Efficient K-medoids Algorithms Using Multi-Centroids With Multi-Runs Sampling Scheme

Shu-Chuan Chu[1], John F. Roddick[1], and J.S. Pan[2,3]

[1] School of Informatics and Engineering,
Flinders University of South Australia,
PO Box 2100, Adelaide 5001, South Australia.
Email: {jan, roddick}@cs.flinders.edu.au
[2] Department of Electronic and Electrical Engineering,
University of Adelaide,
Adelaide 5005, South Australia.
[3] Department of Electronic Engineering,
Kaohsiung University of Applied Sciences
415 Chien Kung Road,
Kaohsiung, Taiwan
Email: jspan@cc.kuas.edu.tw

Abstract. Clustering in data mining is used to group similar objects based on their distance, connectivity, relative density, or some specific characteristics. The k-medoids-based algorithms have been shown to be effective to spherical-shaped clusters with outliers. However, they are not efficient for large database. In this paper, we propose a novel *Multi-Centroids with Multi-Runs Sampling Scheme* $MCMRS$ to improve the performance of many k-medoids-based algorithms, including PAM, $CLARA$ and $CLARANS$. $MCMRS$ is also further improved by combining with the $CLASA$ algorithm presented in earlier work. Experimental results demonstrate the efficiency of the proposed $MCMRS$ and $MCMRS$-$CLASA$ algorithms.

Keywords: k-medoids, PAM, $CLARA$, $CLARANS$, $CLASA$, $MCMRS$.

1 Introduction

Clustering is a form of classification imposed over a finite set of objects. The goal of clustering is to group sets of objects into classes such that similar objects are placed in the same cluster while dissimilar objects are in separate clusters. Clustering (or classification) is a common form of data mining [1, 2] and has been applied in many fields including data compression [3], texture segmentation [4], vector quantization [5], computer vision [6] and various business applications. Clustering algorithms can be classified into partitioning and hierarchical algorithms. Partitioning algorithms create a partitioning of objects into

a set of clusters. Hierarchical algorithms construct a hierarchical decomposition of objects. The hierarchical decomposition is represented by a tree strategy that separates the objects into small subsets until each subset consists only of sufficiently similar objects. There exist a large number of clustering algorithms in the literature including k-means [7], k-medoids [8], $CACTUS$ [9], $CURE$ [10], $CHAMELEON$ [11] and $DBSCAN$ [12]. No single algorithms is suitable for all types of objects, nor are all algorithms appropriate for all problems, however, the k-medoids algorithms have been shown to be robust to outliers [8] compared with centroid-based clustering. Assume T objects $x_1, x_2, \ldots x_T$ and k objects chosen from these T objects as the representative objects (medoids) $o_1, o_2, \ldots o_k$ then the total distance for partitioning these $(T-k)$ objects based on the k representative objects is

$$D_t = \sum_{x_m \in S_p} d(x_m, o_p) \tag{1}$$

where S_p is the pth partitioned set (or cluster) such that $d(x_m, o_p) \leq d(x_m, o_n)$, $n = 1 \ldots k$, and $m = 1 \ldots T$. Partitioning Around Medoids (PAM) [8], Clustering LARge Applications ($CLARA$) [8] and Clustering Large Applications based on RANdomized Search ($CLARANS$) [1] are three popular k-medoids-based algorithms while the Clustering Large Applications based on Simulated Annealing ($CLASA$) algorithm applies simulated annealing to select better medoids [13]. The drawback of the k-medoids algorithms is the time complexity of determining the medoids. In this paper, a novel sampling scheme based on *Multi-Centroids with Multi-Runs Sampling scheme* ($MCMRS$) is proposed to improve k-medoids-based algorithms. Significantly, all existing k-medoids algorithms, including $CLASA$, can be improved by combining with $MCMRS$. Both the $MCMRS$ and the combined $MCMRS$-$CLASA$ are shown to be superior to PAM, $CLARA$, $CLARANS$ and $CLASA$. In the following section we review the k-medoids algorithms including PAM, $CLARA$, $CLARANS$, $CLASA$ and genetic k-medoids algorithm. The proposed Multi-Centroids with Multi-Runs Sampling scheme ($MCMRS$) and the combined version, $MCMRS$-$CLASA$ algorithm are described in Section 4. Section 5 will present some experimental results using three artificial databases and a real image database. Section 6 contains the conclusions and the future work.

