Genetic algorithms for large scale clustering problems

(published in *The Computer Journal*, **40** (9), 547-554, 1997)

Pasi Fränti[1], Juha Kivijärvi[2], Timo Kaukoranta[2] and Olli Nevalainen[2]

[1] Department of Computer Science,
University of Joensuu, PB 111
FIN-80101 Joensuu, FINLAND
Email: franti@cs.joensuu.fi

[2] Turku Centre for Computer Science (TUCS)
Department of Computer Science
University of Turku
Lemminkäisenkatu 14 A
FIN-20520 Turku, FINLAND

Abstract

We consider the clustering problem in a case where the distances of elements are metric and both the number of attributes and the number of the clusters are large. Genetic algorithm approach gives in this environment high quality clusterings at the cost of long running time. Three new efficient crossover techniques are introduced. The hybridization of genetic algorithm and k-means algorithm is discussed.

Indexing terms: clustering problem, genetic algorithms, vector quantization, image compression, color image quantization.

1. Introduction

Clustering is a *combinatorial problem* where the aim is to partition a given set of data objects into a certain number of clusters [1, 2]. In this paper we concentrate on large scale data where the number of data objects (N), the number of constructed clusters (M), and the number of attributes (K) are relatively high. Standard clustering algorithms work well for very small data sets but often perform much worse when applied to large scale clustering problems. On the other hand, it is expected that a method that works well for large scale data would also work well for problems of smaller scale.

Clustering includes the following three subproblems: (1) the selection of the cost function, (2) the decision of the number of classes used in the clustering, and (3) the choice of the clustering algorithm. We consider only the last subproblem and assume that the number of classes (clusters) is fixed beforehand. In some application (such as *vector quantization* [3]) the question is merely about resource allocation, i.e. how many classes can be afforded. The data set itself may not contain clearly separate clusters but the aim is to partition the data into a given amount of clusters so that the cost function is minimized.

Many of the clustering algorithms can be generalized to the case where the number of classes must also be solved. For example, the clustering algorithm can be repeatedly applied for the data using all reasonable number of clusters. The clustering best fitting the data is then chosen according to any suitable criterion. The decision is typically made by the researcher of the application area but analytical methods have also been

considered. For example, by minimizing the *stochastic complexity* [4] one can determine the clustering for which the entropy of the intracluster diversity and the clustering structure is minimal.

Due to the high number of data objects we use a metric distance function instead of a distance matrix to approximate the distances between the objects. The attributes of the objects are assumed to be numerical and of the same scale. The objects can thus be considered as points in a K-dimensional *Euclidean space*. The aim of the clustering in the present work is to minimize the intracluster diversity (*distortion*).

Optimization methods are applicable to the clustering problem [5]. A common property of these methods is that they consider several possible solutions and generate a new solution (or a set of solutions) at each step on the basis of the current one.

In a *genetic algorithm* (GA) [6] we use a model of the natural selection in real life. The idea is the following. An *initial population* of solutions called *individuals* is (randomly) generated. The algorithm creates new *generations* of the population by *genetic operations*, such as *reproduction*, *crossover* and *mutation*. The next generation consists of the possible *survivors* (i.e. the best individuals of the previous generation) and of the new individuals obtained from the previous population by the genetic operations.

Genetic algorithms have been considered previously for the clustering problem in vector quantization by Delport and Koschorreck [7], and by Pan, McInnes and Jack [8]. Vector quantization was applied to DCT-transformed images in [7], and to speech coding in [8]. Scheunders [9] studied genetic algorithms for the scalar quantization of gray-scale images, Murthy and Chowdhury [10] for the general clustering problem. These studies concentrate on special applications [7, 8, 9], or the algorithms have been applied to very small scale data sets [10] only, and there is no guarantee that the methods work for large scale problems in different application domain. In addition, the parameters of the proposed methods should be studied in more detail.

In this paper we present a systematic study on genetic algorithms for the clustering problem. In the design of the algorithms, the key questions are:

- Representation of the solution.
- Selection method.
- Crossover method.

The efficiency of the GA is highly dependent on the coding of the individuals. In our case a natural representation of a solution is a pair (*partitioning table*, *cluster centroids*). The partitioning table describes for each data object the index of the cluster where it belongs. The cluster centroids are representative objects of the clusters and their attributes are found by averaging the corresponding attributes among the objects in the particular cluster.

