[8] H. H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *Comput. J.*, vol. 3, pp. 175–184, 1960.
[9] C. G. Schaefer, "The ARGOT strategy: Adaptive representation genetic optimizer technique," in *Genetic Algorithms Applications: Proc. 2nd Int. Conf.*, 1987, pp. 50–58.
[10] N. N. Schraudolph and R. K. Belew, "Dynamic parameter encoding for genetic algorithms," *Mach. Learn.*, vol. 9, no. 1, pp. 9–21, 1992.
[11] R. J. Streifel, R. von Doenhoff, R. J. Marks II, J. J. Choi, and M. Healy, "Application of genetic algorithms to hydraulic brake system parameter identification," The Boeing Company, Seattle, WA, Doc. D6-81795TN.
[12] E. C. K. Tsao, J. C. Bezdek, and N. R. Pal, "Fuzzy Kohonen clustering networks," *Pattern Recognit.*, vol. 27, no. 5, pp. 757–764, 1994.
[13] D. Whitley, K. Mathias, and P. Fitzhorn, "Delta coding: An iterative search strategy for genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms*, L. Booker and R. Belew, Eds. San Mateo, CA: Morgan Kauffman, 1991, pp. 77–84.
[14] L. A. Zadeh, "Fuzzy sets," *Inf. Contr.*, vol. 8, pp. 338–353, 1965.

Genetic K-Means Algorithm

K. Krishna and M. Narasimha Murty

***Abstract*—In this paper, we propose a novel hybrid genetic algorithm (GA) that finds a globally optimal partition of a given data into a specified number of clusters. GA's used earlier in clustering employ either an expensive crossover operator to generate valid child chromosomes from parent chromosomes or a costly fitness function or both. To circumvent these expensive operations, we hybridize GA with a classical gradient descent algorithm used in clustering viz., K-means algorithm. Hence, the name genetic K-means algorithm (GKA). We define K-means operator, one-step of K-means algorithm, and use it in GKA as a search operator instead of crossover. We also define a biased mutation operator specific to clustering called distance-based-mutation. Using finite Markov chain theory, we prove that the GKA converges to the global optimum. It is observed in the simulations that GKA converges to the best known optimum corresponding to the given data in concurrence with the convergence result. It is also observed that GKA searches faster than some of the other evolutionary algorithms used for clustering.**

***Index Terms*— Clustering, genetic algorithms, global optimization, K-means algorithm, unsupervised learning.**

I. Introduction

Evolutionary algorithms are stochastic optimization algorithms based on the mechanism of natural selection and natural genetics [1]. They perform parallel search in complex search spaces. Evolutionary algorithms include genetic algorithms, evolution strategies and evolutionary programming. We deal with genetic algorithms in this paper. Genetic algorithms (GA's) were originally proposed by Holland [2]. GA's have been applied to many function optimization problems and are shown to be good in finding optimal and near optimal solutions. Their robustness of search in large search spaces and their domain independent nature motivated their applications in various fields like pattern recognition, machine learning, VLSI design, etc. In this paper, we propose an algorithm, that is a modification of GA, for clustering application.

Clustering has been effectively applied in a variety of engineering and scientific disciplines such as psychology, biology, medicine, computer vision, communications, and remote sensing. Cluster analysis organizes data (a set of patterns, each pattern could be a vector measurements) by abstracting underlying structure. The grouping is done such that patterns within a group (cluster) are more similar to each other than patterns belonging to different groups. Thus, organization of data using cluster analysis employs some dissimilarity measure among the set of patterns. The dissimilarity measure is defined based on the data under analysis and the purpose of the analysis. Various types of clustering algorithms have been proposed to suit different requirements. Clustering algorithms can be broadly classified into hierarchical and partitional algorithms based on the structure of abstraction. Hierarchical clustering algorithms construct a hierarchy of partitions, represented as a *dendrogram* in which each partition is nested within the partition at the next level in the hierarchy. Partitional clustering algorithms generate a single partition, with a specified or estimated number of nonoverlapping clusters, of the data in an attempt to recover natural groups present in the data. In this paper, we confine our attention to partitional clustering of a given set of real-valued vectors, where the dissimilarity measure between two vectors is the Euclidean distance between them.

One of the important problems in partitional clustering is to find a partition of the given data, with a specified number of clusters, that minimizes the total within cluster variation (TWCV) (which is defined below). We address this problem, viz., minimization of TWCV, in the present paper. In general, partitional clustering algorithms are iterative and hill climbing and usually they converge to a local minimum. Further, the associated objective functions are highly nonlinear and multimodal. As a consequence, it is very difficult to find an optimal partition of the data using hill climbing techniques. The algorithms based on combinatorial optimization such as integer programming, dynamic programming and, branch and bound methods are expensive ever for moderate number of data points and moderate number of clusters. A detailed discussion on clustering algorithms can be found in [3].

The simplest and most popular among iterative and hill climbing clustering algorithms is the K-means algorithm (KMA). As mentioned above, this algorithm may converge to a suboptimal partition. Since stochastic optimization approaches are good at avoiding convergence to a locally optimal solution, these approaches could be used to find a globally optimal solution. The stochastic approaches used in clustering include those based on simulated annealing, genetic algorithms, evolution strategies and evolutionary programming [4]–[11]. Typically, these stochastic approaches take a large amount of time to converge to a globally optimal partition. In this paper, we propose an algorithm based on GA, prove that it converges to the global optimum with probability one and compare its performance with that of some of these algorithms.

