A Provably Convergent Inhomogenous Genetic Annealing Algorithm

Griff L. Bilbro
Lester C. Hall
Lawrence Ray

Center for Communications and Signal Processing
Department of Electrical and Computer Engineering
North Carolina State University

TR-92/3
June 1992

A Provably Convergent Inhomogeneous Genetic Annealing Algorithm[1]

Griff L. Bilbro and Lester C. Hall
North Carolina State University
Raleigh, NC 27695-7911

Lawrence A. Ray
Eastman Kodak Company
Rochester, NY 14653-5719

ABSTRACT

We define Genetic Annealing as simulated annealing applied to a population of several solutions when candidates are generated from more than one (parent) solution at a time. We show that such genetic annealing algorithms can inherit the convergence properties of simulated annealing. We present two examples, one that generates each candidate by crossing pairs of parents and a second that generates each candidate from the entire population. We experimentally apply these two extreme versions of genetic annealing to a problem in vector quantization.

INTRODUCTION

Combinatorial optimization is the problem of minimizing a function $f : X \rightarrow R$ where $X = \{1, ..., N\}$ is a finite set of feasible solutions. We define a neighborhood structure for X as some $X_i \subset X$ for each $i \in X$ where $j \in X_i$ implies $i \in X_j$. We will consider algorithms that iteratively replace a current solution i with neighboring solution $j \in X_i$ and so we also require that the neighborhood structure permit the global minimum to be reached in a finite number of such steps from any initial solution as in the following descent algorithm:

```
1. Select initial solution $i$.
2. Select candidate solution $j \in X_i$.
3. Replace $i \leftarrow j$ if and only if $f_j < f_i$.
4. If not done, go to 2.
5. Report $i$, $f_i$.
6. Stop.
```

[1]SPIE Vol. 1766 Neural and Stochastic Methods in Image and Signal Processing (1992)

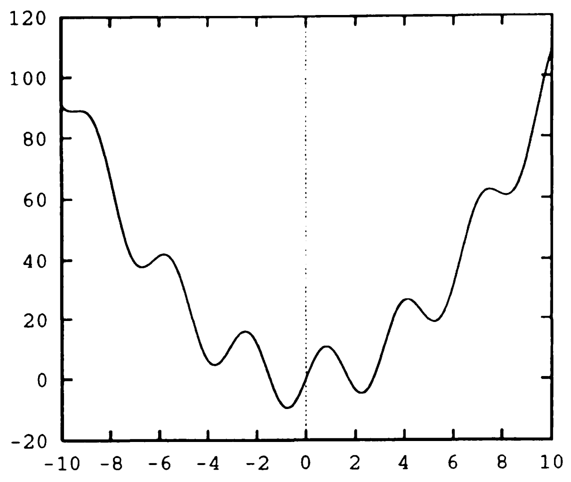


Figure 1: One dimensional slice through a Rastrigin-like function.

It is easy to construct a problem that will defeat this greedy descent algorithm, which generates a path from the initial point to the terminal point. If the resulting path is deterministic and not exhaustive, the algorithm will fail for the modified objective into which a global minimum has been introduced in some unvisited region of X.

Nor does mere randomization guarantee global minimization. Consider a randomized descent algorithm which selects candidate j given current i from generator probabilities g_{ij}, and replaces i with j according to acceptance probabilities a_{ij}. This simple random descent algorithm with uniform $g_{ij} = 1/|X_i|$ and greedy $a_{ij} = 1$, if $f_j < f_i$ and $a_{ij} = 0$ otherwise, can also fail if f has local minima that are not neighbors.

Simulated annealing guarantees global minimization by carefully choosing $a_{ij} > 0$ for $f_j > f_i$ with a Metropolis procedure or a Gibbs sampler[6]. A typical scheme employs uniform $g_{ij} = 1/|X_i|$ and Metropolis $a_{ij} = \min(1, \exp(f_i - f_j)/T_k)$ in the k^{th} step where $T_k = C/\ln(1+k)$ for large k. Simulated annealing is fastest when C and $|X_i|$ are small. In practice, the premises of convergence are often violated to make SA run faster. It is faster than exhaustive search when good solutions occur in regions of slightly poorer solutions[4] as in Figure 1 of a Rastrigin-like function[8]. Simulated Annealing algorithms are particularly efficient on functions like this with smooth global structure perturbed by smaller local variations.