2 Existing k-medoids algorithms

K-medoids clustering algorithms evaluate a set of k objects for those that can be considered to be representative objects (medoids) for k clusters within T objects such that non-selected objects are clustered with the medoid to which it is the most similar. The total distance between non-selected objects and their medoid may be reduced by the swap of one of the medoids with one of the objects iteratively.

The PAM (Partitioning Around Medoids) algorithm can be depicted as follows:

Step 1: Initializaion - choose k medoids from T objects randomly.
Step 2: Evaluation - calculate the cost $D'_t - D_t$ for each swap of one medoid with one object, where D_t is the total distance before swap and D'_t is the total distance after swap.
Step 3: Selection - accept the swap with the best cost and if the cost is negative, go to step 2; otherwise record the medoids and terminate the program.

The computational complexity of the PAM algorithm is $O((1+\beta)k(T-k)^2)$ where β is the number of successful swaps. Obviously, it is time consuming even for a moderate number of objects and small number of medoids. $CLARA$ (Clustering LARge Applications) algorithm reduces the computational complexity by drawing multiple samples of the objects and applying PAM algorithm on each sample. The final medoids are obtained from the best result of these multiple draws. The $CLARA$ algorithm can be expressed as follows:

Step 1: Repeat the following steps q times.
Step 2: Call PAM algorithm with a random sample, s objects from the original set of T objects.
Step 3: Partition the T objects based on the k medoids obtained from previous step. Update the better medoids based on the average distance of the partition.

The clustering process in $CLARANS$ [1] is formalized as searching through a graph where each node is represented by a set of k medoids, and two nodes are neighbours if they only differ by one medoid. Each node has $k(T-k)$ neighbours, where T is the total number of objects. The $CLARANS$ algorithm starts with a randomly selected node. It moves to the neighbour node if one test for the $maxneighbour$ number of neighbours is successful; otherwise it records the current node as a local minimum. If the node is found to be a local minimum, it restarts with a new randomly selected node and repeats the search for a new local minimum. The procedure continues until the $numlocal$ numbers of local minima have been found, and return the best node. The $CLARANS$ algorithm can be described as follows:

Step 1: Repeat the following steps for $numlocal$ times.
Step 2: Select a current node randomly and calculate the average distance of this current code, where node is the collection of k medoids.
Step 3: Repeat the following step for $maxneighbour$ times.
- Select a neighbour node randomly and calculate the average distance for this node. If the average distance is lower, set current node to be the neighbour node.

Simulated annealing [14, 15] is a random search method that has been proved to be efficient for optimization problems. In our previous work, the simulated annealing was applied to generate medoids for the $CLASA$ algorithm [13]. The $CLASA$ algorithm can be illustrated as follows:

Step 1 : Choose an initial state s of the medoids at random and set the initial temperature $Temp = T_0$.

Step 2 : Randomly choose another state s' (a perturbation of state s) by swapping the medoids with the objects. Calculate the difference in total distortion $\Delta D = D_t(s') - D_t(s)$. If $\Delta D < 0$, replace the state s by s'; otherwise replace s by s' with probability $e^{\frac{-\Delta D}{Temp}}$ and go to step 3.

Step 3 : If the times of distance drops *dis_drop* (ie. the number of times $\Delta D < 0$) exceeds a prescribed parameter or the fixed number of perturbations *per* is reached, go to step 4; otherwise go to step 2.

Step 4 : Terminate the program and return the selected medoids if the temperature $Temp$ is below some prescribed freezing temperature T_f or the total number of perturbation *total_per* is reached; otherwise lower the temperature $Temp$ and go to step 2.

There are several possible methods for the annealing schedule, it is convenient to set $Temp = T_0\eta^t$, where t is the number of iterations, η is a constant coefficient, $0 < \eta < 1$.