Three methods for selecting individuals for crossover are considered: *a probability-based* method and two *elitist* variants. In the first one, a candidate solution is chosen to crossover with a probability that is a function of its *distortion*. In the latter two variants only the best solutions are accepted while the rest are dropped.

For the crossover phase, we discuss several problem oriented methods. These include two previously reported (*random crossover* [7, 10] and *centroid distance* [8, 9]) and three new techniques (*pairwise crossover*, *largest partitions* and *pairwise nearest neighbor*). It turns out that, due to the nature of the data, none of the studied methods is efficient when used alone but the resulting solutions must be improved by applying few steps of the conventional *k-means* clustering algorithm [11]. In this *hybrid method*, new solutions are first created by crossover and then fine-tuned by the k-means algorithm. In fact, all previously reported GA methods [7-10] include the use of k-means in a form or another.

The rest of the paper is organized as follows. In Section 2 we discuss the clustering problem, the applications and data sets of the problem area. Essential features of the GA-solution are outlined in Section 3. Results of the experiments are reported in Section 4. A comparison to other clustering algorithms is made. Finally, conclusions are drawn in Section 5.

2. Clustering problem and applications

Let us consider the following six data sets: *Bridge*, *Bridge-2*, *Miss America*, *House*, *Lates mariae*, and *SS2*, see Fig. 1. Due to our vector quantization and image compression background, the first four data sets originate from this context. We consider these data sets merely as test cases of the clustering problem.

In vector quantization, the aim is to map the input data objects (vectors) into a representative subset of the vectors, called *codevectors*. This subset is referred as a *codebook* and it can be constructed using any clustering algorithm. In data compression applications, reduction in storage space is achieved by storing the index of the nearest codevector instead of each original data vector. More details on the vector quantization and image compression applications can be found in [3, 12, 13].

Bridge consists of 4×4 spatial pixel blocks sampled from the gray-scale image (8 bits per pixel). Each pixel corresponds to a single attribute having a value in the range [0, 255]. The data set is very sparse and no clear cluster boundaries can be found. *Bridge-2* has the blocks of *Bridge* after a BTC-like quantization into two values according to the average pixel value of the block [14]. The attributes of this data set are binary values (0/1) which makes it an important special case for the clustering. According to our experiments, most of the existing methods do not apply very well for this kind of data.

The third data set (*Miss America*) has been obtained by subtracting two subsequent image frames of the original video image sequence, and then constructing 4×4 spatial pixel blocks from the residuals. Only the first two frames have been used. The application of this kind of data is found in video image compression [15]. The data set is similar to the first set except that the data objects are presumably more clustered due to the motion compensation (subtraction of subsequent frames).

The fourth data set (*House*) consists of the *RGB* color tuples from the corresponding color image. This data could be applied for palette generation in color image quantization [16, 17]. The data objects have only three attributes (red, green and blue color values) but there are a high number of samples (65536). The data space consists of a sparse collection of data objects spread into a wide area, but there are also some clearly isolated and more compact clusters.

The fifth data set (*Lates mariae*) records 215 data samples from pelagic fishes on Lake Tanganyika. The data originates from a research of biology, where the occurrence of 52 different DNA fragments were tested from each fish sample (using *RAPD analysis*) and a binary decision was obtained whether the fragment was present or absent. This data has applications in studies of genetic variations among the species [18]. From the clustering point of view the set is an example of data with binary attributes. Due to only a moderate number of samples (215), the data set is an easy case for the clustering, compared to the first four sets.

The sixth data set is the standard clustering test problem SS2 of [2], pp. 103-104. The data sets contain 89 postal zones in Bavaria (Germany) and their attributes are the number of self-employed people, civil servants, clerks and manual workers in these areas. The dimensions of this set are rather small in comparison to the other sets. However, it is a commonly used data set and serves here as an example of a typical small scale clustering problem.

The data sets and their properties are summarized in Table 1. In the experiments made here, we will fix the number of clusters to 256 for the image data sets, 8 for the DNA data set, and 7 for the SS2 data set.

Bridge (256×256) | *Bridge-2* (256×256) | *Miss America* (360×288) | *House* (256×256) | *Lates mariae* (52×215)

Figure 1. Sources for the first five data sets.

Table 1. Data sets and their statistics.