Simulated annealing can be accelerated by biasing the g_{ij} to concentrate the search in promising regions, but this bias will cause erroneous convergence unless the a_{ij} are adjusted to compensate. One early approach used a biased generator without modification of the acceptance probabilities to obtain promising but erratic results: the covariance matrix of the Markov chain was adaptively estimated and used to orient the search in anisotropic search spaces. That approach discarded the scaling information contained in the covariance matrix and it required complicated heuristics to prevent premature convergence[9]. In a later image restoration, a non-adaptive biased generator was shown to reduce run times without compromising theoretical convergence. In that application, the Bayesian *maximum a posteriori* objective contained a likelihood part suitable for constructing a fixed generator appropriate to the problem[3]. More recently, when good g_{ij} are not available *a priori*, they have been adaptively accumulated in a binary tree but without guaranteed results[4, 5].

In this paper we propose to secure the efficiency of an adaptive generator without loss of provable convergence by by enlarging the search space of the usual (fixed-generator) simulated annealing in a suitable manner. This enlargement will amount to annealing a population of several solutions. A simple way of adaptively exploiting the structure of the search space is to preferentially generate new candidates in regions of high population density. We will exploit theoretical guarantees for asymptotic convergence which exist for a large class of generators if the acceptance probabilities obey

$$\sum_k \min_{i \in X\, j \in X_i} a_{ij}(T_k) = \infty \tag{1}$$

and if the steady-state distribution $\pi_i(T)$

$$\lim_{k \to \infty} \pi_i(T_k) = 0 \text{ for } f_i \text{ not minimal} \tag{2}$$

eventually vanishes for i not optimal. In the usual case, this is satisfied for simulated annealing based on the Metropolis procedure with fixed generator if

$$T_k = C / \ln(1+k) \tag{3}$$

with large enough constant C[1].

ANNEALING A POPULATION

Consider a population of solutions $I = i_1, i_2, \ldots, i_M$. We will take this population I as the current state of a simulated annealing algorithm with

1. Objective $F_I = \sum_{m \in I} f_m$.
2. Generation probabilities for population state J as zero except when I and J differ only in some single member m as $G_{IJ} = g_{i_m j_m}$ where g may depend arbitrarily on the remaining members $I \backslash i_m$ of the population; in the present paper we choose $m \in \{1, 2, \ldots, M\}$ uniformly.
3. Acceptance probabilities $A_{IJ} = a_{i_m j_m}$ as before with $a_{ij} = \min\left(1, \frac{g_{ji}}{g_{ij}} \exp\left((f_i - f_j)/T\right)\right)$ and $i = i_m$, $j = j_m$.

For annealing a single solution with fixed g_{ij}, this choice of a_{ij} and g_{ij} satisfies Anily and Fedegruen's sufficient conditions for convergence[4, 3]. We propose here to let g_{ij} depend on $I \backslash i_m$ and preserve the condition $\sum_k \min_{i \in X\, j \in X_i} a_{ij} = \infty$ by requiring $g_{i_m j_m} \equiv 0$ for $j_m \in I \backslash i_m$.

VECTOR QUANTIZATION

Vector Quantization (VQ) is the problem of partitioning a set of d dimensional vectors into subsets or clusters that optimally represent their members. For image coding, an $N \times M$ image is partitioned into windows of $n \times m$ pixels to form $V = \frac{NM}{nm}$ vectors in nm dimensions. VQ is used to cluster the vectors into $K \ll |V|$ groups. The centroids of the groups provide an efficient approximate code for representing the original image or similar images.

