Tournament Selection, Niching, and the Preservation of Diversity

Christopher K. Oei
David E. Goldberg,
Shau-Jin Chang
University of Illinois at Urbana-Champaign
Urbana, IL 61801

IlliGAL Report No. 91011
December 1991

Illinois Genetic Algorithms Laboratory
Department of General Engineering
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, Illinois 61801

Tournament Selection, Niching, and the Preservation of Diversity

Christopher K. Oei, David E. Goldberg, & Shau-Jin Chang
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

This article examines the possibility of combining tournament selection with the method of sharing in an attempt to create another selection scheme that can stably maintain multiple niches in genetic algorithms (GAs). Initial efforts have confirmed the previous observation that the naive combination of these two techniques fails because the rescaling of sharing and the autoscaling of tournament selection fight one another. This paper demonstrates empirically and analytically that the dynamics of the naive combination are chaotic and result in a rapid loss of the number of niches that can be supported over time. The paper also shows that when tournament selection is used with a modified form of sharing that is continuously updated in the target population, the dynamics become stable and the resulting selection scheme is able to promote and maintain multiple subpopulations over many generations, virtually without loss. Further investigation is necessary, but this new scheme, *tournament selection with continuously updated sharing*, is ready for trial in those GAs where reliable niching is a must.

1 Introduction

Among selection schemes commonly used in genetic algorithms (GAs), tournament selection has a number of advantages (Goldberg & Deb, 1991): (1) it often involves only local pairwise or k-way interactions between individuals, (2) it naturally and inexpensively implements a means of obtaining rank-based selection (Baker, 1985), and (3) it can be performed so that the noise of selection is minimized and can be all but forgotten in the sizing of populations (Goldberg, Deb, & Clark, 1991). On the other hand, in its simplest form, tournament selection implements a simple or purely competitive selection scheme, relentlessly driving the population toward the best. In problems with solution sets of cardinality greater than one, in multimodal problems, and in classifier systems and other genetics-based machine learning systems, it is desirable to adopt selection schemes that preserve multiple solutions, thereby promoting a representative covering of solution structures. Among the most widely used of such niching schemes is the *method of sharing functions* (Deb, 1989; Deb & Goldberg, 1989; Goldberg & Richardson, 1987), but it has been observed (K. Deb, personal communication, 1989) that when sharing is used naively with tournament selection, the autoscaling of the tournament and the imposed scaling of the sharing mechanism fight one another. The lack of success with the naive combination of sharing and tournament selection has led to the development of so-called *Boltzmann tournament selection* (BMTS) (Goldberg, 1990) that combines tournament selection with the Boltzmann distribution widely used in simulated annealing, but an appendix to this paper casts doubt upon BMTS's ability to stably maintain large number of niches at high temperatures, and the development of an entirely reliable, tournament-based scheme of niched selection has remained an open research direction.

In this paper, we concentrate on (1) understanding why the naive combination of tournament selection and sharing doesn't work and (2) devising a modified tournament-sharing scheme that permits stable niching. Specifically, we find that the naive combination of sharing and tournament selection has chaotic dynamics and that this leads to a rapid decline in the number of niches that a population of given size

can support. We find when we perform sharing continuously over the target rather than the present population that this difficulty is corrected. In other words, *tournament selection with continuously updated sharing* promotes stable niching in finite populations. Further experimentation in practical problems is required, but the technique is ready for trial in problems where stable niching is a must. We suggest a number of ways to generalize and improve the technique, including the use of sampled sharing and the use of a niching threshold.

In the remainder, we first examine the chaotic dynamics of the naive combination of sharing and binary tournament selection and show how these lead to a rapid decline in the number of niches that can be sustained over time. We then show how the use of continuously updated sharing overcomes this difficulty and promotes a stable scheme. Extensions to the method are then discussed. A brief appendix contains an analysis of unimpeded genetic drift and explores its ramifications for unmodified Boltzmann tournament selection.