Data set	Attributes	# Objects	# Clusters
Bridge	16	4096	256
Bridge-2	16	4096	256
Miss America	16	6480	256
House	3	65536	256
Lates mariae	52	215	8
SS2	4	89	7

In the clustering problem we are given a set $Q = \{X^{(i)} | i = 1,\ldots,N\}$ of K dimensional vectors $X^{(i)} = (X_1^{(i)}, X_2^{(i)},\ldots,X_K^{(i)})$. A clustering $\Pi = \{C_1, C_2,\ldots,C_M\}$ of Q has the properties

a) $C_i \neq \varnothing$, for i=1,…,M
b) $C_i \bigcap C_j = \varnothing$, for $i \neq j$ and
c) $Q = \bigcup_{i=1}^{M} C_i$

Each cluster C_i (i=1,…,M) has a representative element, called the centroid $\overline{Z}^{(i)}$, $\overline{Z}^{(i)} = \left(\sum_{X \in C_i} X\right) \Big/ |C_i|$. Here $|C_i|$ stands for the number of objects in C_i and the summation is over the objects X which belong to the cluster C_i.

Assuming that the data objects are points of an Euclidean space, the distance between two objects $X^{(i)}$ and $X^{(j)}$ can be defined by:

$$d(X^{(i)}, X^{(j)}) = \sqrt{\sum_{k=1}^{K} (X_k^{(i)} - X_k^{(j)})^2} \qquad \textbf{(1)}$$

where $X_k^{(i)}$ and $X_k^{(j)}$ stand for the k'th attribute of the objects. Let $f_\Pi(X^{(i)})$ be a mapping which gives the closest centroid in solution Π for a sample $X^{(i)}$. The distortion of the solution Π for objects $X^{(i)}$ is

$$distortion(\Pi) = \frac{1}{NK} \sum_{i=1}^{N} d(X^{(i)}, f_\Pi(X^{(i)}))^2 \qquad \textbf{(2)}$$

The problem is to determine a clustering Π_0 for which $distortion(\Pi_0) = \min$.

When we use (1) as the distance measure we assume that the attributes in the data set are numerical and have the same scale. This is the case for our first five data sets, but we note that it does not hold for the sixth set. In this case the attributes are scaled in order to have similar value ranges. Formula (2) measures the distortion of a solution by the mean square distance of the data objects and their cluster centroids. Again, this is only one of possible distortion measures.

3. Genetic algorithm

The general structure of GA is shown in Fig. 2. Each individual of the population stands for a clustering of the data. An individual is initially created by selecting M random data objects as cluster representatives and by mapping all the other data objects to their nearest representative, according to (1). In each iteration, a predefined number (S_B) of best solutions will survive to the next generation. The rest of the population is replaced

by new solutions generated in the crossover phase. We will discuss the different design alternatives of the algorithm in the following subsections.

1. Generate S random solutions for the initial generation.
2. Iterate the following T times
   2.1. Select the S_B surviving solutions for the next generation.
   2.2. Generate new solutions by crossover.
   2.3. Generate mutations to the solutions.
3. Output the best solution of the final generation.

Figure 2. A sketch for a genetic algorithm.

3.1 Representation of a solution

A solution to the clustering problem can be expressed by the pair (*partitioning table*, *cluster centroids*). These two depend on each other so that if one of them has been given, the optimal choice of the other one can be uniquely constructed. This is formalized in the following two optimality conditions [3]:

- *Nearest neighbor condition*: For a given set of cluster centroids, any data object can be optimally classified by assigning it to the cluster whose centroid is closest to the data object in respect to the distance function.

- *Centroid condition*: For a given partition, the optimal cluster representative, that is the one minimizing the distortion, is the *centroid* of the cluster members.

It is therefore sufficient to determine only the partitioning or the cluster centroids to define a solution. This implies two alternative approaches to the clustering problem:

- Centroid-based (CB)
- Partitioning-based (PB)

In the *centroid-based* variant, the sets of centroids are the individuals of the population, and they are the objects of genetic operations. Each solution is represented by an M-length array of K-dimensional vectors (see Fig. 3). The elementary unit is therefore a single centroid. This is a natural way to describe the problem in the context of *vector quantization*. In this context the set of centroids stands for a *codebook* of the application and the partitions are of secondary importance. The partitioning table, however, is needed when evaluating the distortion values of the solutions and it is calculated using the nearest neighbor condition.

In the *partitioning-based* variant the partitionings are the individuals of the population. Each partitioning is expressed as an array of N integers from the range $[1..M]$ defining cluster membership of each data object. The elementary unit (gene) is a single membership value. The centroids are calculated using the centroid condition. The partitioning-based variant is commonly used in the traditional clustering algorithms because the aim is to cluster the data with no regard to the representatives of the clusters.

