AUSTRIAN JOURNAL OF STATISTICS
Volume 32 (2003), Number 1&2, 13–27

On The Use of Genetic Algorithm with Elitism in Robust and Nonparametric Multivariate Analysis

Biman Chakraborty[1] and Probal Chaudhuri[2]
[1]National University of Singapore, Singapore
[2]Indian Statistical Institute, Kolkata, India

Abstract: In this paper, we provide a general formulation for the problems that arise in the computation of many robust and nonparametric estimates in terms of a combinatorial optimization problem. There is virtually no hope for solving such optimization problems exactly for high dimensional data, and people usually resort to various approximate algorithms many of which are based on heuristic search strategies. However, for such algorithms it is not guaranteed that they will converge to the global optimum as the number of iterations increases, and there are always possibilities for such algorithms getting trapped in some local optimum. Here we propose genetic algorithm with elitism as a way to solve that general problem by probabilistic search method. We establish convergence of our algorithm to the global optimal solution and demonstrate the performance of this algorithm using some numerical examples.

Keywords: Combinatorial Optimization, Crossover, Fitness, Iterated Conditional Modes, Halfspace Depth, Markov Chains, MCD Estimator, Mutation, Transformation Retransformation.

1 Introduction

It is well known that computation of several estimates in robust and nonparametric multivariate analysis poses a very challenging problem. Many of these estimates require solving combinatorial optimization problems to determine some optimal subsets of the data points from the collection of all possible subsets with certain specified sizes that depend on the dimension of the data. Consider $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \in \mathbb{R}^d$. Our problem is to

$$\text{maximize or minimize } h(\boldsymbol{X}_{i_1}, \ldots, \boldsymbol{X}_{i_k})$$

where $\{\boldsymbol{X}_{i_1}, \ldots, \boldsymbol{X}_{i_k}\}$ is a subset of $\{\boldsymbol{X_1}, \ldots, \boldsymbol{X}_n\}$ of size k and h is a non-negative real valued function. Before we proceed with further discussion, let us present some interesting examples of multivariate estimates which can be formulated as a combinatorial optimization problem as above.

Example 1: Tukey's Halfspace Depth: Tukey (1975) proposed the halfspace depth of a d-dimensional point $\boldsymbol{x}$ relative to a data set $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \in \mathbb{R}^d$ as the smallest proportion of observations in any closed halfspace with boundary through $\boldsymbol{x}$. For the bivariate case, some efficient exact algorithms to compute the depth contours and the deepest point are proposed by Rousseeuw and Ruts (1996, 1998) and Ruts and Rousseeuw (1996). For dimension $d > 2$, Struyf and Rousseeuw (2000) proposed some approximation algorithm for computing the halfspace depth of a point but it still remains a challenging problem.

Let us consider a subset of indices $\beta = \{i_1, \ldots, i_{d-1}\}$ of size $d-1$. Then we will denote by $H(\beta)$, the unique hyperplane in $\mathbb{R}^d$ containing $\boldsymbol{x}$ and $\boldsymbol{X}_i$'s with $i \in \beta$. Let $h(X_{i_1}, \ldots, X_{i_{d-1}})$ denotes the absolute difference between the number of data points that fall in one side of $H(\beta)$ and the number of data points that fall in its other side. It is interesting to note that maximizing $h(X_{i_1}, \ldots, X_{i_{d-1}})$ yields the multivariate Hodges's sign test statistic (Hodges, 1955). Chaudhuri and Sengupta (1993) showed that this is equivalent to finding the halfspace depth at $\boldsymbol{x}$, which is obtained by minimizing the smallest proportion of data points on either side of $H(\beta)$. In other words, the halfspace depth of the point $\boldsymbol{x}$ can be written as

$$Depth(\boldsymbol{x}) = \frac{n+d-1}{2n} - \frac{1}{2n} \max h(X_{i_1}, \ldots, X_{i_{d-1}}).$$

Example 2: Transformation Retransformation (TR) Medians: Chakraborty and Chaudhuri (1996, 1998) proposed an affine equivariant version of multivariate median based on a transformation retransformation technique applied to the vector of coordinate-wise medians. Chakraborty et al. (1998) used the similar TR strategy to define an affine equivariant version of spatial median and angle test. The method can be described as follows: Consider the transformation matrix

$$\mathbf{X}(\alpha) = [\boldsymbol{X}_{i_1} - \boldsymbol{X}_{i_0} \vdots \cdots \vdots \boldsymbol{X}_{i_d} - \boldsymbol{X}_{i_0}]$$

where α denotes the subset of indices $\{i_0, \ldots, i_d\}$ of size $d+1$ and d is the dimension of the observations. Then we transform the data points as $\boldsymbol{Y}_i^{(\alpha)} = \{\mathbf{X}(\alpha)\}^{-1}\boldsymbol{X}_i$ for $i \notin \alpha$ and let $\hat{\boldsymbol{\theta}}^{(\alpha)}$ be the computed coordinatewise median/spatial median based on the transformed observations $\boldsymbol{Y}_i^{(\alpha)}$. Then the affine equivariant median is defined as $\mathbf{X}(\alpha)\hat{\boldsymbol{\theta}}^{(\alpha)}$. Chakraborty and Chaudhuri (1999) showed that this transformation retransformation technique with the optimal transformation matrix leads to location estimates with simultaneously high efficiency and high breakdown point. Note that, the efficiency of the estimate depends on the transformation matrix $\mathbf{X}(\alpha)$. Chakraborty and Chaudhuri (1998) and Chakraborty et al. (1998) have shown that in the case of both coordinatewise medians and spatial median, the optimal subset α is obtained by minimizing

