Clustering with Niching Genetic *K*-means Algorithm

Weiguo Sheng, Allan Tucker, and Xiaohui Liu

Department of Information System and Computing
Brunel University, Uxbridge, Middlesex, UB8 3PH
London, UK
{weiguo.sheng, allan.tucker, xiaohui.liu}@brunel.ac.uk

Abstract. GA-based clustering algorithms often employ either simple GA, steady state GA or their variants and fail to consistently and efficiently identify high quality solutions (best known optima) of given clustering problems, which involve large data sets with many local optima. To circumvent this problem, we propose Niching Genetic *K*-means Algorithm (NGKA) that is based on modified deterministic crowding and embeds the computationally attractive *k*-means. Our experiments show that NGKA can consistently and efficiently identify high quality solutions. Experiments use both simulated and real data with varying size and varying number of local optima. The significance of NGKA is also shown on the experimental data sets by comparing through simulations with Genetically Guided Algorithm (GGA) and Genetic *K*-means Algorithm (GKA).

1 Introduction

Clustering is useful in exploratory data analysis. Cluster analysis organizes data by grouping individuals in a population in order to discover structure or clusters in the data. In some sense, we would like the individuals within a group to be similar to one another, but dissimilar from individuals in other groups. Various types of clustering algorithms have been proposed to suit different requirements. For clustering large data sets, there is a general consensus that partitional algorithms are imperative. Partitional clustering algorithms generate a single partitioning, with a specified or estimated number of clusters of the data in an attempt to recover natural groups present in the data. Among the partitional clustering algorithms, the *k*-means [5] has been popularly used because of its simplicity and efficiency. However, it highly depends on the initial choice of cluster centers and may end up in a local optimum.

A possible way to deal with local optimality of clustering problems is to use stochastic optimization schemes, such as Genetic Algorithms (GAs), which are believed more insensitive to initial conditions. There have been many attempts to use GAs for clustering problems. Roughly, the attempts can be classified as GA approaches such as [12,4,7] and hybrid GA approaches such as [15,9]. In most cases, the above approaches are reported performing well on small data sets with few local optima. However, real clustering problems may involve large data sets with many local optima. On such clustering problems, both GA and hybrid GA approaches can run into problems. First, they have difficulties in consistently identifying high quality solutions mainly because they employ either the Simple GA (SGA) [6], the Steady State GA (SSGA)

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 162–173, 2004.

[16] or their variants, which may suffer from premature convergence to local optima. Second, they are either very expensive or not efficient enough to identify high quality solutions.

To overcome these problems, we propose a method called *Niching Genetic K-means Algorithm* (NGKA). In NGKA, instead of employing SGA or SSGA, modified deterministic crowding is proposed and the computationally attractive *k*-means is incorporated. Our experiments show that NGKA can consistently and efficiently identify high quality solutions of given clustering problems, which involve large data sets with many local optima. Experiments use both simulated and real data with varying size and varying number of local optima. The performance of NGKA is also shown on the experimental data sets by comparing through simulation with Genetically Guided Algorithm (GGA) [4] and Genetic *K*-means Algorithm (GKA) [9].

The outline of the paper is as follows. We briefly review deterministic crowding in section 2. Section 3 provides the details of our proposed NGKA. Data sets employed in this work are described in section 4. Section 5 contains a description of how various parameters for NGKA can be set. Section 6 details the experiments and compares NGKA against GGA and GKA. Lastly, section 7 presents our conclusions and future work.

2 Deterministic Crowding

Different niching genetic algorithms have been developed [3,13,10]. In this work we focus on one niching genetic algorithm known as Deterministic Crowding (DC) [10]. In DC, selection pressure at the selection stage is eliminated by allowing individuals to mate at random with any other individual in the population. After crossover and eventually mutation, each child replaces the nearest parent if it has higher fitness. It is expected DC can maintain the population diversity and permit the GA to investigate many peaks in parallel. On the other hand, it prevents the GA from being trapped in local optima of the search space. In this work, we modify the DC and incorporate the computationally attractive *k*-means for clustering.