Two methods reported in the literature apply the centroid-based approach [8, 9] and the other two methods apply the partitioning-based approach [7, 10]. From the genetic algorithm's point of view, the difference between the two variants lies in the realization of the crossover and mutation phases. The problem of the partitioning-based representation is that the clusters become non-convex (in the sense that objects from different parts of the data space may belong to the same cluster) if a simple random crossover method is applied, as proposed in [7, 10]. The convexity of the solutions can be restored by applying the k-means algorithm (see Section 3.5), but then the resulting cluster centroids tend to move towards to the centroid of the data set. This moves the solutions systematically to the same direction, which slows down the search. It is therefore more effective to operate with the cluster centroids than with the partitioning table. Furthermore, all practical experiments have indicated that the PB-variant is inferior to the CB-variant. We will therefore limit our discussion to the CB-variant in the rest of the paper.

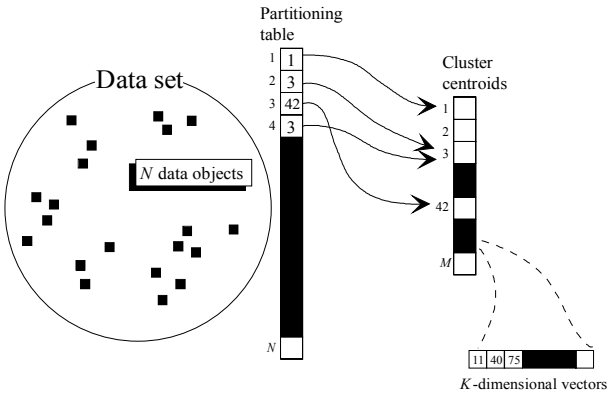

Figure 3. Illustration of a solution.

3.2 Selection methods

Selection method defines the way a new generation is constructed from the current one. It consists of the following three parts:

- Determining the S_B survivors.
- Selecting the *crossing set* of S_C solutions.
- Selecting the pairs for crossover from the crossing set.

We study the following selection methods:

- Roulette wheel selection
- Elitist selection method 1
- Elitist selection method 2

The first method is a *probability based* variant. In this variant, only the best solution survives (S_B=1) and the crossing set consists of all the solutions (S_C=S). For the

crossover, S-1 random pairs are chosen by the *roulette wheel selection*. The weighting function [7] for solution Π is

$$w(\Pi) = \frac{1}{1 + distortion(\Pi)} \quad \textbf{(3)}$$

and the probability that the ith solution is selected to crossover is

$$p(\Pi_i) = \frac{w(\Pi_i)}{\sum_{j=1}^{S} w(\Pi_j)} \quad \textbf{(4)}$$

where Π_j, $j = 1 \ldots S$, are the solutions of the current population.

In the first elitist variant S_B best individuals survive. They also compose the crossover set, i.e. S_C=S_B. All the solutions in the crossover set are crossed with each other so that the crossover phase produces $S_C \cdot (S_C\text{-}1)/2$ new solutions. The population size is thus $S = S_C \cdot (S_C+1)/2$. Here we use S_C=9 resulting in the population size of S=45.

In the second elitist variant only the best solution survives (S_B=1). Except for the number of the survivors, the algorithm is the same as the first variant; the S_C best solutions are crossed with each other giving a population size of $S = 1 + S_C \cdot (S_C\text{-}1)/2$. Here we use S_C=10 which gives a population size of S=46. Note that we can select the desired population size by dropping out a proper number of solutions.

3.3 Crossover algorithms

The object of the crossover operation is to create a new (and hopefully better) solution from the two selected parent solutions (denoted here by A and B). In the CB-variants the cluster centroids are the elementary units of the individuals. The crossover can thus be considered as the process of selecting M cluster centroids from the two parent solutions. Next we recall two existing crossover methods (*random crossover*, *centroid distance*) and introduce three new methods (*pairwise crossover*, *largest partitions*, *pairwise nearest neighbor*).

Random crossover:

Random *multipoint crossover* is performed by picking M/2 randomly chosen cluster centroids from each of the two parents in turn. Duplicate centroids are rejected and replaced by repeated picks. This is an extremely simple and quite efficient method, because there is (in the unsorted case) no correlation between neighboring genes to be taken advantage of. The method works in a similar way to the random single point crossover methods of the PB-based variants [7, 10] but it avoids the non-convexity problem of the PB approach.