$$v(\alpha) = \frac{trace\left[\{\mathbf{X}(\alpha)\}^T\hat{\Sigma}^{-1}\mathbf{X}(\alpha)\right]/d}{\left\{det\left[\{\mathbf{X}(\alpha)\}^T\hat{\Sigma}^{-1}\mathbf{X}(\alpha)\right]\right\}^{1/d}},$$

over α, where $\hat{\Sigma}$ is some consistent affine equivariant estimate of the scatter matrix Σ. So again, we have a combinatorial optimization problem with $h(\boldsymbol{X}_{i_0}, \ldots, \boldsymbol{X}_{i_d}) = v(\alpha)$, where $\alpha = \{i_0, \ldots, i_d\}$.

Example 3: Minimum Covariance Determinant (MCD) Estimator: A popular high breakdown estimator of multivariate location and scale matrix is minimum covariance determinant estimator proposed by Rousseeuw and Leroy (1987). The objective is to find k observations out of n observations whose classical covariance matrix has the lowest determinant. The MCD estimate of location is then the average of these k points, and the

MCD estimate of scale matrix is their covariance matrix. If $k = [(n+d+1)/2]$, the maximum breakdown point is attained by the MCD estimates. In this case, the combinatorial optimization problem is

$$\text{minimize } h(\boldsymbol{X}_{i_1}, \ldots, \boldsymbol{X}_{i_k}) = det\left[cov\left(\boldsymbol{X}_{i_1}, \ldots, \boldsymbol{X}_{i_k}\right)\right].$$

Example 4: Least Median of Squares (LMS) Regression: Consider the linear multiple regression model

$$y_i = \alpha + \boldsymbol{x}_i^T\boldsymbol{\beta} + \epsilon_i, \quad i = 1, \ldots, n \tag{1}$$

The p-dimensional vectors $\boldsymbol{x}_i$ contain the explanatory variables, y_i is the response and ϵ_i is the error term. Rousseeuw (1984) proposed least median of squares (LMS) regression method as a robust alternative to least squares to find an estimate of the parameters $(\alpha, \boldsymbol{\beta})$. To attain the maximum possible breakdown point, take $k = [(n + p + 1)/2]$ and define the LMS estimate of α and $\boldsymbol{\beta}$ as

$$(\hat{\alpha}_{LMS}, \hat{\boldsymbol{\beta}}_{LMS}) = arg\min_{\alpha,\boldsymbol{\beta}} |y_i - \alpha - \boldsymbol{x}_i^T\boldsymbol{\beta}|_{k:n}$$

where $|y_i - \alpha - \boldsymbol{x}_i^T\boldsymbol{\beta}|_{k:n}$ is the k-th order statistic of $y_1 - \alpha - \boldsymbol{x}_1^T\boldsymbol{\beta}, \ldots, y_n - \alpha - \boldsymbol{x}_n^T\boldsymbol{\beta}$.

Rousseeuw and Hubert (1997) suggested an algorithm called PROGRESS to compute the above LMS estimate of regression coefficients. Their method can be outlined as follows: Consider a hyperplane passing through $p+1$ data points $(y_{i_1}, \boldsymbol{x}_{i_1}), \ldots, (y_{i_{p+1}}, \boldsymbol{x}_{i_{p+1}})$ and adjust the intercept term α to get the best fitting hyperplane parallel to the above hyperplane through the data points. Take it as a candidate fit and then minimize $h_1[(y_{i_1}, \boldsymbol{x}_{i_1}), \ldots, (y_{i_{p+1}}, \boldsymbol{x}_{i_{p+1}})]$, over all possible hyperplanes passing through $p+1$ data points, where h_1 is $[(n + p + 1)/2]$-th ordered absolute residual obtained from this candidate fit.

For $p = 1$, it is an exact algorithm to find the LMS estimates. However, for $p \geq 2$, it gives only an approximate solution to the LMS regression. To resolve this issue, we can consider two parallel hyperplanes $y = \tilde{\alpha}_1 + \boldsymbol{x}^T\tilde{\boldsymbol{\beta}}$ and $y = \tilde{\alpha}_2 + \boldsymbol{x}^T\tilde{\boldsymbol{\beta}}$ containing $p+2$ data points $(y_{i_1}, \boldsymbol{x}_{i_1}), \ldots, (y_{i_{p+2}}, \boldsymbol{x}_{i_{p+2}})$ among themselves. Let $h_2[(y_{i_1}, \boldsymbol{x}_{i_1}), \ldots, (y_{i_{p+2}}, \boldsymbol{x}_{i_{p+2}})]$ be the $[(n + p + 1)/2]$-th ordered absolute residual obtained from the fit

$$y = \frac{\tilde{\alpha}_1 + \tilde{\alpha}_2}{2} + \boldsymbol{x}^T\tilde{\boldsymbol{\beta}}.$$

Then LMS estimates are obtained by minimizing

$$h_2[(y_{i_1}, \boldsymbol{x}_{i_1}), \ldots, (y_{i_{p+2}}, \boldsymbol{x}_{i_{p+2}})]$$

over all possible subsets of $p + 2$ observations.