It is natural to formulate VQ as an optimization problem with an optimization objective that measures the expected cost of representing a vector by properties of the cluster it is assigned to[2, 7]. The usual squared error objective is typical and can be defined for an assignment A as

$$f(A) = \sum_{i \in V} \sum_{k \in K} y_{ik} |v_i - c_k|^2$$

where the matrix $y_{ik} = 1$ if and only if vector $k = A(i)$ where V is the set of vectors, K is the set of clusters, and $A : V \rightarrow K$. Here v_i is the i^{th} data vector and

$$c_k = \frac{\sum_{i \in V} y_{ik} x_i}{\sum_{i \in V} y_{ik}}$$

is the code vector for the k^{th} cluster. Each c_k is itself a d-vector like the data.

Vector quantization can be formulated as a combinatorial optimization of the assignment vector A or as a continuous optimization problem with the real-valued code vectors c_k as the independent variable instead of the discrete assignment vector A_i. We will consider both formulations in the next two sections.

MUTATION/CROSSOVER GENETIC ANNEALING

In this section we consider VQ as combinatorial optimization of the assignment A. There are a finite number of feasible assignments and we number each as before $X = \{1, 2, ..., N\}$ and, as before, construct a population of solutions $I = \{i_1, ..., i_M\}$ with M members.

The mutation operator can be chosen as in simulated annealing: A member index m is chosen uniformly from $\{1, 2, ..., M\}$ and current solution $i_m = a_1, a_2, ..., a_{|V|}$. A candidate j_m generated by uniformly selecting some assignment $a_p \in j_m$ and uniformly selecting a new cluster from K to assign it to. We update $i_m \leftarrow j_m$ according to the usual (unbiased) Metropolis test because we chose all selections uniformly so that $g_{i_m j_m} = g_{j_m i_m}$.

Crossover is complicated by the following registration problem. A good crossover genetic operator must "cross" pairs of assignments in such a way that good features of parents may be preserved in the resulting candidate. But the objective $f(A)$ for VQ is unchanged by relabeling clusters: Let $A = a_1, a_2, ..., a_{|V|}$ and let $\pi : Z \rightarrow Z$ be any permutation on the first $|K|$ integers. Then

$$A' = \pi(a_1), \pi(a_2), ..., \pi(a_K).$$

is "really" the same clustering as A and $f(A) = f(A')$. No simple crossover will respect this invariance and will scramble features in the offspring.

We propose the following catalytic crossover as a solution in Figure 2. This particular crossover operator is *reversible*: for a population I, if there exists a population J that can be obtained from I in one crossover operation then there exists a unique crossover operation that will convert population J back into I. Reversibility is convenient for showing that the resulting Markov chain has the proper steady state distribution. In this case the distribution is Boltzmann as can be shown by analysis of the detailed balance of probability mass between any two states.

In Figure 2, the upper and lower solid boxes represent vectors of color k_1 and k_2 in the first parent solution p_1. The dashed box on the right represents vectors of color k' in the second parent p_2. The vertically hatched intersection comprises the vectors that are assigned to k_1 in p_1 and to k'

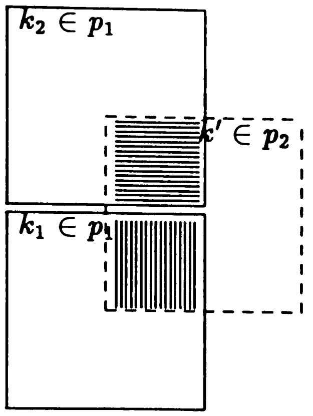


Figure 2: Proposed crossover operator.

in p_2. The horizontally hatched intersection comprises the vectors that are assigned to k_2 in p_1 and to k' in p_2. The new (candidate) solution is obtained by swapping the colors in the two hatched subsets in p_1 . The justification is that these vectors are "close" to each other because they have the same color in p_2. If p_1 the first parent is chosen uniformly from the population and the second parent p_2 is chosen uniformly from the remaining population $I \backslash i_{p_1}$, and the candidate q is chosen as above from $I \backslash \{i_{p_1} \bigcup i_{p_2}\}$, then the generator $g_{p_1,p_2,q} = g_{q,p_2,p_1}$ is symmetric and uniform and cancels out of the acceptance expression as before.

