GENETIC ALGORITHMS WITH SHARING FOR MULTIMODAL FUNCTION OPTIMIZATION

David E. Goldberg, The University of Alabama,
Tuscaloosa, AL 35487

and

Jon Richardson, The University of Tennessee
(formerly at The University of Alabama),
Knoxville, TN 37996

ABSTRACT

Many practical search and optimization problems require the investigation of multiple local optima. In this paper, the method of sharing functions is developed and investigated to permit the formation of stable subpopulations of different strings within a genetic algorithm (GA), thereby permitting the parallel investigation of many peaks. The theory and implementation of the method are investigated and two, one-dimensional test functions are considered. On a test function with five peaks of equal height, a GA without sharing loses strings at all but one peak; a GA with sharing maintains roughly equally sized subpopulations clustered about all five peaks. On a test function with five peaks of different sizes, a GA without sharing loses strings at all but the highest peak; a GA with sharing allocates decreasing numbers of strings to peaks of decreasing value as predicted by theory.

INTRODUCTION

Genetic algorithms (GAs) are finding increasing application in a variety of problems across a spectrum of disciplines (Goldberg & Thomas, 1986). This is so, because GAs place a minimum of requirements and restrictions on the user prior to engaging the search procedure. The user simply codes the problem as a finite length string, characterizes the objective (or objectives) as a black box, and turns the GA crank. The genetic algorithm then takes over, seeking near-optima primarily through the combined action of reproduction and crossover. These so-called simple GAs have proved useful in many problems despite their lack of sophisticated machinery and despite their total lack of knowledge of the problem they are solving. Yet as their usage has grown, several objections to their performance have arisen. Simple GAs have been criticized for sub-par performance on multimodal (multiply-peaked) functions. They have also been criticized for so-called premature convergence where substantial fixation occurs at most bit positions before obtaining sufficiently near-optimal points (Cavicchio, 1970; De Jong, 1975; Mauldin, 1984; Baker, 1985).

In this paper, we examine the first of these maladies and propose a cure borrowed from nature. In particular, our herbal remedy causes the formation of niche-like and species-like subdivision of the environment and population through the imposition of sharing functions. These sharing functions help mitigate unbridled head-to-head competition between widely disparate points in a search space. This reduction in competition between distant points thereby permits better performance on multimodal functions. As a side benefit we find that sharing helps maintain a more diverse population and more considered (and less premature) convergence. In the remainder of this paper, we review the problem and past efforts to solve it; we consider the theory of niche and speciation through Holland's modified two-armed bandit problem, and we compare the performance of a genetic algorithm both with and without the sharing function feature. Finally we examine extensions of the sharing function idea to permit its implementation in a wide array of problems.

MULTIMODAL OPTIMIZATION, GENETIC DRIFT, AND A SIMPLE GA

The difficulty posed by a multimodal problem for a simple genetic algorithm may be illustrated by a straightforward example. Figure 1 shows a bimodal function of a single variable: $f(x) = 4(x-0.5)^2$ coded by a normalized, five-bit binary integer. In this problem, the two optima are located at extreme ends of the one-dimensional space. If we start a genetic algorithm with a population chosen initially at random and let it run for a large number of generations, our fondest hope is that stable subpopulations cluster about the two optima (about 00000 and 11111). In fact if we perform this experiment, we find that the simple GA eventually clusters all of its points about one peak or the other.

Why does this happen? After all, doesn't the fundamental theorem of genetic algorithms (Holland, 1975; De Jong, 1975; Goldberg, 1986) tell us that exponentially increasing numbers of trials will be given to the observed best schemata? Yes it does, but the theorem assumes an infinitely large population size. In a finite size popula-