2 The Naive Combination of Tournaments and Sharing

In this section, we assume that binary tournament selection operates on a shared fitness value calculated using an appropriate sharing function over the current population as described elsewhere (Deb, 1989; Deb & Goldberg, 1989; Goldberg & Richardson, 1987). We consider the dynamics of the two-niche case first and then consider the mean-field dynamics in the multiple-niche setting. The results show that the naive combination is unsuitable for practical applications, because the detailed dynamics are chaotic and because the mean-field performance exhibits a steady decline in the number of niches the population can support.

2.1 Chaotic dynamics of the naive combination

Intuitively, we might expect to have difficulty when we combine tournament selection with sharing. Tournament selection by itself imposes a form of ranking on the members of a population, rewarding even the smallest differences in fitness with greater numbers in the target population. On the other hand, sharing is trying to adjust the number of individuals in the target population in proportion to their absolute fitness level. In theory, if sharing worked perfectly, the population should stabilize at niche sizes that make the shared fitness equal among the niches, but sharing does not work perfectly, and the accentuation of even the smallest—and largely insignificant—differences by the action of the tournament should cause the population to careem about.

To put this in more mathematical terms, consider the following two-niche model, concretely imagined using a population of single-bit strings, each with equal fitness. Sharing makes the subpopulation with the least number of members the most fit, after the sharing is taken into account. Let n be the number of 1's in the population, which is of size N. Then the time evolution of the system is given by the following:

$$n_+ = \begin{cases} \frac{n^2}{N} & \text{for } n > N/2; \\ \frac{n^2}{N} + 2\frac{(N-n)n}{N} & \text{for } n < N/2. \end{cases} \tag{1}$$

where n_+ is the number of 1's in the next generation. It is easier to see what is happening if we look at the ratio of the number of 1's to the total population: $r = n/N$. The evolution becomes

$$r_+ = \begin{cases} r^2 & \text{for } r > 1/2; \\ -r^2 + 2r & \text{for } r < 1/2. \end{cases} \tag{2}$$

If sharing works, then $r = 1/2$ should be a stable fixed point of this mapping. Notice, however, that the mapping is actually discontinuous at $r = 1/2$. If we start with almost the same number of 0's as 1's, with slightly more 1's, then in a single generation the ratio would jump to 3:1 in favor of the 0's. The method of sharing overcompensates and overshoots the equilibrium.

The system has two unstable fixed points of order 1 at 0 and 1, and two unstable fixed points of order 2 at $\frac{3-5^{1/2}}{2}$ and $\frac{5^{1/2}-1}{2}$. In fact, since the slope of the mapping is always greater than 1, there

are no stable fixed points anywhere and thus the trajectory of the system is always aperiodic. We have verified numerically that the system is in fact chaotic with a Lyapunov exponent of about 0.21.

If we think of the system in terms of control theory, what we are trying to do is to control a system that would, if left by itself, drift away from the desired equilibrium. In this case, our controlling force is too strong, and the one-generation time lag between the measurement of the system and the application of the control causes the controlled system to overshoot the equilibrium, causing feedback and chaos. We will use this point of view to correct the situation later in this paper.

2.2 Mean-field analysis of multiple niches

Actually, the fact that the system is chaotic does not really tell us that the method of sharing does not work in this context. In order to show that the naive combination of sharing and tournament selection fails, we must estimate the rate of loss of genetic diversity, and to do this, we will use a more involved model. Instead of thinking of two subpopulations, we will consider a system with many subpopulations and assume that what happens in one subpopulation does not directly affect what happens in another population. This model is similar to the mean-field theory that is often used to describe the behavior of physical systems. Let us assume we have a large population of size N with k_0 sub-populations at time $t = 0$. Again we will take all the fitness values to be equal.