Genetic algorithms (GA's) work on a coding of the parameter set over which the search has to be performed, rather than the parameters themselves. These encoded parameters are called *solutions* or *chromosomes* and the objective function value at a solution is the objective function value at the corresponding parameters. GA's solve optimization problems using a population of a fixed number, called the *population size*, of solutions. A solution consists of a string of symbols, typically binary symbols. GA's evolve

Manuscript received September 4, 1995; revised August 27, 1997 and March 10, 1998.

K. Krishna is with the Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012, India (e-mail: kkrishna@ee.iisc.ernet.in).

M. N. Murthy is with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India (e-mail: mnm@csa.iisc.ernet.in).

Publisher Item Identifier S 1083-4419(99)00770-0.

over *generations*. During each generation, they produce a new population from the current population by applying genetic operators viz., *natural selection, crossover, and mutation*. Each solution in the population is associated with a figure of merit (fitness value) depending on the value of the function to be optimized. The selection operator selects a solution from the current population for the next population with probability proportional to its fitness value. Crossover operates on two solution strings and results in another two stings. Typical crossover operator exchange the segments of selected stings across a crossover point with a probability. The mutation operator toggles each position in a string with a probability, called the *mutation probability*. For a detail study on GA, readers are referred to [12]. Recently, it has been shown that the GA's that maintain the best discovered solution either before or after the selection operator asymptotically converge to the global optimum [13].

There have been many attempts to use GA's for clustering [7], [8], [11]. Even though all these algorithms, because of mutation, may converge to the global optimum, they face the following problems in terms of computational efforts. In the algorithms where the representation of chromosome is such that it favors easy crossover, the fitness evaluation is very expensive as in [7]. In the algorithms where the fitness evaluation is simple, either the crossover operation is complicated or it needs to be repeatedly applied on chromosomes to get legal strings [8], [11]. In this sense, selection and crossover are complementary to each other in terms of computational complexity.

GA's perform most efficiently when the representation of the search space under consideration has a natural structure that facilitates efficient coding of solutions. Also, genetic operators defined on these codes must produce valid solutions with respect to the problem. Thus, in order to efficiently use GA's in various applications, one has to specialize GA's to the problems under consideration by hybridizing them with the traditional gradient descent approaches. A hybrid GA that retains, if possible, the best features of the existing algorithm, could be the best algorithm for the problem under consideration. Davis also made these observations in his handbook [14]. Since KMA is computationally attractive, apart from being simple, we chose this algorithm for hybridization. The resulting hybrid algorithm is called the *genetic K-means algorithm* (GKA). We use the K-means operator, one step of KMA, in GKA instead of the crossover operator used in conventional GA's. We also define a biased mutation operator specific to clustering, called distance based mutation, and use it in GKA. Thus, GKA combines the simplicity of the K-means algorithm and the robust nature of GA's. Using finite Markov chain theory, we derive conditions on the parameters of GKA for its convergence to a globally optimal partition.

We conduct experiments to analyze the significance of the operators used in GKA and the performance of GKA on different data sets and varying sizes of search spaces. We show through simulations that even if many duplicates of KMA starting with different initial partitions are run, the best partition obtained is not necessarily a global optimum, whereas almost every run of GKA eventually converge to a globally optimal partition. We also compare the performance of GKA with that of some of the algorithms based on GA, evolution strategies and evolutionary programming, which possibly converge to a global optimum, and show that GKA is faster than them. In the next section, the statement of the problem under consideration along with a brief description of KMA is given. The proposed algorithm is explained in Section IV. In Section V, the conditions on the parameters of GKA are derived which ensure its convergence to the global optimum. The algorithm is tested on British town data (BTD) and German town data (GTD). Details of simulations and results are presented in Section VI. We conclude with a summary of the contributions of this paper in Section VII.

II. Partitional Clustering

The main objective of the clustering algorithm under consideration is to partition a collection of n given patterns, each pattern is a vector of dimension d, into K groups such that this partition minimizes the TWCV, which is defined as follows.

Let $\{x_i,\ i = 1,\ 2,\ \cdots,\ n\}$ be the set of n patterns. Let x_{ij} denote jth feature of x_i. Define for $i = 1,\ 2,\ \cdots,\ n$ and $k = 1,\ 2,\ \cdots,\ K$,

$$w_{ik} = \begin{cases} 1, & \text{if } i\text{th pattern belongs to } k\text{th cluster,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Then, the matrix $W = [w_{ij}]$ has the properties that

$$w_{ij} \in \{0,\ 1\} \text{ and }, \sum_{j=1}^{K} w_{ij} = 1. \tag{2}$$

Let the centroid of the kth cluster be $c_k = (c_{k1},\ c_{k2},\ \cdots,\ c_{kd})$, then

$$c_{kj} = \frac{\sum_{i=1}^{n} w_{ik} x_{ij}}{\sum_{i=1}^{n} w_{ik}}. \tag{3}$$

The within-cluster variation of kth cluster is defined as

$$S^{(k)}(W) = \sum_{i=1}^{n} w_{ik} \sum_{j=1}^{d} (x_{ij} - c_{kj})^2 \tag{4}$$

and the total within-cluster variation (TWCV) is defined as

$$S(W) = \sum_{k=1}^{K} S^{(k)} = \sum_{k=1}^{K} \sum_{i=1}^{n} w_{ik} \sum_{j=1}^{d} (x_{ij} - c_{kj})^2 \tag{5}$$

Sometimes this is also called *square-error (SE) measure*. The objective is to find a $W^* = [w_{ik}^*]$ which minimizes $S(W)$, i.e.,

$$S(W^*) = \min_{W} \{S(W)\}.$$

KMA is the most popularly used algorithm to find a partition that minimizes SE measure. There are many variations of the KMA [3]. We briefly explain below one of its simple variant that will be used in the development of GKA. KMA is an iterative algorithm. It starts with a random configuration of cluster centers. In every iteration, each pattern is assigned to the cluster whose center is the closest center to the pattern among all the cluster centers. The cluster centers in the next iteration are the centroids of the patterns belonging to the corresponding clusters. The algorithm is terminated when there is no reassignment of any pattern from one cluster to another or the SE measure ceases to decrease significantly after an iteration. A major problem with this algorithm is that it is sensitive to the selection of initial partition and may converge to a local minimum of SE if the initial partition is not properly chosen.