Note that, whether we use PROGRESS or the exact algorithm described above, the problem of finding LMS estimates can be reduced to a combinatorial optimization problem of the same type as described earlier.

The naive way to solve any combinatorial optimization problem is to search all possible configurations and then the number of times the objective function is evaluated is of order n^k, where n is the number of data points and k is the size of the subset. Moreover, many of these optimization problems are $NP - hard$ (Garey and Johnson, 1979),

that is they belong to the class of problems for which there is no known algorithm that solves each instance of the problem to optimality in computing time bounded by a fixed degree (irrespective of the dimension, etc.) polynomial function of n, where n is the number of data points. Computing MCD estimator is one such example. For that reason many combinatorial optimization problems are tackled by constructing approximate rather than exact optimization algorithms. In that case the goal is to construct an approximation algorithm that runs in low order polynomial time and has the property that final solutions are "close" to globally optimal ones. In this paper, our objective is to construct an approximation algorithm, which works reasonably well for most of the combinatorial optimization problems in statistics literature and in many other fields. In Section 2, we discuss complete random search and many other commonly available heuristic optimization algorithms (e.g. iterated conditional mode algorithm). We also discuss some problem specific algorithms like FAST-MCD (Rousseeuw and van Driessen, 1999) for computing MCD estimates. In Section 3, we introduce genetic algorithm with elitism (EGA) as a stochastic optimization algorithm, which can be used to solve general combinatorial optimization problems considered here. We also discuss the Markov chain nature of the successive populations in the genetic algorithm and discuss its convergence. We present some simulation and real data examples in Section 4 to compare EGA with completely random search and commonly available algorithms. In Section 5, we make some concluding remarks and discuss some possible merits and demerits of the proposed algorithm.

2 Some Specific and General Purpose Algorithms

2.1 Completely Random Search

The simplest probabilistic algorithm is the completely random search, which chooses a subset of data points at random and evaluates the objective function at that subset and then repeat this procedure for a large number of times. The maxima or minima of the objective function values in these iterations is taken as the approximate global optimum of the original problem. The biggest advantage of this method is that it is very simple to implement and if the number of iterations (say, I) goes to infinity, the probability of hitting the true optimum goes to one. In other words, it is a convergent algorithm. But in most of the problems, especially when n and k are large, the number of iterations required to get the true optimum is considerably high. If the total number of possible configurations is $\binom{n}{k}$, the expected number of iterations to hit the true value is also $\binom{n}{k}$. If we stop after a finite number of step, the approximation may be nowhere near to the true optimum.

2.2 Deterministically Guided Search

Let us first discuss the algorithm FAST-MCD proposed by Rousseeuw and van Driessen (1999) for computing MCD estimates. To describe the method briefly, it starts with a subset of data points and then proceeds in a deterministic way until the objective function value stabilizes in the subsequent steps. Then it repeats the same procedure for many randomly chosen initial subset of observations and takes the minimum of these iterations as approximate solution to the optimization problem. While this algorithm runs quite

fast, it can only be used to solve a specific problem of finding MCD estimates. It is not probabilistic in nature after the initial subset is fixed and there is no guarantee of convergence of the algorithm to the true solution. If we repeat with randomly chosen initial subsets, of course the probability of obtaining the true solution converges to one as the number initial subsets goes to infinity, but then the algorithm becomes similar to the completely random search with may be a slight improvement in the expected hitting time.

A general purpose optimization algorithm of this type is iterated conditional modes algorithm (ICM) proposed by Besag (1986). A single iteration consists of k steps, where k is the size of the subset. At each step, it updates only one element of the subset given the rest of the element. Suppose at the m-th iteration, the system is at the subset S, then it updates the i-th element by choosing $\boldsymbol{X}_j$, $1 \leq j \leq n$, which maximizes the objective function value evaluated at the subset S with the i-th element replaced by X_j. When there is no improvement in the objective function value for a long time, we stop. Again there is no randomness in this ICM algorithm once the initial choice is fixed, one proceeds in a purely deterministic manner from the initial choice. This kind of algorithm very easily gets trapped in local optima and there is no in-built mechanism to get out of local optima (see Winkler, 1995). There is no guarantee of convergence to the global optimum.

We have reviewed two types of algorithms. For the first type the probability of hitting a true optimum converges to one but the expected hitting time is very large and if we stop after a fixed stopping time, we may have a very high variance for the approximate solution. The second type of algorithm will run very fast but deterministic in nature and may get trapped in a local optimum without ever getting out of it.

3 Genetic Algorithm With Elitism

Genetic algorithms (GA) are stochastic search methods based on the principles of natural genetic systems (Goldberg, 1989). The basic idea is to maintain a population of possible solutions that evolves and improves over time through a process of competition and controlled variation.