Material protegido por derechos de autor

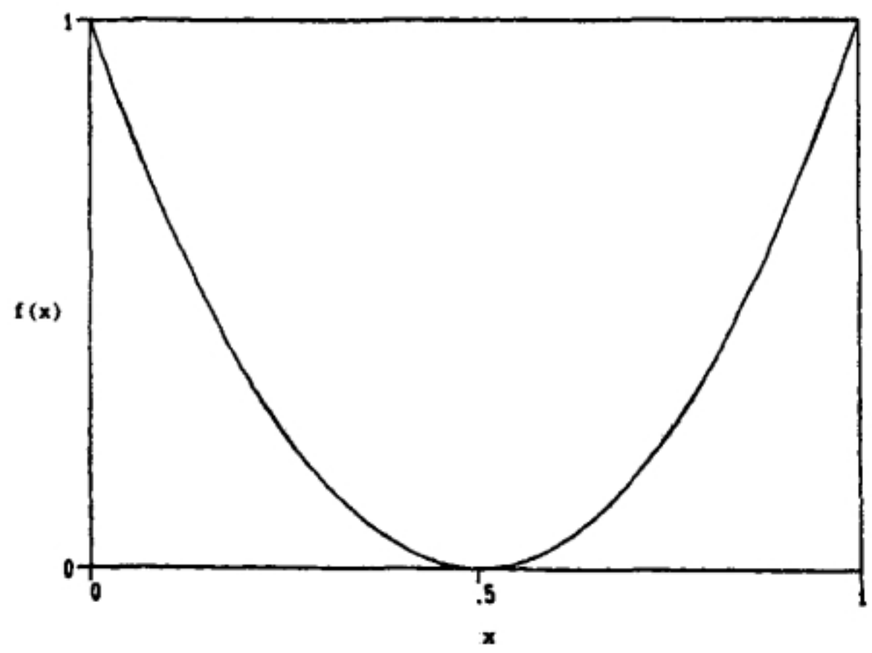


Figure 1. Bimodal function with equal peaks.

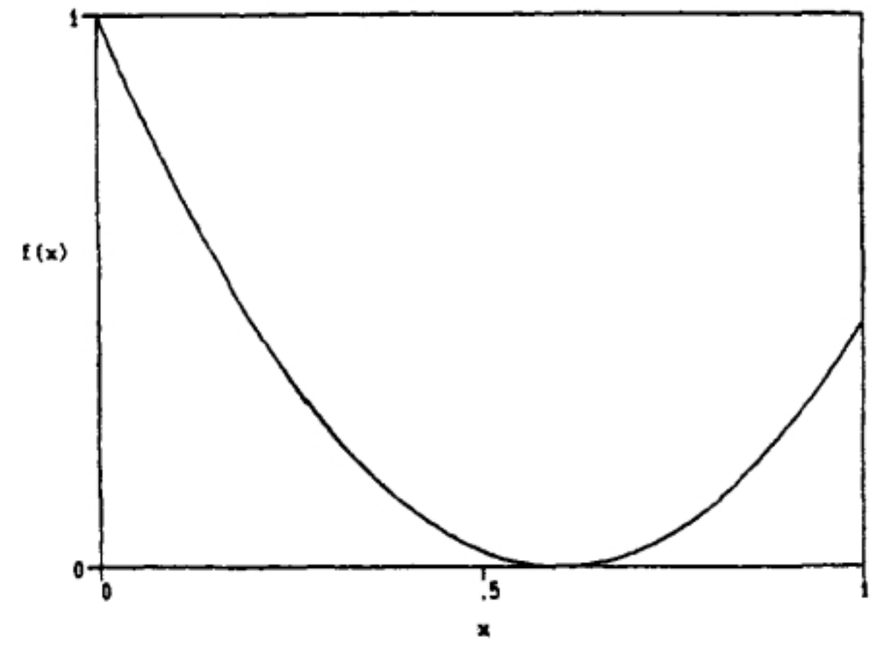


Figure 2. Bimodal function with unequal peaks.

tion, even when there is no selective advantage for either of two competing alternatives (as is the case for schemata 11*** and 00*** in the example problem) the population will converge to one alternative or the other in finite time (De Jong, 1975; Goldberg & Segrest, this volume). This problem of finite populations is so important that geneticists have given it a special name, genetic drift. Stochastic errors tend to accumulate, ultimately causing the population to converge to one alternative or another.

The convergence toward one optimum or another is clearly undesirable in the case of peaks of equal value. In multimodal problems where peaks of different altitudes exist, the desirability of convergence to the globally best peak is not so clear cut. In Figure 2 we see a bimodal function with unequal peaks: $f(x) = 2.8(x-0.6)^2$ with a five-bit normalized coding. If we are interested in obtaining only the global optimum, we should not mind the eventual convergence of the population to the leftmost point; however, this convergence is not always guaranteed. Small initial populations may allow sampling errors which overestimate the schemata of the rightmost points thereby permitting convergence to the wrong peak. Furthermore, in real world optimization we are often interested in having information about good, better, and best solutions. When this is so, it might be nice to see a form of convergence that permits stable subpopulations of points to cluster about both peaks according to peak fitness.