III. Genetic K-Means Algorithm

As in GA, GKA maintains a population of coded solutions. The population is initialized randomly and is evolved over generations; the population in the next generation is obtained by applying genetic operators on the current population. The evolution takes place until a terminating condition is reached. The genetic operators that are used in GKA are the selection, the distance based mutation and the K-means operator. In this section we explain GKA by specifying the coding and initialization schemes and, the genetic operators.

1) Coding: Here the search space is the space of all W matrices that satisfy (2). A natural way of coding such W into a string, s_W, is to consider a chromosome of length n and allow each allele in the chromosome to take values from $\{1, 2, \cdots, K\}$. In this case, each allele corresponds to a pattern and its value represents the cluster number to which the corresponding pattern belongs. This is possible because [refer to (1)] for all i, $w_{ik} = 1$ for only one k. This type of coding is *string-of-group-numbers encoding* [8]. GKA maintains a population of such strings.

2) Initialization: The initial population $\mathcal{P}(0)$ is selected randomly. Each allele in the population can be initialized to a cluster number randomly selected from the uniform distribution over the set $\{1, \cdots, K\}$. In this case, we may end up with *illegal strings*, strings representing a partition in which some clusters are empty, with some nonzero probability. This is avoided by assigning p, the greatest integer which is less than n/K, randomly chosen data points to each cluster and the rest of the points to randomly chosen clusters.

3) Selection: The selection operator randomly selects a chromosome from the previous population according to the distribution given by

$$P(s_i) = \frac{F(s_i)}{\sum_{j=1}^{N} F(s_j)} \tag{6}$$

where $F(s_i)$ represents fitness value of the string s_i in the population and is defined in the next paragraph. We use the roulette wheel strategy for this random selection.

Solutions in the current population are evaluated based on their merit to survive in the next population. This requires that each solution in a population be associated with a figure of merit or a fitness value. In the present context, the fitness value of a solution string s_W depends on the total within-cluster variation $S(W)$. Since the objective is to minimize $S(W)$, a solution string with relatively small square error must have relatively high fitness value. There are many ways of defining such a fitness function [12]. We use the σ-truncation mechanism for this purpose. Let $f(s_W) = -S(W)$, $g(s_W) = f(s_W) - (\overline{f} - c \cdot \sigma)$, where $\overline{f}$ and σ denote the average value and standard deviation of $f(s_W)$ in the current population, respectively. c is constant between 1 and 3. Then, the fitness value of s_W, $F(s_W)$, is given by

$$F(s_W) = \begin{cases} g(s_W), & \text{if } g(s_W) \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

4) Mutation: Mutation changes an allele value depending on the distances of the cluster centroids from the corresponding data point. It may be recalled that each allele corresponds to a data point and its value represents the cluster to which the data point belongs. An operator is defined such that the probability of changing an allele value to a cluster number is more if the corresponding cluster center is closer to the data point. To apply the mutation operator to the allele $s_W(i)$ corresponding to pattern x_i, let $d_j = d(x_i, c_j)$ be the Euclidean distance between x_i and c_j. Then, the allele is replaced with a value chosen randomly from the following distribution:

$$p_j = \Pr\{s_W(i) = j\} = \frac{c_m d_{\max} - d_j}{\sum_{i=1}^{K} (c_m d_{\max} - d_i)} \tag{8}$$

where c_m[1] is a constant usually ≥ 1 and $d_{\max} = \max_j \{d_j\}$. In case of a partition with one or more than one singleton clusters, the above mutation may result in the formation of empty clusters with a nonzero probability. It may be noted that smaller the number of clusters, larger the SE measure; so empty clusters must be avoided. A quick way of detecting the possibility of empty cluster formation is check whether the distance of the data x_i from its cluster center $c_{s_W(i)}$ is greater than zero. It may be noted that $d_{s_W(i)} = 0$ even in the case of nonsingleton clusters wherein the data point and the center of the cluster are the same. Thus, an allele is mutated only when $d_{s_W(i)} > 0$. The strings that represent K nonempty clusters are called *legal strings*; otherwise, they are called *illegal strings*. Each allele in a chromosome is mutated as described above with a probability P_m, called mutation probability. We call this mutation DBM1 in the sequel. It will be shown in Section V that this mutation helps in reaching better solutions. A pseudo-code of the operator is given below.

```
Mutation(s_W)
{ for i = 1 to n
  { if (drand() < P_m)
    { Calculate cluster centers, c_j's,
    corresponding to s_W;
     for j = 1 to K, d_j = d(x_i, c_j);
     if (d_{s_W(i)} > 0)
       { d_max = max{d_1, d_2, ..., d_K}
         for j = 1 to K,
             p_j = (c_m d_max - d_j) / sum_{k=1}^{K} (c_m d_max - d_k)
         s_W(i) = a number, randomly selected from
                  {1, 2, ..., K} according to the
                  distribution {p_1, p_2, ..., p_K};
       }
    }
  }
}
( drand() returns a uniformly distributed random
number in the range [0, 1])
```

5) K-Means Operator: The algorithm with the above selection and mutation operators may take more time to converge, since the initial assignments are arbitrary and the subsequent changes of the assignments are probabilistic. Moreover, the mutation probability is forced to assume a low value because high values of P_m lead to oscillating behavior of the algorithm. To improve this situation, a one-step K-means algorithm, named K-means operator (KMO), is introduced. Let s_W be a string. The following two steps constitute KMO on s_W which yields $s_{\tilde{W}}$:

1) calculate cluster centers using (3) for the given matrix W;
2) reassign each data point to the cluster with the nearest cluster center and thus form $\tilde{W}$.