Let $p(k, x, t)$ be the number of individuals at time t in the x-th niche, where the niches are sorted according to their sizes, and $k = k(t)$ is the number of niches remaining at time t. The largest niche will have $p(k, k, t)$ members and the smallest niche will have $p(k, 1, t)$ members. What we expect is that for large k, the system will quickly collapse onto a slow manifold that does not depend on the initial conditions except for the value of N and k_0. Let us work in the rescaled population distribution:

$$F(x/k, t) = \frac{k}{N} p(k, x, t). \tag{3}$$

What we are guessing is that as time progresses, the system loses more and more niches, but the rescaled population distribution remains unchanged. If such a quasi-steady-state solution exists, it is given by

$$F^*(y) = \lim_{t \to \infty} F(y, t). \tag{4}$$

So if our quasi-steady-state assumption holds, when we have a large number of niches ($k \gg 1$) and evolution has taken place for a while ($t \gg 1$), $p(k, x, t)$ follows a simple scaling law:

$$p(k, x, t) \approx \frac{N}{k} F^*(x/k). \tag{5}$$

We have verified this scaling law numerically. It turns out that it is possible to solve for this quasi-steady-state solution, F^* analytically. The key is to postulate the existence of a solution $F^*(y)$ that is symmetric about $y = 1/2$ after the tournament reproduction phase; that is, the most endangered niches remain highly endangered, the most populated niches become endangered, and the niches in between the extremes remain about the same. Numerical studies have pointed out this symmetry, and using these assumptions, we can write the time evolution of the system as follows.

Since all the fitness values have been assumed to be equal, the outcome of the tournaments depends only on how many individuals are in the niches. Let F_t be the rescaled population distribution after the tournament phase. Since the niches are ranked according to their sizes, we can take the expectation:

$$F_t(x, t) = 2 \int_x^1 F(x, t) F(y, t) dy. \tag{6}$$

After the tournament phase, some of the niches that were heavily populated are no longer heavily populated, and some of the niches that were endangered are no longer endangered. This means that we must sort the niches again according to their new sizes. Ordinarily, this would be extremely difficult to do analytically, but because the solution is symmetric by assumption, the sorting phase is given by

$$F_s(x, t) = F_t(x/2, t). \tag{7}$$

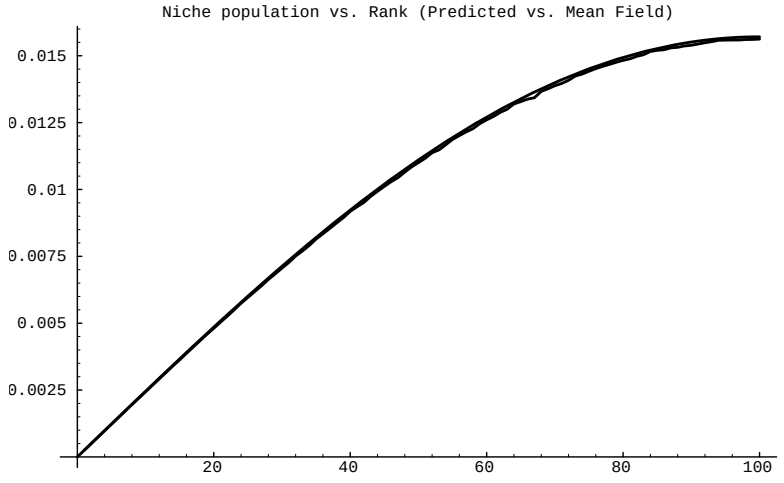

Figure 1: Continuum and mean field calculations agree when compared on the basis of distribution of niche proportion.

Now we just renormalize the probability distribution so that the integral is unity:

$$F(x,t+1) = CF_s(x,t). \tag{8}$$

We can write the time evolution of the system as an operator equation:

$$F(x,t+1) = \Psi[F(x,t)], \tag{9}$$

where the operator Ψ is given as follows:

$$\Psi[w(x)] = \Phi[w(x)] / \int_0^1 \Phi[w(x)]dx; \tag{10}$$

$$\Phi[w(x)] = w(x/2) \int_{x/2}^1 w(y)dy. \tag{11}$$

For the quasi-steady-state solution, we must solve

$$F^*(x) = \Psi[F^*(x)]. \tag{12}$$

These equations are straightforward to solve, yielding the quasi-steady-state solution:

$$F^*(x) = \frac{\pi}{2}\sin(\frac{\pi x}{2}). \tag{13}$$

It is straightforward to verify that after the tournament phase, this solution is symmetric about $x = 1/2$, and so we have a self-consistent solution. The fact that numerical experiments agree with this analysis indicates that this is indeed a stable solution. Figure 1 demonstrates the validity of the assumption that the behavior of a large but finite number of niches is similar to that of an infinite number of niches. Figure 2 demonstrates the validity of the assumption that a large population with many niches is similar to an infinite population with infinitely many niches. Now that we have the distribution of the population, it is straightforward, using the mean-field approximation, to compute the probability that a certain subpopulation will lose every tournament that it enters. This will tell us how the number of niches, k, depends on time. The rate of loss is

$$\frac{\partial k}{\partial t} = -\int_0^k \left[\int_0^{x/k} \frac{\pi}{2}\sin(\frac{\pi}{2}y)dy\right]^{\frac{\pi N}{2k}2\sin(\frac{\pi x}{2k})} dx. \tag{14}$$

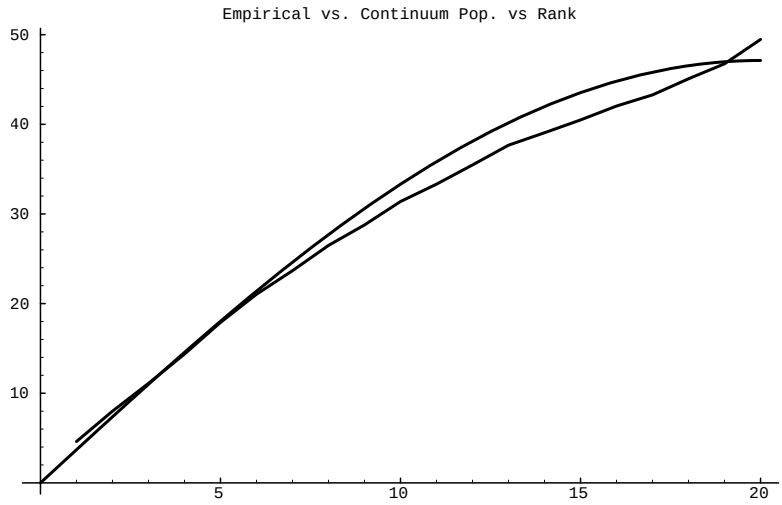

Figure 2: Analytical and empirical distributions of niche count are consistent.

The factor of two in the exponent comes because each individual enters twice in tournaments.

A little massaging gives

$$\frac{\partial k}{\partial t} = -k \int_0^1 \left[(1 - \cos(\frac{\pi x}{2})\right]^{\frac{\pi N}{k} \sin(\frac{\pi x}{2})} dx, \tag{15}$$

or

$$\frac{\partial k}{\partial t} = -kG(N/k), \tag{16}$$

where G is determined by the previous equation. It can be shown that only the values of the integrand near the endpoint at $x = 1$ matters in the integral when N/k is large. Doing the asymptotic expansion yields the following:

$$\frac{\partial k}{\partial t} = -\frac{2}{\pi^2} \frac{k^2}{N}. \tag{17}$$

Using the initial condition $k(0) = k_0$ we obtain

$$k = \frac{1}{\frac{1}{k_0} + \frac{2t}{\pi^2 N}}. \tag{18}$$

The following are two numerical studies to check this result. We used a fixed number of individuals per niche, varied the number of niches in the initial population, and asked how many niches survive fifty generations. The results are shown in figure 3. We also took a single run and asked how many niches survived as a function of time. These results are shown in figure 4.

A simple rule of thumb for computing the required population size for supporting k_0 niches is to set $\frac{\partial k}{\partial t} \ll 1$ at $t = 0$, which gives

$$N \gg k_0^2. \tag{19}$$

The number of niches we can support for a fixed number of generations scales as the square root of the population size. Thus, if we want to support twice as many niches, we need four times as many individuals in the population.

As a practical matter, the chaotic dynamics together with the rapid and relentless loss of niches is disconcerting and tolls the death knell for the naive combination of tournament selection and sharing. In the next section, we show that sharing and tournament selection can become more simpatico with a simple yet inobvious modification to the combined scheme.

