mutation operator on the object;
    End-if;
  End-for;
  Set the new generation as the current generation P;
Until the termination criterion is satisfied;

Figure 3: GAC-R^3 and GAC-RAR_w algorithms

3. Design of empirical experiments

This section details the design of the empirical experiments for evaluating the target clustering algorithms. It includes the procedure of generating synthetic data sets, evaluation criteria, experimental procedure, and parameter tuning experiments.

3.1. Generation of synthetic data

Since the performance of the target clustering algorithms was evaluated over a range of data characteristics described by data size, number of clusters, cluster distinctness, cluster asymmetry, and data randomness, finding real-world data sets for our evaluation would have been extremely difficult, if not impossible. Thus, synthetic data sets were generated and employed for the evaluation.

Without loss of generality, we assumed that objects in experimental data sets could be described by two interval-scaled attributes, each of which ranged from 0 to 1000 (i.e., objects in each data set were generated in [0, 1000] × [0, 1000]). We assumed all clusters in a data set to have the same radius, defined as $rs = \sqrt{n/(k * ds * \pi)}$ where ds was the density of an area (i.e., the maximal number of objects allowed in an area divided by the total number of feasible objects in the area). The k objects $\vec{C}_1$, $\vec{C}_2$, …, and $\vec{C}_k$ first were selected randomly as medoids of the k clusters, any two of which had to satisfy the constraint of $t-0.05 \le rs/d \le t+0.05$ where d was the distance between the two medoids and t was the degree of cluster distinctness ($0<t<1$). The number of objects in each cluster was determined as follows. Clusters were assumed to be randomly and evenly divided into two groups with all clusters in the same group having the same cluster size. The ratio between the cluster size from one group and the cluster size from another group was defined as the degree of cluster asymmetry (denoted as a, where $a \ge 1$). When $a = 1$, all clusters were said to be of the same size. If $a > 1$, the size of a larger cluster was a times of that of a smaller cluster. In addition, the data set was assumed to contain $r*n$ (where $r< 1$) random objects. Thus, the number of objects in a smaller cluster was $\left(\frac{n(1-r)}{\lceil \frac{k}{2} \rceil + a \lfloor \frac{k}{2} \rfloor} \right)$, while that in a larger cluster was $\left(\frac{n(1-r)}{\lceil \frac{k}{2} \rceil + a \lfloor \frac{k}{2} \rfloor} \right) \times a$.

Subsequently, an object in a cluster with the medoids $\vec{C}_i$ was generated by first randomly selecting the polar coordinates γ_j, θ_j (where γ_j was uniform in [0, r] and θ_j was uniform in [0, 2π]). This object in the data set was then $\vec{C}_i + (\gamma_j \sin\theta_j, \gamma_j \cos\theta_j)$. After generating all non-random objects for all clusters, $r*n$ random objects then were randomly generated. To reduce the effect of random objects on clustering results, random objects were limited within the area of $[x_L, x_H] \times [y_L, y_H]$ where x_L and x_H (or y_L and y_H) were the minimal value and the maximal value in the first (or second) attribute among all the non-random objects.

The parameters for the synthetic data generation program are summarized in Table 2. The default values of these parameters for synthetic data sets were n=3000, k=20, t=0.2, a=1, and r=1%. Depending on the type of experiment conducted, the respective parameter was varied, while the rest of parameters adopted their default values. For example, to evaluate the data size effect on the target clustering algorithms, the parameter n took values from 500 to 7000, while default values were used for the

rest of parameters.

