Chapter 5

Michael de la Maza
Numinous Noetics Group
Room NE43-815
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square
Cambridge, MA 02139

mdlm@ai.mit.edu

The Boltzmann Selection Procedure[1]

Abstract

This chapter investigates Boltzmann selection, a tool for controlling the selective pressure 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

[1]This chapter is an elaboration of two previously published papers. de la Maza and Tidor (1992) describe Boltzmann selection and its application to a problem in molecular biology, while de la Maza and Tidor (1993) develop the theoretical analysis of Boltzmann selection.

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 empirical analysis is presented that compares the selective pressure of this algorithm to a standard selection procedure.

Then, in order to understand these results in a broader context, an analytical discussion of selection procedures used in genetic algorithms is presented. A unified framework for discussing and comparing procedures is developed and used to compare proportional, Boltzmann, power law, and sigma truncation selection procedures. Two properties, translation and scale invariance, are defined and studied for each of these procedures. Selective pressure is investigated for proportional and Boltzmann selection. It is proven that, for a normal distribution of individuals in the optimization space, proportional scaling decreases selective pressure during the course of an optimization run.

5.1 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 [16]. 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 [22], and Monte Carlo methods, which do not require derivative information [19]. 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" [6]. 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.

Genetic searches generally converge from a heterogeneous starting population of random individuals to a more homogeneous population in which the individuals are nearly identical. Once the population has converged to near homogeneity, a predominantly local search is performed. The selection scheme can be used to exert control over the rate of convergence of the population, as is shown in this work using a function with multiple optima.

There is a modest literature on methods of controlling the selective pressure in a genetic algorithm, and some systematic studies have been performed [3,24,1,17]. We feel that it is important [1] to develop methods in the genetic algorithm that allow specific control of the selective pressure, and [2] to study the best ways of varying the selective pressure during the course of a genetic algorithm run to obtain rapid convergence to an optimal solution.

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 chapter is as follows. Section 2 describes Boltzmann selection and compares it to proportional selection on two test problems. Section 3 explores the scale and translation invariance of several selection procedures and proves a theorem which might explain why Boltzmann selection outperforms proportional selection. Section 4 summarizes related work and Section 5 concludes the chapter. The purpose of Section 2 is to provide the reader with two examples of how Boltzmann selection can be used to optimize functions. Each step in the approach to both problems is described in detail so the reader can adapt these methods to other optimization problems. Section 3 then provides theoretical justification for the use of Boltzmann selection.

5.2 Empirical Analysis

5.2.1 Framework

This section describes a general framework for defining selection procedures. Here we use it to define the proportional selection procedure which will be empirically compared to Boltzmann selection on two test problems. Later, in Section 3 this definition will be used to prove properties about several selection procedures.

Definition 2.1 defines a selection procedure in terms of four functions. To determine the number of copies of each individual in one generation that will be propagated into the next, the four functions are composed as follows: W(P(F(U(X)))), where U is the problem dependent objective function, F is called

the fitness function, P usually produces a vector, $\vec{R}$, of probabilities multiplied by the number of individuals in the population, and W is usually the roulette wheel procedure [10] . The values produced by U are often called raw fitnesses. In some sense, this choice of functions is arbitrary; however, we feel that it captures our intuitions about the salient components of selection procedures. Notice that the fitness function used by the genetic algorithm, F, is not always the same as the function to be optimized, U. The key transformation, F(U(X)) produces the fitnesses from the objective function. Frequently this involves simply adding a constant so the fitness function is positive. Defining proportional selection, which is one of the most widely used selection procedures [15,10], will give an intuitive understanding of the definition.

Definition 2.1 *We define a selection procedure to be a quadruple (W, P, F, U) where*

$$W: \vec{R} \to \vec{I}$$

$$P: \vec{R} \to \vec{R}$$

$$F: \vec{R} \to \vec{R}$$

$$U: X \to \vec{R}$$

and where X is a population, R is the set of reals, and I is the set of integers.

Definition 2.2 defines proportional selection. Note that the transformation from objective function to fitness is the identity function. The P used here is what gives proportional selection its name. This P is used in many selection procedures, including proportional selection, power law selection [9] , and sigma truncation [8] . These other procedures are all different, however, because they use different functions for F. While these selection procedures all proportionally scale the fitness function, traditional proportional selection is the only one that proportionally scales the objective function. In Section 3, we will briefly consider two other selection procedures, rank selection [3] and tournament selection [4], that are substantially different.

Definition 2.2 *The proportional selection procedure is:*

$U(X)$ *is the problem dependent objective function*

$$F(\vec{R}) = \vec{R}$$

$$P_i(\vec{R}) = \frac{\vec{R}_i}{\left\langle \vec{R} \right\rangle}$$

$W(\vec{R})$ *is the roulette wheel procedure* [12]

where the notation P_i is used to indicate the ith element of P and $\left\langle \vec{R} \right\rangle$ is a scalar equal to the average of all of the elements in $\vec{R}$ in the current generation.

This high-level description leaves out some implementation details which are specified in the Lisp code listing in Figure 5.1.

5.3 Introduction to Boltzmann Selection

Having described proportional selection in terms of our definition, we now turn to an explanation of Boltzmann selection.