A genetic annealing algorithm can be constructed with this mutation and crossover operator from the previously discussed scheme for population annealing by choosing with specified probability whether to take a mutation step or a crossover step.

Figure 4 summarizes the performance of the resulting genetic annealing algorithm for a vector quantization of the standard test image shown in Figure 3. Each trace in Figure 4 is a plot of the logarithm of the objective function versus the logarithm of the number of iterations for a specified probability of crossover and mutation. This particular format of presentation provides a useful means of comparing the performance (quality and efficiency) of optimization algorithms. On such a log-log plot, uniform random search produces a straight line as in trace 1. As can be seen, genetic annealing of the assignment vectors, trace 2, is superior to uniform random search through the assignment vector space, trace 1. When cooled according to $T_k = C/\ln(k)$ for large enough C, genetic annealing will eventually find the global minimum.

Traces 3 and 4 are variations of greedy descent algorithms based on optimizing the code vectors instead of the assignments. Descent algorithms fall rapidly until they become constant at some local minimum. However the efficiency of these particular algorithms indicate that optimization of code vectors may be a more attractive starting point than optimization of assignment vectors: Trace 1 and trace 3 are both greedy descent algorithms based on uniformly generated random vectors. We address this version of genetic annealing in the next section.

GAUSSIAN GENETIC ANNEALING

In this section we treat vector quantization as optimization of real valued code vectors. We regard the c_k independent variables and define the assignment of vector i to cluster $a_i = \text{argmin}_k |v_i - c_k|$,

Figure 3: A section of a standard test image in image processing literature used to exercise vector quantization algorithms.

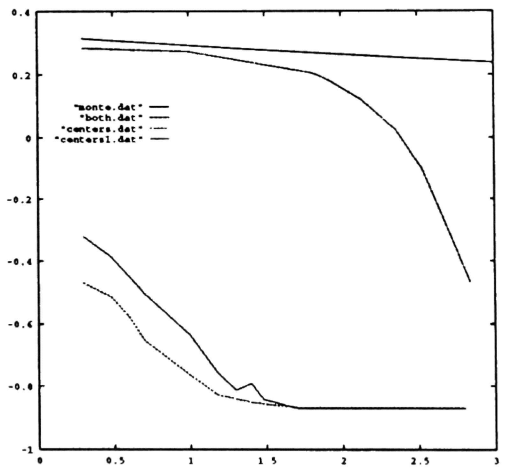
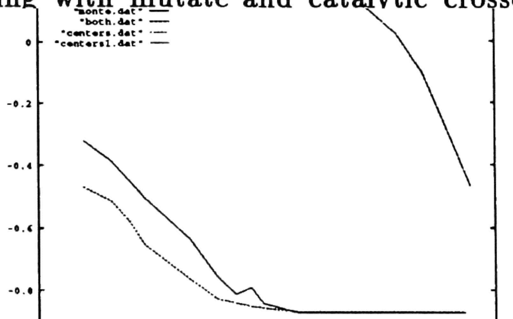


Figure 4: Logarithm of objective function versus logarithm of iteration for several vector quantization algorithm in order of increasing performance: **1** Greedy descent with uniformly random assignments. **2** Genetic annealing with mutate and catalytic crossover. **3** Greedy descent with



where c_k is the k^{th} code vector with components c_{dk}. We constrain the search to some hyperbox $c_{d,min} < c_{dk} < c_{d,max}$ for each code vector k and each dimension d. The c_{dk} are real, but we are only interested in objectives that smoothly varying so we can discretize each component uniformly to some specified tolerance and apply the preceding discrete theory.