Table 2: Parameters and default values for synthetic data generation

Symbols	Meanings	Default
n	Number of objects in a data set	3000
k	Number of clusters	20
t	Degree of cluster distinctness	0.2
a	Degree of cluster asymmetry	1
r	Level of data randomness	1%

3.2. Evaluation criteria and procedure

In addition to execution time, the effectiveness of the target fast clustering algorithms was evaluated. To cluster a data set D into k clusters, any of the clustering algorithms described generates a set of k medoids. As mentioned, the quality of medoids which is measured by the average dissimilarity between every object in D and the medoid of its cluster is adopted by the clustering algorithms as the cost function for selecting the appropriate set of medoids. However, when adopted to evaluate the effectiveness of the clustering algorithms, this measure may not be appropriate because two non-identical sets of medoids may yield the same clustering but may have different average dissimilarity between every object in D and its respective medoid. In this research, we employed the silhouette measure [9], as the effectiveness measure of the clustering algorithms. Assume the cluster to which object i is assigned be A. Let $a(i)$ = average dissimilarity of i to all other objects in cluster A. For any cluster C different from A, let $d(i, C)$ be the average dissimilarity of object i in the cluster A to all objects in C. After computing $d(i, C)$ for all clusters C, the smallest among them is selected and denoted as

$$b(i) = \min_{C \neq A} d(i, c)$$

The cluster B for which this minimum is attained (i.e., $d(i, B) = b(i)$) is called the neighbor of object i. In fact, the cluster B can be viewed as the second-best choice for object i. The silhouette of object i, $s(i)$, is then obtained as follows:

$$s(i) = \begin{cases} 1 - a(i)/b(i) & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \text{ or object } i \text{ is the only object in cluster A} \\ a(i)/b(i) - 1 & \text{if } a(i) > b(i) \end{cases}$$

$s(i)$ measures how well the object i matches the cluster to which the object is currently assigned. Apparently, $-1 \leq s(i) \leq 1$. When $s(i)$ is close to 1, this implies that the "within" dissimilarity $a(i)$ is much smaller than the smallest "between" dissimilarity $b(i)$. Therefore, the object i can be regarded as "well classified." When $s(i)$ is close to -1 (i.e., $a(i)$ is much larger than $b(i)$), it would have seemed much more natural to assign the object i to cluster B. The average of $s(i)$ for all objects in the data set is called the *silhouette width* of the data set under a given set of k medoids and denoted by $\bar{s}(k)$. $\bar{s}(k)$ measures the degree of cluster separation and has the property that two non-identical sets of medoids yielding the same clustering result will have the same silhouette width. Thus, it was adopted as the measure for evaluating the effectiveness of the clustering algorithms.

All experiments in this study were conducted on an IBM SP2 machine with a CPU clock rate of 120 MHz, 2 GB of main memory and running AIX 4.1. To obtain unbiased estimates, each experiment was performed 30 times and an overall performance estimate was calculated by averaging the performance of the 30 individual runs.

3.3. Parameter tuning

Parameter tuning is concerned with selecting for each clustering algorithm appropriate parameter values, by considering the two evaluation criteria described above. The synthetic data set for parameter tuning experiments was generated by using default values (i.e., n=3000, k=20, t=0.2, a=1, and r=1%).

Sample size and the number of samplings are two parameters of CLARA. As indicated in [9, 12], setting sample size as $40+2k$ and the number of samplings as 5 would give satisfactory clustering performance. Using the number of samplings suggested, the performance of the clustering result produced by CLARA over different sample sizes is shown in Figure 4. When the sample size grew from $40+0.5k$ to $40+2.5k$, the silhouette width improved progressively. The silhouette width flattened out when the sample size was over $40+2.5k$. On the other hand, when the sample size increased, the execution time increased exponentially. Considering the tradeoff between clustering quality and execution time, we decided on the sample size of $40+3k$ for CLARA.

The maximal number of neighbors (denoted as maxneighbor) and the number of local minima obtained (denoted as numlocal) are two parameters of CLARANS. According to a study by Ng and Han [12], $1.25\% * k(n-k)$ for maxneighbor and 2 for numlocal would be appropriate. As shown in Figure 5, increasing maxneighbor or numlocal did not improve the clustering quality. On the other hand, given a specific numlocal, increasing maxneighbor from $1\% * k(n-k)$ to $1.35\% * k(n-k)$ increased the execution time only slightly. However, the effect of numlocal on the execution time was more significant. Thus, we selected $1.25\% * k(n-k)$ for maxneighbor and 1 for numlocal.