Centroid distance [8, 9]:

If the clusters are sorted by some criterion, *single point crossover* may be advantageous. In [8], the clusters were sorted according to their distances from the centroid of the entire data set. In a sense, the clusters are divided into two subsets. The first subset (*central clusters*) consists of the clusters that are close to the centroid of the data set, and the second subset (*remote clusters*) consists of the clusters that are far from the data set centroid. A new solution is created by taking the central clusters from solution A and the remote clusters from solution B. Note that only the cluster centroids are taken, the data objects are partitioned using the nearest neighbor condition. The changeover point can be anything between 1 and M; we use the halfpoint ($M/2$) in our implementation. A simplified version of the same idea was considered in [9] for scalar quantization of images (K=1).

Pairwise crossover:

It is desired that the new individual should inherit different genes from the two parents. The sorting of the clusters by the centroid distance is an attempt of this kind but the idea can be developed even further. The clusters between the two solutions can be paired by searching the "nearest" cluster (in the solution B) for every cluster in the solution A. Crossover is then performed by taking one cluster centroid (by random choice) from each pair of clusters. In this way we try to avoid selecting similar cluster centroids from both parent solutions. The pairing is done in a greedy manner by taking for each cluster in A the nearest available cluster in B. A cluster that has been paired cannot be chosen again, thus the last cluster in A is paired with the only one left in B. This algorithm does not give the optimal *pairing* (*2-assignment*) but it is a reasonably good heuristic for the crossover purpose.

Largest partitions:

In the *largest partitions* algorithm the M cluster centroids are picked by a greedy heuristic based on the assumption that the larger clusters are more important than the smaller ones. This is a reasonable heuristic rule since our aim is to minimize the intracluster diversity. The cluster centroids should thus be assigned to a large concentration of data objects.

Each cluster in the solutions A and B is assigned with a number, *cluster size*, indicating how many data objects belong to it. In each phase, we pick the centroid of the largest cluster. Assume that cluster i was chosen from A. The cluster centroid C_i is removed from A to avoid its reselection. For the same reason we update the cluster sizes of B by removing the effect of those data objects in B that were assigned to the chosen cluster i in A.

Pairwise nearest neighbor:

An alternative strategy is to consider the crossover phase as a special case of the clustering problem. In fact, if we combine the cluster centroids A and B, their union can be treated as a data set of $2M$ data objects. Now our aim is to generate M clusters from this data set. This can be done by any existing clustering algorithm. Here we consider

the use of *pairwise nearest neighbor* (*PNN*) [19]. It is a variant of the so-called *agglomerative nesting algorithm* and was originally proposed for vector quantization.

The PNN algorithm starts by initializing a clustering of size $2M$ where each data object is considered as its own cluster. Two clusters are combined at each step of the algorithm. The clusters to be combined are the ones that increase the value of the distortion function least. This step is iterated M times, after which the number of the clusters has decreased to M.

3.4 Mutations

Each cluster centroid is replaced by a randomly chosen data object with a probability p. This operation is performed before the partitioning phase. We fix this probability to $p = 0.01$, which has given good results in our experiments.

3.5 Fine-tuning by the k-means algorithm

One can try to improve the algorithm by applying a few steps of the k-means algorithm for each new solution [3, 11]. The crossover operation first generates a rough estimate of the solution which is then fine-tuned by the k-means algorithm. This modification allows faster convergence of the solution than pure genetic algorithm.

Our implementation of the k-means algorithm is the following. The initial solution is iteratively modified by applying the two optimality conditions (of Section 3.1) in turn. In the first stage the centroids are fixed and the clusters are recalculated using the nearest neighbor condition. In the second stage the clusters are fixed and new centroids are calculated. The optimality conditions guarantee that the new solution is always at least as good as the original one.

4. Test results

The performance of the genetic algorithm is illustrated in Fig. 4 as a function of the number of generations for *Bridge* and *Bridge-2*. The inclusion of the k-means algorithm is essential; even the worst candidate in each generation is better than any of the candidates without k-means. The drawback of the hybridization is that the running time considerably grows as the number of k-means steps increases. Fortunately, it is not necessary to perform the k-means algorithm to its convergence but only a couple of steps (two in the present work) suffice. The results are similar for the other data sets not shown in Fig. 4.

