MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1345 December 1991

Boltzmann Weighted Selection Improves Performance of Genetic Algorithms

Michael de la Maza and Bruce Tidor

Abstract

Modifiable Boltzmann selective pressure is investigated as a tool to control variability in optimizations using genetic algorithms. An implementation of variable selective pressure, modeled after the use of temperature as a parameter in simulated annealing approaches, is described. The convergence behavior of optimization runs is illustrated as a function of selective pressure; the method is compared to a genetic algorithm lacking this control feature and is shown to exhibit superior convergence properties on a small set of test problems. An analysis is presented that compares the selective pressure of this algorithm to a standard selection procedure.

Keywords: genetic algorithms; simulated annealing; function optimization; hybrid search strategies.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology and at the Whitehead Institute for Biomedical Research. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124. The Whitehead Institute Fellow's Program is supported by generous grants from the Lucille P. Markey Charitable Trust and the W. M. Keck Foundation.

1. Introduction

A number of problem solving methods in current use are based on paradigms derived from natural phenomena. Examples include simulated annealing, neural networks, and genetic algorithms. The first of these is modeled after physical systems that are remarkably successful at finding global optima by sampling a potential energy surface as the temperature is slowly reduced (Kirkpatrick *et al.*, 1983). At higher temperatures, relatively larger excursions over the potential energy surface are permitted. During cooling, the system evacuates less favorable optima and becomes trapped in the neighborhood of more favorable ones; the amount of parameter space sampled effectively decreases with the temperature, and the system generally converges to very good local solutions. Simulated annealing has been implemented using both first-derivative methods, in which equations of motion on the potential energy (or general optimization) surface are integrated to produce the search path (Verlet, 1967), and Monte Carlo methods, which do not require derivative information (Metropolis *et al.*, 1953). In both cases a temperature parameter is used to control the optimization. Artificial neural networks, inspired by the highly interconnected, relatively simple, non-linear processing units found in biological nervous systems, are proving useful in areas of machine learning and pattern recognition. Genetic algorithms are based on the same principles of natural selection that describe the evolution of sizable biological populations over time scales covering a large number of generations. A fitness function describes the success of each member of the population in terms of that member's parameters (genetic makeup or "genes"); the fitness is a direct measure of an individual's reproductive potential, which follows in some measure the imperative, "Survival of the fittest" (Darwin, 1859). Mechanisms for creating diversity are also incorporated, including, but not limited to, mutation and crossover.

Genetic algorithms are atypical in that many solutions are followed in parallel and these are recombined in search of improved ones. The evolutionary aspect provides for the elimination of trial solutions that are relatively unsuccessful, but a variety of selection criteria are possible. The quality of the overall result and the computational effort required depend critically on the selection criteria used. Here we compare the standard proportional scaling method with a new Boltzmann-based protocol. As an illustration, note that one extreme selection scheme would allow only copies of the fittest individual to survive. Variability would be introduced only by mutation (and, if so desired, by crossover of mutant siblings); this would correspond to a highly parallel Monte Carlo search, but at zero temperature (i.e., a simple "always improving" optimization). While this might be efficient to perfect the best optimum once it had been located, it would be extraordinarily inefficient for most problems at the start of an optimization. In fact, for small enough mutational steps in the parameter space, it would lead to the local optimum closest to the fittest individual in the starting population. This corresponds to a "zero tolerance" evolutionary system, in which the slightest advantage of one individual over another results in the loss of the less fit individual from the gene pool. The other extreme would be an "infinitely tolerant" environment; i.e., one that permits all individuals to survive to reproduction, regardless of fitness. If the total number of individuals in the population is fixed, this corresponds to a random walk in the space with no preference for optima. Good solutions that are found are likely to be lost to mutation and crossover. Between these two extremes lies a continuum of evolutionary tolerance. Early in an optimization, it would be useful to have a high tolerance, so that the search is carried out over a large portion of the space (like the initially high temperature used for simulated annealing) and a large variety of individuals are retained in the population so that, even if they, themselves, are not of high fitness, they might donate to a crossover that produces an exceptionally fit individual. Later in the procedure, when the major optima have been located and partially refined, it would be reasonable to eliminate the lesser optima and concentrate on refining the better ones, so a lower tolerance would be useful.

We have implemented a genetic algorithm using such a scheme for varying the evolutionary tolerance of the environment with Boltzmann scaling. The plan of the rest of this paper is as follows. In Section 2 we outline the theory of the Boltzmann scaling method. In Section 3 we describe a set of trial problems and present the empirical design of a tolerance schedule, a comparison between Boltzmann and standard scaling, and an analysis of the variability of selective pressure with standard scaling. Section 4 contains a discussion of the results, and Section 5 presents our conclusions.

2. Theory

Generally there is a function to be optimized, $U(\mathbf{R})$, which depends upon the parameters, $\mathbf{R}$. A transformation is applied to produce a fitness function, $F(\mathbf{R})$, which ensures non-negativity, provides a sign change when minimization, rather than maximization, is desired, and introduces variable parameters to aid in the optimization (De Jong, 1975; Baker, 1985; Grefenstette and Baker, 1989).

Commonly, in the selection step, the number of offspring propagated into the next generation by an individual, j, with genetic makeup, $\mathbf{R}_j$, and fitness, $F(\mathbf{R}_j)$, is,

$$N_j^{i+\frac{1}{2}} = \frac{F(\mathbf{R}_j)}{\bar{F}_i} \tag{1}$$

where $\bar{F}_i$ is the average fitness in generation i and we have used the notation $i + \frac{1}{2}$ to indicate that genetic operators, such as mutation and crossover, are applied to the resulting set of individuals to produce the population in generation $i + 1$. This selection technique is known as proportional scaling applied to fitness, but how it selects on the optimization function depends greatly on the nature of the transformation mapping $U(\mathbf{R})$ to $F(\mathbf{R})$ as well as on the distribution of individuals in optimization space.