To begin with, each subset of k indices is represented by a binary string S of length L and a random sample of size M is drawn from 2^L strings to form the initial population. For a maximization problem of the non-negative objective function h, we define the *fitness* of a string as $fit(S) = h(\boldsymbol{X}_{i_1}, \ldots, \boldsymbol{X}_{i_k})$ or equivalently, for the minimization problem, $fit(S) = \{1 + h(\boldsymbol{X}_{i_1}, \ldots, \boldsymbol{X}_{i_k})\}^{-1}$ or any other decreasing function of h, where the string S corresponds to the subset of indices $\{i_1, \ldots, i_k\}$. When the total number of strings, 2^L, is greater than the total number of subsets, we assign the fitness value 0 to the strings, which do not correspond to any subset. Next we generate a *mating pool* by coping the individual strings of the current population using probability proportional to size sampling with the sizes given by their fitness function values. Many strategies for the generation of mating pool are available in the literature (Goldberg, 1989; Michalewicz, 1996). To perform crossover on the mating pool, one chooses pairs of strings randomly at a time from M strings until $M/2$ pairs are obtained. Then for each pair of strings, one performs *single point crossover* with probability p_c. *Mutation* is the mechanism which changes the string randomly and allows the process to come out of any possible local

optima. Every character β_i, $i = 1, \ldots, L$ in the string S changes to $(\beta_i + 1) \text{mod } 2$ with probability p_m.

3.1 The Algorithm

In this paper, we take the strategy of replacing the worst string of the new population with the best string of the current population. Genetic algorithms with this strategy are referred as *genetic algorithms with elitism* or EGA. The basic steps in an EGA is described as follows:

Step 1: Let l be the smallest integer such that $n \leq 2^l$. Then each element of the subset $\{i_1, \ldots, i_k\}$ can be represented as a binary string of length l. Concatenate these strings representing elements to get a string of length $L = kl$ to represent a particular subset of indices.

Step 2: Generate an initial population P of size M and calculate the fitness of each string S of P.

Step 3: Find the best string S_{best} of P. If the best strings are not unique, then call anyone of the best string in P as S_{best}.

Step 4: Construct the mating pool. Perform crossover and mutation operation on the strings of the mating pool and obtain a population P_{tmp}.

Step 5: Compare the fitness of each string S of P_{tmp} with S_{best}. Replace the worst string of P_{tmp} with S_{best} if the fitness of each string in P_{tmp} is less than the fitness of S_{best}. Otherwise no replacement takes place in P_{tmp}. Rename P_{tmp} as P.

Step 6: Go to step 3.

3.2 A Markov Chain Representation of EGA And Its Convergence

Let $F_1 > F_2 > \cdots > F_s$ be the ordered fitness values of all possible strings and s is the number of distinct fitness values. Since the total number of strings is 2^L, we have $s \leq 2^L$. Let P denotes a population of M strings and $\mathcal{P}$ denotes the collection of all populations. We can immediately partition $\mathcal{P}$ as

$$\mathcal{P}_i = \{P : \ P \in \mathcal{P} \text{ and } \max_{S \in P} fit(S) = F_i\}, \quad i = 1, \ldots, s.$$

Note that, F_1 is the maximum possible fitness and $\mathcal{P}_1$ is the collection of populations containing the strings with maximum fitness. Thus, if our population is in $\mathcal{P}_1$, we know that we have got an optimal subset of data points. Let p_i be the number of populations in $\mathcal{P}_i$ and P_{ij} be the j-th population in $\mathcal{P}_i$, for $j = 1, \ldots, p_i$ and $i = 1, \ldots, s$.

In any generation, the algorithm creates a population P_{kl} from some P_{ij}. Since we are preserving the previous best in the population, we cannot generate a population P_{kl} from P_{ij}, if $k > i$. Because the maximum fitness value of the new population is at least F_i. The

creation of a population P_{kl} from P_{ij} can be viewed as a transition from P_{ij} to P_{kl} and let $u_{(ij)(kl)}$ denotes the corresponding transition probability. Then

$$u_{(ij)k} = \sum_{l=1}^{p_k} u_{(ij)(kl)}$$

denotes the probability of transition from P_{ij} to any population in $\mathcal{P}_k$, for $j = 1, \ldots, p_i$ and $i, k = 1, \ldots, s$.

Proposition 3.1 *For all $j = 1, \ldots, p_i$ and $i = 1, \ldots, s$*

$$\begin{aligned} u_{(ij)k} &> 0 \quad \textit{if } k \leq i \\ &= 0 \quad \textit{otherwise}. \end{aligned}$$

It is clear from the above discussion that one can consider any population P_{ij} as a state of a Markov chain. Let $u^{(N)}_{(ij)(kl)}$ be the probability that EGA results in P_{kl} at the N-th step given that the initial state is P_{ij} and $u^{(N)}_{(ij)k}$ denotes the probability of reaching a population in $\mathcal{P}_k$ in N steps with the starting population as P_{ij}, then

$$u^{(N)}_{(ij)k} = \sum_{l=1}^{p_k} u^{(N)}_{(ij)(kl)}.$$

The following Theorem asserts that starting from any population P_{ij}, EGAs eventually result in one of the populations P_{1l} of $\mathcal{P}_1$. In other words, as the number of generations $N \longrightarrow \infty$, the EGAs converge to a population containing the optimal subset.