3 Niching Genetic *K*-means Algorithm

3.1 Algorithm Overview

To consistently and efficiently identify high quality solutions of given clustering problems, which involve large data sets with many local optima, we propose NGKA as follows. After initialization, one parent p_1 is selected randomly from the population, and its mate p_2 is selected not from the entire population, but from a group of individuals called *Comparing Factor Group* (CFG), picked randomly (without replacement) from the population. The one most similar to p_1 (e.g., the one whose cluster centers encoded in the chromosome are the closest to p_1's) is chosen as mate p_2. This procedure is repeated until $P/2$ parent pairs are selected. Each of these parent pairs is crossed to form not two but only one single offspring, after undergoing mutation and applying one step of *k*-means the single offspring is then paired with a more similar parent (a parent whose cluster centers encoded in the chromosome are closer to the offspring's), and if the fitness of the offspring is better than its paired parent, its parent is replaced.

The details of NGKA are shown as below. The algorithm is terminated when the stopping criterion is met. The output of the algorithm is the best solution encountered during the evolution.

Step 1. Randomly initialize P sets of k cluster centers using floating-point representation. Constrain the initial values to be within the space defined by the vectors to be clustered. Only valid individuals that have at least one data point in each cluster are considered to be included in the initial population.

Step 2. Calculate *MSE* according to equation (3) for each individual in the initial population and set the fitness value as $f = 1/MSE$.

Step 3. Repeat following (a) to (e) until the *stopping criterion* is met.

a) One parent p_1 is selected randomly from the population, and its mate p_2 is selected from CFG (see section 5 for the size setting). The one most similar (determined by the Euclidean distance based on *phenotypic metric*) to p_1 is chosen as mate p_2. This procedure is repeated until $P/2$ parent pairs are selected.

b) Do real *arithmetic crossover* on each paired parent to form not two but only one single offspring with probability of one and then perform *Gaussian mutation* on each feature of the offspring with some low probability.

c) Run k-means one step on the new offspring and update the offspring (i.e. each data point in the data set is assigned to its closest cluster center encoded in offspring's chromosome after which the cluster centers are updated as the centers of mass of the data points that are assigned to the clusters).

d) Compare the offspring with both parents, and paired with a more similar parent.

e) Calculate MSE' according to equation (4) for the offspring and set fitness of the offspring equal to $1/MSE'$. If the fitness of offspring is better than its paired parent, then replace parent.

Step 4. Provide the cluster centers for the terminal population member with the best fitness.

3.2 Representation

In most of the GA clustering applications, the binary, integer or floating-point representation [14,12,2] are commonly used. In the binary and integer representation, binary codes or integers are usually used to represent the membership or permutation of data points and cluster assignment is done explicitly based on the value of binary codes or integers. In most cases, both binary and integer representations suffer from problems of redundancy and context insensitivity with traditional crossover and mutation methods. Furthermore, they are not a scalable representation for clustering large data sets due to the length of the genomes. In this work, we use a floating-point presentation, which represents the cluster centers. Cluster assignment is done implicitly based on distance. In this context, [11] showed that a real-valued representation moves the problem closer to the problem representation which offers higher precision with more consistent results across replications.

Our representation consists of a vector of $k \times d$ features of real numbers, where d is the number of dimensions in the data and k is the number of clusters. The first d positions represent the d dimensions of the first cluster center, the next d positions represent those of the second cluster center, and so on.

3.3 Crossover and Mutation

The crossover and mutation operators are chosen after a number of experimental trials. Arithmetic crossover [11] and Gaussian mutation perform well and are selected as reproduction operators for NGKA. Traditional arithmetic crossover linearly combines two parent chromosome vectors to produce two new offspring according to the following equations:

$$Offspring1 = a{*}Parent1 + (1\text{-}\ a){*}Parent2, \tag{1}$$

$$Offspring2 = (1 - a){*}Parent1 + a{*}Parent2. \tag{2}$$

Where $a \in [0,1]$ is a random weighting factor. In NGKA, we use $a = 0.5$ as a weighting factor, however, to produce not two but only one single offspring.