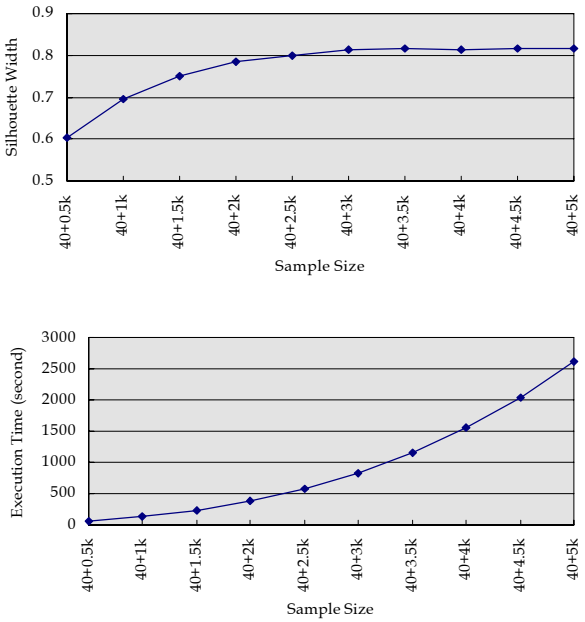

Figure 4: Parameter tuning for CLARA

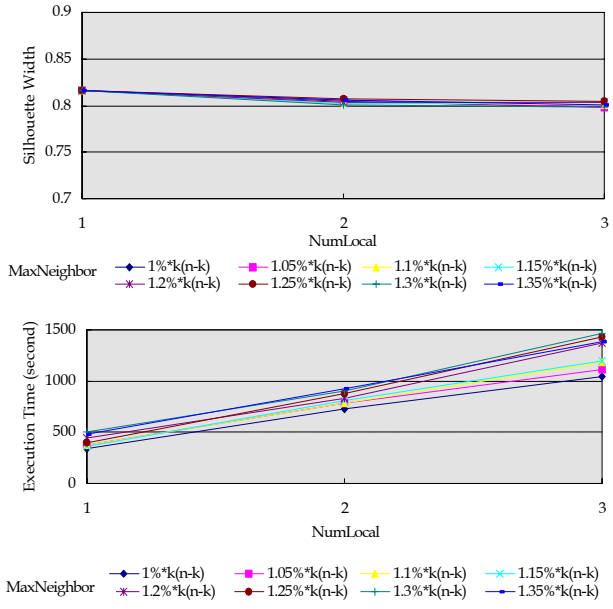

Figure 5: Parameter tuning for CLARANS

GAC-R^3 involves four major parameters: population size, maximal number of generations, crossover rate and mutation rate. To reduce the extent of experiments, we decided on 50 as the population size (i.e., 50 chromosomes per generation) and 500 as the maximal number of generations. Accordingly, the performance of GAC-R^3 over different crossover and mutation rate is shown in Figure 6. When the mutation rate was above 0.01, the resulting clustering quality decreased significantly and, at the same time, the resulting execution time increased significantly. This poor performance can be attributed to higher mutation rates over-exploiting the search space that resulted in the search process terminating at the maximal number of generations without finding an optimal (or near optimal) solution. When the mutation rate was less than or equal to 0.01, increasing the crossover rate generally resulted in improved clustering quality but increased execution time. Accordingly, we decided on the crossover rate of 0.8 and the mutation rate of 0.008 for GAC-R^3.