Theorem 3.2 *For an EGA with the probability of mutation $p_m \in (0, 0.5]$,*

$$\lim_{N\to\infty} u^{(N)}_{(ij)k} = 0 \textit{ for } 2 \leq k \leq s; \quad j = 1, \ldots, p_i \textit{ and } i = 1, \ldots, s$$

and

$$\lim_{N\to\infty} u^{(N)}_{(ij)1} = 1 \textit{ for } j = 1, \ldots, p_i \textit{ and } i = 1, \ldots, s.$$

The proof of the above theorem follows from the results in Bhandari et al. (1996) and we give an outline in the appendix. The proof does not depend on the crossover operation, but mutation should be performed with probability $p_m > 0$. Note that, the transition probabilities from one population to another depend on the process of mating pool generation, but these probabilities will always obey the conditions in Theorem 3.2.

4 Some Illustrative Numerical Examples

Let us first consider finding out the optimal transformation to be used in the transformation retransformation methodology (Chakraborty and Chaudhuri, 1998; Chakraborty et al., 1998). For that purpose, we simulate data sets from standard normal distribution with means equal to zero and covariance matrix equal to the identity matrix for dimensions

$d = 2$, 3, 4, 5 and 10 and sample sizes $n = 50$, 100, 500 and 1000. We have compared EGA with a completely random search with 60000 subsets for each data sets and for dimension $d = 2$ and sample size $n = 50$, we have performed a complete search. The parameters of the EGA are: the population size $M = 300$, probability of crossover, $p_c = 0.7$, probability of mutation, $p_m = 0.4$, and number of generations, $g = 500$. In the following Table 1, we report the percentage of times EGA returns a subset which is as good as the best one obtained in completely random search in 1000 simulations.

Table 1: Percentage of times EGA returned a subset which is as good as the best subset obtained from the complete random search

	Dimension, d				
n	2	3	4	5	10
50	88.4	87.6	90.5	91.8	94.9
100	89.6	90.3	93.2	94.8	97.4
500	89.4	89.4	95.3	96.1	99.1
1000	88.4	92.7	96.7	97.8	99.3

We note that, for a small data set ($n = 50$, $d = 2$), when a complete search is possible, the EGA returns the best value more than 88% of the times, which is quite encouraging. For higher dimensions, when a complete search is practically impossible, we see that EGA is performing much better compared to a complete random search.

To illustrate the effect of different values of the parameters, we consider a synthetic data set about physical characteristics of pollens used for the 1986 ASA data exposition and available in Statlib (http://lib.stat.cmu.edu/datasets/pollen.data). There are 5 physical measurements, 'ridge', 'nub', 'crack', weight and density of 3848 pollens. We compute Tukey's halfspace depth (Tukey, 1975) for this data at the coordinatewise median $(-0.1639, -0.2317, -0.0562, -0.1493, -0.0304)$. We have found that the probability of crossover, p_c does not have a significant effect on the rates of convergence and we fix it at 0.6 for this problem and vary the population size and the probability of mutation, p_m.

In the case of halfspace depth, we see that when the probability of mutation is large ($p_m = 0.1$) and the population size is not so large ($M = 100$ or 200), the EGA gets trapped in a local optimum and does not come out of it for a long time (Figure 1). Whereas for a smaller mutation probability $p_m = 0.005$ and a large population size $M = 500$, it achieves better objective function values quite early and does no seem to get trapped in a local optimum. For a still smaller mutation probability, $p_m = 0.001$ and not so large population size $M = 100$, the EGA starts from a relatively high objective function value but gradually decreases to the stable value and never gets trapped in a local optimum. By a complete enumeration, we have found that the true halfspace depth of this point is 0.426715 and this value is attained be the last case. It suggests that, if we have a larger population size, the EGA converges faster, however, that requires a larger computation time in each generation and if we have a smaller mutation probability, it is less likely to get trapped in a local optimum.

In our next example, we consider computing MCD estimates using the proposed EGA

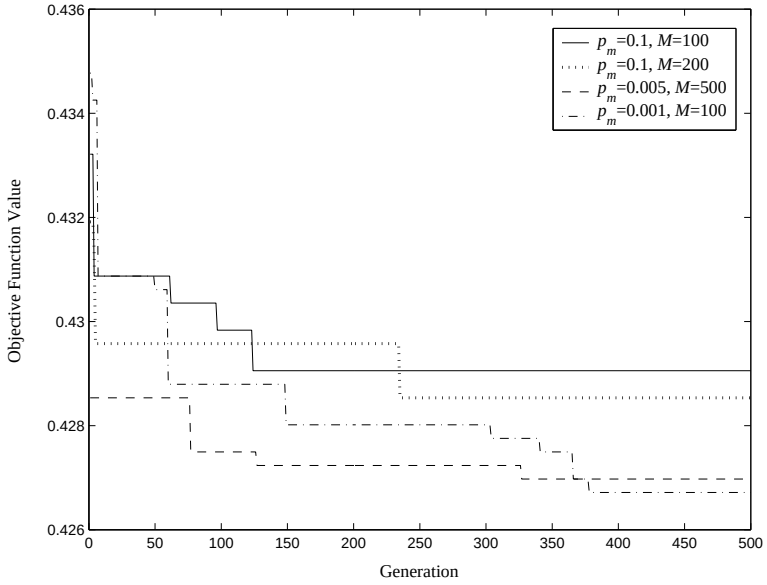

Figure 1: The plot of the objective function corresponding to the best subset in populations of successive generations.