After crossover, a very low probability of Gaussian mutation will apply on the offspring. Gaussian mutation adds a unit Gaussian distributed random value to the chosen feature. The new feature value is clipped if it falls outside of the lower or upper bounds of that feature.

3.4 *K*-means Hybridization

K-means is an iterative scheme attempting to minimize the sum of squared Euclidean distances between data points and cluster centers. Let x_i i=1,2,…n be the set of n data points, k, the number of clusters, m_j, the centroid of cluster C_j. Then the algorithm tries to minimize the cost function Mean Square Error (MSE)

$$\boldsymbol{MSE} = \sum_{i=1}^{n} \sum_{j=1,x_i \in C_j}^{k} \left| x_i - m_j \right|^2 . \tag{3}$$

Starting from an initial distribution of cluster centers in the data space, each data point is assigned to the cluster with closest center, after which each center itself is updated as the center of mass of all data points belonging to that particular cluster. The procedure is repeated until convergence. This iterative scheme is known to converge sufficiently fast.

In order to improve computational efficiency dramatically, one step of k-means is applied to all new offspring during each generation, after the regeneration step. This is done by assigning each data point to one of the clusters with the nearest centre encoded in the individual's chromosome. After that, the cluster centers encoded in the chromosome are replaced by the mean points of the respective clusters.

3.5 Fitness Function

The fitness calculation of an individual is based on the clusters formed according to the centers encoded in the chromosome. It was defined as $f = 1/MSE$, in which MSE is computed according to equation (3), so that maximization of the fitness leads to minimization of MSE.

During our experiments, sometimes the resulting individuals may represent a partitioning with empty clusters (called *illegal individuals*). To penalize illegal individuals, we use the following heuristic proposed in [4]. If a partitioning, defined by its cluster centers, has b clusters with no data points assigned to them and the MSE value for the partitioning evaluates to a value Tot, the new value will be $(b+1)*Tot$. The resulting fitness function will be $f = 1/MSE'$ where

$$MSE' = MSE + b*MSE, \tag{4}$$

$b \in \{0,\ldots,k\}$ is the integer number of empty clusters. The heuristic penalizes illegal partitionings by decreasing their fitness value. This makes them less likely to replace paired parents and less likely to enter the population pool for the next generation.

4 Data Set Description

4.1 Simulated Data

The simulated data created for our study comprises 3,300 data points with nine clusters. The nine clusters are generated according to a spherical bivariate normal distribution with given mean vector $u=(u_x,u_y)$ and standard deviation σ for both x and y according to:

Cluster1: 300 objects	$u_x = 1.0$	$u_y = 0.5$	$\sigma = 0.2$
Cluster2: 300 objects	$u_x = 2.0$	$u_y = 0.5$	$\sigma = 0.2$
Cluster3: 600 objects	$u_x = 5.0$	$u_y = 0.5$	$\sigma = 0.6$
Cluster4: 600 objects	$u_x = 1.5$	$u_y = 3.0$	$\sigma = 0.6$
Cluster5: 300 objects	$u_x = 4.2$	$u_y = 3.5$	$\sigma = 0.2$
Cluster6: 300 objects	$u_x = 5.5$	$u_y = 3.5$	$\sigma = 0.2$
Cluster7: 300 objects	$u_x = 3.5$	$u_y = 2.0$	$\sigma = 0.8$
Cluster8: 300 objects	$u_x = 3.0$	$u_y = -1.0$	$\sigma = 0.7$
Cluster9: 300 objects	$u_x = 3.0$	$u_y = 5.0$	$\sigma = 0.7$

4.2 Subcellcycle Data

The subcellcycle is a subset of the yeast cell cycle data set provided by [1]. The yeast cell cycle data set contains time-course expression profiles for more than 6220 genes, with 17 time points for each gene taken at 10-min intervals covering nearly two yeast cell cycles (160min). This data set is very attractive because a large number of genes contained in it are biologically characterized and have been assigned to different phases of the cell cycle. The subcellcycle data used in this work consists of 384 genes whose expression levels peak at different time points corresponding to the five phases (early G1, late G1, S, G2 and M) of cell cycle. We expect clustering results to approximate this five-class partitioning.