Instead of a mutation/crossover pair, we propose to generate candidates from $g_{i_m j_m}$, a d-dimensional Gaussian with mean and covariance equal to the diagonal part of the sample covariance of $I \backslash i_m$. This results in the following algorithm:

1. Select initial populations of solutions I, $k = 1$
2. Set $T_k = C/\ln(1+k)$.
3. Select a member i of I with coordinates c_i
4. Compute the means $\mu = \mu_1, \mu_2, ..., \mu_d$ and variances $\sigma^2 = \sigma_1^2, \sigma_2^2, ..., \sigma_d^2$ of the rest of the population $I \backslash i$.
5. Generate a vector u from a multivariate Gaussian density G(u) with mean μ and covariance $\text{diag}(\sigma_1^2, \sigma_2^2, ..., \sigma_d^2)$.
6. Replace the coordinates of member $c_i \leftarrow u$ with probability $\min\left(1, \frac{G(c_i)f(u)}{G(u)f(c_i)}\right)$
7. Go to step 3

This algorithm can be shown to have the right steady state distribution if the covariance does not happen to get too small[5]. Under the same condition, the sum in Equation 1 diverges. Strictly speaking therefore, it is necessary to constrain the covariance in step 4 away from zero for guaranteed convergence, but this does not seem necessary in practice. Using the complete covariance instead of its diagonal part should produce a more efficient algorithm in general (and permit a smaller C in Equation 3), but the diagonal part of the covariance provides a simple initial implementation.

Preliminary results on a 4-dimensional problem are shown in the following figures. Figure 5 shows a slice through the objective function with a single global minimum and several local minimum. In the next figures, we present the results of genetic annealing for a population of 50 solutions. This cooling schedule is more aggressive than the guaranteed logarithmic schedule of the preceding sections, but slow enough to allow the solutions to converge to the global minimum about 90 percent of the time. We present only the first of the coordinates. The value of this first coordinate at the global minimum is $c_1 = 17$. Figure 6 shows the values of c_1 of the solutions generated in genetic annealing as a function of iteration. Note that the values are initially scattered over the entire allowed domain of c_1 which some preference for values around the deepest non-global minimum at $c_1 \approx 10$. Figure 7 shows the first coordinate of the mean of the the population converges more quickly than the sequence of new solutions. The variance of the population also converges quickly as in Figure 8. The objective function as a function of iteration in Figure Objectives reflects four coordinates of each new candidate.