algorithm and we compare our method with FAST-MCD algorithm for some simulated data sets. We have simulated data sets from standard multivariate normal distribution with means equal to zero and covariance matrix equal to the identity matrix for dimensions $d = 2$, 3, 4, 5 and 10 and for sample sizes $n = 50$, 100, 500 and 1000. Table 2 summarizes the percentage of the times EGA got a better (smaller) value of the objective function compared to the FAST-MCD algorithm in 1000 simulations of the each case. For the FAST-MCD, we have used the default choice of 500 initial guesses and for the EGA, the parameters taken are: size of a population, $M = 200$, the probability of crossover $p_c = 0.7$, the probability of mutation, $p_m = 0.001$ and number of generations, $g = 100$.

Table 2: Percentage of times EGA returned a smaller covariance determinant compared to FAST-MCD

	Dimension, d				
n	2	3	4	5	10
50	99.8	98.5	96.3	85.6	71.2
100	99.8	100.0	99.9	99.5	98.2
500	97.2	98.5	99.2	99.6	99.7
1000	82.0	87.0	91.3	92.1	95.6

We note that, in all cases EGA performed better than FAST-MCD in at least 70% of the times. In high dimensions, with a large sample size, the performance of EGA is almost always better. However, we note that, this simulation study has been carried out with some specific values of the EGA parameters. The performance of EGA can be improved in real

data set by trying out several combinations of parameters.

It might be of interest to see that when EGA returns a smaller value compared to FAST-MCD, do we really gain a lot or the improvement is only marginal. For that purpose, we use the pollen data with 5 variables and 3848 observations as discussed before. In Figure 2, we plot the value of the objective function corresponding to the best string in a population in successive generations of EGA with parameters, population size, $M = 100$, probability of crossover, $p_c = 0.8$ and probability of mutation, $p_m = 0.0001$. For comparison, we have also plotted a dotted line indicating the best value obtained by the FAST-MCD algorithm with 5000 trial subsamples and we note that the EGA gets a better value after about 1500 generations and if we continue for a large number of generations, the improvement is substantial. However, still the final results are not satisfactory for this data set. One needs to investigate further with different choices of the parameters to achieve a faster convergence and stability in EGA.

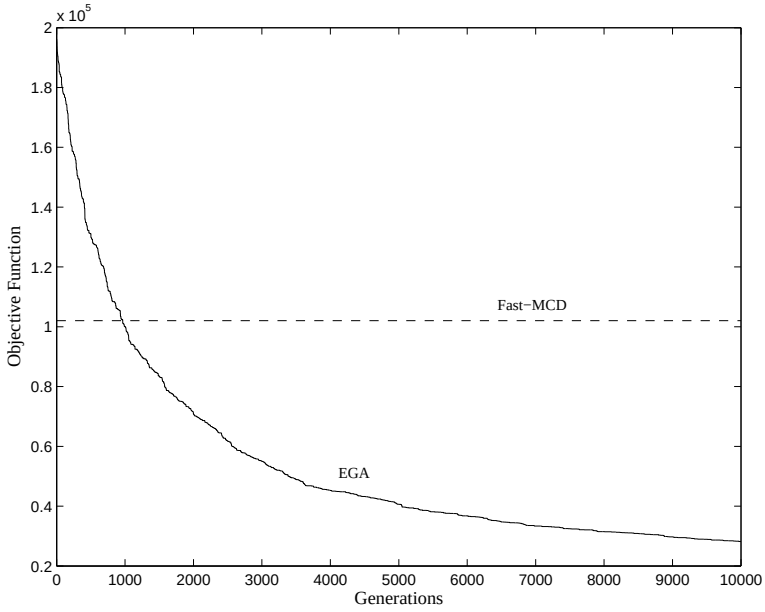

Figure 2: The plot of objective function corresponding to the best subset of the populations in successive generations of the EGA

In this example, the FAST-MCD algorithm evaluates the objective function about 18200 times and EGA with population size $M = 100$ and generation $g = 200$ evaluates the objective function 20000 times. However, with such a large number of trial subsamples, FAST-MCD is no longer very fast compared to EGA.

5 Concluding Remarks

We have noticed that the choice of the population size effects the convergence, but there is no strict guidelines to choose M, however it is convenient to choose M to be an even integer, since in the crossover operation, we need pairs of strings. The algorithm does not depend much on the choice of the probability of crossover and hence it can be chosen to be

any reasonable number in $[0, 1]$, but it heavily depends on the selection of the probability of mutation, p_m. We must choose $p_m \in (0, 0.5]$ for the algorithm to converge, the rate of convergence depends on its choice and we suggested that for a particular problem the user should try a few values of p_m to understand the behaviour of the problem regarding the choice. We assumed that p_c and p_m remain fixed during the process, however, it is possible to change these parameters during the process and then the underlying Markov chain becomes nonhomogeneous.