4.3 Serum Data

This data set is described and used in [8] and corresponds to the selection of 517 genes whose expression vary in response to serum concentration in human fibroblasts. Here, we expect eight clusters from it.

Both gene data sets are normalized so that every gene has an average expression value of zero and a standard deviation equal to one. Normally, to measure the dissimilarity between two genes one tends to choose correlation coefficients which capture the similarity of the "shapes" of two expression profiles, and ignores differences between their magnitudes. However, Euclidean distance metric is used for all results reported here since it has been shown that the correlation coefficient and Euclidean distance are equivalent on a standardized gene expression data set [17].

5 Parameters Configuration

Before running NGKA, there are some parameters that need to be set including the crossover probability, mutation rate, population size, stopping criterion and *Comparing Factor Group* (CFG) size.

The crossover probability (p_c) is set to 100%. Normally one tends to choose a lower value for the crossover probability in order to allow individuals to survive from one generation to the other. However, in NGKA we have found that a crossover rate of 95-100% offers best results, since only better offspring can replace its paired parent so the individuals with highest fitness will survive multiple generations. The mutation probability (p_m) is set at 0.01. Mutation is useful to allow individuals to explore other areas that haven't been explored by the algorithm.

For population size setting, we find NGKA performs well on a generally small population size. However, a too small population would heavily retard the search ability of NGKA. Based on trial runs, we find a population size of 50 for simulated data and subcellcycle data (with 10s to 100s local optima), 75 for serum data (with 100s to 1000s local optima) to be acceptable lower limit, and better performance can be obtained by using a large population size.

Our stopping criterion for NGKA is that the fitness value of the best population member has not changed for *n* generations, with n=10 being a reasonable choice for NGKA on all the three experimental data sets.

A final variable that needs to be set is the *CFG* size. The idea of using *CFG* is to get a group that will likely contain an individual from the same niche of the first parent. However, an appropriate value should be set to allow both thorough exploration of the search space and competition between different niches. When the group size value equals one, it is basically random selection. As the group size increases there is more possibility of selecting a mate from the same niche as the first parent. For our experimental data sets, the *CFG* size for NGKA has been determined empirically. We use *CFG*=20%**P*, where *P* is the population size.

6 Experiments

In order to be of real use, the solutions supplied by a clustering algorithm must be of high quality (i.e. best known optima) and discovered within a reasonable time. More importantly, to have confidence in the clusters supplied by a clustering algorithm, the algorithm must consistently deliver such high quality solutions. In this respect, we compare the performance of NGKA with the recently proposed GGA and GKA in terms of consistency and efficiency of identifying high quality solutions both on simulated and real data. As well as fitness value, we report the *MSE* value for all experiments. This allows us to compare performance of the different algorithms with similar but different fitness functions. All results reported in this section were obtained on a PC with AMD Athlon 1800 running Windows2000 operation system.

We are primarily interested in clustering problems that involve large data sets with many local optima. In order to determine the exact number of different optima of clustering the above three data sets, one would in principle be required to start a local search algorithm such as *k*-means from every possible initial configuration of the cluster centers. However even if we restrict the experiment to consider only the configurations where each cluster center is initially placed in one of the data points, there would still be too many initial configurations to be tested. Therefore we run 5000 trials of *k*-means using random initial configuration on each of the three data sets. There are 395, 71 and 965 different optima found on simulated data, subcellcycle data and serum data respectively. And more trials lead to more different optima, for example 1324 optima found out of 10,000 trials on serum data, *figure 1* plots all 1324 different optima associated with the final partitionings. The figure also illustrates that *k*-means is very far from being a consistent technique to identify high quality solutions. Since it has been reported in [4] and [9] that GGA and GKA outperform *k*-means, we will not consider *k*-means for further comparison.