In an equilibrated simulated annealing ensemble, the probability of visiting a point in optimization space, *Xj,* is,

$$P(X_j) = \frac{e^{-u(X_j)/T}}{\sum_i e^{-u(X_i)/T}}$$

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. Definition 2.3 specifies Boltzmann selection (not to be confused with Boltzmann tournament selection, which is defined in [11]). The T parameter in the definition is a variable that corresponds to evolutionary tolerance (analogous to temperature in simulated annealing) and the plus sign is changed to minus when minimization, rather than maximization, is desired.

```
(defun  Boltzmann-selection  (individual-list  temperature)
       (let,
              ((beta  (/  temperature))
              (get_exp  (mapcar  #(lambda  (anindiv)
                     (list
                     (exp  (*  beta  (first   anindiv)))
                     (second  anindiv)))
                     individual-list))
              (total  (reduce  #'+  (mapcar  #'first  get_exp))
              (boltzmann_fitness
                     (mapcar  #'(lambda  (anindiv)
                            (list
                            (/  (first  anindiv)  total)
                            (second  anindiv)))
                     get_exp)))
              (roulette-wheel  boltzmann_fitness)))
```

Figure 5.1: Implementation of Boltzmann selection in Lisp. This procedure implements the fitness function, F, of Boltzmann selection. The two inputs to the Boltzmann-selection procedure are a list of individuals and the temperature. Each element of the list of individuals is itself a list with two components. The first component is $U_i(X)$ and the second is the genotype of the individual (X_j). The procedure ends by calling the `roulette-wheel` procedure which selects the individuals that will be propagated into the next generation. This roulette wheel function should take as input a list each of whose elements is a list of two components. The first component is $P_j(X)$ and the second component is the genotype of the individual (X_j). The function of this roulette wheel procedure is identical to the one used in proportional selection and the same code could be used for both. The Lisp code given here has been tested on a Sun SPARCstation running Lucid Common Lisp (version 4.1) and Austin Kyoto Common Lisp (version 1.530).

Definition 2.3 *The Boltzrnann selection procedure is:*

$$F_i(U(X)) = e^{U_i(X)/T}$$

and U, P, and W are defined as in proportional selection.

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. Multiplying the optimization surface by a constant, so that U'(X) = cU(X), which corresponds to changing the units in which U(X) is measured, is also invariant so long as the parameter, T, which has the same units as U(X), is similarly scaled. Moreover, there is no requirement that the optimization function be non-negative, as there is with proportional selection, since the exponential provides an appropriate transformation. The proof of the translational invariance of Boltzmann selection will be given in Section 3.

5.3.1 Experiments with Boltzmann Selection

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).

5.3.1.1 Description of Model Problems

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 5.1. 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.

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

Table 5.1: Typical database. Each instance has sixteen letters. The instances in the positive class all contain the sequence "RIEY".

The optimization function, *U(X),* 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 *Xj* is calculated using,

$$U_j(X) = \frac{1}{|P|}\sum_{I_k \in P} \max(match(X_j, I_k)) - \frac{1}{|N|}\sum_{I_k \in N} \max(match(X_j, I_k))$$

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 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 a traditional l-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

selection) was performed, creating generation i + 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 + 1/2 was crossed over with any of the individuals in that generation (including itself, 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 is approximately 0.7%, which is similar to that of other genetic algorithms.

F2 Function. Deb and Goldberg's F2 function [9] is:

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

On the interval [0.0, 1.0], F2 has five peaks, each one smaller than the previous one (see Figures 5.4, 5.5, and 5.6).

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 + 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 + 1/2 was crossed over with any of the individuals in that generation (including himself, producing the identity transformation) with equal probability.

5.3.1.2 Finding the Initial Tolerance

The appropriate initial tolerance value was determined by performing a series of experiments. The tolerance schedule is shown in Figure 5.2. 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 5.3. Each experiment was repeated eight times; the numbers shown are averages.

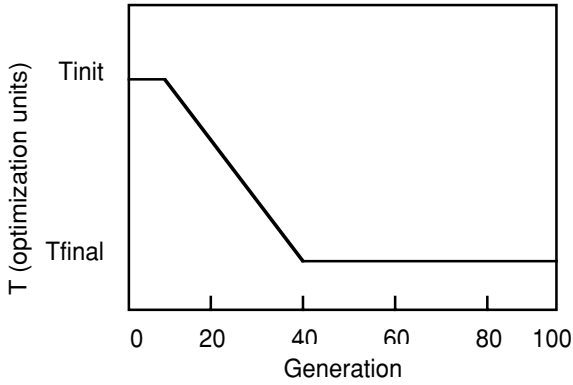


Figure 5.2: The tolerance schedule used in the Boltzman selection genetic algorithm. The tolerance was set to T_{init} for the first ten generations, ramped down to T_{final} over thirty generations, and maintained at T_{final} until completion. In all runs, T_{final} = 0.5 optimization units.

The U-shaped curve in Figure 5.3 is in accordance with our intuition about how the initial tolerance should affect search behavior. If the initial tolerance is too high, then the genetic algorithm spends too much time performing a random search and requires a long time to focus on the few good solutions. If the initial tolerance is too low, then the genetic algorithm performs a local search around the individual with the highest fitness in the initial population and, therefore, risks never finding the solution.