There is a penalty to be paid for the simplicity of this operator. The resulting string $s_{\tilde{W}}$ may represent a partition with empty clusters, i.e., KMO may result in illegal strings. We convert illegal strings to legal strings by creating desired number of new singleton clusters. This is done by placing in each empty cluster a pattern x from the cluster C with the maximum within-cluster variation [refer to (4)]. x is the farthest from the cluster center of the cluster C. Since KMO and DBM1 are applied again and again on these strings, it should not matter how the splitting is done. We chose to do as above because this technique is found to be effective and computationally less expensive.

6) GKA: A pseudo-code for GKA is given in Fig. 1. To start with, the initial population is generated as mentioned above and the subsequent populations are obtained by the application of selection, DBM1 and KMO over the previous population. The algorithm is terminated when the limit on the number of generations is exceeded.

[1] c_m is introduced because, in Section IV, we need p_j to be nonzero for all j to prove the convergence of GKA. This forces c_m to be strictly greater than 1.

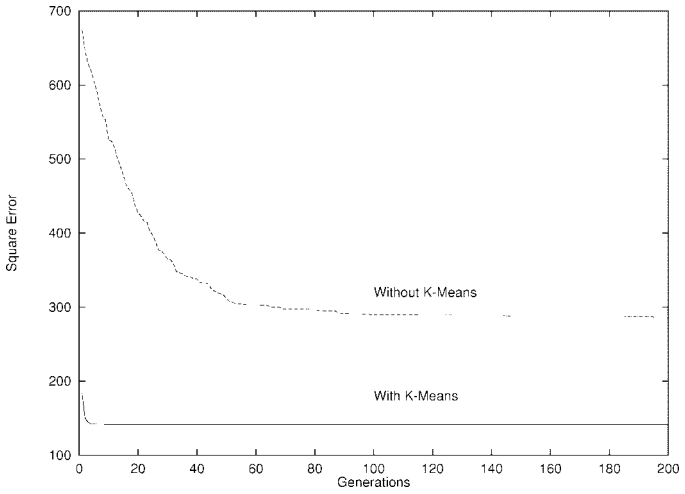

Fig. 1. GKA with and without K-mean pass on BTD.

The output of the algorithm is the best solution encountered during the evolution of the algorithm.

IV. Analysis

It has been shown using finite Markov chain theory that the canonical genetic algorithms converge to the global optimum [13]. We prove the global convergence of GKA along similar lines by deriving conditions on the parameters of GKA that ensure the global convergence.

Consider the process $\{\mathcal{P}(t)\}_{t\geq 0}$, where $\mathcal{P}(t)$ represents the population maintained by GKA at generation t. The state space of this process is the space of all possible populations $\mathcal{S}$ and the states can be numbered from 1 to $|\mathcal{S}|$. As mentioned earlier, the state space is restricted to the populations containing legal strings, i.e., strings representing partitions with K nonempty clusters. From the definition of GKA, $\mathcal{P}(t+1)$ can be determined completely by $\mathcal{P}(t)$, i.e.,

$$\begin{aligned}\Pr\{\mathcal{P}(t) = p_t | \mathcal{P}(t-1) = p_{t-1}, \cdots, \mathcal{P}(0) = p_0\} \\ = \Pr\{\mathcal{P}(t) = p_t | \mathcal{P}(t-1) = p_{t-1}\}\end{aligned}$$

```
GeneticK-Means Algorithm
Input:
   Mutation Probability, P_m;
   Population size, N;
   Maximum number of generation, MAX_GEN;
Output: Solution string, s*;
{  Initialize the population, P;
   geno = MAX_GEN;
   s* = P_1; (P_i is the ith string in P)
   while (geno > 0)
   {     Calculate Fitness values of strings in P;
         P̂ = Selection (P);
         for i = 1 to N, P_i = Mutation(P̂_i);
         for i = 1 to N, K-Means(P_i);
         s = string in P such that the corresponding weight
               matrix W_s has the minimum SE measure;
         if (S(W_s*) > S(W_s)), s* = s;
         geno = geno - 1;
   }
   output s*;
}
```

Hence $\{\mathcal{P}(t)\}_{t\geq 0}$ is a Markov chain. Also, the transition probabilities are independent of the time instant, i.e., if

$$p_{ij}(t) = \Pr\{\mathcal{P}(t) = p_j | \mathcal{P}(t-1) = p_i\}$$

then $p_{ij}(s) = p_{ij}(t)$ for all $p_i, p_j \in \mathcal{S}$ and for all $s, t \geq 1$. Therefore, $\{\mathcal{P}(t)\}_{t\geq 0}$ is a time-homogeneous finite Markov chain. Let $\mathbf{P} = (p_{ij})$ be the *transition matrix* of the process $\{\mathcal{P}(t)\}_{t\geq 0}$. The entries of the matrix $\mathbf{P}$ satisfy $p_{ij} \in [0, 1]$ and $\sum_{j=1}^{|\mathcal{S}|} p_{ij} = 1\ \forall i \in \mathcal{S}$. Any matrix whose entries satisfy the above conditions is called a *stochastic matrix*. Some definitions are given below which will be used in the rest of this section.