CONCLUSIONS

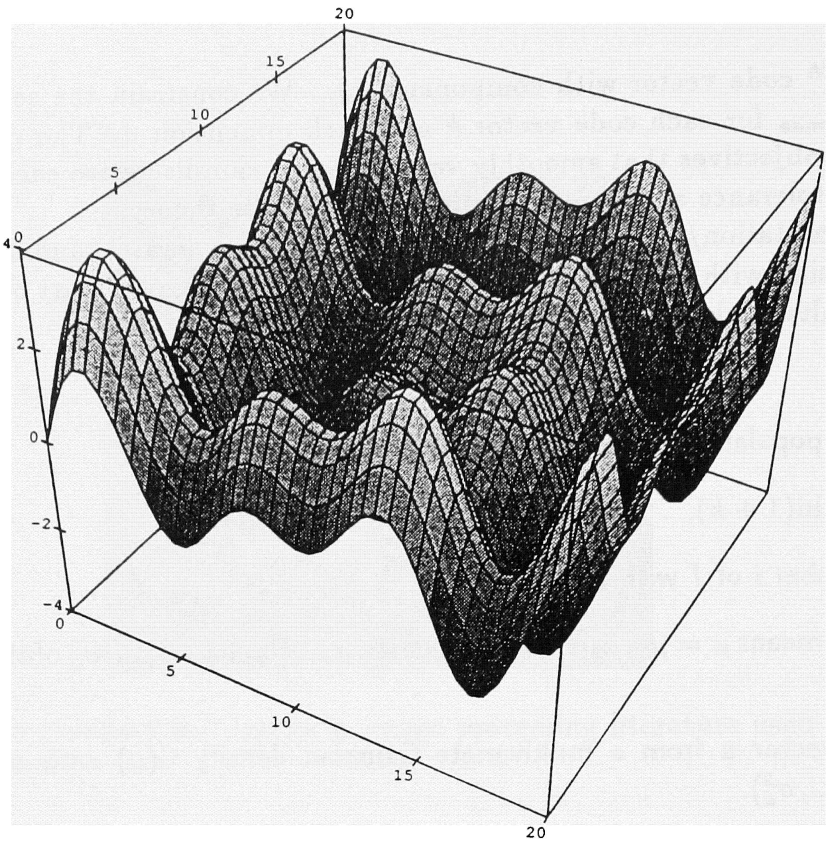
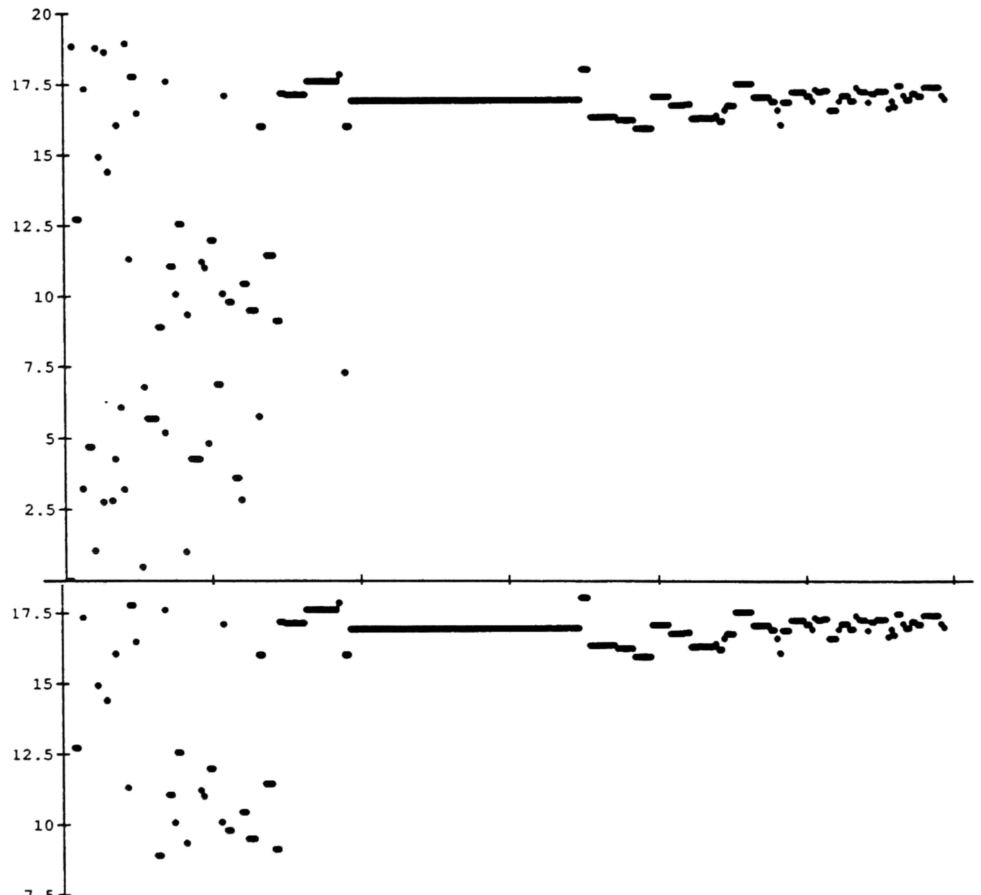


Figure 5: A two-dimensional slice through the 4 dimensional objective.