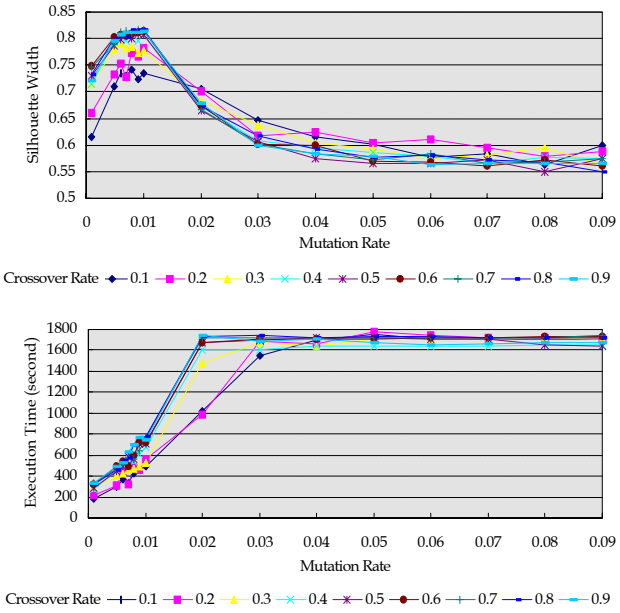

Figure 6: Parameter tuning for GAC-R^3

Similarly, we decided on 50 as the population size and 500 as the maximal number of generations for GAC-RAR$_w$. Accordingly, the performance of GAC-RAR$_w$ over different cut, crossover rate and mutation rate is shown in Figure 7. When cut increased, the clustering quality improved at any level of crossover and mutation rate investigated. On the other hand, the execution time with cut between 0.1 and 0.8 varied marginally. As cut increased to 0.9, the execution time decreased significantly because the over-exploiting problem was avoided. At cut 0.9, increasing the crossover rate from 0.7 to 0.9 or the mutation rate from 0.007 to 0.009 yielded similar clustering quality. Conversely, as the crossover or mutation rate increased, the execution time increased as well. Based on the experimental results, cut 0.9, crossover rate 0.8 and mutation rate 0.008 were selected for GAC-RAR$_w$.

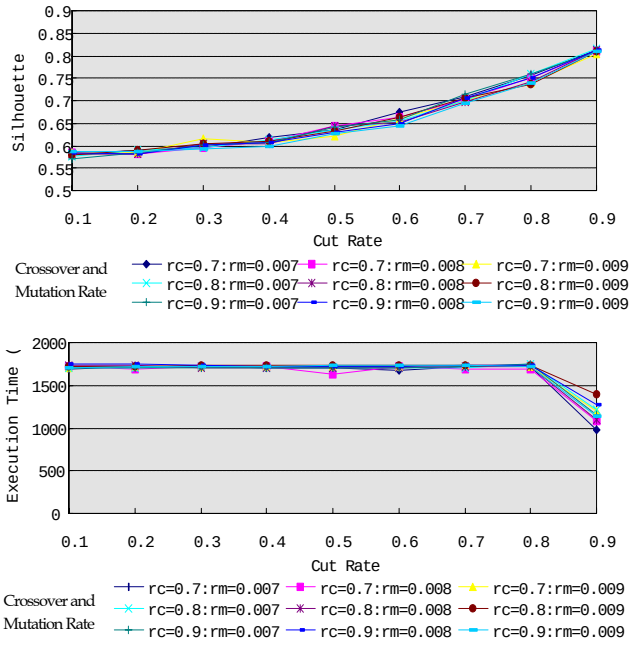

Figure 7: Parameter tuning for GAC-RAR$_w$

4. Experimental results

Based on the parameters experimentally determined, the effect of data size, number of clusters, degree of cluster distinctness, degree of cluster asymmetry and level of data randomness on the execution time and clustering quality of the four fast clustering algorithms were evaluated empirically.

4.1. Effect of data size

Synthetic data sets were generated for various data sizes, ranging from 500 to 7,000 (i.e., n=500, 1,000, 2,000, …, 7,000). Remaining parameters received their default values, as defined in Table 2. Figure 8 shows the performance of the target clustering algorithms as a function of the data size.