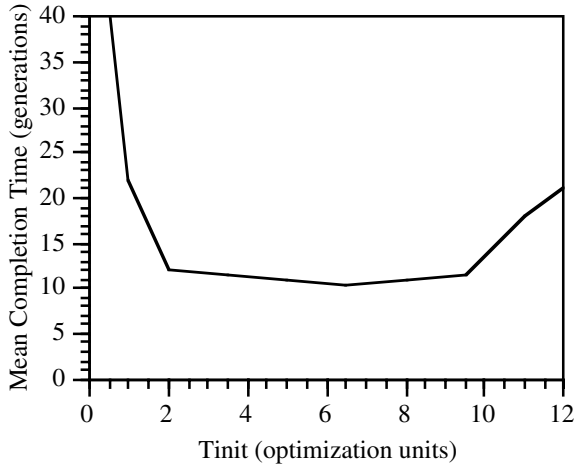


Figure 5.3: The average number of generations to completion of the Boltzmann scaling genetic algorithm for the problem of pattern length four. Experiments that required more than fifty generations to complete were stopped at fifty generations and combined to compute the average as if they had completed in fifty generations.

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 section. The observation that other optimization surfaces were searched reasonably quickly with the same schedule suggests that the method is robust with respect to small changes in the tolerance schedule.

5.3.2 Comparison

This section compares Boltzmann scaling and proportional scaling on a small set of molecular biology problems and Deb and Goldberg's F2 function [7] .

5.3.2.1 Molecular Biology Problem

The results of comparing the Boltzmann and proportional GAs are shown in Table 5.2. The first and second columns give the number of characters in the instances and in the patterns, the third shows the size of the population, the fourth indicates how many times each experiment was performed and the fifth and sixth 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 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.

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

Table 5.2: 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 significance test to the results.

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

Figure 5.4 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 5.2 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.

```
(.......%%%%%%%.................................)
(rvftsdtRIEYGKSDawvqekhmkwiqfyprfateshkyiiitgvscvp)Gen     Score
(...........cvSwiwwk ............................)0         1.60
(............Svswiwwk ..........................)1         1.60
(......swIwYGfg..................................)2-3       5.80
(.....gswIwYGf...................................)4         7.60
(.....gnwIwYGf...................................)5         8.80
(.........wYGKSswi...............................)6        13.80
(........IwYGKSsw................................)7-13     24.20
(........IwYGKSs*................................)14       25.60
(......s*IwYGKS..................................)15-18    27.20
(.......*IwYGKS*.................................)19-22    30.00
(.......*IhYGKS*.................................)23-33    31.40
(.......*IEYGKS*.................................)34-42    43.60
(.......RIEYGKS*.................................)43       52.00
(......*RIEYGKS..................................)44-48    52.00
(.......RIEYGKS*.................................)49-53    52.00
(.......RIEYGKSD.................................)54       63.40
```

Figure 5.4: Simple 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.

5.3.2.2 F2 Function

Two experiments were performed using the F2 function [7] 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 5.5 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 5.6 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.

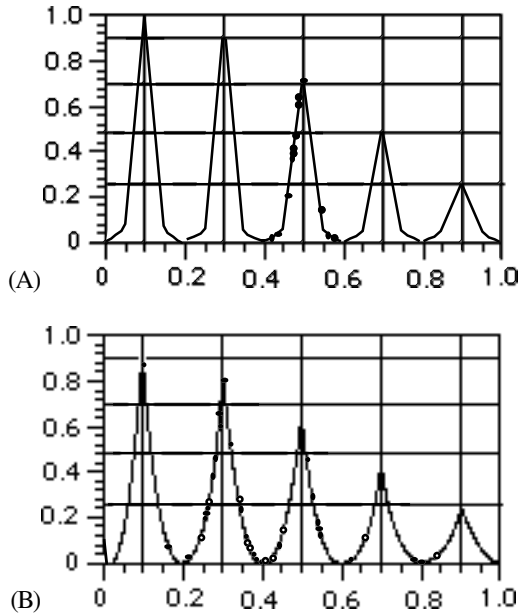
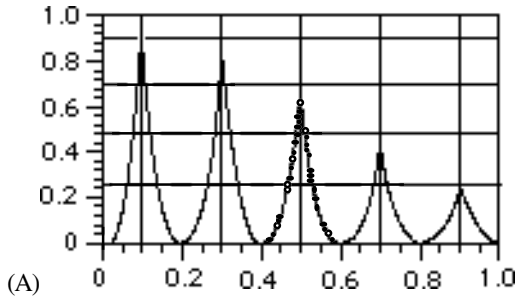
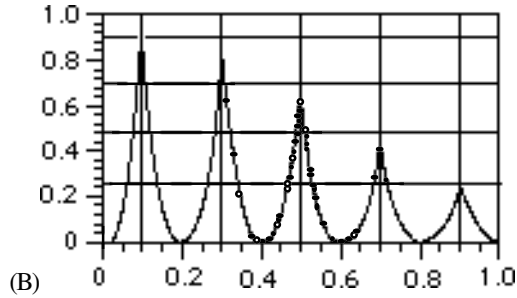


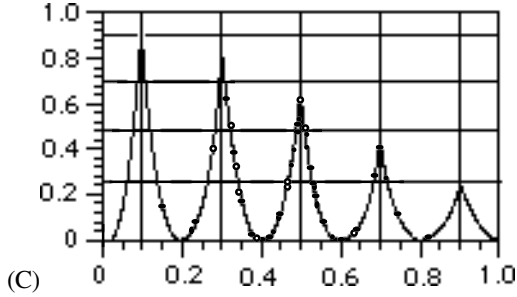
Figure 5.5: The proportional and Boltzmann GA populations at generation 50. (A) proportional GA population, (B) Boltzmann GA population. The initial population was randomly distributed between 0.400 and 0.600. The individuals are represented by small circles and the F2 function is the dark, continuous 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.