1) Definition 1: A square matrix $\mathbf{A}: m \times m$ is said to be *positive*, if $a_{ij} > 0 \forall i, j \in \{1, 2, \cdots, m\}$ and, is said to be *primitive*, if there exists a positive integer k such that $\mathbf{A}^k$ is positive. A square matrix is said to be *column-allowable*, if it has at least one positive entry in each column. □

In the following theorem, it is required that $\mathbf{P}$ be a primitive matrix. So, first we investigate the conditions on the operators which make the matrix $\mathbf{P}$ primitive. The probabilistic changes of the chromosome within the population caused by the operators used in GKA are captured by the transition matrix $\mathbf{P}$, which can be decomposed in a natural way into a product of stochastic matrices $\mathbf{P} = \boldsymbol{K} \cdot \boldsymbol{M} \cdot \boldsymbol{S}$, where $\boldsymbol{K}$, $\boldsymbol{M}$, and $\boldsymbol{S}$ describe the intermediate transitions caused by K-means, mutation and selection operators respectively.

2) Proposition 2: Let $\boldsymbol{K}$, $\boldsymbol{M}$, and $\boldsymbol{S}$ be stochastic matrices, where $\boldsymbol{M}$ is positive and $\boldsymbol{S}$ is column-allowable. Then the product $\boldsymbol{K} \cdot \boldsymbol{M} \cdot \boldsymbol{S}$ is positive.

Since every positive matrix is primitive, it is therefore, enough to find the conditions which make $\boldsymbol{M}$ positive and $\boldsymbol{S}$ column-allowable.

3) Conditions on Mutation: The matrix $\boldsymbol{M}$ is positive if any string $s \in \mathcal{S}$ can be obtained from any another string on application of the corresponding mutation operator. The mutation operator (DBM1) defined in the previous section does not ensure this because the alleles are not mutated if $d_{s_{W(i)}} = 0$. DBM1 is slightly modified to make the corresponding transition matrix $\boldsymbol{M}$ positive. The modified operator is referred to as DBM2. DBM2 changes each allele, irrespective of the value of $d_{s_{W(i)}}$, according to the distribution in (8). This may result in an illegal string, with some small nonzero probability, in the cases where $d_{s_{W(i)}} = 0$. If this operation results in an illegal string, the above procedure is repeated till we get a legal string. If c_m in (8) is strictly greater than one then all p_j's are strictly greater than zero. This implies that DBM2 can change any legal string to any other legal string with nonzero probability. Hence, the transition matrix $\boldsymbol{M}$ corresponding to DBM2 is positive.

4) Conditions on Selection: The probability of survival of a string in the current population depends on the fitness value of the string; so is the transition matrix due to selection, $\boldsymbol{S}$. Very little can be said about $\boldsymbol{S}$ if the fitness function is defined as in (7). The following modification to the fitness function will ensure the column-allowability of $\boldsymbol{S}$. Let

$$F(s_W) = c_s \cdot S_{\max} - S(W) \tag{9}$$

where $S_{\max}$ is the maximum square error that has been encountered till the present generation and $c_s > 1$. Then the fitness values of all the strings in the population are strictly positive and hence the probability of survival of any string in the population after selection is also strictly positive. Therefore, the probability that selection does

not alter the present state, s_{ii}, can be bounded as follows:

$$s_{ii} \geq \frac{F(s_1)}{\sum_{l=1}^{N} F(s_l)} \cdots \frac{F(s_N)}{\sum_{l=1}^{N} F(s_l)}$$
$$= \frac{\prod_{l=1}^{N} F(s_l)}{\left(\sum_{l=1}^{N} F(s_l)\right)^N} > 0 \; \forall \, i \in \mathcal{S}$$

where s_l is the lth string in the population under consideration. Even though this bound changes with the generation, it is always strictly positive. Hence, under this modification $\boldsymbol{S}$ is column-allowable.

5) Theorem 3: Let W_s be the weight matrix (1) corresponding to the string s. Let $X(t) = S(W_{s^*(t)})$, where $s^*(t)$ is the string, with the least SE measure, encountered during the evolution of GKA till the time instant t. Let DBM2, with $c_m > 1$, be the mutation operator and the fitness function be as defined in (9). Then

$$\lim_{t \to \infty} \Pr \{X(t) = S^*\} = 1 \tag{10}$$

where $S^* = \min \{S(i) | i \in \mathcal{T}\}$, $\mathcal{T}$ is the set of all legal strings.

6) Sketch of the Proof: It is proved [13, Theorem 6] that a canonical GA, whose transition matrix $\mathbf{P}$ is primitive and, which maintains the best solution found over time, converges to the global optimum in the sense given in (10). Under the hypothesis of the theorem, the transition matrix of GKA is primitive. This is evident from the above discussion. It may be noted that the GKA defined in Fig. 1 maintains the best solution found till the current time instant. Thus, the theorem follows from [13, Theorem 6]. □

The above theorem implies that $X(t)$, the least SE measure of the strings encountered by GKA till the instant t, converges to the global optimum S^*, with probability 1.

7) Remark 1—On Mutation: DBM1 is less computationally expensive than DBM2. Since the performance of GKA remains same with either of the mutation operators, we have used the earlier one in simulations. It has been observed that small variations in the value of c_m do not significantly affect the performance of GKA. Hence, in simulation results reported in the next section, c_m is set to one.

8) Remark 2—On Fitness Function: GKA was simulated using (7) as well as (9) as fitness functions. It has been noted that σ-truncation mechanism performed much better than the other technique. So, we use σ-truncation mechanism for the experimental study.