As shown in Figure 8, GAC-R^3 and CLARA slightly outperformed the others in terms of clustering quality when given only a small data size (i.e., fewer than 1,000). When the data size was increased, the clustering quality of GAC-R^3 degraded as compared to that of others. CLARANS, and GAC-RAR$_w$ generally achieved compatible clustering quality at any level of data size. The clustering quality of CLARA was similar to that of GAC-R^3 when the data size was less than 3,000 and similar to that of CLARANS, and GAC-RAR$_w$ when the data size was more than 3,000. In terms of execution time, CLARANS, GAC-R^3, and GAC-RAR$_w$ were more efficient than CLARA when the data size was fewer than 2,000. As the data size increased, CLARA increased its execution time slightly and, eventually, outperformed others. The insensitivity of CLARA to data size is attributed to its execution time's being greatly influenced by a sample size that is a function of k rather than n. On the other hand, as the data size was increased, the execution time of CLARANS and GAC-RAR$_w$ increased more rapidly than that of GAC-R^3 which, in turn, increased more rapidly than that of CLARA.

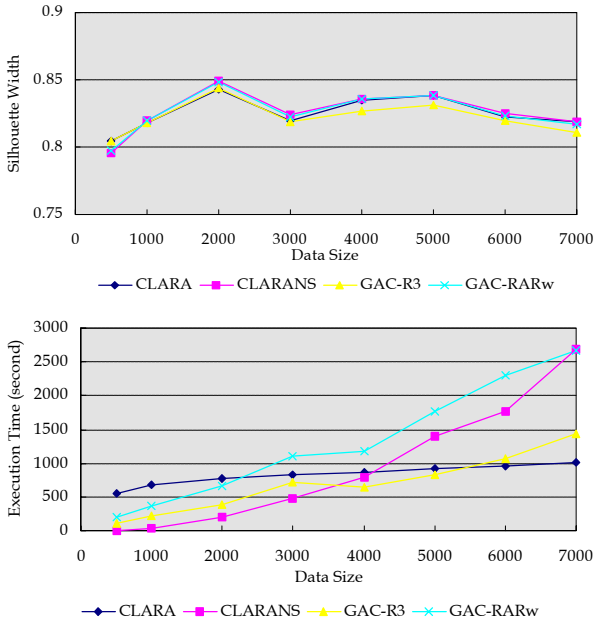

Figure 8: Effect of data size

4.2. Effect of number of clusters

Synthetic data sets were generated for various numbers of clusters (k), ranging from 2, 3, 4, 5, 10, 20, 30, to 40. As shown in Figure 9, all the algorithms reached similar clustering quality when the number of clusters was 20 or fewer. When the number of clusters grew from 20 to 40, GAC-R^3 and GAC-RAR$_w$ were inferior to others in clustering quality, while CLARANS achieved the best clustering quality. On the other hand, when the number of clusters was 10 or fewer, all the algorithms' execution times were comparable. When the number of clusters increased, the execution time of CLARA increased significantly due to its sensitivity to the number of clusters, and the execution time of CLARANS increased at a greater speed than that of GAC-RAR$_w$ and GAC-R^3. It can be expected that when the number of clusters keeps growing, CLARANS still can achieve the best clustering quality, but requires longer execution time than GAC-RAR$_w$ or GAC-R^3.