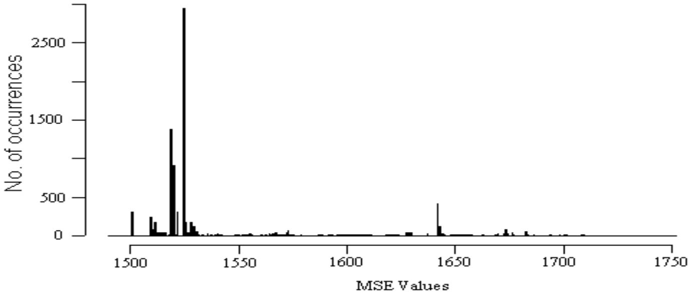

Fig. 1. All of the 1324 different optima with *MSE* value found by applying *k*-means 10,000 trials on serum data

Before comparing the performance of the three algorithms, we give a brief description of the GA used in GGA and GKA. In GGA, a GA based on tournament selection, two-point crossover and flip bit mutation is applied with binary gray coding for clustering. The GA used in GKA is based on roulette wheel selection and a biased mutation with string-of-group-number coding which is one kind of integer representation discussed in section 3.2. Moreover GKA uses one step of *k*-means instead of the traditional crossover operators, which we think may restrict the GA's search capability. A further investigation is out of the scope of this paper.

To make the comparison between the three algorithms more meaningful, the same stopping criterion (i.e. the fitness value of the best population member has not changed for *n* generations) is used for all experiments. And a reasonable *n* value is set to be *n*=10 for GKA and *n*=80 for GGA on all the three experimental data sets. The remaining parameters of GGA and GKA are set in *table 1*. The population size of GGA and GKA on the three data sets is identical to that in NGKA. Other parameters (crossover rate, mutation rate and order of tournament) of GGA and GKA are specified according to the original papers for best performance.

Table 1. Population size, crossover rate, mutation rate and order of tournament setting of the three algorithms (GGA, GKA and NGKA) for tests on three experimental data sets (*data1*: simulated data, *data2*: subcellcycle data, *data3*: serum, *P* is the population size and *bits* is the number of bits in an individual [4])

Parameter	**Algorithm**								
	GGA			**GKA**			**NGKA**		
Population size	Data1	Data2	Data3	Data1	Data2	Data3	Data1	Data2	Data3
	50	50	75	50	50	75	50	50	75
Crossover rate (P_c)	0.9			N/A			1.0		
Mutation rate (P_m)	$1.75/(P\times\sqrt{(bits)})$			0.05			0.01		
Order of tournament	2			N/A			N/A		

To compare performance of the three algorithms in terms of consistency and efficiency of identifying high quality solutions, we recorded the average *MSE* value, standard deviation, distribution of the optima and average run time to identify the high quality solution found from tests on the three data sets. The average *MSE* values and corresponding standard deviations are obtained by generating 20 random starting populations, and running each experiment once with each of the 20 random starting populations. While the average run time to identify the high quality solution is obtained by running each experiment iteratively with each of the 20 random starting populations until a high quality solution is identified. All the results were then averaged over the 20 trials. All sets of experiments use the same 20 random starting populations. *Table 2* lists the average values of *MSE* and standard deviation found from the tests. *Figures 2*, *3* and *4* show the distribution of different optima found out of 20 trials on the three data sets respectively. The results of average running time to identify the high quality solution are shown in *table 3*.