In an equilibrated simulated annealing ensemble, the probability of visiting a point in optimization space, $\mathbf{R}_j$, is,

$$P(\mathbf{R}_j) = \frac{\mathrm{e}^{-U(\mathbf{R}_j)/T}}{\sum_i \mathrm{e}^{-U(\mathbf{R}_i)/T}} \tag{2}$$

where the minus sign in the exponent is necessary because a minimization is performed, T is the temperature, the numerator contains the Boltzmann weighting term, and the denominator is a normalization factor. The Boltzmann function has the property that at higher temperatures the system visits more of phase space, whereas at lower temperatures the probability of visiting points more unfavorable than the global minimum is lower. We have implemented an analogous equation in the selection step of our genetic algorithm. The transformation from optimization function to fitness function is,

$$F(\mathbf{R}) = \mathrm{e}^{+U(\mathbf{R})/T} \tag{3}$$

where T is a variable parameter corresponding to evolutionary tolerance (analogous to temperature in simulated annealing) and the plus sign is changed to minus when minimization, rather than maximization, is desired. Thus, in terms of the optimization function,

$$N_j^{i+\frac{1}{2}} = \frac{\mathrm{e}^{+U(\mathbf{R}_j)/T}}{\left\langle \mathrm{e}^{+U(\mathbf{R})/T} \right\rangle_i} \tag{4}$$

where $\langle\ \rangle_i$ indicates an average over the population at generation i. Equation (4) is analogous to the proportion of time that a simulated annealing optimization spends in the neighborhood of $\mathbf{R}_j$, Equation (2), and we refer to it as Boltzmann scaling applied to the optimization function. In what follows, we compare this method to proportional scaling applied to the optimization function, in which,

$$F(\mathbf{R}) = U(\mathbf{R}) \tag{5}$$

and so,

$$N_j^{i+\frac{1}{2}} = \frac{U(\mathbf{R}_j)}{\bar{U}_i} \tag{6}$$

where $\bar{U}_i$ is the average value of the optimization function in the i-th generation.

The Boltzmann formulation provides a number of attractive features. The result of the selection step is independent of overall translational shifts in the optimization surface (i.e., the offspring from a given generation using the function $U'(\mathbf{R}) = U(\mathbf{R}) + c$, for any constant, c, are equivalent to the offspring produced using the function $U(\mathbf{R})$). Scaling the optimization surface by a constant, so that

$U'(\mathbf{R}) = cU(\mathbf{R})$, which corresponds to changing the units in which $U(\mathbf{R})$ is measured, is also invariant so long as the parameter, T, which has the same units as $U(\mathbf{R})$, is similarly scaled. Moreover, there is no requirement that the optimization function be non-negative, since the exponential provides an appropriate transformation.

3. Practice

In this section we first describe two model problems used to compare Boltzmann and proportional scaling, we then explain how a tolerance schedule for the Boltzmann GA was chosen and present comparative results showing faster convergence for the Boltzmann GA. Finally, we provide an empirical analysis that illustrates that a genetic algorithm with proportional scaling increases, rather than decreases, evolutionary tolerance as the point of completion nears (contrary to what one might wish).

3.1 Description of Model Problems

3.1.1 Molecular Biology Problem. This section describes a problem, inspired by molecular biology, in which a pattern must be built that distinguishes between functional and non-functional protein sequences.

A database of instances, composed of the twenty letters used to represent the twenty amino acids, is divided into positive and negative classes. A random pattern is generated and the same pattern is embedded in a random location in all of the positive instances. The goal is to find this pattern or an acceptable substitute. In addition to containing any character that can appear in the instances, the substrings, also called individuals, can contain a "don't care" symbol, which matches any character.

A typical database is shown in Table 1. All of the positive instances contain the substring *RIEY* while none of the negative instances do. Each database has ten instances, each having a 0.5 probability of being in the positive class.

The optimization function, $U(\mathbf{R})$, is a measure of the difference between how well an individual matches the positive instances and how well it matches the negative instances. The score of individual i_j is calculated using,

$$U(i_j, I) = \frac{1}{|P|} \sum_{I_k \in P} \max(\text{match}(i_j, I_k)) - \frac{1}{|N|} \sum_{I_k \in N} \max(\text{match}(i_j, I_k)) \qquad (7)$$

where I is the set of all instances, P is the subset containing $|P|$ positive instances and N is the subset containing $|N|$ negative instances. The matching function returns a list of numbers that indicate how well an individual matches each substring of an instance. A point is given for each character that correctly matches

Table 1. Typical database. Each instance has sixteen letters. The instances in the positive class contain the sequence "RIEY".

Instance																Class
K	D	G	R	W	E	G	G	G	H	W	V	M	A	R	M	negative
N	H	I	T	R	H	Y	V	Q	P	C	C	C	Y	D	K	negative
T	I	C	N	V	S	D	Q	W	L	F	F	K	L	W	S	negative
E	E	P	S	R	I	E	Y	T	I	M	G	I	E	V	T	positive
V	P	Y	K	P	C	P	K	H	S	L	S	G	A	F	K	negative
R	I	E	Y	R	W	P	V	K	V	R	H	Q	N	Y	G	positive
V	A	H	R	C	K	N	W	Q	M	T	W	I	T	H	Q	negative
T	L	Q	F	Y	K	E	N	D	L	T	K	C	G	L	K	negative
R	C	W	Y	K	N	A	Y	I	G	Q	Y	V	C	P	H	negative
G	V	M	Q	G	T	R	I	E	Y	Y	F	F	C	G	S	positive

and half a point is given for the "don't care" symbol. For example, the individual **INE*, when matched against the instance *THISISFINE*, returns 0.5 when matched against *THIS*; 0.5 when matched against *ISIS*; and 3.5 when matched against *FINE*.

Both the Boltzmann and proportional GAs shared the following properties. There were three recombination operators: crossover, mutation, and shift. The crossover operator was traditional 1-point crossover. Mutation was accomplished by randomly switching exactly one character in an individual to another character. The shift operator performed a cyclic permutation. To create the next generation from the present one, first a selection step (either Boltzmann or proportional scaling) was performed, creating generation $i + \frac{1}{2}$ from generation i. Each of these individuals was examined in turn and one of the three operators was chosen (at random in the ratio crossover:mutation:shift of 2:1:1) and applied to create an individual for generation $i + 1$. In the case of crossover, an individual in generation $i + \frac{1}{2}$ was crossed over with any of the individuals in that generation (including himself, producing the identity transformation) with equal probability.

The shift operator was introduced because many runs converged to a local optimum that was a cyclic permutation away from the global optimum (correct answer). With mutation and crossover alone, the rate of moving from the local optima to the global optimum is negligible because it requires crossing deep valleys. The cyclic permutation shift operator crosses these valleys in a single step.

The mutation rate (25%) seems deceptively high. For individuals with eight characters, each character was mutated with an average probability of 3.125%.

If the twenty-one characters are represented as bits, then approximately 4.4 bits are needed to represent each character. Thus, the mutation rate per bit was approximately 0.7%, which is similar to that of other genetic algorithms.