3 Genetic *k*-medoids algorithm

Genetic algorithm [16] has been applied to k-medoids algorithm [17]. In paper [17], the genetic k-medoids algorithm is called GCA (genetic clustering algorithm). The basic idea of GCA is to generate P individuals initially and each individual consists of k different parameters selected from T objects randomly. The fitness function can be the inverse of the total distortion. The fitness is also modified using linear scaling technique. The modified fitness of each individual is evaluated and pair of individuals is selected based on the roulette selection. These two selected individuals are used for crossover operation to generate temporary individual with half parameters from each selected individuals. If the temporary individual is a wrong one, i.e., two parameters are the same, then simply replace by one of the selected individual. The mutation technique is applied to this temporary individual. After getting the same population size, the evaluation, selection, crossover and mutation are applied again until the maximum number of generations is reached or the satisfied fitness is obtained. The computational complexity of CGA based on the number of distance calculation is $O'(PGk(T-k))$, where G is the number of generations.

4 *MCMRS* sampling scheme

K-means and k-medoids are both partitioning clustering algorithms. For k-means, each cluster is represented by the mean value of the objects in the cluster whereas each cluster is represented by one of the objects located near the centre of the cluster in the k-medoids algorithm. K-means can be sensitive while k-medoids is generally more robust to outliers (or noise). One of the main factors to limit the use of the k-medoids algorithm is the inefficiency of k-medoids algorithms comparing with k-means – k-means algorithm can be several orders of magnitude faster than the k-medoids algorithm. In general, it is not efficient

for k-medoids algorithm even for moderate sized datasets. This drawback can be overcome with the aid of an efficient sampling scheme. The idea for the sampling scheme in this paper is motivated from the efficiency of the centroid-based clustering algorithms [15, 18]. From our empirical observations, we noticed that there is a higher probability of better medoids being selected within some distance from the centroids of the clusters. Based on this observation and the efficiency of the centroid-based clustering, we can generate k groups of medoid candidates with each group containing $NumCandidate$ nearest objects from the centroid for each centroid-based cluster. K medoids can be collected from each object in each group randomly. This process interates $NumSample$ times. This sampling scheme can be more robust by repeating the above procedure many times. The proposed $MCMRS$ can be depicted as follows:

Step 1 : Repeat the following steps for $NumRun$ times.
Step 2 : Get representative centroids by calling the centroid-based clustering algorithm (such as k-means or GLA algorithm[19]) with random initialization.
Step 3 : Get $NumCandidate$ objects for each cluster by selecting the $NumCandidate$ nearest objects from the centroid in each cluster.
Step 4 : Repeat the following steps $NumSample$ times.
Step 5 : Generate medoids by selecting one object from the $NumCandidate$ nearest objects in each cluster.
Step 6 : Calculate the average distance per object and update the best medoids.

$MCMRS$ sampling scheme can be combined with PAM, $CLARA$, $CLARANS$ and $CLASA$ to further improve the performance. The combined version of $MCMRS$ and $CLASA$ for example, (referred here as $MCMRS$-$CLASA$) can be described as follows:

Step 1 : Choose the best centroids whose average distance per object is a minimum by running the centroid-based clustering $NumRun$ times.
Step 2 : Get $NumCandidate$ objects for each cluster by selecting the $NumCandidate$ nearest objects from the centroid in each cluster.
Step 3 : Call $CLASA$ by using the nearest object from the centroid in each cluster as the initial medoids. The candidate medoid is swapped from the $NumCandidate$ nearest objects in the same group.
Step 4 : Terminate the program and return the medoids when the predefined criterion is satisfied.

5 Experimental Results

Three artificial databases and one real image database were used for the experiments are as follows:

1. 1,500 objects collected from four elliptic clusters shown in Fig. 1.
2. 12,000 objects collected from twelve elliptic clusters shown in Fig. 2.
3. 3,100 objects collected from five compact clusters shown in Fig. 3.