The performance of the different crossover methods is illustrated in Fig. 5 as a function of the number of generations. The pairwise crossover and the PNN method outperform the centroid distance and random crossover methods. Of the tested methods the PNN algorithm is the best choice. It gives the best clustering with the fewest number of

iterations. Only for binary data, the pairwise crossover method obtains slightly better results in the long run.

The performance of the different selection and crossover methods is summarized in Table 2. The selection method seems to have a smaller effect on the overall performance. In most cases the elitist variants are better than the roulette wheel selection. However, for the best crossover method (PNN algorithm) the roulette wheel selection is a slightly better choice.

The above observations demonstrate two important properties of genetic algorithms for large scale clustering problems. A successful implementation of GA should direct the search efficiently but it should also retain enough genetic variation in the population. The first property is clearly more important because all ideas based on it (inclusion of k-means, PNN crossover, elitist selection) gave good results. Their combination, however, reduces the genetic variation so that the algorithm converges too quickly. Thus, the best results were reached only for the binary data sets. In the best variant, we therefore use the roulette wheel selection to compensate the loss of the genetic variation.

Among the other parameters, the amount of mutations had only a small effect on the performance. Another interesting but less important question is whether extra computing resources should be used to increase the generation size or the number of iteration rounds. Additional tests have shown that the number of iteration rounds has a slight edge over the generation size but the difference is small and the quality of the best clustering depends mainly on the total number of candidate solutions tested.

The best variant of GA is next compared to other existing clustering algorithms. *Simulated annealing algorithm* (SA) is implemented here as proposed in [20]. The method is basically the same as the k-means algorithm but random noise is added to the cluster centroids after each step. A logarithmic temperature schedule decreases the temperature by 1 % after each iteration step.

The results for k-means, PNN, SA and GA are summarized in Table 3. We observe that GA clearly outperforms the other algorithms used in comparison. SA can match the GA results only for the two smallest test sets, and if the method is repeated several times, as shown in Table 4. The statistics show also that GA is relatively independent on the initialization whereas the results of k-means have much higher variation. According to the *Student's t-test* (independent samples with no assumptions on the equality of the variances) the difference between the GA results and the k-means/SA results are significant (with risk of wrong decision $p<0.05$) except for SA and GA results for *Bridge*.

It was proposed in [10] that the initial population would be constructed as the output of S independent runs of the k-means algorithm. However, this approach had in our tests no benefits compared to the present approach where k-means is applied in each generation. Only moderate improvement is achieved if k-means was applied to the initial population only. Furthermore, if k-means iterations are already integrated in each iteration, random initialization can be used as well. For a more detailed discussion of various hybridizations of GA and k-means, see [21].

The drawback of GA is its high running time. For *Bridge* the running times (min:sec) of the algorithms (k-means, SA, PNN, GA) were 0:38, 13:03, 67:00 and 880:00 respectively. Higher quality clusterings are thus obtained at the cost of larger running time.

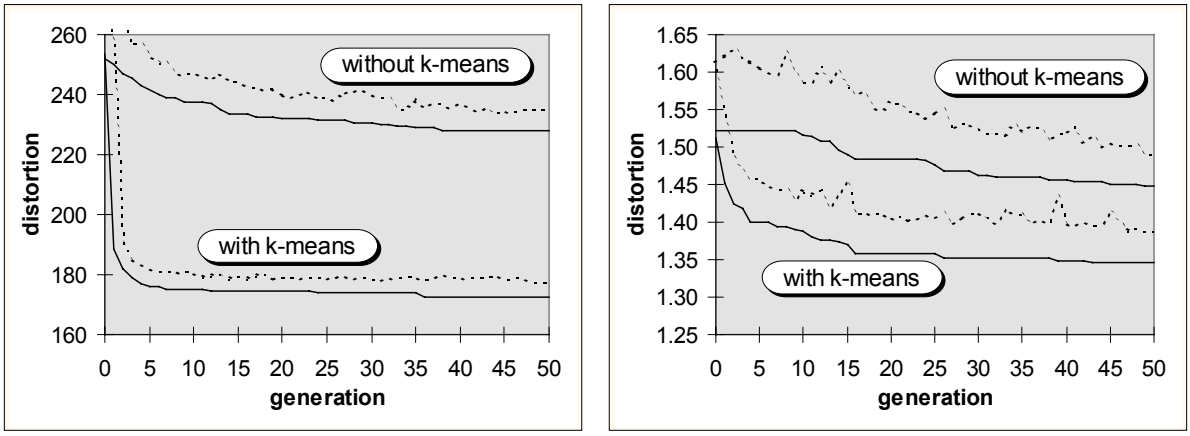

Figure 4. Quality of the best (solid lines) and worst candidate solutions (broken lines) as a function of generation number for *Bridge* (left) and for *Bridge-2* (right). The elitist selection method 1 was applied with the random crossover technique. Two steps of the k-means algorithm were applied.