The reason for using simulated data set in this work is that the structure is known a priori, thus enabling a better validation of the algorithm proposed here. Furthermore, the bivariate simulated data allows us to graphically display the clustering results. So, here we will examine the experimental results on simulated data first. In terms of consistency, *figure 2* shows that NGKA consistently identifies the best known optimum with *MSE*=1516.41 in each of 20 trials. However GGA and GKA can only identify

Table 2. Comparing average *MSE* values and standard deviation found by the three algorithms(GGA, GKA and NGKA) on simulated data, subcellcycle data and serum data. The results are averaged over the 20 trials

Algorithm	Simulated data		Subcellcycle data		Serum data	
	MSE	**St.dv.**	***MSE***	**St.dv.**	***MSE***	**St.dv.**
GGA	1573.23	45.89	2361.12	12.15	1511.79	8.28
GKA	1548.32	29.89	2350.22	5.52	1507.65	6.74
NGKA	1516.41	0.0	2345.56	0.0	1500.84	0.11

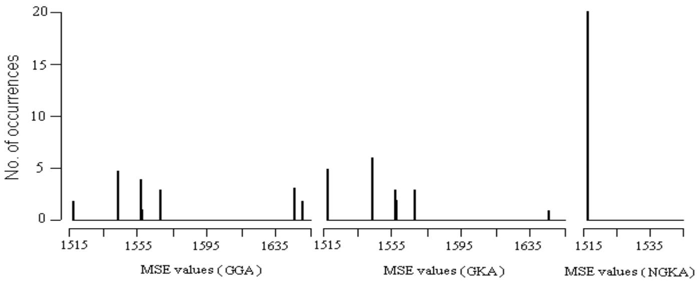

Fig. 2. Distribution of the optima with *MSE* value found for 20 trials of GGA, GKA and NGKA on simulated data

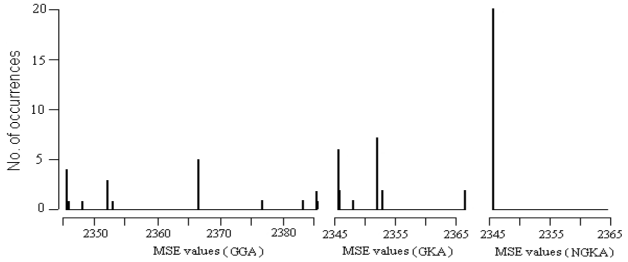

Fig. 3. Distribution of the optima with *MSE* value found for 20 trials of GGA, GKA and NGKA on subcellcycle data

the best known optimum 2 times and 5 times respectively. Two typical local optima that were identified by GGA and GKA are *MSE*=1543.08 and *MSE*=1556.13. *Figure 5(a)* and *5(b)* show the simulated data and its best known optimum clustering result (different clusters represented by different symbols). The two local optima and their corresponding clustering results are shown in *figure 5(c)* and *5(d)* respectively. We can see that the two local optima cannot succeed in finding the correct clusters, as two clusters at the lower-left corner are joined together (represented by 'Δ' in *figure 5(c)* and '+' in *figure 5(d)*) while one other cluster is split up. In terms of efficiency, *table 3* shows that to identify the best known optimum, by average, GGA needs 6945.2 seconds, which is around 118 times longer than that of NGKA. GKA is quite efficient and takes only 169.2 seconds, however it is still more than 3 times longer than that of NGKA.

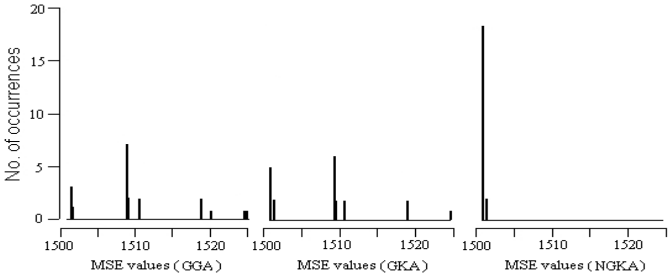

Fig. 4. Distribution of the optima with *MSE* value found for 20 trials of GGA, GKA and NGKA on serum data

Table 3. Comparing average run time to identify the high quality solution found by the three algorithms (GGA, GKA and NGKA) on simulated data, subcellcycle data and serum data. The results are averaged over the 20 trials

Algorithm	**Average run time to identify the high quality solution** (seconds)		
	Simulated data	**Subcellcycle data**	**Serum data**
GGA	6945.2	3805.3	12587.4
GKA	192.7	72.1	281.6
NGKA	58.6	29.7	92.1