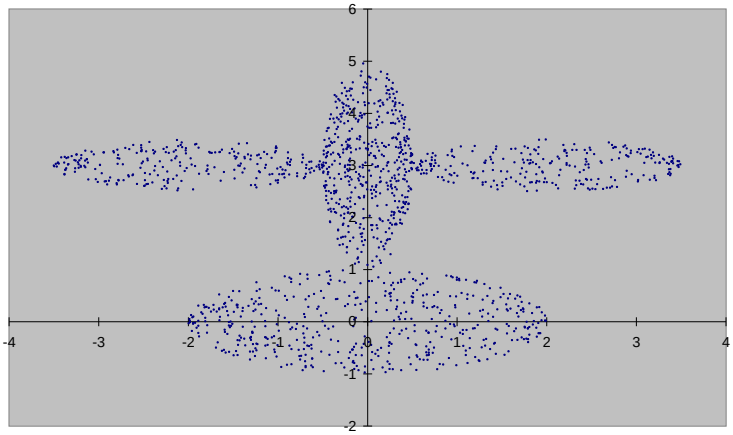

Fig. 1. Four elliptic clusters

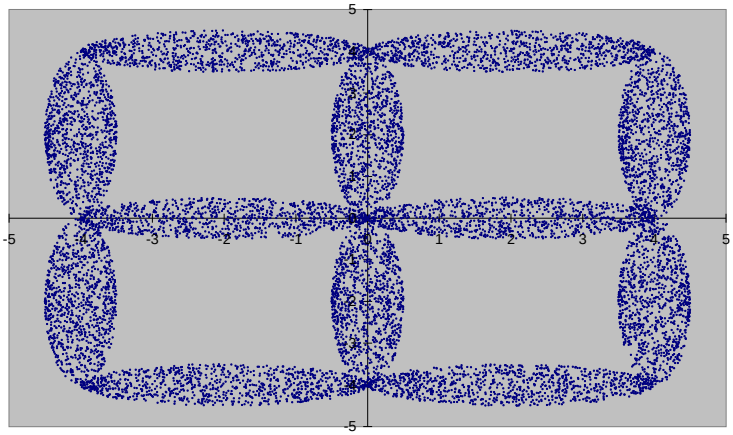

Fig. 2. Twelve elliptic clusters

4. 16,384 objects with 16 dimensions are generated from the $LENA$ grey-level image with size 512 by 512 .

Experiments were carried out to test the number of distance calculations and the average distance per object for the $CLARA$, $CLARANS$, $CLASA$ algorithms and the proposed $MCMRS$ and $MCMRS$-$CLASA$ algorithms. Squared Euclidean distance measure is used in this paper. The four elliptic clusters were used for the first experiment and 12 medoids are selected from 1500 objects. For $CLARA$, the parameter q was set to 5 and s was set to $160 + 2 * k$. For $CLARANS$, the parameter $numlocal$ was set to 5 and parameter