3.1.2 F2 Function. The F2 function (Deb and Goldberg, 1989) is:

$$F2(x) = \sin^6(5\pi x) \exp\left[2 \ln 2 (\frac{x - 0.1}{0.8})^2\right] \tag{8}$$

On the interval [0.0, 1.0], F2 has five peaks, each one smaller than the previous one (see Figures 4, 5, and 6).

Individuals for both the Boltzmann and proportional GAs were composed of three decimal digits and represent a value between 0.000 and 0.999 (inclusive). The optimization function was simply the value of F2 for the x value encoded by the individual. The population consisted of 100 individuals. The 1-point crossover rate was 90% and the mutation rate was 10%. The mutation operator added a uniform random number between 0.1 and -0.1 to the individual. To create the next generation from the present one, first a selection step (either Boltzmann or proportional scaling) was performed, creating generation $i + \frac{1}{2}$ from generation i. Each of these individuals was processed in turn and one of the two operators was chosen (at random in the ratio crossover:mutation of 9:1) and applied to create an individual for generation $i + 1$. In the case of crossover, an individual in generation $i + \frac{1}{2}$ was crossed over with any of the individuals in that generation (including himself, producing the identity transformation) with equal probability.

3.2 Finding the Initial Tolerance

The appropriate initial tolerance value was determined by performing a series of experiments. The tolerance schedule is shown in Figure 1. This tolerance schedule was chosen by adapting a successful simulated annealing cooling schedule to genetic algorithms. The tolerance is constant for the first ten generations and then ramps down over the next thirty generations to a final value. The final tolerance was set to be 0.5.

Experiments using the molecular biology problem with a four character pattern were used to determine the initial tolerance. Ten initial tolerances were tested: 0.5, 1.25, 2, 3.5, 5, 6.5, 8, 9.5, 11, and 12. The number of generations required for convergence (see Section 3.3.1) was recorded; the results are shown in Figure 2. Each experiment was repeated eight times; the numbers shown are averages.

The U-shaped curve in Figure 2 is in accordance with our intuition about how the initial tolerance should affect search behavior. If the initial tolerance was too high, then the genetic algorithm spent too much time performing a random search and required a long time to focus on the few good solutions. If the initial tolerance was too low, then the genetic algorithm performed a local search around

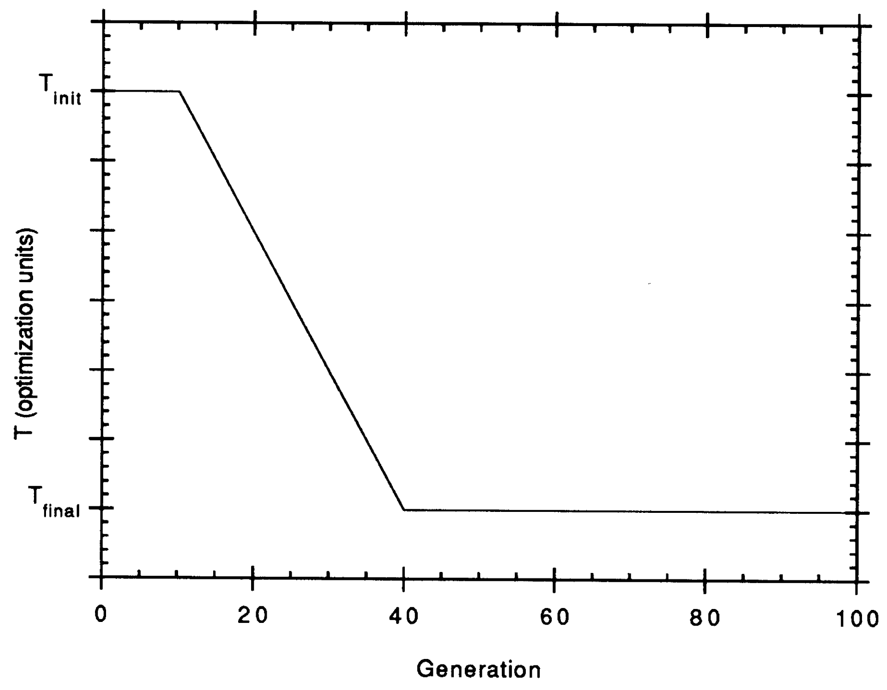


Figure 1. The tolerance schedule used in the Boltzmann selection genetic algorithm. A constant tolerance of T_{init} is used for the first ten generations, followed by a linear ramp down to T_{final} over thirty generations, and finally at a constant tolerance of T_{final} until completion. In all runs, $T_{final} = 0.5$ optimization units.

the individual with the highest fitness in the initial population and, therefore, risked never finding the solution.

On the basis of these results, an initial tolerance of 4 was chosen for the next series of experiments. Unless otherwise noted, this tolerance schedule was used for all of the problems discussed in this paper. The observation that other optimization surfaces were searched reasonably quickly with the same schedule suggests that the method is robust with respect to fine details of the tolerance schedule.

3.3 Comparison

This section compares Boltzmann scaling and proportional scaling on a small set of molecular biology problems and the F2 function (Deb and Goldberg, 1989).

3.3.1 Molecular Biology Problem. The results of comparing the Boltzmann and proportional GAs are shown in Table 2. The first and second columns give the number of characters in the instances and in the patterns. The third column shows the size of the population. The fourth column indicates how many times each experiment was performed. The fifth and sixth columns give the average number of generations for the Boltzmann and proportional GAs to converge. For this purpose convergence is defined as finding a pattern that is a perfect match