There are many other variations of the genetic algorithms. One important modification is that instead of considering populations of binary strings, one can consider populations of strings over a finite alphabet $\mathcal{A} = \{a_1, \ldots, a_l\}$. Our main theorem on convergence is still valid for this situation. Other modifications could be on the method by which the mating pools are generated or on the method we perform the crossover operation. These changes in the algorithm will change the transition probabilities of the algorithm but the convergence of EGA is still valid. We would also like to note that the GAs without elitism can also be modelled as a Markov chain and Davis and Principe (1991) proved their convergence to the limiting distributions under some conditions on the mutation probabilities. However, it does not guarantee the convergence to the global optimum. With the introduction of elitism or by keeping the best string in the population allows us to show the convergence of the EGA to the global optimal solution starting from any arbitrary initial population.

Todorov (1992) suggested a simulated annealing based approach for computing MCD estimates and Woodruff and Rocke (1993) proposed some heuristic search algorithms for computing minimum volume ellipsoid estimators, which includes a brief discussion on genetic algorithm too. Neither of these works proved any convergence results of their algorithms and they did not report any detailed study for the general combinatorial optimization problem considered here.

Boček and Lachout (1995) proposed a probabilistic algorithm based on simplex method for computing LMS regression estimates and Tichavský (1991) suggested some optimal shift of the randomly chosen hyperplanes. There is another approximation algorithm suggested by Olson (1997) too. However, none of them can be used as a general purpose algorithm to solve other combinatorial optimization problems and a detailed study is necessary to compare their performances in solving large regression problems.

In this paper, our objective is to provide a general algorithm which works for a large class of combinatorial optimization problems in statistics. While it is still possible to have an algorithm which works much better than EGA for a particular problem, we conclude this note by pointing out that EGA is a general purpose algorithm and one can safely use it without doing a great deal of research on computing every time a new combinatorial optimization problem appears.

A Appendix: Proofs

Proof of Proposition 3.1: let $P = \{S_m : \quad m = 1, \ldots, M\}$ be a population generated using the genetic operators on the population P_{ij}. Note that, the best string $S_{best} \in P_{ij}$ ($fit(S_{best}) = F_i$) is copied in the generated population P if the fitness of all the generated

strings are less than F_i and hence

$$\max_m fit(S_m) \geq F_i.$$

This implies $u_{(ij)(kl)} = 0$ for all $l = 1, \ldots, p_k$, if $k > i$ and consequently

$$u_{(ij)k} = \sum_{l=1}^{p_k} u_{(ij)(kl)} = 0, \quad \text{for } k > i.$$

Now for $k \leq i$, consider a population , say P_{kl}, which contains M copies of a string S' such that $fit(S') = F_k$. Since $k \leq i$, $fit(S) \leq fit(S')$ for all $S \in P_{ij}$. If all the strings $S \in P_{ij}$ are changed to S' by the genetic operators then we need not copy S_{best}, the best string in P_{ij}, in the new population. Now we show that the probability of such a transition is greater than 0.

It is obvious that the probability of generating any string S_1, from a given string S_2 is $p_m^{\nu}(1-p_m)^{L-\nu}$, where $\nu(0 \leq \nu \leq L)$ is the number of places where the strings S_1 and S_2 have distinct characters. Since $p_m < 0.5$, the probability of generating $S' \in P_{kl}$ from any string $S \in P_{ij}$ is not less than p_m^L. Hence the minimum probability to obtain P_{kl} from P_{ij} is p_m^{ML}, that is, $u_{(ij)(kl)} \geq p_m^{ML}$. Hence,

$$u_{(ij)k} = \sum_{l_1=1}^{p_k} u_{(ij)(kl_1)} \geq p_k (p_m)^{ML} > 0 \quad \text{for all } k \leq i. \tag{2}$$

□

Proof of Theorem 3.2: From Proposition 3.1, we have $u_{(ij)k}^{(n)} \geq 0$ and $u_{(ij)1} > 0$ for all $j = 1, \ldots, p_i$ and $i = 1, \ldots, s$. Let $\min_{i,j} u_{(ij)1} = \delta$. Now,

$$\sum_{k \neq 1} u_{(ij)k}^{(1)} = \sum_{k=2}^{s} u_{(ij)k} = 1 - u_{(ij)1} \tag{3}$$

$$\begin{aligned}
\sum_{k \neq 1} u_{(ij)k}^{(2)} &= \sum_{k=2}^{s} \sum_{i_1=2}^{s} \sum_{j_1=1}^{p_{i_1}} u_{(ij)(i_1 j_1)} u_{(i_1 j_1)k} \quad (\text{since, } u_{(1 j_1)k} = 0 \text{ for } k > 1) \\
&= \sum_{i_1=2}^{s} \sum_{j_1=1}^{p_{i_1}} u_{(ij)(i_1 j_1)} \sum_{k=2}^{s} u_{(i_1 j_1)k} = \sum_{i_1=2}^{s} \sum_{j_1=1}^{p_{i_1}} u_{(ij)(i_1 j_1)} (1 - u_{(i_1 j_1)1}) \\
&\leq (1-\delta) \sum_{i_1=2}^{s} \sum_{j_1=1}^{p_{i_1}} u_{(ij)(i_1 j_1)} \\
&= (1-\delta) \sum_{i_1=2}^{s} u_{(ij)i_1} = (1-\delta)(1 - u_{(ij)1})
\end{aligned} \tag{4}$$