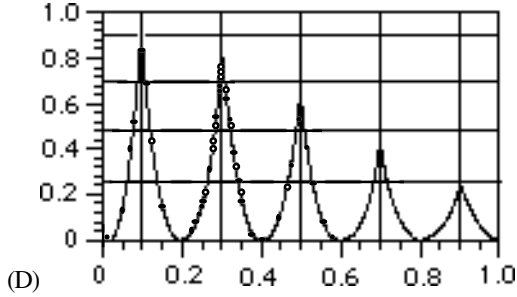
(A)



(B)



(C)



(D)

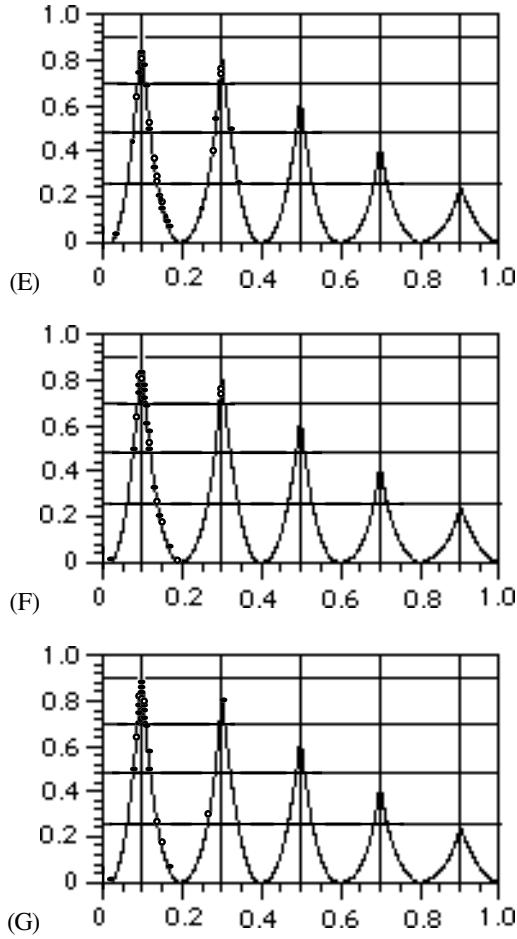


Figure 5.6: Boltzmann GA population time series. The initial population was randomly distributed between 0.400 and 0.600. The individuals are represented by small circles and the F2 function is the dark, continuous line. These graphs show the population immediately after the recombination operators have 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, (E) generation 70, (F) generation 80, (G) generation 90.

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 5.7 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.

5.3.3 Tolerance in the Proportional GA

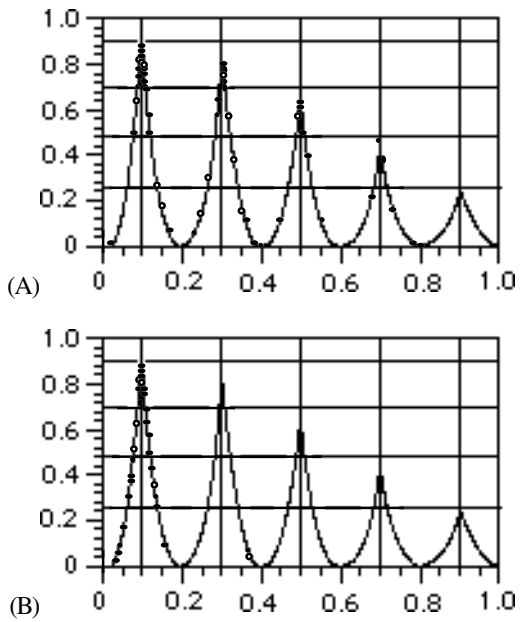
Given the formalism that has been presented to modify evolutionary tolerance, it is possible to study how the proportional GA sets an effective tolerance value at a given generation by choosing the tolerance that minimizes,

$$\sum_j \left[\frac{U_j(X)}{\sum_i U_i(X)} - \frac{e^{U_j(X)/T}}{\sum_i e^{U_i(X)/T}} \right]^2$$

where $U_j(X)$ 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.[21].

The results are shown in Figure 5.8. They indicate that in the proportional GA the effective tolerance increases, rather than decreases, as a function of the number of generations. This result, which runs contrary to both intuition and theory, strongly suggests that the traditional proportional scaling technique may need reconsideration.



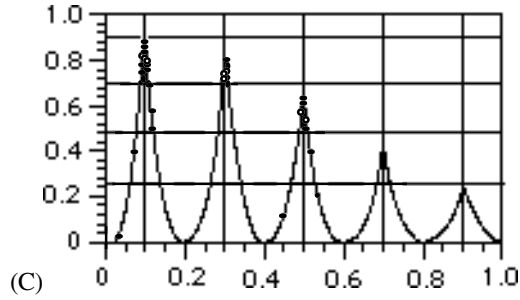


Figure 5.7: Random initial population. The initial population was randomly distributed between 0.000 and 0.999. The individuals are represented by small circles and the F2 function is the dark, continuous line. These graphs show the population immediately after the recombination operators have been applied 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).