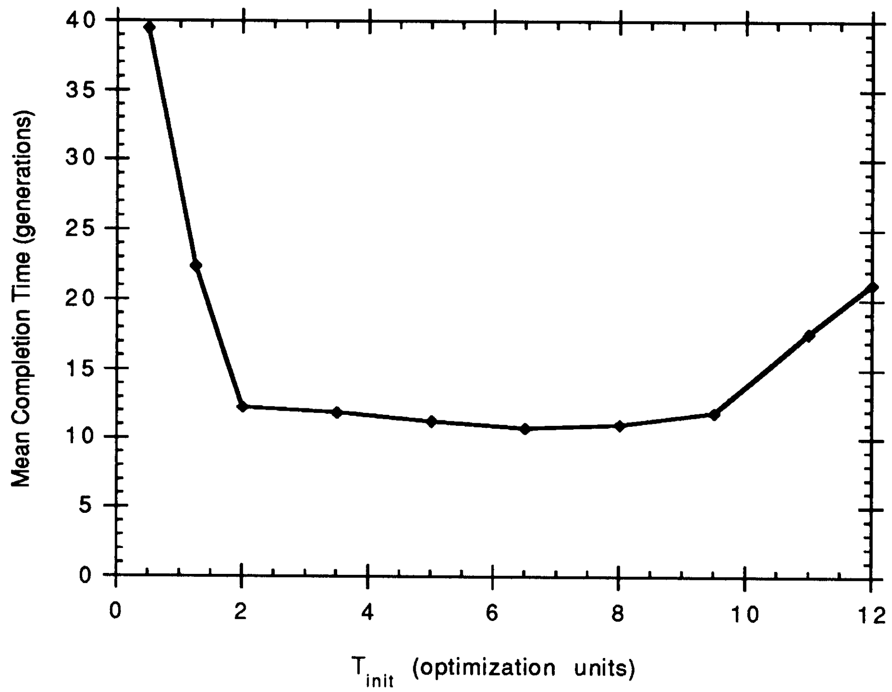


Figure 2. The average number of generations to completion of the Boltzmann scaling genetic algorithm for the problem of pattern length four. Experiments that required more than fifty generations to complete were stopped at fifty generations and combined to compute the average as if they had completed in fifty generations.

in each of the positive instances ("don't care" matches everything) but in none of the negative instances. Note that the GA is not required to find the optimal or characteristic pattern. The last column is the result of applying a one-tailed statistical test: $z = (\mu_1 - \mu_2)/\sqrt{\sigma_1^2/n_1 + \sigma_2^2/n_2}$. If this number is greater than 2.326 then the Boltzmann GA is better than the proportional GA at the $p < 0.01$ level. If it is greater than 2.576 then it is significant at the $p < 0.005$ level. A one-tailed (rather than two-tailed) test was used to show that the performance of the Boltzmann GA was superior to (rather than different from) the performance of the proportional GA.

The results are clear. On all three versions of this problem, the Boltzmann GA is far superior to the proportional GA.

Figure 3 shows the progress of the top individual in a Boltzmann scaling experiment. At generation 0, the score is very low and the individual does not match the target pattern, "RIEYGKSD", very well. But after a series of mutations, crossovers, and shifts, the instance is perfectly aligned with the target pattern at generation 19. After this point, the top individual is changed, one position at a time, until it matches the target pattern perfectly. Note that in collecting the data for Table 2 this run would have been considered to converge at generation 34, when the pattern matches all of the positive instances and none of the negative instances.

```
(-------%%%%%%%%-----------------------------------)
(rvftsdtRIEYGKSDawvqekhmkwiqfyprfateshkyiiitgvscvpp)
(-----------cvSwiwwk-------------------------------) Gen:  0      Score:  1.60
(-------------Svswiwwk-----------------------------) Gen:  1      Score:  1.60
(------swIwYGfg------------------------------------) Gen:  2      Score:  5.80
(------swIwYGfg------------------------------------) Gen:  3      Score:  5.80
(-----gswIwYGf-------------------------------------) Gen:  4      Score:  7.60
(-----gnwIwYGf-------------------------------------) Gen:  5      Score:  8.80
(---------wYGKSswi---------------------------------) Gen:  6      Score: 13.80
(--------IwYGKSsw----------------------------------) Gen:  7      Score: 24.20
(--------IwYGKSsw----------------------------------) Gen:  8      Score: 24.20
(--------IwYGKSsw----------------------------------) Gen:  9      Score: 24.20
(--------IwYGKSsw----------------------------------) Gen: 10      Score: 24.20
(--------IwYGKSsw----------------------------------) Gen: 11      Score: 24.20
(--------IwYGKSsw----------------------------------) Gen: 12      Score: 24.20
(--------IwYGKSsw----------------------------------) Gen: 13      Score: 25.60
(--------IwYGKSs*----------------------------------) Gen: 14      Score: 25.60
(------s*IwYGKS------------------------------------) Gen: 15      Score: 27.20
(------s*IwYGKS------------------------------------) Gen: 16      Score: 27.20
(------s*IwYGKS------------------------------------) Gen: 17      Score: 27.20
(------s*IwYGKS------------------------------------) Gen: 18      Score: 27.20
(-------*IwYGKS*-----------------------------------) Gen: 19      Score: 30.00
(-------*IwYGKS*-----------------------------------) Gen: 20      Score: 30.00
(-------*IwYGKS*-----------------------------------) Gen: 21      Score: 30.00
(-------*IwYGKS*-----------------------------------) Gen: 22      Score: 30.00
(-------*IhYGKS*-----------------------------------) Gen: 23      Score: 31.40
(-------*IhYGKS*-----------------------------------) Gen: 24      Score: 31.40
(-------*IhYGKS*-----------------------------------) Gen: 25      Score: 31.40
(-------*IhYGKS*-----------------------------------) Gen: 26      Score: 31.40
(-------*IhYGKS*-----------------------------------) Gen: 27      Score: 31.40
(-------*IhYGKS*-----------------------------------) Gen: 28      Score: 31.40
(-------*IhYGKS*-----------------------------------) Gen: 29      Score: 31.40
(-------*IhYGKS*-----------------------------------) Gen: 30      Score: 31.40
(-------*IhYGKS*-----------------------------------) Gen: 31      Score: 31.40
(-------*IhYGKS*-----------------------------------) Gen: 32      Score: 31.40
(-------*IhYGKS*-----------------------------------) Gen: 33      Score: 31.40
(-------*IEYGKS*-----------------------------------) Gen: 34-42   Score: 43.60
(-------RIEYGKS*-----------------------------------) Gen: 43      Score: 52.00
(------*RIEYGKS------------------------------------) Gen: 44      Score: 52.00
(------*RIEYGKS------------------------------------) Gen: 45      Score: 52.00
(------*RIEYGKS------------------------------------) Gen: 46      Score: 52.00
(------*RIEYGKS------------------------------------) Gen: 47      Score: 52.00
(------*RIEYGKS------------------------------------) Gen: 48      Score: 52.00
(-------RIEYGKS*-----------------------------------) Gen: 49      Score: 52.00
(-------RIEYGKS*-----------------------------------) Gen: 50      Score: 52.00
(-------RIEYGKS*-----------------------------------) Gen: 51      Score: 52.00
(-------RIEYGKS*-----------------------------------) Gen: 52      Score: 52.00
(-------RIEYGKS*-----------------------------------) Gen: 53      Score: 52.00
(-------RIEYGKSD-----------------------------------) Gen: 54      Score: 63.40
```

Figure 3. Sample Boltzmann scaling experiment. On each line the top individual, the generation number, the number of perfect alignments with the positive instances, and the score of the top individual is shown. The top line shows a positive instance with the target region, "RIEYGKSD", in capitals and the rest of the string in lower case. Each line shows the top individual and where it matches the instance. When a letter matches with the target sequence it is capitalized. "*" is the "don't care" character. The complete data base had five positive and five negative instances, so the maximum number of perfect alignments is five.