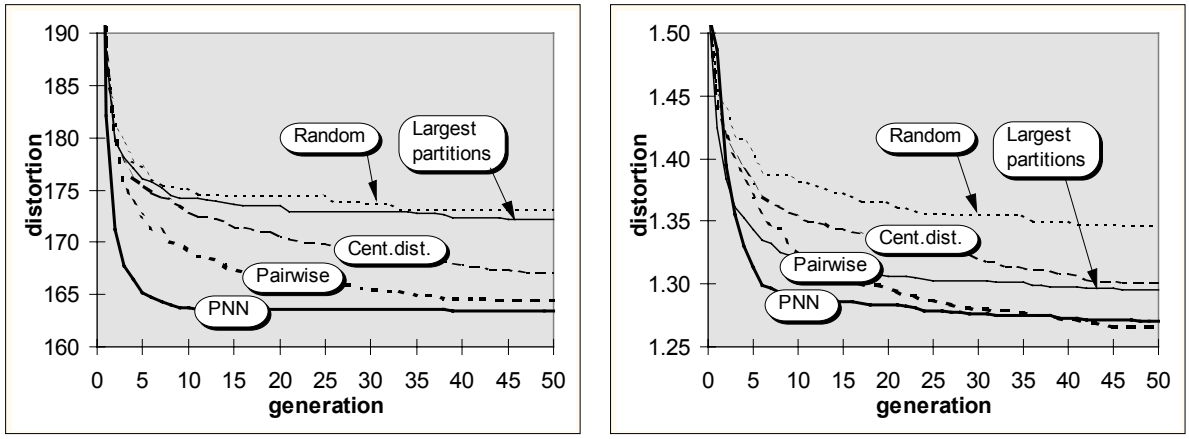

Figure 5. Convergence of the various crossover algorithms for *Bridge* (left) and for *Bridge-2* (right). The elitist selection method 1 was used, and two steps of the k-means algorithm were applied.

Table 2. Performance comparison of the selection and crossover techniques. GA results are averaged from five test runs. The distortion values for *Bridge* are due to (2). For *Bridge-2* the table shows the average number of distorted attributes per data object (varying from 0 to *K*). Population size is 45 for elitist method 1, and 46 for method 2.

Bridge	**Random crossover**	**Centroid distance**	**Pairwise crossover**	**Largest partitions**	**PNN algorithm**
Roulette wheel	174.77	172.09	168.36	178.45	**162.09**
Elitist method 1	173.46	168.73	164.34	172.44	**162.91**
Elitist method 2	173.38	168.21	164.28	171.93	**162.90**
Bridge-2	**Random crossover**	**Centroid distance**	**Pairwise crossover**	**Largest partitions**	**PNN algorithm**
Roulette wheel	1.40	1.34	1.34	1.30	**1.28**
Elitist method 1	1.34	1.30	**1.27**	1.30	1.28
Elitist method 2	1.35	1.30	**1.26**	1.29	1.27

Table 3. Performance comparison of various algorithms. In GA, the roulette wheel selection method and the PNN crossover with two steps of the k-means algorithm were applied. The k-means and SA results are averages from 100 test runs; GA results from 5 test runs.

	k-means	**PNN**	**SA**	**GA**
Bridge	179.68	169.15	162.45	**162.09**
Miss America	5.96	5.52	5.26	**5.18**
House	7.81	6.36	6.03	**5.92**
Bridge-2	1.48	1.33	1.52	**1.28**
Lates mariae	5.28	5.41	5.19	**4.56**
SS2	1.14	0.34	0.32	**0.31**

Table 4. Statistics (min, max, and standard deviation) of the test runs, see Table 3 for parameter settings.

min - max st. dev.	**k-means**	**SA**	**GA**
Bridge	176.85 - 183.93 1.442	162.08 - 163.29 0.275	161.75 - 162.39 0.305
Miss America	5.80 - 6.11 0.056	5.24 - 5.29 0.013	5.17 - 5.18 0.005
House	7.38 - 8.32 0.196	5.97 - 6.08 0.023	5.91 - 5.93 0.009
Bridge-2	1.45 - 1.52 0.015	1.43 - 1.50 0.014	1.28 - 1.29 0.002
Lates mariae	4.56 - 6.67 0.457	4.56 - 5.28 0.166	4.56 - 4.56 0.000
SS2	0.40 - 2.29 0.899	0.31 - 0.35 0.009	0.31 - 0.31 0.000