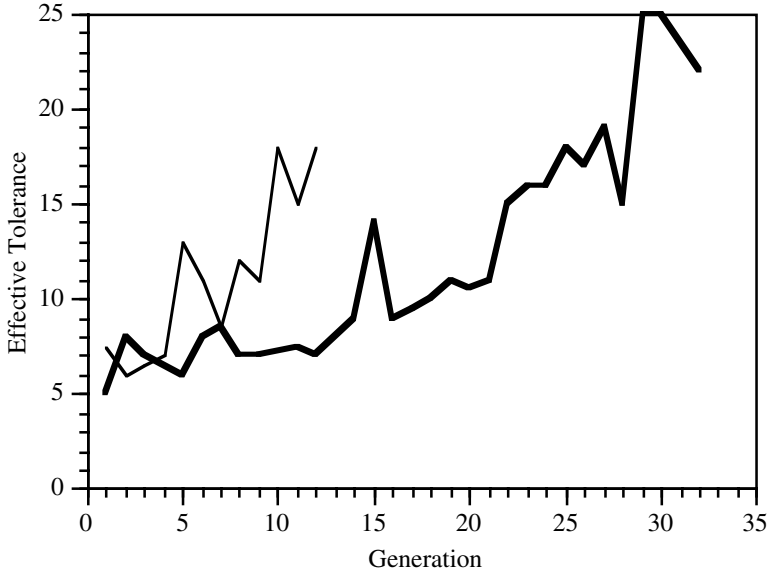


Figure 5.8: Effective tolerance in proportional GA. The dark line is for an experiment in which the Boltzmann GA outperformed the proportional GA; the light line is for an experiment with the opposite outcome. Both experiments are for patterns of length eight.

5.4 Theoretical Analysis

5.4.1 Definitions of Scale and Translation Invariance

Section 5.2 gives empirical evidence showing that a genetic algorithm with Boltzmann selection converges faster than an algorithm with proportional selection. Moreover, the parameter T in the above transformation is a variable parameter that can be used to control selective pressure during the course of a genetic algorithm run. At the end of this theoretical analysis section, we will give an analytical result that may account for this difference in convergence.

The primary properties of selection procedures that we will explore are scale and translation invariance. As we discuss below, these invariances are important properties not only because we feel intuitively that an optimal algorithm should follow the same search path on a simple transformation of a problem as on the original problem, but also because selective pressure should be carefully chosen by the user and the algorithm and not imposed by the objective function.

Definition 3.1 *A selection procedure is scale invariant exactly when:*

$P(F(U(X))) = P(F(kU(X)))$

where $k > 0$ and X is an arbitrary population.

Intuitively, a selection procedure is scale invariant if multiplying the objective function by a constant does not change the values produced by P.

Definition 3.2 *A selection procedure is translation invariant exactly when:*

$$P(F(U(X))) = P(F(U(X) \oplus \vec{C}))$$

where C is a vector of identical constant elements Ci, $\oplus$ is vector addition, and X is an arbitrary population.

Intuitively, a selection procedure is translation invariant if adding a constant to the objective function does not change the values produced by P.

In the next section we show that the Boltzmann selection procedure is translation invariant, but not scale invariant.

5.4.2 Scale and Translation Invariance of some Selection Procedures

In this section we explore the scale and translation invariance of four selection procedures. Because proportional selection is the best known and most widely used selection procedure, we carefully examine its lack of translational invariance.

5.4.2.1 Proportional Selection

Proportional selection is the most widely used selection procedure. We show that the proportional selection function is scale invariant, but not translation invariant, and we examine the nature of its lack of translational invariance.

Observation 3.3 *The proportional selection procedure is scale invariant.*

Proof:

$$P_i(F(kU(X))) = \frac{F_i(kU(X))}{\langle F(kU(X))\rangle}$$
$$= \frac{kU_i(X)}{\langle kU(X)\rangle} = \frac{U_i(X)}{\langle U(X)\rangle}$$
$$= \frac{F_i(U(X))}{\langle F(U(X))\rangle} = P_i(F(U(X)))$$

Observation 3.4 *The proportional selection procedure is not translation invariant*

Proof:

$$P_i(F(U(X))) = P_i(F(U(X) \oplus \vec{C})) \Leftrightarrow$$
$$\frac{F_i(U(X))}{\langle F(U(X))\rangle} = \frac{F_i(U(X) + \vec{C}}{\left\langle F(U(X)) \oplus \vec{C}\right\rangle} \Leftrightarrow$$
$$\frac{\left\langle U(X) \oplus \vec{C}\right\rangle}{\langle U(X)\rangle} = \frac{U_i(X) + \vec{C}_i}{U_i(X)} \Leftrightarrow$$
$$1 + \frac{\left\langle \vec{C}\right\rangle}{\langle U(X)\rangle} = 1 + \frac{\vec{C}_i}{U_i(X)} \Leftrightarrow$$
$$\frac{\left\langle \vec{C}\right\rangle}{\langle U(X)\rangle} = \frac{\vec{C}_i}{U_i(X)} \Leftrightarrow$$
$$\langle U(X)\rangle = U_i(X)$$

In general, the last equation is false.

The same result has been shown by Grefenstette and Baker [2] . In some sense, the proportional selection function is trivially not translation invariant because a negative constant Ui can be chosen such that $(U_i(X) + \vec{C}_i) < 0$. But this proof shows that proportional selection is not translation invariant even when $\vec{C}_i > 0$.

We can further investigate the role of this constant by asking how it affects selective pressure. We prove that as the constant increases, selective pressure decreases. For this purpose we define selective pressure on an individual to be the fitness of the individual divided by the average fitness of the population [24] .