By using mathematical induction,

$$\sum_{k=2}^{s} u_{(ij)k}^{(n+1)} = \sum_{k=2}^{s} \sum_{i_1=2}^{s} \sum_{j_1=1}^{p_{i_1}} u_{(ij)(i_1 j_1)}^{(n)} u_{(i_1 j_1)k} \leq (1-\delta)^n (1 - u_{(ij)1}) \tag{5}$$

Note that $(1-\delta)^n(1-u_{(ij)1}) \to 0$ as $n \to \infty$ since $0 < \delta \leq 1$. Hence we have

$$\lim_{n \to \infty} u_{(ij)k}^{(n)} = 0$$

for $k = 2, \ldots, s$, $i = 1, \ldots, s$ and $j = 1, \ldots, p_i$. Thus,

$$\lim_{n\to\infty} u^{(n)}_{(ij)1} = \lim_{n\to\infty} \left(1 - \sum_{k=2}^{s} u^{(n)}_{(ij)k}\right) = 1. \tag{6}$$

□

References

J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*, 48:259–302, 1986.

D. Bhandari, C.A. Murthy, and S.K. Pal. Genetic algorithms with elitist model and its convergence. *International Journal of Pattern Recognition and Artificial Intelligence*, 10:731–747, 1996.

P. Boček and P. Lachout. Linear programming approach to LMS–estimation. *Computational Statistics & Data Analysis*, 19:129–134, 1995.

B. Chakraborty and P. Chaudhuri. On a transformation and re-transformation technique for constructing affine equivariant multivariate median. *Proceedings of the American Mathematical Society*, 124:2539–2547, 1996.

B. Chakraborty and P. Chaudhuri. On an adaptive transformation retransformation estimate of multivariate location. *Journal of the Royal Statistical Society, Series B*, 60: 145–157, 1998.

B. Chakraborty and P. Chaudhuri. A note on robustness of multivariate median. *Statistics and Probability Letters*, 45:269–276, 1999.

B. Chakraborty, P. Chaudhuri, and H. Oja. Operating transformation retransformation on spatial median and angle test. *Statistica Sinica*, 8:767–784, 1998.

P. Chaudhuri and D. Sengupta. Sign tests in multidimension: Inference based on the geometry of the data cloud. *Journal of the American Statistical Association*, 88:1363–1370, 1993.

T. E. Davis and J.C. Principe. A simulated annealing-like convergence theory for the simple genetic algorithm. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 174–181. Morgan Kaufmann, San Mateo, CA, 1991.

M.R. Garey and D.S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, 1979.

D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, New York, 1989.

J.L. Hodges. A bivariate sign test. *The Annals of Mathematical Statistics*, 26:523–527, 1955.

Z. Michalewicz. *Genetic Algorithms + Data Structure = Evolution Programs*. Springer, New York, 1996.

C.F. Olson. An approximation algorithm for least median of squares regression. *Information Processing Letters*, 63:237–241, 1997.

P.J. Rousseeuw. Least median of squares regression. *Journal of the American Statistical Association*, 79:871–880, 1984.

P.J. Rousseeuw and M. Hubert. Recent developments in progress. In Y. Dodge, editor, L_1*-Statistical Procedures and Related topic*, pages 201–214. Institute of Mathematical Statistics, Hayward, California, 1997.

P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection*. Wiley, New York, 1987.

P.J. Rousseeuw and I. Ruts. As 307: Bivariate location depth. *Applied Statistics*, 45: 516–526, 1996.

P.J. Rousseeuw and I. Ruts. Constructing the bivariate Tukey median. *Statistica Sinica*, 8:827–839, 1998.

P.J. Rousseeuw and K. van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223, 1999.

I. Ruts and P.J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23:153–168, 1996.

A. Struyf and P.J. Rousseeuw. High-dimensional computation of the deepest location. *Computational Statistics and Data Analysis*, 34:415–426, 2000.

P. Tichavský. Algorithm for and geometrical character of solutions in the LMS and the LTS linear regression. *Computational Statistics Quarterly C*, 6:139–151, 1991.

V. Todorov. Computing the minimum covariance determinant estimator (MCD) by simulated annealing. *Computational Statistics & Data Analysis*, 14:515–525, 1992.

J.W. Tukey. Mathematics and the picturing of data. In R.D. James, editor, *Proceedings of the International Congress of Mathematicians, vol. 2*, pages 523–531. Canadian Mathematical Congress, Vancouver, 1975.

G. Winkler. *Image Analysis, Random Fields and Dynamic Monte Carlo Methods. A Mathematical Introduction*. Springer, New York, 1995.

D.L. Woodruff and D.M. Rocke. Heuristic search algorithms for the minimum volume ellipsoid. *Journal of Computational and Graphical Statistics*, 2:69–95, 1993.

Authors' addresses:

Dr. Biman Chakraborty
Department of Statistics and Applied Probability
The National University of Singapore
10, Kent Ridge Crescent
Singapore 119260
Singapore

Tel. +65 68745239
Fax +65 68723919
E-mail: stabc@nus.edu.sg
http://www.stat.nus.edu.sg/∼biman/

Prof. Probal Chaudhuri
Theoretical Statistics and Mathematics Unit
Indian Statistical Institute
203, B. T. Road
Kolkata 700108
India

E-mail: probal@isical.ac.in
http://www.isical.ac.in/∼probal/