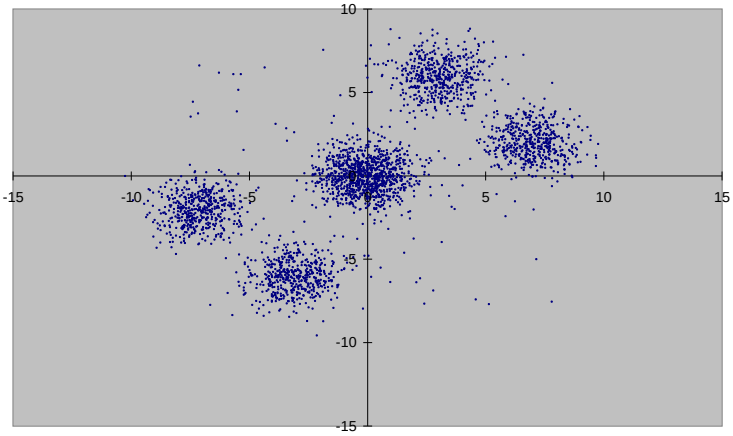

Fig. 3. Compact clusters with noise

maxneighbour was set to 270 (i.e. 1.5% of $k * T$). For $CLASA$, the parameters for the final temperature T_f, the initial temperature T_0, η, the times of distance drops *dis_drop*, *per* and the total number of perturbations *total_per* were set to 0.000025, 0.0001, 0.95, 10, 200, 50000, respectively. For $MCMRS$, a k-means algorithm is used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $MCMRS$-$CLASA$ algorithm, k-means algorithm is also used to generate 12 centroid-based clusters. The parameters $NumRun$ and $NumCandidate$ in $MCMRS$-$CLASA$ algorithm were set to 60 and 10, respectively. The other parameters in $MCMRS$-$CLASA$ were the same as for $CLASA$. The experimental results of $CLARA$, $CLARANS$, $MCMRS$ and $MCMRS$-$CLASA$ are averaged for 10 seeds are shown in Table 1 and the result of the first seed of $CLASA$ are shown in Fig. 4. As shown in Fig. 4, both $MCMRS$-$CLASA$ and $MCMRS$ algorithms outperform $CLASA$, $CLARANS$ and $CLARA$ algorithms.

Twelve elliptic clusters were used for the second experiment. 12 medoids are selected from 12,000 objects. For $CLARA$, the parameter q was set to 5 and s was set to $960 + 2 * k$. For $CLARANS$, the parameters *numlocal* and *maxneighbour* are set to 5 and 1800, respectively. For $MCMRS$, a k-means algorithm is used to generate 12 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $MCMRS$-$CLASA$, a k-means algorithm is also used to generate 12 centroid-based clusters. The parameters $NumRun$ and $NumCandidate$ in $MCMRS$-$CLASA$ were set to 60 and 10, respectively. The parameters for the final temperature T_f, the initial temperature T_0, η, the number of total distance drop *dis_drop*, *per* and the total number of perturbations *total_per* were set to 0.000025, 0.001, 0.95, 5, 20, 500000, respectively. Experimental results for 10

Table 1. Results of Experiment for four elliptic clusters

seed	*CLARA*		*CLARANS*		*MCMRS*		*MCMRS-CLASA*	
	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)
1	0.228	2004	0.228	1609	0.216	808	0.212	541
2	0.221	2302	0.232	1067	0.217	821	0.213	567
3	0.236	2388	0.236	1351	0.218	825	0.212	527
4	0.230	2686	0.233	1154	0.218	811	0.213	572
5	0.227	2430	0.232	1572	0.216	807	0.213	546
6	0.237	2260	0.227	1223	0.217	803	0.213	549
7	0.232	2646	0.234	895	0.217	821	0.213	591
8	0.236	2516	0.231	1429	0.217	803	0.213	580
9	0.233	2345	0.231	1148	0.216	803	0.213	524
10	0.238	2728	0.231	1167	0.216	805	0.212	581
Ave.	0.232	2431	0.232	1140	0.218	811	0.213	558

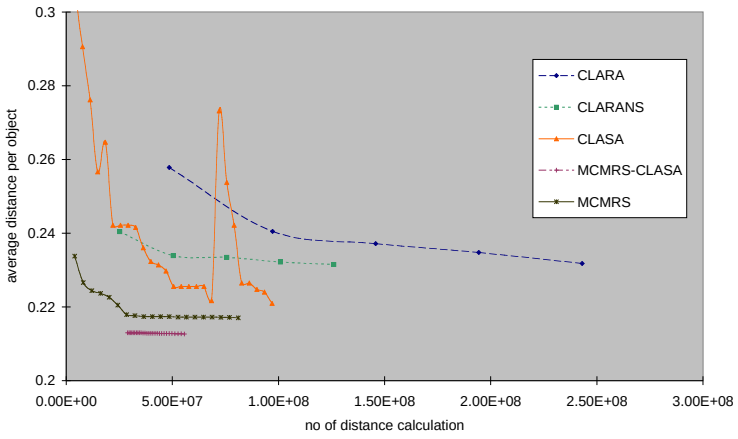

Fig. 4. Performance comparison of *CLARA*, *CLARANS*, *CLASA*, *MCMRS-CLASA* and *MCMRS* for four elliptic clusters

runs for *CLARA*, *CLARANS*, *MCMRS* and *MCMRS-CLASA* are shown in Table 2 and Fig. 5.