Theorem 3.5 *If* $F(\vec{R})$ *is bounded, then*

$$\lim_{C\to\infty} \frac{F_i(\vec{R})+c}{\left\langle F(\vec{R})+c \right\rangle} = 1$$

Proof.

$$\lim_{C\to\infty} \frac{F_i(\vec{R})+c}{\left\langle F(\vec{R})+c \right\rangle} = \lim_{C\to\infty} \frac{\frac{F_i(\vec{R})}{c}+1}{\frac{\left\langle F(\vec{R}) \right\rangle}{c}+1} = 1$$

Intuitively, the theorem says that as the constant c increases the differences among individuals are blurred and therefore there is less and less selective pressure. This result is of particular importance because adding a constant to an objective function is recommended in the genetic algorithm literature as a way to make negative fitness functions positive (see, e.g., [10,20]). This theorem says that the choice of this constant can greatly affect the selective pressure, and therefore the performance, of the genetic algorithm. Michalewicz makes a similar, albeit more informal, argument [20]. Thus, this seemingly cosmetic change in the fitness function, F, can have wide-ranging consequences.

Moreover, as a genetic algorithm progresses, individuals will become increasingly fit. Intuitively, this might act like adding a constant to the fitness function which decreases selective pressure [24] .

The following theorem makes the effect of the constant on selective pressure more precise.

Theorem 3.6 *If F is bounded and greater than zero and c > 0, then*

$$\frac{F^{\min}+c}{F^{\max}+c} > 1 - \frac{F^{\max}}{c}$$

where F^{min} *is the fitness assigned to the least fit individual and* F^{max} *is the fitness assigned to the most fit individual in a particular generation.*

Proof.

$$\frac{F^{\min}+c}{F^{\max}+c} > \frac{c-F^{\max}}{c} \Leftrightarrow$$
$$cF^{\min}+c^2 > c^2-(F^{\max})^2 \Leftrightarrow$$
$$cF^{\min} > -(F^{\max})^2$$

This last equation is clearly true.

If F^{min} and F^{max} are interpreted to be the lower and upper bound on F, then the theorem can be used to make general statements about an entire run. For example, assume that the constant c is an order of magnitude greater than the upper bound on F, then the least fit individual in the population will never be less than 1 - 1/10 = 0.9 as fit as the most fit individual in the population.

We now consider Boltzmann selection [17] , power law selection [9] , and sigma truncation selection [8].

5.4.2.2 Boltzmann Selection

Observation 3.7 *The Boltzmann selection procedure is not scale invariant.*

Proof:

$$P(F(U(X))) = P(F(kU(X))) \Leftrightarrow$$

$$\frac{e^{kU_i(X)/T}}{\left\langle e^{kU(X)/T} \right\rangle} = \frac{e^{U_i(X)/T}}{\left\langle e^{U(X)/T} \right\rangle} \Leftrightarrow$$

$$e^{(k-1)U_i(X)/T} = \frac{\left\langle e^{kU(X)/T} \right\rangle}{\left\langle e^{U(X)/T} \right\rangle}$$

This last equation is clearly false.

Although Boltzmann selection is not scale invariant, any changes in scale can be offset by multiplying the temperature parameter by the scaling constant, k.

Observation 3.8 *The Boltzmann selection procedure is translation invariant.*

Proof.

$$P_i(F(U(X) \oplus \vec{C})) = \frac{e^{(U_i(X)+\vec{C}_i)/T}}{\left\langle e^{(U(X)\oplus\vec{C})/T} \right\rangle}$$

$$= \frac{e^{\vec{C}_i/T} e^{U_i(X)/T}}{\left\langle e^{\vec{C}/T} e^{U(X)/T} \right\rangle} = \frac{e^{U_i(X)/T}}{\left\langle e^{U(X)/T} \right\rangle}$$

$$= P_i(F(U(X)))$$

Thus, the Boltzmann selection procedure is translation invariant but not scale invariant.

5.4.2.3 Power Law Selection

Definition 3.9 *The power law selection procedure is*

$$F_i(U(X)) = U_i(X)^b$$

where b is a constant and U, P, and W are defined as in proportional selection.

Observation 3.10 *The power law selection procedure is scale invariant.*

Proof:

$$P_i(F(kU(X))) = \frac{F_i(kU(X))}{\langle F(kU(X))\rangle}$$

$$= \frac{(kU_i(X))^b}{\langle (kU(X))^b \rangle} = \frac{k^b (U_i(X))^b}{k^b \langle (U(X))^b \rangle}$$

$$= \frac{F_i(U(X))}{\langle F(U(X))\rangle} = P_i(F(U(X)))$$

Observation 3.11 *The power law selection procedure is not translation invariant.*

Proof:

$$P_i(F(kU(X))) = P_i(F(U(X) \oplus \vec{C})) \Leftrightarrow$$

$$\frac{F_i(U(X) \oplus \vec{C})}{\left\langle F(U(X) \oplus \vec{C}) \right\rangle} = \frac{F_i(U(X))}{\langle F(U(X))\rangle} \Leftrightarrow$$

$$\frac{(U_i(X) + \vec{C}_i)^b}{\left\langle (U(X) \oplus \vec{C})^b \right\rangle} = \frac{(U_i(X))^b}{\langle (U(X))^b \rangle} \Leftrightarrow$$

$$\frac{\langle (U(X))^b \rangle}{\left\langle (U(X) \oplus \vec{C})^b \right\rangle} = \frac{(U_i(X))^b}{(U_i(X) + \vec{C})^b}$$

The last equation is false.