V. Experimental Study

We conducted experiments using GKA on two data sets. The two data sets were German town data (GTD) [15] and British town data (BTD) [16]. GTD consists of Cartesian coordinates of 59 towns in Germany. BTD consists of 50 samples each of four variables corresponding to the first four principal components of the original data [16]. We report the results of four sets of experiments in this section. First, the significance of KMO and DBM1 in finding the global optimum is examined. The performance of GKA on GTD and BTD for different number of clusters is considered next. Third, we compare the performance of many duplicates of KMA with that of GKA. Finally, the performance of GKA is compared with that of the algorithms based on evolutionary strategies (ES) and evolutionary programming (EP) [11], GA and the greedy algorithms viz., alternating first-best (AFB) and, alternating best-first (ABF) [7].

Since GKA is a stochastic algorithm, the average SE value reported in all the simulation results is the average of the SE values of the output strings of ten different runs of GKA. In all the experiments, the value of N, the population size, was set to 50 and the value of c, the constant in (7), was set to 2. The mutation rate P_m was set to 0.05 wherever DBM1 is used.

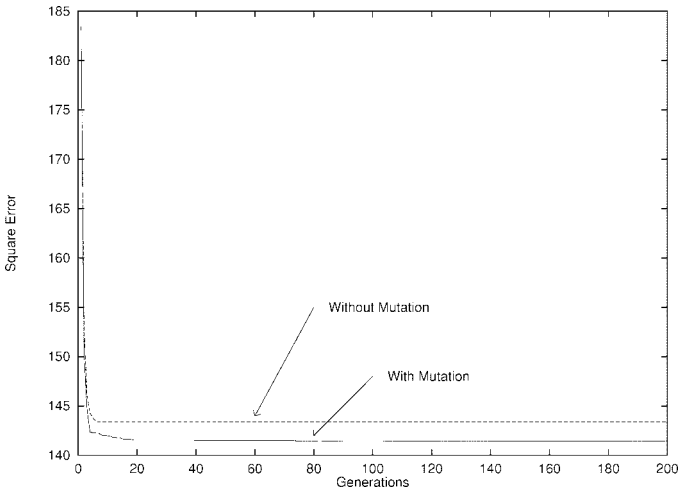

Fig. 2. GKA with and without the distance based mutation on BTD.

1) Significance of KMO and DBM1: GKA with and without KMO were applied on BTD. Fig. 1 shows the SE measure over generations corresponding to this experiment. Although it can be analytically shown that GKA without KMO asymptotically converges to the global optimum, the algorithm becomes very slow after some initial generations as shown in Fig. 1. It can be observed from Fig. 1 that KMO significantly increases the speed of convergence of the algorithm. In fact, this is the main reason for using a gradient descent step in GA's.

In the next experiment, GKA with and without DBM1 were applied on BTD. The corresponding results are shown in Fig. 2. It is to be emphasized at this point that the SE shown in the graph is the average over 10 runs of GKA. It was observed, in the case of GKA without DBM1, that even though the minimum of SE values among these 10 runs reached the best known optimum, the average is still at a higher value as shown in this figure. This shows that GKA without mutation is not always ensured to reach the global optimum. It is observed, as in the case of any evolutionary algorithm, that lower mutation rates make the GKA converge slowly and higher rates also slow down the convergence of GKA. This is so because at low mutation rate, mutation is hardly applied on the strings and at higher rates, the chromosomes keep changing frequently and give no time to the algorithm to exploit the search space around the solution. In fact, this is the only parameter that controls the performance of GKA.

Thus, the two operators, KMO and DBM1, play very important roles; KMO helps in speeding up of convergence and DBM1 in the global convergence of GKA.

2) Performance of GKA: In the next set of experiments, GKA was applied on both the data sets for different number of clusters. The SE measures corresponding to BTD and GTD with different number of clusters are given in the second columns of Tables I and II, respectively. It is observed that GKA took more time to reach the optimal partition as the number of clusters increases. This is very much expected since the increase in the number of clusters increases the size of the search space combinatorially, hence it is more difficult to find a globally optimal solution. However, it is also observed that in all the cases, average SE eventually converged to the global optimum. This is in concurrence with the convergence result derived in the previous section.

TABLE I
COMPARISON OF ACCURACY OF DIFFERENT ALGORITHMS WITH THAT OF GKA ON GTD

K	GKA	ES	EP	GAX	ABF
	Avg (min)	Avg (min)	Avg (min)		
4	49600.59(49600.59)	49600.59(49600.59)	49600.59(49600.59)	49600	49600
5	38716.02(38716.02)	38716.02(38716.02)	38716.02(38716.02)	38716	38716
6	30535.39(30535.39)	30535.39(30535.39)	30543.66(30535.39)	30535	30535
7	24432.57(24432.57)	24432.57(24432.57)	24434.70(24432.57)	24432	24432
8	21497.81(21483.02)	21523.10(21483.02)	21491.95(21483.02)	21483	21499
9	18787.21(18550.44)	18550.44(18550.44)	18722.56(18550.44)	18550	18970
10	16526.23(16307.97)	16429.86(16307.97)	16498.89(16307.97)	16353	16711

TABLE II
COMPARISON OF ACCURACY OF DIFFERENT ALGORITHMS WITH THAT OF GKA ON BTD

K	GKA	ES	EP	KMA
	Avg (min)	Avg (min)	Avg (min)	Avg (min)
4	180.91(180.91)	180.91(180.91)	180.91(180.91)	189.51(180.91)
5	160.23(160.23)	160.30(160.23)	160.26(160.23)	168.83(160.23)
6	141.46(141.46)	141.85(141.46)	141.65(141.46)	152.27(141.46)
7	126.29(126.29)	127.70(126.28)	126.79(126.28)	140.01(128.00)
8	113.99(113.50)	115.13(113.50)	114.25(113.50)	128.90(115.65)
9	103.52(102.74)	103.92(103.21)	104.39(102.74)	119.31(105.01)
10	93.40(92.68)	94.75(92.68)	93.97(92.68)	110.92(95.20)

3) Comparison with K-Means Algorithm (KMA): In this experiment, we consider partitioning of BTD into 10 clusters ($K = 10$). Fig. 3 shows the average and the best SE values obtained in ten independent runs of GKA during the first 100 generations. The values corresponding to KMA are also plotted in Fig. 3. Since each GKA is maintaining a population 50 solutions and, we considered ten independent runs of GKA's to obtain the values plotted in the figure, to make the comparison more meaningful, we considered 500 duplicates of KMA starting with different random initial configurations. Since all theses trials converged well within 20 iterations, we have plotted the corresponding values up to only 50 iterations.