The compact clusters with noise were used for the third experiment. 5 medoids are selected from 3,100 objects. For *CLARA*, the parameter q was set to 5 and s was set to $200 + 2 * k$. For *CLARANS*, the parameters *numlocal* and *maxneighbour* are set to 5 and 200, respectively. For *MCMRS*, a k-means algorithm is used to generate 5 centroid-based clusters. The parameters *NumRun*, *NumCandidate* and *NumSample* in *MCMRS* were set to 20, 10 and 200, respectively. For *MCMRS-CLASA*, a k-means algorithm was again used to generate 5 centroid-based clusters. The parameters *NumRun* and *NumCandidate* in *MCMRS-CLASA* were set to 60 and 10, respectively. The parameters for the final temperature T_f, the initial temperature T_0, η, the times of distance

Table 2. Results of Experiment for twelve elliptic clusters

seed	*CLARA*		*CLARANS*		*MCMRS*		*MCMRS-CLASA*	
	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)
1	0.940	115659	0.931	91792	0.922	6587	0.920	5217
2	0.942	92528	0.943	63512	0.920	6554	0.918	5479
3	0.956	122462	0.935	76873	0.930	6496	0.920	5312
4	0.951	103413	0.928	79582	0.928	6447	0.918	5373
5	0.946	111577	0.949	90014	0.921	6483	0.918	5374
6	0.960	108856	0.936	72069	0.922	6632	0.918	5407
7	0.972	77562	0.936	84914	0.920	6426	0.920	5509
8	0.944	93889	0.934	93092	0.920	6549	0.918	5413
9	0.954	84365	0.930	59242	0.923	6573	0.918	5426
10	0.942	84365	0.931	75053	0.923	6541	0.920	5171
Ave.	0.951	99467	0.935	78615	0.923	6529	0.919	5368

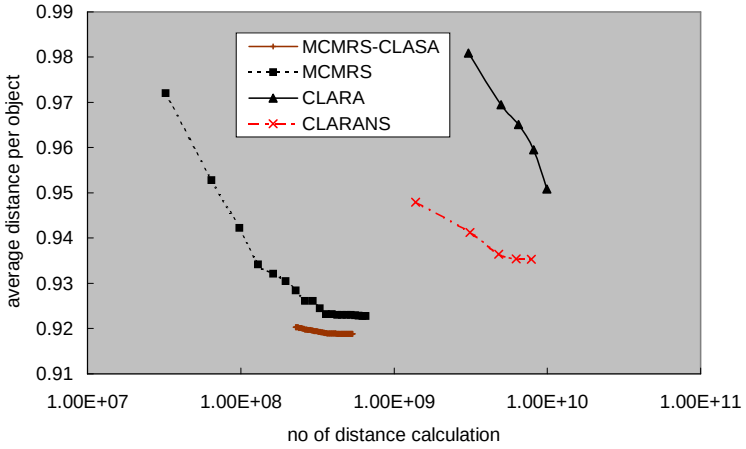

Fig. 5. Performance comparison of *CLARA*, *CLARANS*, *MCMRS* and *MCMRS-CLASA* for twelve elliptic clusters

drops *dis_drop*, *per* and the total number of perturbations *total_per* were set to 0.00025, 0.001, 0.85, 10, 200, 50000, respectively. Experimental results based on 10 runs for *CLARA*, *CLARANS*, *MCMRS* and *MCMRS-CLASA* are shown in Fig. 6. If the database is not large and the medoid size is small, the performance of *CLARA* is better than *CLARANS* shown in Table 3 and Fig. 6. Both *MCMRS* and *MCMRS-CLASA* are more efficient and effective than *CLARA* and *CLARANS*.