Table 2. Results of comparison between Boltzmann GA and proportional GA. The first two columns give the length of the instance and pattern. The third column shows the size of the population of individuals. The Runs column indicates how many times each experiment was repeated. The Boltzmann and Proportional columns show the average number of generations needed for each algorithm to converge. The final column gives the result of applying a statistical function to the results.

Instance Length	Pattern Length	Population Size	Runs	Boltzmann	Proportional	Stat
25	4	100	49	12.0	16.3	2.4
35	6	100	44	12.0	25.8	6.5
50	8	100	36	16.9	34.2	6.7

3.3.2 F2 Function. Two experiments were performed using the F2 function (Deb and Goldberg, 1989) to explore the properties of tolerance. The experiments differed only in the distribution of the initial population. The first experiment, performed with a population randomly distributed around the middle peak, demonstrates that the proportional GA does not allow individuals to jump from the middle peak to the second highest peak and then onto the highest peak, while the Boltzmann GA does. It also illustrates how the Boltzmann GA searches the F2 space. The second experiment, performed with a random initial population, shows how tolerance affects the search of the Boltzmann GA and compares it to how the proportional GA searches the F2 space.

In the first experiment, the 100 individuals were randomly distributed between 0.400 and 0.600. The middle peak is at approximately 0.5. Figure 4 shows a snapshot of the proportional GA and Boltzmann GA populations after 50 generations have passed. Notice that the proportional GA was not able to move any individuals from the middle peak, while the Boltzmann GA fully explored the second highest peak and had an individual on the highest peak. Figure 5 shows a time series of the progress of the Boltzmann GA. The population of individuals began, at generation 0, with the 100 individuals on the middle peak. By generation 23, some of the individuals began to explore the second highest peak. At generation 60, there were few individuals left on the middle peak, many individuals on the second highest peak, and a few individuals on the highest peak. By generation 90, almost all of the individuals were on the highest peak.

The second experiment, with a random initial population, demonstrates that the behavior of the Boltzmann GA can be altered by changing tolerance. The first graph in Figure 6 shows the distribution of individuals in the Boltzmann GA subject to a constant tolerance of 10. The second graph repeats the same experiment but with a tolerance of 1. As expected, in the experiment with the higher tolerance, the individuals were comparatively more distributed throughout the space than in the experiment with the lower tolerance. The lower tolerance caused more copies of the highest fitness individuals to be made and therefore there was much more pressure to explore the highest peak than the other peaks. For purposes of comparison, the same experiment done with the proportional GA is also shown.

3.4 Tolerance in the Proportional GA

Given the formalism that has been presented to modify evolutionary tolerance, it is possible to study how the proportional GA sets an effective tolerance value at a given generation by choosing the tolerance that minimizes,

$$\sum_j \left[\frac{U(\mathbf{R}_j)}{\sum_i U(\mathbf{R}_i)} - \frac{e^{U(\mathbf{R}_j)/T}}{\sum_i e^{U(\mathbf{R}_i)/T}} \right]^2 \tag{9}$$

where $U(\mathbf{R}_j)$ is the score of individual j and T is the tolerance.

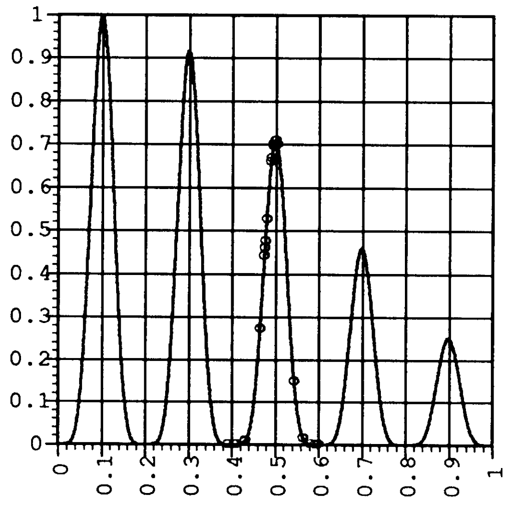
Minimizing this function gives the tolerance which best characterizes the behavior of the proportional GA in the framework of the Boltzmann GA. For runs of the molecular biology problem, the function was minimized using the golden section search described by Press *et al.* (1988).

The results are shown in Figure 7. They indicate that in the proportional GA the effective tolerance increased, rather than decreased, as a function of generation. This result, which runs contrary to both intuition and theory, strongly suggests that the traditional proportional scaling technique may need reconsideration.

4. Discussion

We have implemented Boltzmann scaling on the optimization function to select the number of offspring each individual in the current population contributes to the next generation; the procedure outperforms a standard proportional scaling method on the small set of problems we have investigated. A broader range of problems should be used to test the generality of this result. The tolerance schedule is robust enough that the same schedule was used successfully for problems of different sizes and correspondingly different scales in optimization space. These results show that, for the molecular biology problem, many Boltzmann experiments completed with a correct solution before the decrease in tolerance that occurred after generation ten and nearly all completed before the schedule leveled off again after generation forty.

(A)



(B)

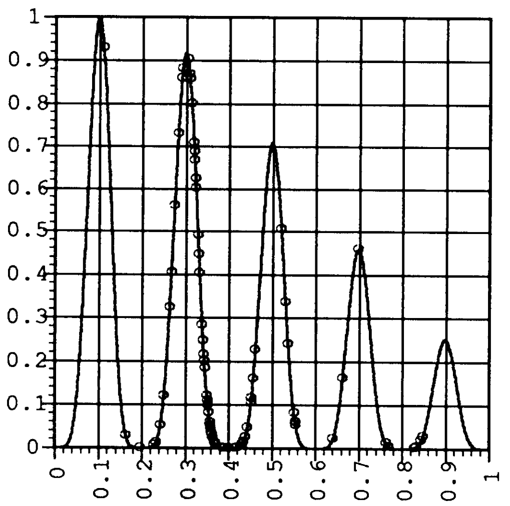


Figure 4. The proportional and Boltzmann GA populations at generation 50. (A) proportional GA population, (B) Boltzmann GA population. The initial population was randomly distributed between 0.400 and 0.600. The individuals are represented by small circles and the F2 function is the dark, continuous line. These graphs show the population immediately after the recombination operators have been applied and before the scaling operation has been done. Notice that none of the individuals in the proportional GA have been able to escape the local optimum of the middle peak.