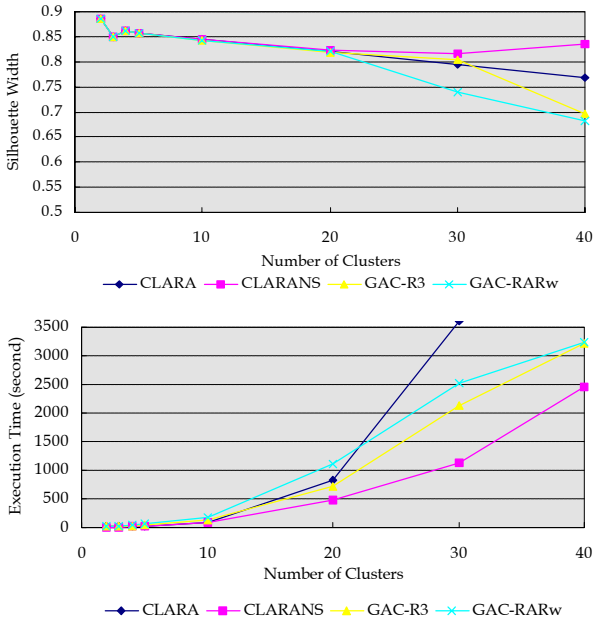

Figure 9: Effect of number of clusters

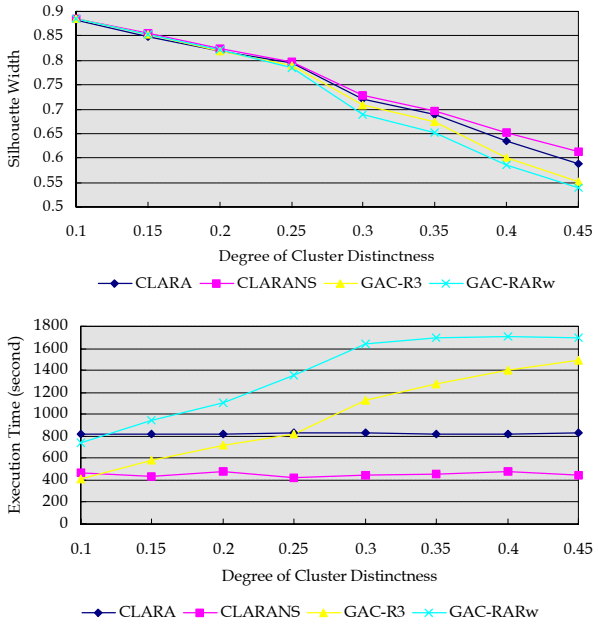

Figure 10: Effect of degree of cluster distinctness

4.3. Effect of degree of cluster distinctness

Synthetic data sets were generated for various degrees of cluster distinctness (t). Smaller t indicated that clusters in the data set were more separated. As shown in Figure 10, the degree of cluster distinctness had no or little impact on the target clustering algorithms' clustering quality. When $t \leq 0.25$, all the algorithms arrived at almost identical clustering quality. As t increased, CLARANS performed slightly better while GAC-RAR_w performed slightly worse than others in clustering quality. The execution time of CLARA or CLARANS remained constant with varying the degree of cluster distinctness. However, the increment of t resulted in increasing execution time of GAC-R^3 and GAC-RAR_w. A plausible explanation is that when clusters come closer together (i.e., t increases) the possibility of creating a set of chromosomes in a generation with similar fit will be higher and their change to survive in the next generation will be alike. Subsequently, the resulting convergence speed will decline and the execution will eventually terminate at the pre-defined maximal number of generations. Among all clustering algorithms, CLARANS was fastest, GAC-RAR_w was the slowest.

4.4. Effect of degree of cluster asymmetry

Synthetic data sets were generated for various degree of cluster asymmetry (a), ranging from 1, 3, …, 17, to 19. A small a denoted that the sizes of clusters in a data set were more alike. As shown in Figure 11, the degree of cluster asymmetry had little impact on the clustering quality of CLARANS. However, as clusters became more asymmetric, the resulting clustering quality of the other three algorithms degraded and became unsatisfactory. Similar to the empirical results of cluster distinctness experiments, the execution time of CLARA or CLARANS remained constant with varying degrees of cluster asymmetry. Moreover, the increment of a resulted in increasing execution time for GAC-R^3 and GAC-RAR_w more quickly than in the cluster distinctness experiments. An explanation is that, as clusters became more asymmetric, objects from larger clusters were more likely to be included in the chromosomes of a generation. Any mutation operation is more likely to create a chromosome with a higher propensity to fit. Hence, the resulting convergence speed will decline and the execution will eventually terminate at the pre-defined maximal number of generations. The search process of GAC-R^3 and GAC-RAR_w became divergent more quickly than in the clustering distinctness experiments. Among all the clustering algorithms, CLARANS was most efficient, while GAC-RAR_w and GAC-R^3 were the least.