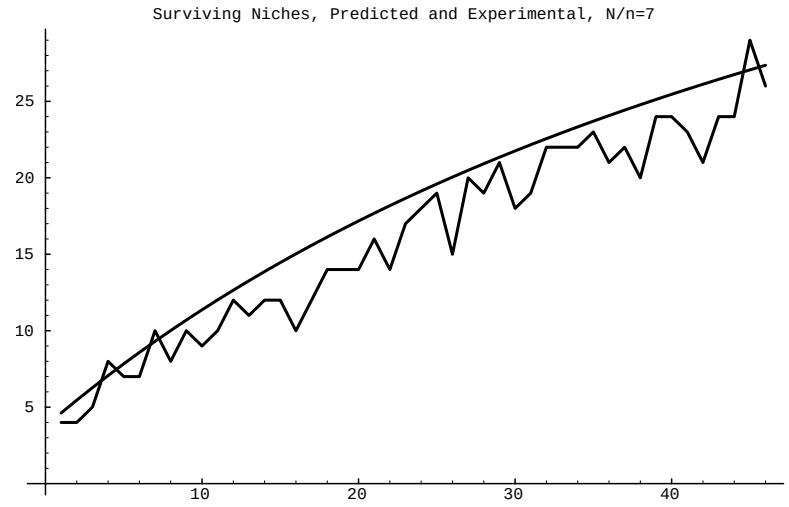

Figure 3: The number of niches that survive 50 generations versus the initial number of niches shows good agreement between mean-field prediction and experiment.

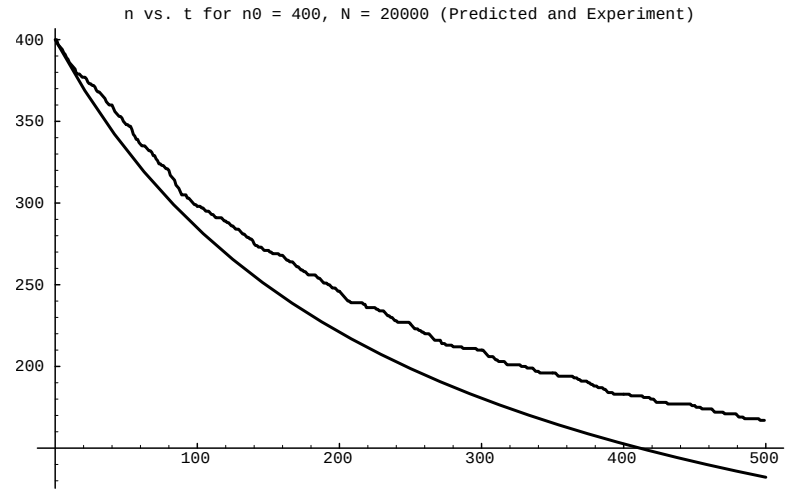

Figure 4: A comparison of experimental versus predicted mean-field loss in the number of niches shows generally good agreement. The total population size is $20,000$ and the initial number of niches is 400.

3 Tournament Selection with Continuously Updated Sharing

In the previous section, we began to understand how the scaling of sharing and the autoscaling of tournament selection seem to fight each other when used in naive combination, but we also saw how the difficulty seems largely to be one of timing. That is, by alternately applying tournaments to shared fitness values calculated over populations generated themselves by tournaments, the combined scheme is in an endless loop of accentuating the inevitable small differences that result from the imperfections of sharing. It is natural to wonder whether there is some way to avoid this rat race through a realignment of the timing. There is, and the answer is remarkably straightforward. Instead of applying tournament selection to a population that has undergone shared fitness calculations en masse in the extant population, we instead apply tournament selection according to shared fitness values that have been updated continuously using only the individuals actually chosen to be members of the target or next generation. In this way, continuous feedback is obtained regarding the actual state of the population, and this feedback is used immediately in the next shared fitness calculation that is used to determine whether one string or another gets chosen to enter the target population. To put this another way, in the naive combination, we tried to create the new generation by looking only at how many individuals existed in each niche in the old generation. What we will try to do now is to create the target generation by looking at how many individuals exist in the new generation as it is being created. That is, when two individuals compete, we look in the new generation and see which one has proportionately fewer than its target number of individuals, choosing that indivdual to win that particular tournament. In this section, we partially test and analyze this proposed scheme.