In either of these cases, we can argue for more controlled competition and less reckless convergence than is possible when we work with a simple, tripartite (reproduction, crossover, and mutation) genetic algorithm. For these reasons we turn to the theory of niche and speciation to find an appropriate model for naturally regulated competition.

THEORY OF SPECIES AND NICHE

The results of our initial gedankenexperimente (thought experiments) with simple GAs and multimodal functions are somewhat perplexing when juxtaposed with natural example. In our problem with equal peaks, the simple GA converges on one peak or the other even though both peaks are equally useful. By contrast, why doesn't nature converge to a single species? In our second problem with unequal peaks, we notice that the simple GA again converges to one peak, usually--but not always--the "correct" peak. How, when faced with a somewhat less fit species, does nature choose to limit population size before resorting to extinction? In both cases, nature has found a way to combat unbridled competition and permit the formation of stable subpopulations. In nature, different species don't go head to head. Instead they exploit separate niches (sets of environmental features) in which other organisms have little or no interest. In this section, we need to bridge the gap between natural example and genetic algorithm practice through the application of some useful theory.

Although there is a well-developed biological literature in both niche and speciation, its transfer to the arena of GA search has been limited. Like many other concepts and operators, the first theories directly applicable to artificial genetic search are due to Holland (1975). To illustrate niche and species, Holland introduces a modification of the two-armed bandit problem with distributed payoff and sharing. Let's examine his argument with a concrete formulation of the same problem.

Imagine a two-armed bandit as depicted in Figure 3. In the ordinary two-armed bandit problem (Holland, 1975; De Jong, 1975), we have two arms, a left arm and a right arm, and we have different payoffs associated with each arm. Suppose we have an expected payoff associated with the left arm of $25 and an expected payoff associated with the right arm of $75; in the standard two-armed bandit, we are unaware initially which arm pays the higher amount, and our dilemma is to min-

Material protegido por derechos de autor

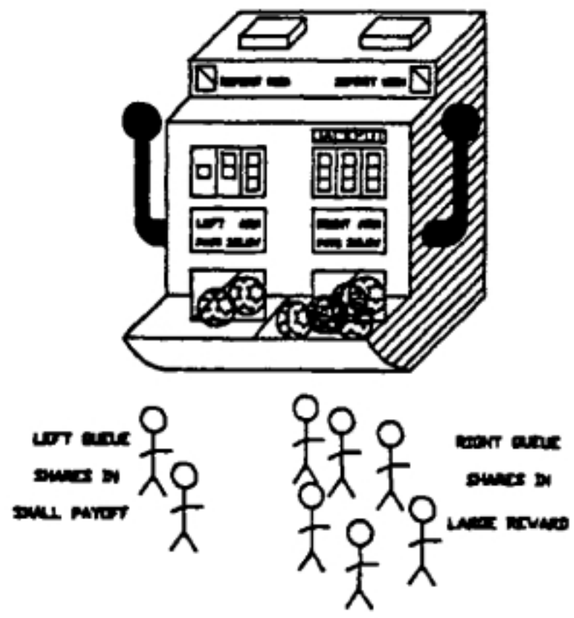


Figure 3. Sketch of the two-armed bandit with queues and sharing.

imize our expected losses over some number of trials. In this form, the two-armed bandit problem puts the tradeoffs between exploration and exploitation in sharp perspective. We can take extra time to experiment, but in so doing risk the possible gain from choosing the right arm, or we can experiment briefly and risk making an error once we choose the arm we think is best. In this way, the two-armed bandit has been used to justify the allocation strategy adopted by the reproductive plans of simple genetic algorithms.