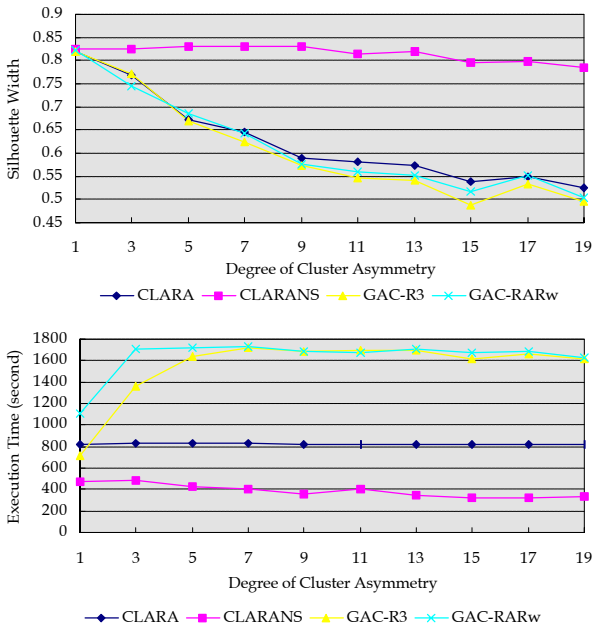

Figure 11: Effect of degree of cluster asymmetry

4.5. Effect of level of data randomness

Synthetic data sets were generated for various percentages of random objects (r), ranging from 1%, 2%, …, to 10%. Figure 12 shows the performance of the fast clustering algorithms as a function of the level of data randomness. As shown, the level of data randomness did not favor any clustering algorithm's clustering quality. However, when the data set had more random objects (i.e., r increased), the execution time of GAC-R^3 and GAC-RAR_w fluctuated more than that of the other two. In general, CLARANS was the most efficient algorithm, followed by GAC-R^3, CLARA and finally GAC-RAR_w.

4.6. Summary

Based on the fine-tuned parameters for each clustering algorithm and synthetic data sets generated, the experimental results for each target clustering algorithm are summarized as follows. Given a specific number of clusters, CLARA can achieve satisfactory clustering quality while maintaining its efficiency when data size increases. However, when the number of clusters increases, the execution time CLARA becomes intolerable. As a sampling approach, CLARA is less susceptible to degree of cluster distinctness, and level of data randomness, both in clustering quality and execution time. Its resulting clustering quality was seen to degrade significantly when the cluster sizes were more asymmetric.

CLARANS outperformed its counterparts both in clustering quality and execution time when the number of clusters increases, clusters are more closely related, more asymmetric clusters were present, or more random objects existed in the data set. CLARANS's superiority may be explained by the local maximum and plateau problem [13], which commonly occurs in a search space but is less often encountered in a clustering problem. Thus the serial randomized search strategy adopted by CLARANS appears to be appropriate.

GAC-R^3 and GAC-RAR_w achieved satisfactory clustering quality and efficiency when the data size was small (i.e., fewer than 4,000), the number of clusters was small (i.e., less than 20), and clusters were more distinct and symmetric. Moreover, both are very vulnerable to the degree of cluster distinctness and cluster asymmetry. Compared with CLARA, each of these genetic algorithm based clustering methods can achieve better performance when the number of clusters increases.