The *LENA* gray-level image data with size 512 by 512 is used for the fourth experiment. 16,384 objects with 16 dimensions are extracted from this image. 8 medoids are selected from these 16,384 objects. For *CLARA*, the parameter q was set to 5 and s was set to $1000 + 2 * k$. For *CLARANS*, the parameters *numlocal* and *maxneighbour* are set to 5 and 1800, respectively. For *MCMRS*, a

Table 3. Results of Experiment for compact clusters

seed	*CLARA*		*CLARANS*		*MCMRS*		*MCMRS-CLASA*	
	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)
1	2.436	253	2.432	584	2.398	646	2.397	243
2	2.425	274	2.457	600	2.397	647	2.397	232
3	2.430	264	2.429	604	2.398	655	2.397	239
4	2.440	295	2.442	504	2.397	649	2.397	251
5	2.445	232	2.431	583	2.398	651	2.397	232
6	2.411	253	2.419	606	2.398	655	2.397	244
7	2.435	243	2.457	530	2.397	642	2.397	256
8	2.422	285	2.470	519	2.397	654	2.397	241
9	2.444	253	2.417	620	2.397	647	2.397	267
10	2.440	274	2.424	581	2.398	645	2.397	283
Ave.	2.429	263	2.438	573	2.398	649	2.397	249

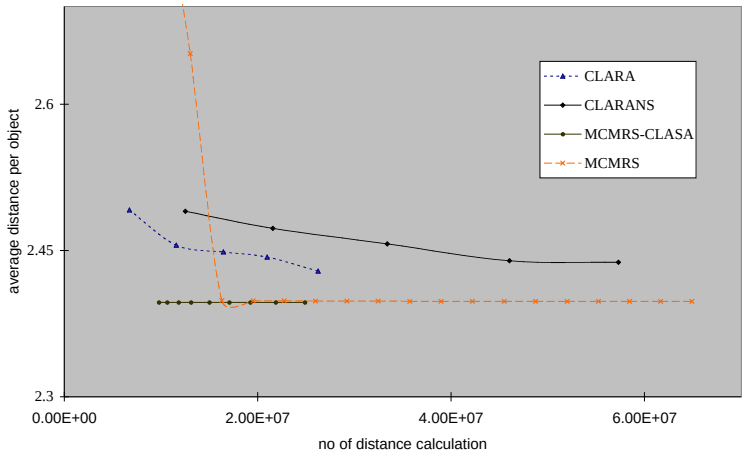

Fig. 6. Performance comparison of *CLARA*, *CLARANS*, *MCMRS* and *MCMRS-CLASA* for compact clusters with noise

k-means algorithm is used to generate 8 centroid-based clusters. The parameters $NumRun$, $NumCandidate$ and $NumSample$ in $MCMRS$ were set to 20, 10 and 200, respectively. For $MCMRS$-$CLASA$, a k-means algorithm is used to generate 8 centroid-based clusters. The parameters $NumRun$ and $NumCandidate$ in $MCMRS$-$CLASA$ were set to 20 and 10, respectively. The parameters for the final temperature T_f, the initial temperature T_0, η, the times of distance drops *dis_drop*, *per* and the total number of perturbations *total_per* were set to 0.000025, 0.0001, 0.85, 10, 200, 50000, respectively. Experimental results based on 10 runs for $CLARA$, $CLARANS$, $MCMRS$ and $MCMRS$-$CLASA$ are shown in Table 4. The proposed $MCMRS$-$CLASA$ algorithm can reduce the computation time by approximately a factor of 20 and obtain better average distance comparing with $CLARANS$.

Table 4. Results of Experiment for $LENA$ image

seed	$CLARA$		$CLARANS$		$MCMRS$		$MCMRS$-$CLASA$	
	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)	Average distance	Count of distances(10^5)
1	2829.56	30057	2816.43	79704	2800.15	6121	2795.64	3135
2	2833.36	31379	2831.77	56796	2802.49	5897	2795.64	3135
3	2839.22	32700	2806.48	76182	2801.81	6138	2795.64	3135
4	2821.52	36003	2814.16	59684	2801.98	6080	2795.76	3264
5	2827.43	28736	2821.41	44216	2802.71	6067	2795.64	3287
6	2842.22	28736	2822.10	54274	2801.52	6158	2793.44	3184
7	2838.23	30718	2817.13	55947	2800.16	6145	2795.76	3192
8	2843.91	28736	2819.38	56845	2803.82	6036	2795.76	3179
9	2820.76	24772	2823.58	64621	2802.98	6229	2796.01	3387
10	2819.52	28736	2826.95	59039	2801.34	6019	2793.54	3065
Ave.	2831.57	30057	2819.94	60731	2801.90	6089	2795.28	3196