It can be observed from Fig. 3 that, in case of GKA, the average SE value is approaching the best SE value, whereas it is not in case of KMA. This again shows that almost every run of GKA eventually converges to a globally optimal partition, numerically verifying the convergence result proved in the previous section. The performance of KMA is not surprising because KMA typically converges to a local optimum. Therefore, from this graph we can infer that even if KMA starts with the same number of initial configurations as in GKA, it is not assured to reach the global optimum. The situation becomes worse when the search space is large and there are many local optima. The figure also shows that in every iteration/generation, the best and average SE corresponding to GKA is less than those corresponding to KMA. The extra computational effort made by GKA in every generation, is that of DBM1 and selection operators.

4) Relative Performance: The accuracy and speed of GKA were compared with those of the following algorithms. In [10] and [11], the performance of different algorithms based on ES and EP applied on GTD and BTD are reported. In each case, one version of the algorithm is identified by the respective researchers, to be the best on these data sets. We use the results of these identified versions to compare with the results of GKA. Different GA's have been used in clustering [7], [8], [11]. We consider a representative among these viz., the one given in [7]. In [7], two crossover operators are defined and the GA's using these operators were applied on GTD. Of the two crossover operators, the GA using the first crossover showed better performance than the other. We refer to the GA with this crossover as GAX in the following text and quote the results reported in [7] using GAX. The performance of GAX was compared with two greedy algorithms AFB, ABF in [7]. Among AFB and ABF, ABF performed better than AFB. So, we quote here the results of ABF. The results on GTD and BTD are compiled in Tables I and II, respectively. In these tables, the columns corresponding to GKA, ES, and EP contain the average and minimum SE measure of the solutions obtained by the algorithms after 50 generations. The columns corresponding to GAX contain the best SE measure of the solutions found by GAX after 40 generations as reported in [7]. ABF algorithm, being a deterministic algorithm, converges to a local optimum. Reported here are the best results obtained from 40 random initial cluster configurations. Since the time taken by ABF to reach the optimum highly depends on the initial configuration, we do not analyze the time complexity of this algorithm. We have applied 500 copies of KMA, starting with different random initial clusters, on BTD for various number of clusters. The obtained results are given in Table II. As mentioned above, the difference between the average and the best SE, in case of KMA, is increasing with the number of clusters. This implies that when the number of clusters is large, the algorithms, that converge to a local optimum, can give a really bad solution. Again in this case, we do not compare the time complexities of GKA and KMA.

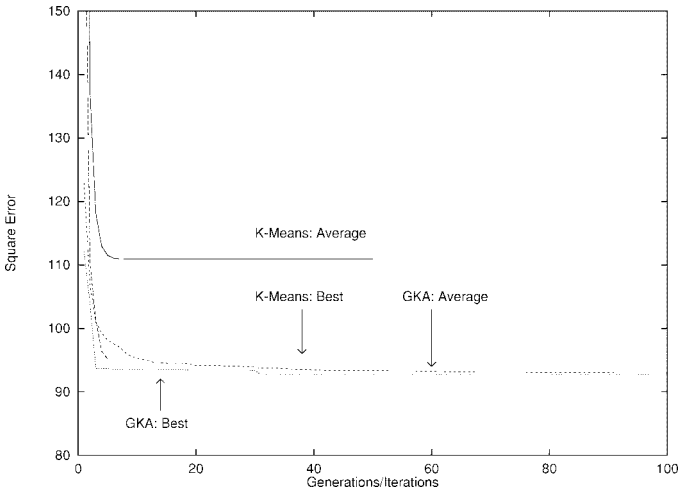

Fig. 3. Numerical demonstration of global convergence property of GKA using BTD.

5) Complexity of GKA: The complexity of evaluation of SE of a given solution string is $O(nd)$, the complexity of mutation operator is $O(n^2d)$ and K-means operator is $O(nKd)$. Since the mutation rate is very small, the effective number of times the operator is applied to an allele in a string is only a fraction (equal to mutation rate) of the total number of alleles. Moreover, K-means operator is a gradient descent operator and it does not change the string once it reached a local optimum unless the string is disturbed by mutation. Also since KMO is deterministic, if a string is not changed in the previous generation then there is no need to apply KMO on the string again. In fact, KMO is operational only in the initial phases of the evolution; in the later phases, it is effective only when the strings are disturbed by mutation. Therefore, the effective computations can be reduced if the changes in strings in a population are stored.

6) Complexity of ES and EP: In ES, each solution contains centroids and strategic parameters associated with each component of the centroids. In each generation, the algorithm calculates fitness values of all the strings in the population. The offspring are generated by recombining randomly selected solutions and mutating each resulting solution by adding Gaussian noise to each component of the centroids. The variance of the Gaussian noise is decided by the strategic parameters. In every generation, the strategic parameters are updated. The next population is obtained by selecting the best among parents and offspring of the current population.

In case of EP, solutions contain just centroids. In each generation EP evaluates the fitness value of each solution in the population and calculates variance of the zero-mean Gaussian noise, that is to be added by the mutation operator to the solutions, based on the fitness value of the solutions. The stochastic tournament strategy is used on the parents and offspring in the current generation to get the next population. Though tournament strategy is very expensive, the presence of recombination operator in ES makes both ES and EP equally computationally expensive.