5. Conclusions

GA solutions for large scale clustering problems were studied. The implementation of a GA-based clustering algorithm is quite simple and straightforward. However, problem specific modifications were needed because of the nature of the data. New candidate solutions are created in the crossover but they are too arbitrary to give a reasonable solution to the problem. Thus, the candidate solutions must be fine-tuned by a small number of steps of the k-means algorithm.

The main parameters of GA for the clustering problems studied here are the inclusion of k-means steps, and the crossover technique. The mutation probability and the choice of selection method seem to be of minor importance. The centroid-based representation for the solution was applied. The results were promising for this configuration of GA. The results of GA (when measured by the intracluster diversity) were better than those of k-means and PNN. For non-binary data sets SA gave competitive results to GA with less computing efforts, but for binary data sets GA is still superior. The major drawback of GA is the high running time, which may in some cases prohibit the use of the algorithm.

Acknowledgements

The work of Pasi Fränti was supported by a grant from the Academy of Finland.

References

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

[2] H. Späth, *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*, Ellis Horwood Limited, West Sussex, UK, 1980.

[3] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[4] M. Gyllenberg, T. Koski and M. Verlaan, Classification of binary vectors by stochastic complexity. *Journal of Multivariate Analysis*, **63** (1), 47-72, October 1997.

[5] K.S. Al-Sultan and M.M. Khan, Computational experience on four algorithms for the hard clustering problem. *Pattern Recognition Letters*, **17**, 295-308, 1996.

[6] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, 1989.

[7] V. Delport and M. Koschorreck, Genetic algorithm for codebook design in vector quantization. *Electronics Letters*, **31**, 84-85, January 1995.

[8] J.S. Pan, F.R. McInnes and M.A. Jack, VQ codebook design using genetic algorithms. *Electronics Letters*, **31**, 1418-1419, August 1995.

[9] P. Scheunders, A genetic Lloyd-Max quantization algorithm. *Pattern Recognition Letters*, **17**, 547-556, 1996.

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

[11] J.B. McQueen, Some methods of classification and analysis of multivariate observations, *Proc. 5th Berkeley Symp. Mathemat. Statist. Probability* **1**, 281-296. Univ. of California, Berkeley, USA, 1967.

[12] N.M. Nasrabadi and R.A. King, Image coding using vector quantization: a review. *IEEE Transactions on Communications*, **36**, 957-971, 1988.

[13] C.F. Barnes, S.A. Rizvi and N.M. Nasrabadi, Advances in residual vector quantization: a review. *IEEE Transactions on Image Processing*, **5**, 226-262, February 1996.

[14] P. Fränti, T. Kaukoranta and O. Nevalainen, On the design of a hierarchical BTC-VQ compression system, *Signal Processing: Image Communication*, **8**, 551-562, 1996.

[15] J.E. Fowler Jr., M.R. Carbonara and S.C. Ahalt, Image coding using differential vector quantization. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3 (5), pp. 350-367, October 1993.

[16] M.T. Orchard, C.A. Bouman, Color quantization of images. *IEEE Transactions on Signal Processing*, **39**, 2677-2690, December 1991.

[17] X. Wu, YIQ Vector quantization in a new color palette architecture. *IEEE Transactions on Image Processing*, **5**, 321-329, February 1996.

[18] L. Kuusipalo, Diversification of endemic Nile perch *Lates Mariae* (Centropomidae, Pisces) populations in Lake Tanganyika, East Africa, studied with RAPD-PCR. *Proc. Symposium on Lake Tanganyika Research*, 60-61, Kuopio, Finland, 1995.

[19] W.H. Equitz, A new vector quantization clustering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37**, 1568-1575, October 1989.

[20] K. Zeger and A. Gersho, Stochastic relaxation algorithm for improved vector quantiser design. *Electronics Letters*, **25** (14), 896-898, July 1989.

[21] P. Fränti, J. Kivijärvi, T. Kaukoranta and O. Nevalainen, Genetic algorithms for codebook generation in vector quantization, *Proc. 3rd Nordic Workshop on Genetic Algorithms*, 207-222, Helsinki, Finland, August 1997.