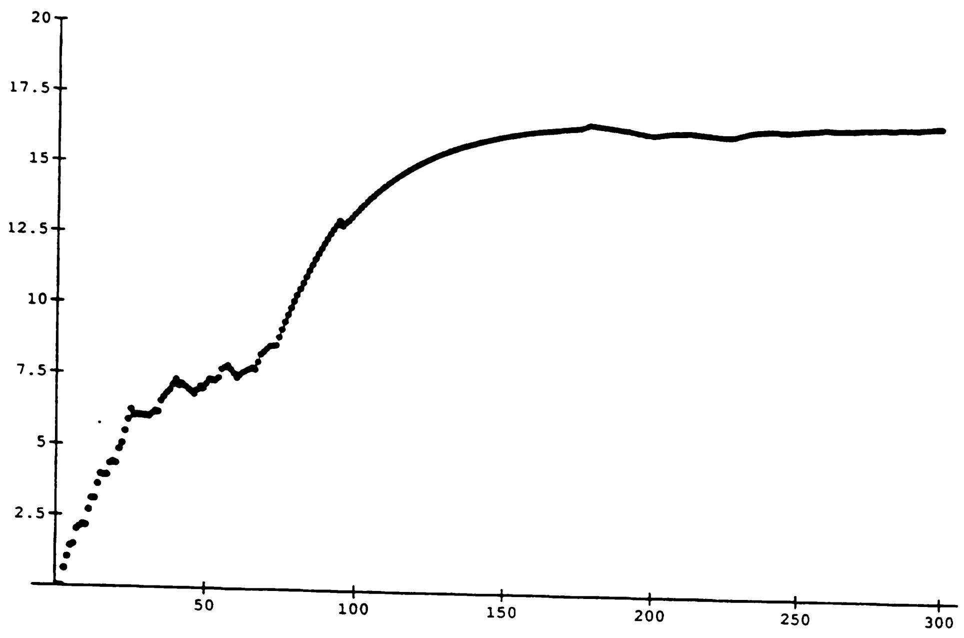


Figure 7: Means of population of current 50 accepted solutions.

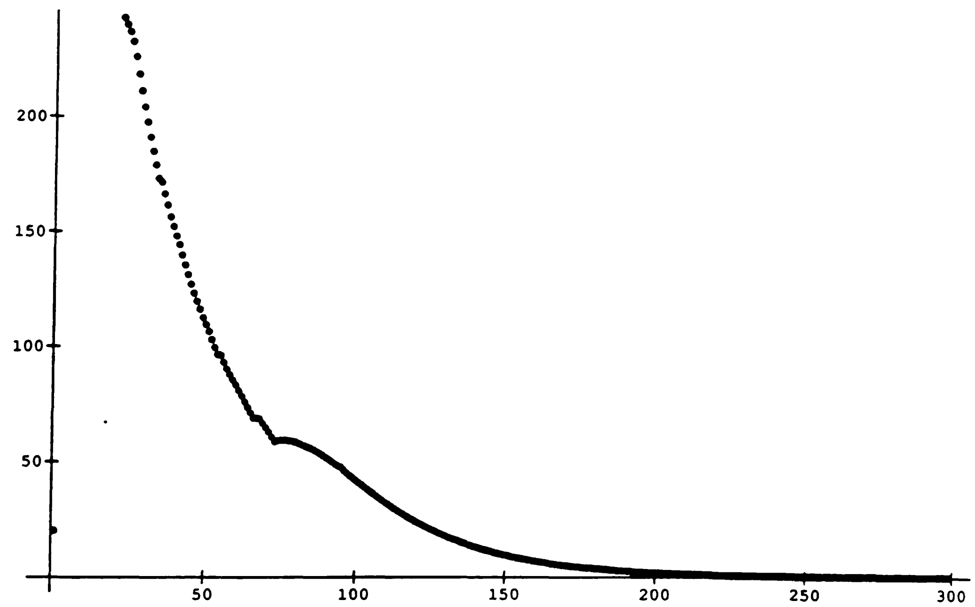


Figure 8: Variances of population of current 50 accepted solutions.