3.1 Testing and analysis

Our initial tests consider the application of tournament selection with continuously updated sharing using equal fitness values. We have found that with ten individuals per niche, we can support 500 niches for 50 generations with no losses, and there seems little reason why many more or many fewer niches can't be supported as long as the number of individuals allocated per niche is sufficiently high. Of course, even with this scheme niches can be lost, but the loss rate is much lower than in the naive case or in the unimpeded drift case examined in the appendix. To understand that losses can still occur, consider a scenario where among a number of healthy niches, there are three niches, A, B, and C, facing extinction, each with only one individual remaining. Let us also assume that the shuffling is so unlucky as to cause these niches to compete only with each other (this argument applies strictly to tournaments performed with replacement). Since all three niches are in identical predicaments, we consider only the case where A fights B in the first tournament and loses. In the second tournament, if A fights C and loses again, it is eliminated. Otherwise, it will have at least one copy in the next generation. Although the possibility of total loss is possible, the scenario suggests that total loss is rather unlikely.

We can analyze the quasi-steady-state of this algorithm in the following way. Let us look at the size of the niche in the new generation during reproduction. We will assume that there are already a substantial number of individuals in the target population. Under these conditions, the algorithm will have the same noise characteristics as the following algorithm: take two individuals at random in the population and replace the more redundant individual with another copy of a less redundant individual. Let $w(x)$ be the probability of finding a niche with $N/k - x$ individuals. Then the rate of production of niches with $N/k - x$ individuals is

$$w(x-1)\left[w(x-1) + w(x) + w(x+1) + w(x+2) + \ldots \right. \tag{20}$$

and the rate of destruction of niches with $N/k - x$ individuals is $w(x)$. The quasi-steady-state condition gives that the rate of production equals the rate of destruction:

$$w(x) = w(x-1)\left[w(x-1) + w(x) + w(x+1) + w(x+2) + \ldots \right. \tag{21}$$

We will assume that $w(x)$ decreases rapidly in x for large x so that we can approximate the sum by the first term:

$$w(x) \approx w(x-1)^2. \tag{22}$$

The general solution to this equation is

$$w(x) \approx C_1 e^{-C_2 2^x}, \tag{23}$$

where C_1 and C_2 are arbitrary constants. We see that $w(x)$ does indeed decreases rapidly in x for large x so our ansatz is self-consistent. From this argument, we conclude that large fluctuations from the equilibrium posititon of N/k individuals per niche are extremely unlikely; the probability of a fluctuation of size x drops as an exponential of an exponential. The rate of loss of niches is proportional to the probability of seeing a fluctuation of size N/k:

$$\frac{\partial n}{\partial t} \approx -C_1 k e^{-C_2 2^{N/k}}. \tag{24}$$

So we see that increasing the niche size by even a single individual represents a dramatic improvement in the ability of the algorithm to sustain niches. To insure that we maintain genetic diversity, we should set $\frac{\partial n}{\partial t} \ll 1$ and solve for N. This gives

$$N = Dk \log(E \log(Fk)). \tag{25}$$

where D, E, and F are constants. We will assume that for any reasonable value of k, the double logarithm is of order unity. This gives

$$N = Dk. \tag{26}$$

Using a population size of ten times the number of niches seems to yield no loss of genetic diversity for any reasonable value of k. Thus, the number of niches we can support goes up almost linearly in the population size.

There are several ways to implement this method in a real application. One way would be to use the usual idea of adjusting the fitness function with a sharing function (Goldberg & Richardson, 1987) and compute the sharing function by using the target population. Another method would be to introduce a niche size parameter n^*. With this method, we would use the usual sharing function to determine the number of individuals in a niche, but we would determine which individual wins a tournament in the following manner: if the two niches that the individuals belong to both have less than n^* members, then the individual with the better fitness value wins; otherwise the more endangered individual wins. The fitness-sharing method evolves to a population that has more individuals in the niches that have higher fitness peaks, and the niche-threshold method evolves to a population with roughly n^* individuals in each of the best N/n^* niches.