5.4.2.4 Sigma Truncation Selection

Definition 3.12 *The sigma truncation selection procedure is:*

$$F_i(X) = g(U_i(X) - (< U(X) > - c\sigma))$$

where

$$g(x) = \begin{cases} x \text{ if } x > 0 \\ 0 \text{ otherwise} \end{cases}$$

σ *is the standard deviation of U(X) in a particular generation, c is a small constant, and U, P, and W are defined as in proportional selection.*

Observation 3.13 *The sigma truncation selection procedure is translation invariant.*

Proof: Note that the standard deviation of *U(X)* is equal to that of *(U(X)* $\oplus\vec{C}$*).*

$$
\begin{aligned}
& P_i(F(U(X)\oplus\vec{C})) \\
&= \frac{g((U_i(X)+\vec{C}_i)-(<U(X)\oplus\vec{C}>-c\sigma))}{\left\langle g((U(X)+\vec{C})-(<U(X)\oplus\vec{C}>-c\sigma))\right\rangle} \\
&= \frac{g(U_i(X)-(<U(X)>-c\sigma))}{\left\langle g(U(X)-(<U(X)>-c\sigma))\right\rangle} \\
&= P_i(F(U(X)))
\end{aligned}
$$

The proof of the translation invariance of sigma truncation is very different from the proof of the translation invariance of the Boltzmann fitness function. In the Boltzmann proof, the structure of P is exploited to cancel the constant. In the sigma truncation proof, the structure of P is irrelevant because $F_i(U(X)) = F_i(U(X)\oplus\vec{C})$. We formalize this notion in the following definition.

Definition 3.14 *A selection procedure is strongly translation invariant exactly when:*

$$F_i(U(X)) = F_i(U(X)\oplus\vec{C})$$

Intuitively, a selection procedure is strongly translation invariant when the invariance is independent of the particular choice of P. It is easy to show that strong translation invariance implies translation invariance with respect to P. Strong scale invariance can be analogously defined.

5.4.3 Rank Selection and Tournament Selection

Rank selection [3] is both scale and translation invariant because the relative position of an individual in a sorted list of raw fitness is not affected by a translation in the raw fitness or a scaling change in the raw fitness. Similarly, the outcome of a head to head competition between the raw fitnesses of two individuals is not changed by a scaling change or a translational change, so tournament selection [4] is both scale and translation invariant.

Since tournament selection and rank selection are interested only in qualitative comparisons of fitness, rather than quantitative numerical values, they are insensitive to many other transformations in the fitness function. For example, cubing the objective function will not change tournament and rank selection.

5.4.4 Understanding the Relationship between Proportional and Boltzmann Selection

The relationships among selection procedures have been studied empirically by many researchers (e.g., [13,1,17] and others). Typically, two identical genetic algorithms, differing only in the type of selection procedure they employ, are

tested on a variety of test functions. Here, we are interested in analytically exploring the relationship between proportional and Boltzmann selection.

We propose a general technique for analytically comparing two selection procedures. All of the selection procedures that we have studied, with the exception of proportional fitness, have problem dependent parameters. The relationship between two selection procedures can be studied by explaining how to set these parameters so that one selection procedure acts like the other. In this particular case, the Boltzmann selection procedure has an extra parameter, T. We are interested in understanding how to set the parameter T so that Boltzmann selection is most like proportional selection. By "most like" we mean the setting of T such that the difference between W(P(F(U(X)))) and W'(P'(F'(U'(X)))) is minimized, where (W, P, F, U) and (W', P', F', U') define the two selection procedures. The following theorem explains how to set the parameter T in order to achieve this goal:

Theorem 3.15 *If U(X) has a normal distribution at a particular generation, then Boltzmann selection is most like proportional selection when*

$$\beta e^{\sigma^2\beta^2} = \frac{1}{\mu}$$

where $\beta = 1/T$, p is $< U(X) >$, and σ^2 is the variance of U(X).

The proof of this theorem can be obtained from the author whose address is supplied at the beginning of this chapter. Notice that since $\beta e^{\sigma^2\beta^2}$ is a strictly increasing function of β, the exact value of β can be found by using a simple binary search strategy (Cormen, Leiserson et al. 1990) . If μ increases as a function of time (there is considerable empirical evidence that it does; see, e.g., (Goldberg 1989)) and σ^2 does not decrease rapidly as a function of time, then β will decrease as a function of time. This corresponds to an increase in the temperature parameter, T. If T is interpreted as controlling selective pressure (when T is high, selective pressure is low and vice versa), then the proportional selection procedure decreases selective pressure with time. In the field of simulated annealing, T decreases with time and therefore selective pressure increases with time. This may partially explain why Boltzmann selection out performs proportional selection on some problems (de la Maza and Tidor 1992).

5.5 Discussion and Related Work

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.

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 outweigh the computational overhead.

A refinement of our method that we have considered is to eliminate all duplicates in the population before applying 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 be, 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 [23] 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. 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 is valuable.

Goldberg [11] 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.

Back and Hoffmeister [1] study the performance of a genetic algorithm as a function of selective pressure on the less fit individuals in the population (referred to as "extinctiveness"). For a unimodal objective function, they find optimum performance with strong selective pressure, which produces relatively little genetic diversity and a gradient-directed search. In contrast, for a multimodal

objective function, they find optimum performance with weaker selective pressure, which produces more genetic diversity and exploration of the search space.