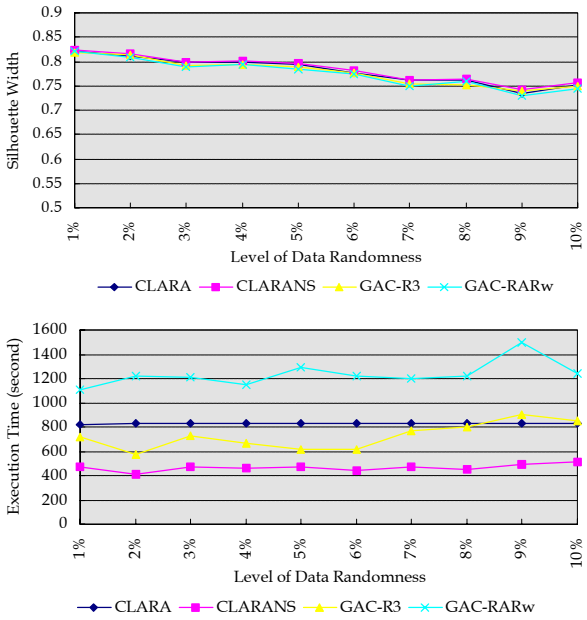

Figure 12: Effect of level of data randomness

5. Conclusions

The most distinct characteristic of clustering analysis is that it often encounters very large data sets, described by various attribute types. Thus, it is desirable that a clustering algorithm be scalable and capable of dealing with different attribute types. In this study, we conducted an empirical comparison of four clustering algorithms (i.e., CLARA, CLARANS, GAC-R^3, and GAC-RAR_w) over a wide range of data characteristics. According to the experimental results, CLARANS outperforms its counterparts both in clustering quality and execution time when the number of clusters increases, clusters are more closely related, more asymmetric clusters are present, or more random objects exist in the data set. With a specific

number of clusters, CLARA can achieve satisfactory clustering quality while maintaining its efficiency when the data size gets larger, while GAC-R^3 and GAC-RAR_w can achieve satisfactory clustering quality and efficiency when the data size is small, the number of clusters is small, or clusters are more distinct or symmetric.

Some ongoing and future research directions are briefly discussed as follows. In this research, synthetic data sets were used to evaluate the performance of the target clustering algorithms. The decision to use synthetic data sets was based on being able to control experimental data sets with certain characteristics. However, it is essential to investigate the performance of the target clustering algorithms with real-world data sets. The clustering algorithms included in this research are medoid-based approaches. Other fast clustering algorithms of different approaches exist. For example, Huang has proposed k-modes and k-prototypes algorithms which belong to the centroid-based approach [7]. It would be highly desirable to empirically compare the performance of fast clustering algorithms of different approaches.

References

[1] M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, Inc., 1973.

[2] M. J. A. Berry and G. Linoff, *Data Mining Techniques: For Marketing, Sales, and Customer Support*, John Wiley & Sons, Inc., New York, NY, 1997.

[3] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi, *Discovering Data Mining: From Concept to Implementation*, Prentice Hall PTR, Upper Saddle River, NJ, 1998.

[4] M. S. Chen, J. Han, and P. S. Yu, "Data Mining: An Overview from a Database Perspective," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, 1997.

[5] V. Estivill-Castro and A. T. Murray, "Spatial Clustering for Data Mining with Generic Algorithms," Technical Report FIT-TR-97-10, Queensland University of Technology, Faculty of Information Management, September 1997.

[6] W. Frawley, G. Piatetsky-Shapiro, and C. Matheus, "Knowledge Discovery in Databases: An Overview," *AI Magazine*, Fall 1992, pp.213–228.

[7] Z. Huang, "Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values," *Data Mining and Knowledge Discovery*, Vol. 2, No. 3, 1998, pp.283–304.

[8] A. K. Jain, and R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Inc., 1988.

[9] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, Inc., New York, NY, 1990.

[10] J. R. Koza, *Genetic Programming II*, The MIT Press, Cambridge, MA, 1994.

[11] G. F. Luger and W. A. Stubblefield, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, The Benjamin/Cummings Publishing Company, Inc., 1993.

[12] R. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," *Proceedings of International Conference on Very Large Data Bases*, Santiago, Chile, Sept. 1994, pp.144–155.

[13] E. Rich and K. Knight, *Artificial Intelligence*, 2nd Ed., McGraw-Hill, Inc., New York, 1991.

[14] H. Spath, *Cluster Analysis Algorithms: For Data Reduction and Classification of Objects*, John Wiley & Sons, Inc., New York, 1980.

[15] C. P. Wei, P. Hu, and L. M. Kung, "Multiple-Level Clustering Analysis for Data Mining Applications," *Proceedings of the 4th INFORMS Joint Conference on Information Systems and Technology*, May 1999.

[16] M. Zait and H. Messatfa, "A Comparative Study of Clustering Methods," *Future Generation Computer System*, Vol. 13, 1997, pp.149–159.