We can make a modification of this algorithm to speed it up. Currently, each generation requires that we do $O(N^2)$ operations to compute the number of individuals in the same niche. Actually, we do not need to sample the entire target population to get a good estimate of the number of individuals in a niche as has been suggested elsewhere (Goldberg & Richardson, 1987). We only need to look at Ak individuals in the target population, where $A \approx 5$ for a good sampling. This reduces the computational complexity to $O(Nk)$ per generation.

This technique is ready for trial in practical problems. It is easy to implement and stably maintains the target subpopulation sizes. It should prove to be an immediate aid to genetic algorithmists looking for alternative means of obtaining effective niching in their GAs.

4 Conclusions

This paper has considered the combination of tournament selection and the sharing-function method and has shown both analytically and empirically that the naive combination of these methods is unable to maintain a significant number of niches stably. On the other hand, by continuously updating the sharing calculation in the target generation, the dynamics settle down and the modified method, *tournament selection with continuously updated sharing*, is able to maintain many niches, almost without loss. A number of extensions of the technique have been suggested, including the use of niching thresholding and partial niche sampling, and more work remains. Nonetheless, the analytical calculations and computer simulations presented herein support the trial of this technique in the many genetic algorithms where stable niching is a must.

Acknowledgments

The authors acknowledge the support provided by the US Army under Contract DASG60-90-C-0153 and by the National Science Foundation under Grant ECS-9022007.

A Analysis of Unimpeded Genetic Drift

Boltzmann tournament selection has been suggested as one way to obtain niching-like behavior in a tournament scheme (Goldberg, 1990), and the original work presented some analytical and empirical results in support of that claim. In its purest form, BMTS uses only fitness information to distinguish between different individuals although the original paper recognized that it might be useful to use other signals such as genotypic or phenotypic difference to distinguish between different individuals as well. At high temperatures, if fitness is the only distinguishing signal, Boltzmann tournament selection is subject to the vagaries of random genetic drift. This is clear and not open to question, because all individuals are otherwise indistinguishable to the algorithm, and if there are differences that are unknown to the selection process, they can only be acted upon by the well-known effects of random drift in a finite population (Goldberg & Segrest, 1987). In this appendix, we calculate a mean-field approximation of such unimpeded drift and apply it to calculate the number of niches that are supported as time goes on. As expected, the calculation shows that a drifting population provides little protection to the undifferentiated niches, and this suggests that at high temperatures that Boltzmann tournament selection is ineffective when no other signal other than fitness is used to distinguish between differing members of different niches. On the other hand, the analysis does not apply to lower temperatures where there is differentiation among members of different niches, and members are chosen to participate in tournaments in a manner that biases that participation toward lower fitness (most different) individuals. Whether the bias of the anti-acceptance step is enough to counteract the bias of the primary tournament stably is an open question, upon which this analysis sheds little light. Nonetheless, these analysis techniques may prove useful in answering this question and—if it proves to be necessary—-restructuring the algorithm to obtain the desired stable, near-Boltzmann performance.

Rather than concentrate on the entire population as we did before, we will concentrate on a single niche. We will assume that at $t = 0$, all niches have the same number of individuals and recognize that we can use generating functions to do the analysis. Let l represent the state with a single individual, l^2 represent the state with two individuals, $\frac{1}{2}(1+l)$ represent the state which has fifty percent probability of having no individuals and fifty precent probability of having one individual, and so on. Using this notation, we can write the time evolution of the niche in the mean-field approximation. At $t = 0$, the niche has n individuals, and so the initial state is represented by the function l^n. Each successive generation, the individuals in the niche can win both tournaments, win one tournament, or lose both tournaments, with probabilities 1/4, 1/2, and 1/4, respectively. We represent this by substituting l by $\frac{1}{4}l^2 + \frac{1}{2}l + \frac{1}{4} = (\frac{l+1}{2})^2$ each generation. The result is an iterative equation:

$$g(l,0) = l^n; \tag{27}$$

$$g(l,t) = g((\frac{l+1}{2})^2, t-1). \tag{28}$$

Note that $g(0,t)$ gives the probability that the niche will have no individuals at time t. Our estimate for the number of niches remaining at a given time is therefore

$$k(t) = k_0[1 - g(0,t)]. \tag{29}$$

We can get a better idea of how the system behaves by using asymptotics on the iterative equation $l(t+1) = (\frac{l(t)+1}{2})^2$. Let $\epsilon(t) = 1 - l(t)$. The iterative equation becomes:

$$\epsilon(t+1) = \epsilon(t)[1 - \frac{1}{4}\epsilon(t)]. \tag{30}$$

For large t, the behavior is given by:

$$\epsilon(t) = 4/t + \ldots. \tag{31}$$

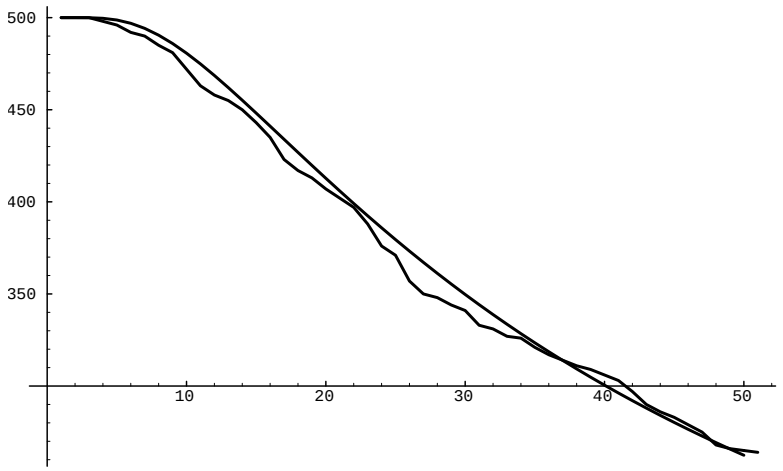

Figure 5: A comparison of analysis versus experiment for the case of unimpeded drift shows good agreement with 500 niches and 10 individuals/niche.

Plugging this back into the definition of the generating function gives:

$$g(0,t) = 1 - 4/t + \ldots, \tag{32}$$

and so the number of niches alive at time t for large t is

$$k(t) = \frac{4k_0}{t} + \ldots. \tag{33}$$

As expected, the loss of niches is a severe problem, and figure 5 shows the number of niches that survive as a function of generation number in a run starting with 500 niches and 10 individuals per niche. It is interesting that the mechanism of loss is in some sense opposite to that observed in the previous analysis of the naive combination of sharing and tournament selection. There the loss resulted from the successive and repeated overshooting corrections. Here, the loss results from the lack of any correction whatsoever. Again, extrapolating these results to low temperatures should be done with caution. There is a restoring pressure of sorts in the anti-acceptance phase of the algorithm; whether it is strong enough to counteract the positive bias of the acceptance phase and the noise of drift is unclear, but this analysis provides a path to answering that open question and repairing the BMTS algorithm should this be necessary.

References

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, 101–111.

Deb, K. (1989) *Genetic algorithms in multimodal function optimization* (MS Thesis and TCGA Report No. 89002). Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms.

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

Deb, K., & Goldberg, D. E. (1991). *Analyzing deception in trap functions* (IlliGAL Report No. 91009). Urbana: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

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

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 69–93.

Goldberg, D. E., Deb, K., & Clark, J. H. (1991). *Genetic algorithms, noise, and the sizing of populations* (IlliGAL Report No. 91010). Urbana: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. *Proceedings of the Second International Conference on Genetic Algorithms*, 41–49.

Goldberg, D. E., & Segrest, P. (1987). Finite Markov chain analysis of genetic algorithms. *Proceedings of the Second International Conference on Genetic Algorithms*, 1–8.