Testing on subcellcycle gene expression data (Table 2 and Fig 3), GGA also takes a long time and result in *ave.MSE*=2361.12 with *std.dev*=12.15. Out of 20 trials, it identifies the best known optimum 4 times with *MSE*=2345.56. When applying GKA, the computational efficiency is significantly improved. On average it takes 72.1 seconds to identify the best known optimum and achieves *ave.MSE*=2350.22 with *std.dev*=5.52. However, out of 20 trials, it only identifies the best known optimum 6 times. Experiments using NGKA show that the computational efficiency of identifying the best known optimum is even more improved with only 29.7 seconds. More importantly, the average *MSE* value and standard deviation are improved to *ave.MSE*=2345.56 with *std.dev*=0.0, which means the best known optimum is consistently identified in each of 20 trials. Testing on serum gene expression data (Table 2 and Fig 4), similar results are obtained. Out of 20 trials, NGKA successfully identifies the best known optimum 18 times with *MSE*=1500.82. However, GGA and GKA can only identify it 3 times and 5 times respectively. Clearly, based on the experiments both on simulated and real data NGKA is the best alternative as it consistently delivers high quality clustering solutions faster.

7 Conclusions and Future Work

In this work, we considered the clustering problem of identifying high quality solutions (best known optima) of given data sets, which involve in large data sets with many local optima. We propose NGKA to solve the problem consistently and efficiently. For clustering of small data sets with few local optima, all GGA, GKA and