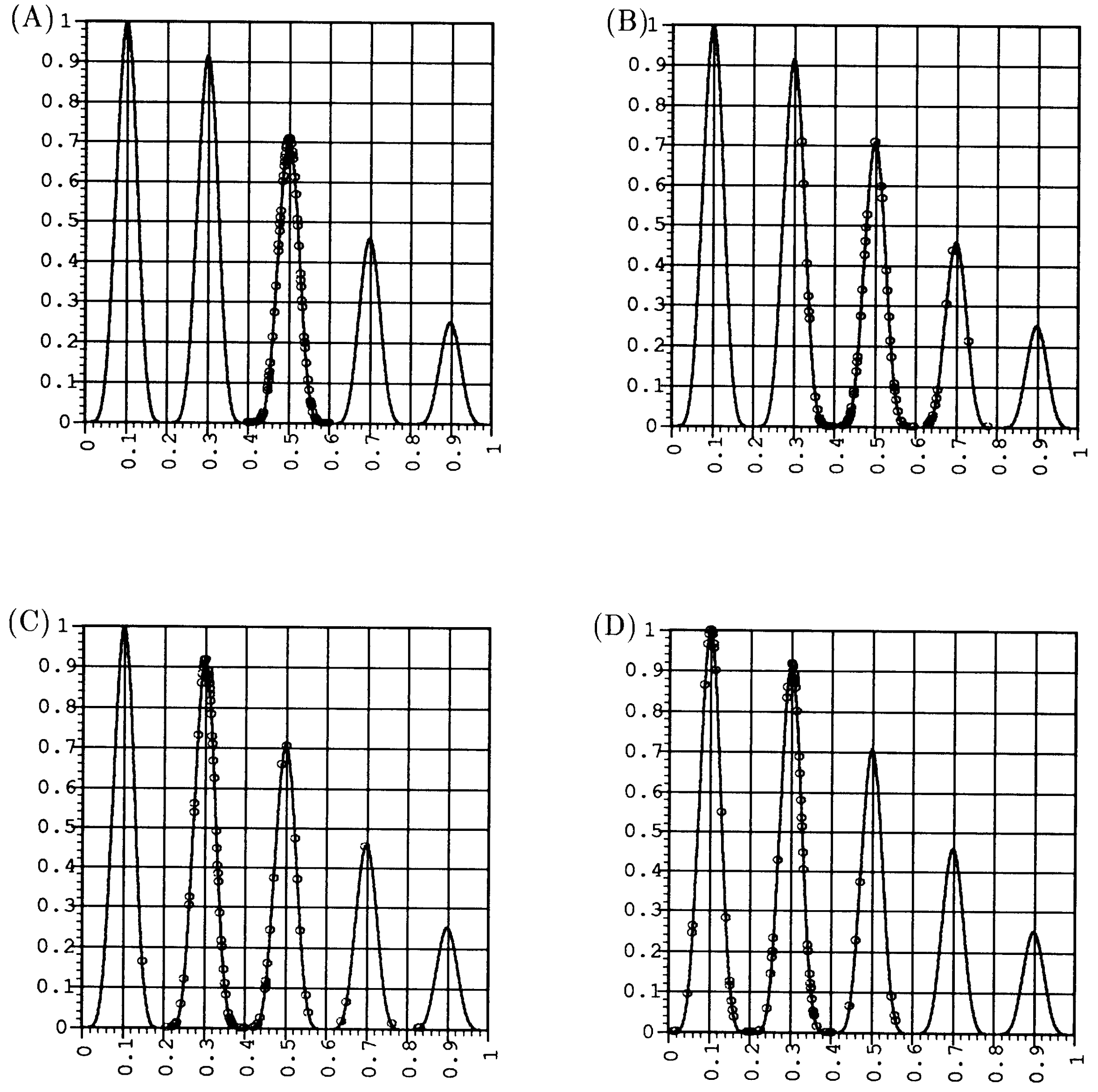


Figure 5. Boltzmann GA population time series. The initial population was randomly distributed between 0.400 and 0.600. The individuals are represented by small circles and the F2 function is the dark, continuous line. These graphs show the population immediately after the recombination operators have been applied and before the scaling operation has been done. Each graph shows the population at a different generation: (A) generation 0, (B) generation 23, (C) generation 49, (D) generation 60, (continued on next page).