The fitness function used in both these algorithms is as follows. Each pattern is assigned to the cluster with the nearest centroid. Based on these assignments new cluster centers are computed and again data are assigned to different clusters as before. The fitness value of the solution is the SE value of this assignment. So, in this case the fitness value computation is twice as expensive as that in GKA. The ES and EP, that gave the above results, evaluate the fitness function of 100 solutions, which include both parents and offspring, in every generation. This makes the fitness computation by these algorithms four times as expensive as that by GKA in every generation. Even if we assume that KMO is applied on every solution in all the generations, computational effort needed to find SE value and apply KMO and DBM1 by GKA is much less than (almost three quarters) that needed to find the fitness value by ES and EP. Therefore, the overall computational effort by GKA is much less than that by ES or EP. Even though the average SE in case of ES and EP appears to have been converging to the best SE value in these examples, there is no formal proof of convergence of these algorithms to the global optimum.

7) Complexity of GAX: GAX manipulate the order representation of the partitions. Each chromosome in this case represents many possible partitions of the data. The fitness value of a chromosome is the SE values of the partition with the least SE value. This was computed using a dynamic programming algorithm whose complexity is $O(n^2Kd)$. This is the most computationally expensive step in this algorithm. The complexity of mutation and crossover are $O(nd)$ and $O(nKd)$, respectively. So, it is evident that GKA is much faster than GAX.

VI. Conclusions

We considered the problem of finding a globally optimal partition, optimum with respect to SE criterion, of a given data into a specified number of clusters. Since the objective function associated with the above problem is nonlinear and multimodal, deterministic gradient descent methods converge to suboptimal solutions. We developed a stochastic gradient descent method by hybridizing the deterministic gradient descent clustering algorithm and GA. The resulting hybrid algorithm, GKA, has KMO, DBM1 and selection as genetic operators. Earlier work on the applications of GA to clustering defined various coding schemes and crossover operators on the encoded strings. In most of the cases either the evaluation of fitness function of the encoded string or the crossover operator is computationally expensive. In this paper, a simple coding scheme is employed and a problem-specific gradient descent operator, KMO, is defined and complicated crossover operators as well as computationally expensive function evaluations are avoided. It has been shown that KMO significantly improved the speed of convergence of GKA. The distance based mutation, DBM1, acts as a generator of biased random perturbations on the solutions, otherwise moving along the gradient with the possibility of getting stuck at a local optimum. Thus, mutation helps GKA avoid local minima. The selection operator carries a focused parallel search. It has been shown by analysis and through simulations that almost every run of GKA eventually converges to a globally optimal partition. The performance of GKA has been compared with that of some representatives of evolutionary algorithms, which are used for clustering and are supposed to converge to a global optimum. It turns out that GKA is faster than these algorithms.

We conjecture that for any complicated search problem, a combination of known best gradient descent step specific to the problem and knowledge-based biased random mutation may form competent operators. These operators along with the selection operator may yield a good hybrid GA for the problem under consideration. In such a case, the resulting hybrid GA would retain all the best features of the gradient descent algorithm. GKA is an instance of this type of hybridization. In this manner, the present work demonstrates with an example a way to obtain good hybrid GA's for a variety of complex problems.

Acknowledgment

The authors would like to thank the anonymous referees for their valuable comments on an earlier version of this paper. The authors also thank M. T. Arvind for his useful comments and suggestions on this paper.

References

[1] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 3–14, 1994.
[2] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: Univ. of Michigan Press, 1975.
[3] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data.* Englewood Cliffs, NJ: Prentice-Hall, 1989.
[4] R. W. Klein and R. C. Dubes, "Experiments in projection and clustering by simulated annealing," *Pattern Recognit.*, vol. 22, pp. 213–220, 1989.
[5] S. Z. Selim and K. Alsultan, "A simulated annealing algorithm for the clustering problem," *Pattern Recognit.*, vol. 10, no. 24, pp. 1003–1008, 1991.
[6] G. P. Babu and M. N. Murty, "Simulated annealing for selecting initial seeds in the k-means algorithm," *Ind. J. Pure Appl. Math.*, vol. 25, pp. 85–94, 1994.
[7] J. N. Bhuyan, V. V. Raghavan, and V. K. Elayavalli, "Genetic algorithm for clustering with an ordered representation," in *Proc. 4th Int. Conf. Genetic Algorithms.* San Mateo, CA: Morgan Kaufman, 1991.
[8] D. R. Jones and M. A. Beltramo, "Solving partitioning problems with genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms.* San Mateo, CA: Morgan Kaufman, 1991.
[9] G. P. Babu and M. N. Murty, "A near-optimal initial seed selection in K-means algorithm using a genetic algorithm," *Pattern Recognit. Lett.*, vol. 14, pp. 763–769, 1993.
[10] ——, "Clustering with evolution strategies," *Pattern Recognit.*, vol. 27, no. 2, pp. 321–329, 1994.
[11] G. P. Babu, "Connectionist and evolutionary approaches for pattern clustering," Ph.D. dissertation, Dept. Comput. Sci. Automat., Indian Inst. Sci., Bangalore, Apr. 1994.
[12] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA: Addison-Wesley, 1989.
[13] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Networks*, vol. 5, no. 1, pp. 96–101, 1994.
[14] L. Davis, Ed., *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold, 1991.
[15] H. Spath, *Clustering Analysis Algorithms.* New York: Wiley, 1980.
[16] Y. T. Chien, *Interactive Pattern Recognition.* New York: Marcel-Dekker, 1978.