6 Conclusions

In this paper, a novel sampling scheme using multi-centroids with multi-runs ($MCMRS$) is presented. This sampling scheme can be applied to PAM, $CLARA$, $CLARANS$ and $CLASA$ algorithms. A $MCMRS$-$CLASA$ algorithm that combines the benefits of $MCMRS$ with the $CLASA$ algorithm [13] is also proposed and evaluated. Experimental results based on three artificial databases and one real image database indicates that the proposed $MCMRS$ and $MCMRS$-$CLASA$ algorithms can not only reduce the average distance but also improve the clustering process. The computation load in $MCMRS$ and $MCMRS$-$CLASA$ can be further released by applying a more efficient centroid-based clustering method [15]. In future, we intend to apply the adaptive concept to further improve this novel sampling scheme by adjusting the $NumSample$, $NumRun$ and $NumCandidate$ automatically. We will also generalize the experimental results to commercial databases.

References

1. R. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *Twentieth International Conference on Very Large Data Bases* (J. B. Bocca, M. Jarke, and C. Zaniolo, eds.), (Santiago, Chile), pp. 144–155, Morgan Kaufmann, 1994.
2. J. F. Roddick and M. Spiliopoulou, "A survey of temporal knowledge discovery paradigms and methods," *IEEE Transactions on Knowledge and Data Engineering*, 2001.
3. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1992.
4. J. S. Pan, J. Wang, H. L. Fang, and C. Chen, "A modifed tabu search approach for texture segmentation using $2-d$ non-separable wavelet frames," in *10th International Conference on Tools with Artificial Intelligence*, pp. 474–481, 1998.
5. J. S. Pan, F. R. McInnes, and M. A. Jack, "vq codebook design using genetic algorithms," *IEE Electronics Letters*, vol. 31, no. 17, pp. 1418–1419, 1995.

6. J. Jolion, P. Meer, and S. Bataouche, "Robust clustering with applications in computer vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 8, pp. 791–802, 1991.
7. J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *5th Berkeley symposium on mathematics, statistics and Probability*, vol. 1, pp. 281–296, 1967.
8. L. Kaufman and P. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. New York: John Wiley and Sons, 1990.
9. V. Ganti, J. Gehrke, and R. Ramakrishnan, "Cactus-clustering categorical data using summaries," in *International Conference on Knowledge Discovery and Data Mining*, (San Diego, USA), pp. 73–83, 1999.
10. S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," in *ACM SIGMOD International Conference on the Management of Data*, (Seattle, WA, USA), pp. 73–84, 1998.
11. G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: a hierarchical clustering algorithm using dynamic modeling," *Computer*, vol. 32, pp. 32–68, 1999.
12. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Second International Conference on Knowledge Discovery and Data Mining* (E. Simoudis, J. Han, and U. Fayyad, eds.), (Portland, Oregon), pp. 226–231, AAAI Press, 1996.
13. S. C. Chu, J. F. Roddick, and J. S. Pan, "A comparative study and extensions to k-medoids algorithms," in *Fifth International Conference on Optimization : Techniques and Applications*, (Hong Kong, China), 2001.
14. S. Kirkpatrick, C. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
15. H. C. Huang, J. S. Pan, Z. M. Lu, S. H. Sun, and H. M. Hang, "Vector quantization based on genetic simulated annealing," *Signal Processing*, vol. 81, no. 7, pp. 1513–1523, 2001.
16. D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, 1989.
17. C. B. Lucasius, A. D. Dane, and G. Kateman, "On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasibility and comparison," *Analytica Chimica Acta*, pp. 647–669, 1993.
18. H. C. Huang, S. C. Chu, J. S. Pan, and Z. M. Lu, "A tabu search based maximum descent algorithm for vq codebook design," *Journal of Information Science and Engineering*, vol. 17, pp. 753–762, 2001.
19. Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, pp. 84–95, 1980.