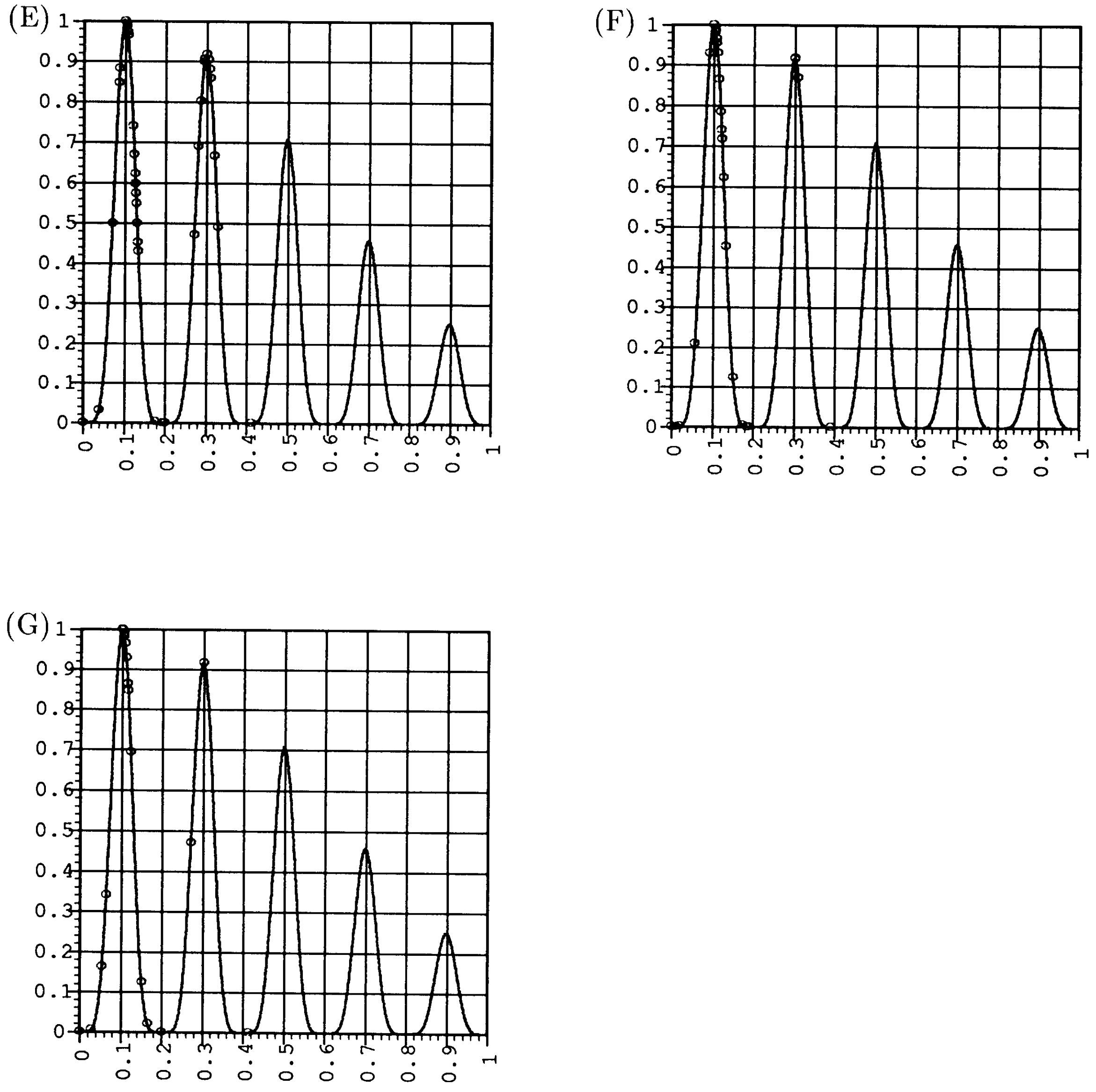


Figure 5. (continued) (E) generation 70, (F) generation 80, (G) generation 90.

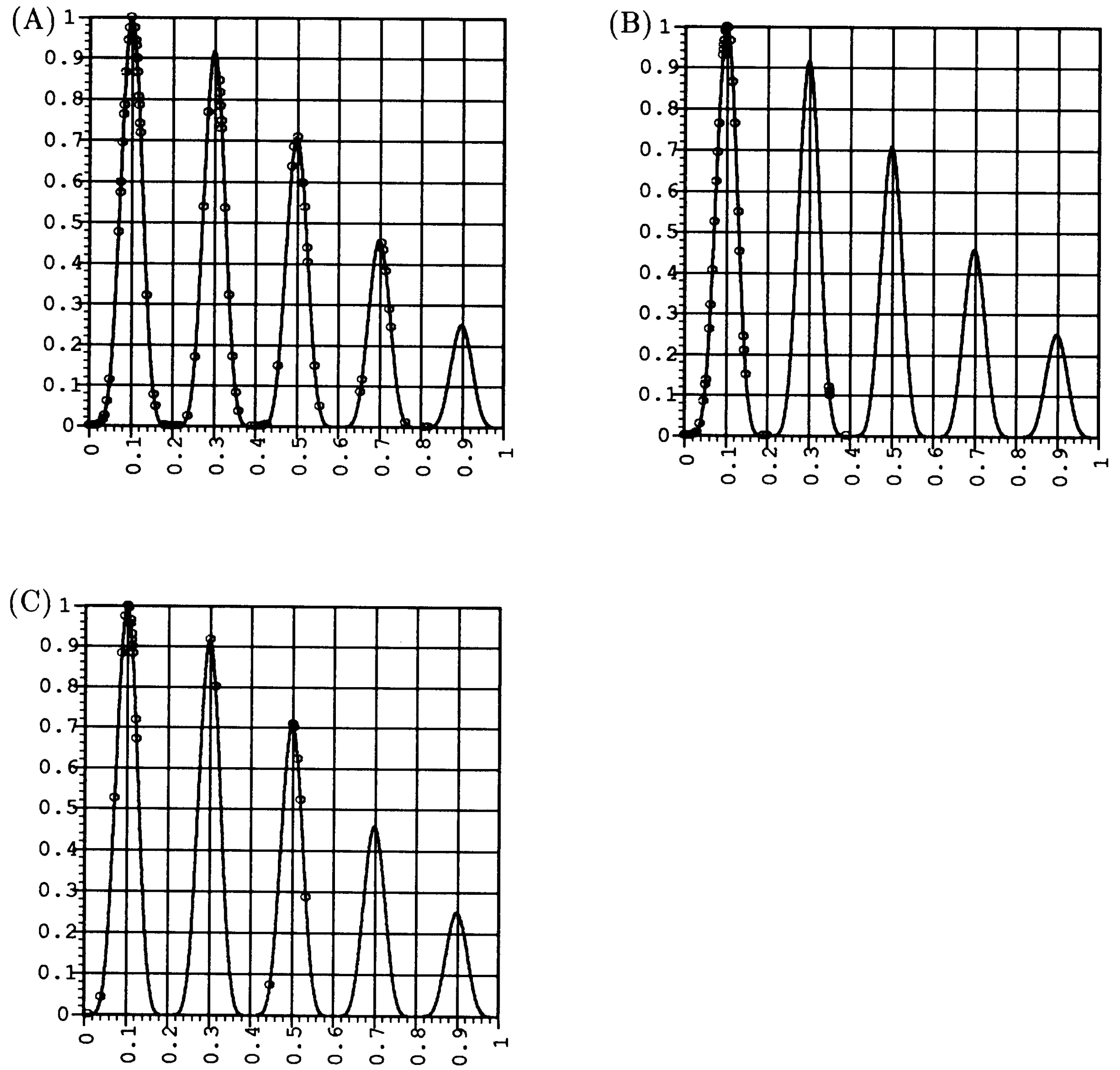


Figure 6. Random initial population. The initial population was randomly distributed between 0.000 and 0.999. The individuals are represented by small circles and the F2 function is the dark, continuous line. These graphs show the population immediately after the recombination operators have been applied and before the scaling operation has been done. (A) Boltzmann GA population at generation 20 with a tolerance of 10, (B) Boltzmann GA population at generation 20 with a tolerance of 1, (C) proportional GA population at generation 20. As expected, the individuals in (A) are comparatively more distributed than the individuals in (B).