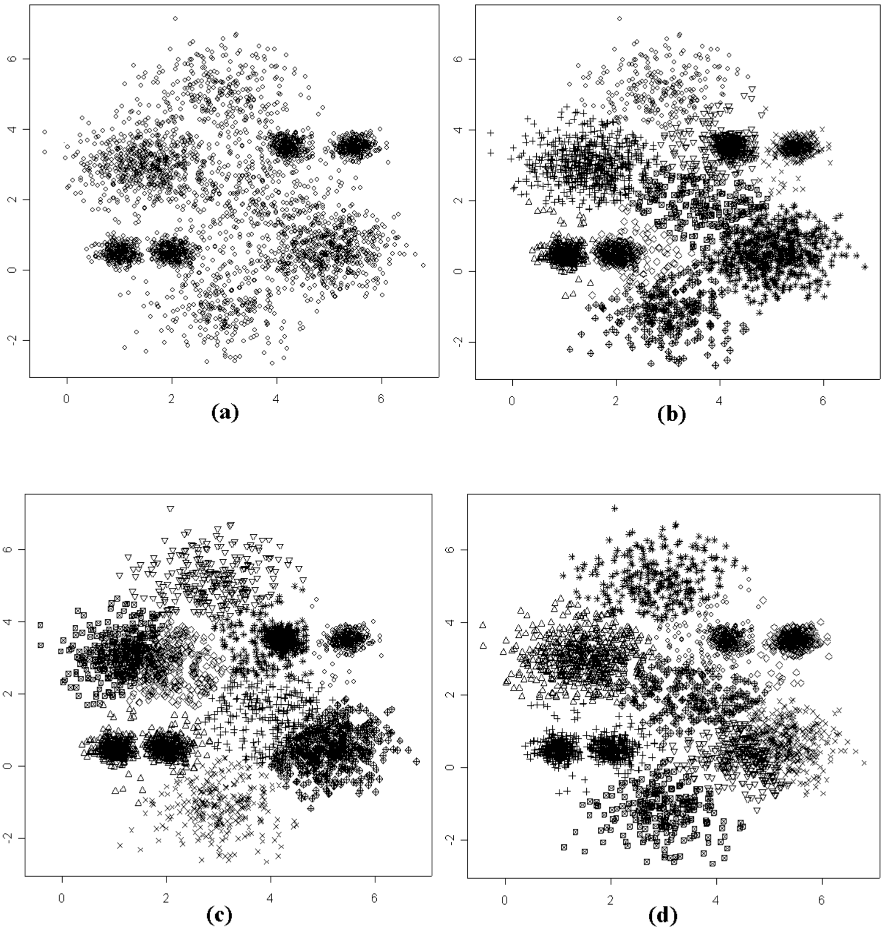

Fig. 5. (*a)* Random generated simulated data. (*b*) Clustering results corresponding to the best known optimum with *MSE*=1516.41. (*c*) Clustering results corresponding to the local optimum with *MSE*=1543.08. (*d*) Clustering results corresponding to the local optimum with *MSE*=1556.13

NGKA perform well. However, as the size of data sets and corresponding number of local optima increase, the picture changes drastically: GGA becomes very expensive and both GGA and GKA have difficulties in consistently identifying high quality solutions. However, our proposed NGKA can identify high quality solutions *faster* and more importantly, it can *consistently* deliver such high quality solutions.

The work presented here represents the first attempt to consistently and efficiently cluster gene expression data sets. Our promising results lead us to believe that NGKA can be extended to a class of real world large clustering problems where other methods have not been appropriate.

Future work will involve the analysis of biological significance of the clustering results found by NGKA based on biological knowledge. A dynamic NGKA clustering which does not require prior specification of the number of clusters will also be investigated, involving perhaps a fitness function which maximizes both the homogeneity within each cluster and the heterogeneity among clusters.

References

1. Cho, R.J. et al.: A Genome-Wide Transcriptional Analysis of the Mitotic Cell Cycle. Molecular Cell 2(1) (1998) 65-73
2. Cucchiara, R.: Genetic Algorithms for Clustering in Machine Vision. Machine Vision and Applications, Vol. 11, No.1 (1998) 1-6
3. Goldberg, D.E. and Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. Proceeding of the 2nd Int. Conference on Genetic Algorithms. Hillsdale, New Jersey (1987) 41-49
4. Hall, L.O., Ozyurt, B. and Bezdek, J.C.: Clustering with a Genetically Optimized Approach. IEEE Transactions on Evolutionary Computation, Vol. 3, No. 2 (1999) 103-112
5. Hartigan, J.A. and Wong, M.A.: A *k*-means clustering algorithm. Applied Statistics, 28 (1979) 100-110
6. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
7. Eduardo, R. Hruschka and Nelson, FF Ebecken: A genetic algorithm for cluster analysis. Intelligent Data Analysis 7 (2003) 15-25
8. Iyer V.R. et al.: The Transcriptional Program in the Response of Human Fibroblasts to Serum. Science, 283 (1999) 83-87
9. Krishna, K. and Murty, M. Narasimha: Genetic *K*-means Algorithm. IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, Vol. 29, No. 3 (1999)
10. Mahfoud, S.W.: Niching methods for genetic algorithms. Ph.D. dissertation, Univ. of Illinois, Urbana-Champaign (1995)
11. Michalewicz, Z.: Genetic algorithms + Data structure = Evolution programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
12. Murthy, C.A. and Chowdhury, N.: In search of optimal clusters using genetic algorithms. Pattern Recognition Letters, 17 (1996) 825-832
13. Petrowski, A.: A clearing procedure as a niching method for genetic algorithms. Proceeding of IEEE Int. Conf. Evolutionary Computation (1996) 798-803
14. Sarkar, M., Yegnanarayana, B. and Khemani, D.: A Clustering Algorithm Using an Evolutionary Programming-based Approach. Pattern Recognition Lett. 18 (1997) 975-986
15. Scheunders, P.: A genetic *c*-means clustering algorithm applied to color image quantization. Pattern Recognition, Vol. 30, No. 6 (1997) 859-866
16. Syswerda, G.: A study of reproduction in generational and steady-state genetic algorithms. In Foundations of Genetic Algorithms, Morgan Kauffmann Publishers (1991) 94-101
17. Yeung, K.Y.: Clustering analysis of gene expression data. PhD Thesis, University of Washington (2001)