The parameters of their genetic algorithm were not modified during the course of a run and they comment that, without knowing the character of the objective function, it is difficult to choose the proper search strategy. One approach to solving this problem is to use a hybrid strategy that is initially more explorative and then becomes more directed as the run proceeds, as is generally done in simulated annealing [17].

Whitley [24] gives an informal argument that explains why proportional selection decreases selective pressure with time and proposes the use of rank-based selection, because it does not rely on the relative arbitrariness of the objective function to define the selective pressure. We have illustrated the use of a transformation from objective function to fitness and identified invariance properties of this key transformation for a number of selection procedures. We have pointed out that Boltzmann selection is invariant to translations and that a simple parameter, T, can be used to control scaling. Moreover, this parameter can be used to vary selective pressure during a run to switch from a more explorative to a more directed search strategy. Theorem 3.15 proves that proportional selection decreases selective pressure with time. In simulated annealing and in Boltzmann selection, selective pressure increases with time.

Baker [2] studies selection algorithms with the goal of overcoming the premature convergence problem. Grefenstette and Baker [14] examine how selection procedures interact with implicit parallelism and argue that the usual application of the k-armed bandit problem to genetic algorithms may be flawed, a view which is disputed by Goldberg and Deb [12]. Grefenstette [13] extends the results of Baker and Grefenstette and strongly argues that because the genetic algorithm has access to a biased sample, instead of an unbiased sample, of points, the standard understanding of implicit parallelism needs to be reexamined. Goldberg and Deb [12] hint that proportional selection may not maintain selective pressure as the point of convergence nears, a suggestion that is supported by Theorem 3.15.

5.6 Conclusion

This chapter 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 proved that proportional scaling, contrary to intuition and annealing methods, actually increases evolutionary tolerance during the experiment.

Translation and scale invariance are powerful properties to examine for selection procedures. Intuitively, it may be desirable for an optimization procedure to solve a problem equally well whether it is expressed in feet or meters and in the Gregorian or the Chinese calendar. If the procedure itself is not translation and scale invariant, parameters could be available that can be adjusted for each

problem. Presumably, setting these parameters properly will result in similar solutions in similar times for different translations and scalings of the same problem.

References

[1] Back, T. and F. Hoffmeister (1991). Extended selection mechanisms in genetic algorithms. *Fourth International Conference on Genetic Algorithms*. 92-99.

[2] Baker, J.E. (1989). *An Analysis of the Effects of Selection in Genetic Algorithms*. Vanderbilt University, Nashville. Ph.D., Thesis.

[3] Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. *International Conference on Genetic Algorithms and Their Applications*. 101-111.

[4] Brindle, A. (1981). Genetic algorithms for function optimization. University of Alberta, Canada. Technical, 81-2.

[5] Cormen, T. H., C. E. Leiserson, et al. (1990). *Introduction to Algorithms*. Cambridge, MA, MIT Press.

[6] Darwin, C., Ed. (1951). *The Origin of Species by Means of Natural Selection; or, The Preservation of Favoured Races in the Struggle for Life*. London, Oxford University Press.

[7] Deb, K. and D. E. Goldberg (1989). An investigation of niche and species formation in genetic function optimization. *Third International Conference on Genetic Algorithms*. 42-50.

[8] Forrest, S. (1985). Documentation for PRISONERS DILEMMA and NORMS programs that use the genetic algorithm. Unpublished manuscript.

[9] Gillies, A. M. (1985). *Machine learning procedures for generating image domain feature detectors*. University of Michigan, Ann Arbor.

[10] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA, Addison-Wesley.

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

[12] Goldberg, D. and K. Deb (1991). "A comparative analysis of selection schemes used in genetic algorithms." *Foundations of Genetic Algorithms* : 69-93.

[13] Grefenstette, J. (1991). "Conditions for implicit parallelism." *Foundations of Genetic Algorithms*. 252-261.

[14] Grefenstette, J. and J. Baker (1989). How genetic algorithms work: A critical look at implicit parallelism. *Third International Conference on Genetic Algorithms*. 20-27.

[15] Holland, J. D. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, University of Michigan Press.

[16] Kirkpatrick, S., J. C.D. Gelatt, et al. (1983). "Optimization by simulated annealing." *Science* **220**: 671-680.

[17] de la Maza, M. and B. Tidor (1992). Increased flexibility in genetic algorithms: The use of variable Boltzmann selective pressure to control propagation. *Research: New ORCA CSTS Conference: Computer Science and Operations Developments in Their Interfaces*. 425-440.

[18] de la Maza, M. and B. Tidor (1993). An analysis of selection procedures with particular attention paid to proportional and Boltzmann selection. *Fifth International Conference on Genetic Algorithms*. San Mateo, Morgan Kaufmann.

[19] Metropolis, N., A. W. Rosenbluth, et al. (1953). "Equation of state calculations by fast computing machines." *Journal of Chemical Physics.* **21**: 1087-1092.

[20] Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Springer-Verlag.

[21] Press, W. H., S. A. Teukolsky, et al. (1992). *Numerical Recipes in C: The Art of Scientific Computation*. Cambridge University Press.

[22] Verlet, L. (1987). "Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules." *Physical Review* **159**: 98-103.

[23] Whitley, D. (1987). Using reproductive evaluation to improve genetic search and heuristic discovery. *Second International Conference on Genetic Algorithms*. 108-115.

[24] Whitley, D. (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. *Third International Conference on Genetic Algorithms*. 116-121.