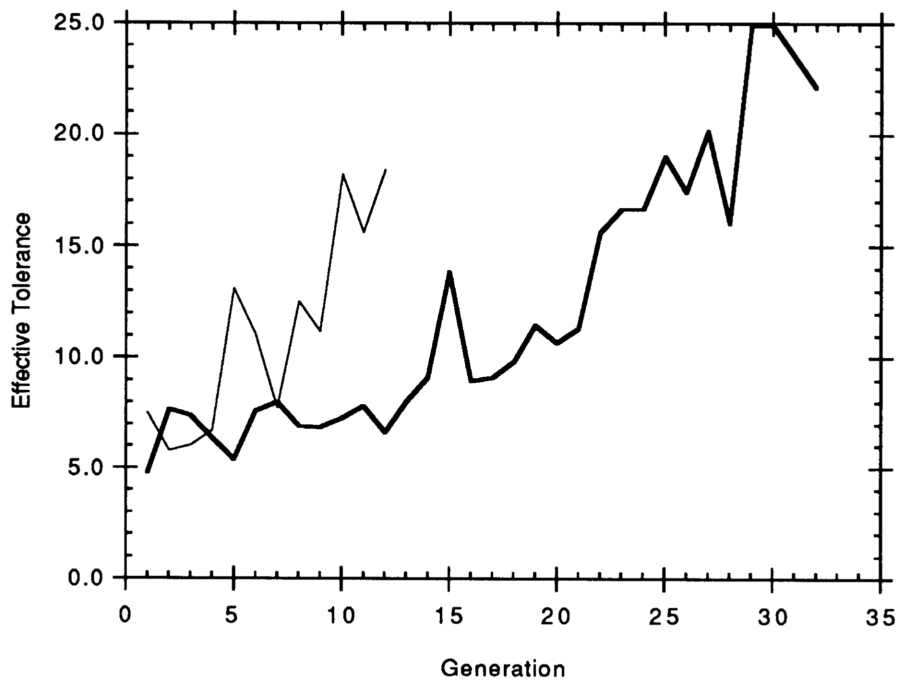


Figure 7. Effective tolerance in proportional GA. The dark line is for an experiment in which the Boltzmann GA outperformed the proportional GA; the light line is for an experiment with the opposite outcome. Both experiments are for patterns of length eight.

One possibility that we have not investigated, but which is used in biological systems, is to vary population size. In high tolerance periods the size of the population could be allowed to increase, and in low tolerance periods it could be forced to decrease. The advantage of such an approach is that more low fitness individuals could be retained for use in crossover during critical stages of the optimization, though it is not clear whether the benefits of this outweight the computational overhead.

A refinement of our method that we have considered is to eliminate all duplicates in the population before applying the Boltzmann selection and adjusting the selection to restore the fixed population size, as would be required by a strict interpretation of the Boltzmann equation. The current distribution of fitness after selection is biased somewhat more toward fit individuals than the refined method would produce, but we expect that any benefit would be small relative to the cost of finding and eliminating duplicates. Moreover, biological systems, particularly those with larger genomes, have no such mechanism. Rather, they use a suite of genetic operators that tend to keep exact duplicates as a low probability event.

Whitley (1987) reports using an exponential selection protocol for a genetic algorithm and found that this increased problems of premature convergence. This contradicts our results and suggests that the use of a reasonable evolutionary tolerance schedule is important (the parameter T in Equations (3) and (4)). It should be noted that the evolutionary tolerance corresponds roughly to the acceptable range of scores, in optimization units, between the best and worst individuals kept after selection; thus, it is expected to vary with the scale of optimization space and the use of trial runs to choose useful parameters (see Figure 2) is valuable.

Goldberg (1990) describes a Boltzmann tournament scheme in which the population of individuals converges to a Boltzmann distribution. The method was developed so that genetic algorithms could benefit from the asymptotic convergence properties enjoyed by simulated annealing and so that simulated annealing procedures might be efficiently implemented on parallel machine architectures. The algorithm includes a non-genetic "anti-acceptance" step that effectively converts between Boltzmann and uniform distributions. Our goal here is to achieve faster convergence to the global optimum rather than to a specific distribution. We use Boltzmann scaling to control the approach to this optimum by varying selective pressure through the tolerance (or its physical analogue, temperature). Indeed, this is found to improve convergence over proportional scaling on at least this set of problems. Moreover, proportional scaling appears to increase, rather than decrease, effective tolerance during the course of an optimization.

5. Conclusion

This paper has illustrated the implementation of a procedure for genetic selection based on Boltzmann scaling of the optimization function and empirically demonstrated that it leads to convergence to the correct solution in fewer generations than traditional proportional scaling on a small set of problems. Furthermore, it was observed that proportional scaling, contrary to intuition and annealing methods, actually increases evolutionary tolerance during the experiment.

We thank Patrick Winston, Richard H. Lathrop, and the MIT AI Biology Reading Group for helpful discussions.

References

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In: *Proceedings of the International Conference on Genetic Algorithms*, pp. 101–111. Lawrence Erlbaum, Hillsdale.

Darwin, C. (1859). *The Origin of Species by Means of Natural Selection; or, The Preservation of Favored Races in the Struggle for Life.* London.

De Jong, K. A. (1975). *Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Ph.D. Thesis, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor.

Deb, K. and D. E. Goldberg (1989). An investigation of niche and species formation in genetic function optimization. In: *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 42–50. Morgan Kaufmann, San Mateo.

Goldberg, D. E. (1990). A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, *4*, 445–460.

Grefenstette, J. J. and J. E. Baker (1989). How genetic algorithms work: A critical look at implicit parallelism. In: *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 20–27. Morgan Kaufmann, San Mateo.

Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi (1983). Optimization by simulated annealing. *Science (Washington, D. C.)*, *220*, 671–680.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, *21*, 1087–1092.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (1988). *Numerical Recipes in C: The Art of Scientific Computing*, pp. 293–298. Cambridge University Press, Cambridge.

Verlet, L. (1967). Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Physical Review*, *159*, 98–103.

Whitley, D. (1987). Using reproductive evaluation to improve genetic search and heuristic discovery. In: *Proceedings of the Second International Conference on Genetic Algorithms*, pp. 108–115. Lawrence Erlbaum, Hillsdale.