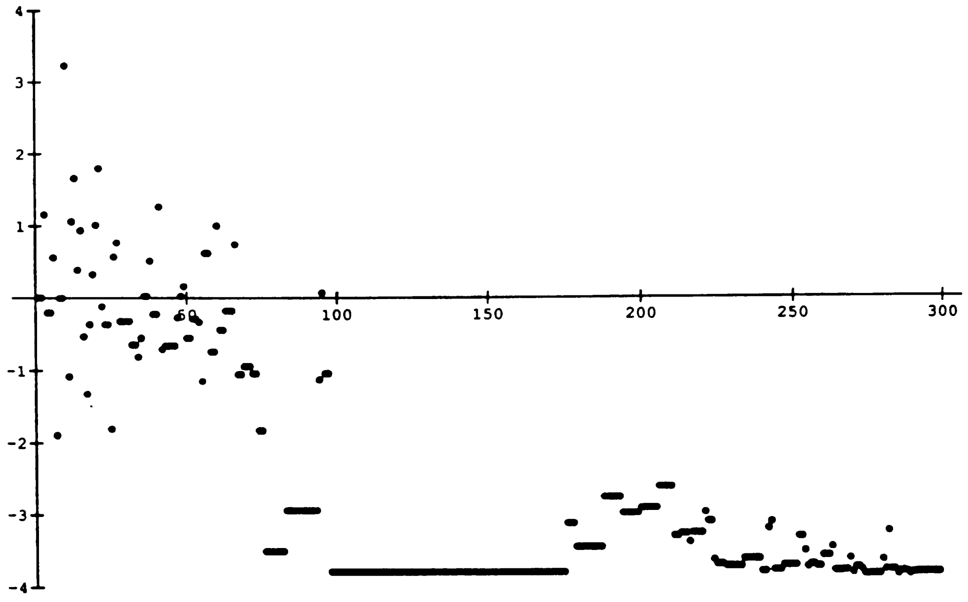


Figure 9: Objectives of accepted solutions.

We have introduced genetic annealing as biased simulated annealing applied to a population of solutions. We show how to construct the generator to treat the population as a single point in a larger search space comprising the combined coordinates of the member solutions. The members of the population were used to generate the next candidates. The steady-state distribution of the the population is Boltzmann in the search space. Convergence for a slow enough logarithm cooling schedule is guaranteed by the detailed construction of the generator and due to existing results for simulated annealing with generalized acceptance probabilities. We have introduced a log-log plot as a means of comparing different algorithms and used it to compare two versions of uniform random optimization with a genetic annealing algorithm.

ACKNOWLEDGEMENTS

This work was supported by *The Center for Communications and Signal Processing* of North Carolina State University and by *The Eastman Kodak Company* under Project Number 590-6411-A-2076.

References

[1] S. Anily and A. Federgruen. Simulated annealing methods with general acceptance probabilities. *J. Appl. Prob.*, 24:657–667, 1987.

[2] J. N. Bhuyan, V. V. Raghavan, and V. K. Elayavalli. Genetic algorithm for clustering with an ordered representation. *See final report.*

[3] G. L. Bilbro. A general method for accelerating sa for bayesian restoration. In *SPIE Conference on Stochastic Methods in Sig. Proc, Image Proc., and Computer vision*, pages 88–98. SPIE, San Diego, CA, 1991.

[4] G. L. Bilbro, M. B. Steer, R. J. Trew, C.-R. Chang, and S. G. Skaggs. Extraction of the parameters of equivalent circuits of microwave transistors using tree annealing. *IEEE Trans. Microwave Theory and Techniques*, Nov. 1990.

[5] Griff L. Bilbro and Wesley E. Snyder. Optimization of functions with many minima. *IEEE Trans. Systems, Man, and Cybernetics*, 21(4), 1991.

[6] D. Geman and S. Geman. Stochastic relaxation, Gibbs Distributions, and the Bayesian restoration of images. *IEEE Transactions on PAMI*, PAMI-6(6):721–741, November 1984.

[7] A. D. Gordon and J. T. Henderson. An algorithm for Euclidean sum of square classification. *Biometrica*, 33:355–362, 1977.

[8] A. Törn and A. Žilinskas. *Global Optimization.* Lecture Notes in Computer Science. Springer-Verlag, 1989.

[9] D. Vanderbilt and S. G. Louie. A Monte Carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 36:259–271, 1984.