This is not our purpose here. Instead we examine the modified two-armed bandit problem to put the concepts of niche and species in sharper focus. In the modified problem, we further suppose that we have a population of some number of players, say 100 players, and that each player may decide to play one arm or the other. If at this point we do nothing else, we simply create a parallel version of the original two-armed bandit problem where we expect that all players eventually line up behind the observed best (and actual best) arm. To produce the subdivision of species and niche, we introduce an important rule change. Instead of allowing a full measure of payoff for each individual, individuals who choose a particular arm are now forced to share the wealth derived from that arm with other players queued up at the arm. At first glance, this change appears to be quite minor. In fact, this single modification causes a strikingly and surprisingly different outcome in the modified two-armed bandit.

To see why and how the results change, we first recall that despite the different rules of the game, we still allocate population members according to payoff. In the modified game, an individual will receive a payoff which depends on the arm payoff value and the number of individuals queued up at that arm. In our concrete example, an individual lined up behind the right arm when all individuals are lined up behind that same arm receives an amount \$75/100 = \$0.75. On the other hand, an individual lined up behind the left arm when all individuals are queued there receives \$25/100 = \$0.25. In both cases, there is motivation for some individuals to shift lines. In the first case, a single individual changing lines stands to gain an amount \$25.00 - \$0.75 = \$24.25. The motivation to shift lines is even stronger in the second case. At some point in between we should expect there to be no further motivation to shift lines. This will occur when the individual payoffs are identical for both lines. If N is the population size, m_{right} and m_{left} are the number of individuals behind the right and left queues, and f_{right} and f_{left} are the expected payoff values from the right and left arms respectively, the equilibrium point may be calculated as follows:

$$\frac{f_{right}}{m_{right}} = \frac{f_{left}}{m_{left}}$$

In our example, this complete equalization of individual payoff occurs when 75 players select the right arm and 25 players select the left arm, because \$75/75 = \$25/25 = \$1.

This problem may be extended to the k-armed case directly, and the extension does not change the fundamental conclusions at all: the system attains equilibrium when the ratios of arm payoff to queue length are equal (Holland, 1975). The incorporation of forced sharing causes the formation of stable subpopulations (species) behind different arms (niches) in the problem. Furthermore, the number of individuals devoted to each niche is proportional to the expected niche payoff. This is exactly the type of solution we had hoped for when we considered the bimodal problems of Figures 1 and 2. Of course the extension of the sharing concept to real genetic algorithm search is more difficult than the idealized case implies. In a real genetic algorithm there are many, many arms and deciding who should share and how much should be shared becomes a non-trivial question. In the next section we will examine a number of current efforts to induce niche and species through indirect or direct sharing.

A BRIEF REVIEW OF CURRENT SCHEMES

A number of methods have been implemented to induce niche and species in genetic algorithms. In some of these techniques the sharing comes about indirectly. Although the two-armed bandit problem is a nice, simple abstract model of niche and species formation and maintenance, nature is not so direct in divvying up her bounty. In natural settings, sharing comes about through crowding and conflict. When a habitat becomes fairly full of a particular organism, individuals are forced to share available resources.

Cavicchio's (1971) dissertation study was one of the first to attempt to induce niche-like and species-like behavior in genetic algorithm search. Specifically, he introduced a mechanism he called

Material protegido por derechos de autor

preselection. In this scheme, an offspring replaces the inferior parent if the offspring's fitness exceeds that of the inferior parent. In this way diversity is maintained in the population because strings tended to replace strings similar to themselves (one of their parents). Cavicchio claimed to maintain more diverse populations in a number of simulations with relatively small population sizes (n=20).

De Jong (1975) has generalized preselection in his crowding scheme. In De Jong crowding, individuals replace existing strings according to their similarity with other strings in an overlapping population. Specifically, an individual is compared to each string in a randomly drawn subpopulation of CF (crowding factor) members. The individual with the highest similarity (on the basis of bit-by-bit similarity count) is replaced by the new string. Early in the simulation, this amounts to random selection of replacements because all individuals are likely to be equally dissimilar. As the simulation progresses and more and more individuals in the population are similar to one another (one or more species have gotten a substantial foothold in the population) the replacements of individuals by similar individuals tends to maintain diversity within the population and reserve room for one or more species. De Jong has had success with the crowding scheme on multimodal functions when he used crowding factors CF=2 and CF=3. De Jong's crowding has subsequently been used in a machine learning application (Goldberg, 1983).

Booker (1982) discusses a direct application of the sharing idea in a machine learning application with genetics-based, classifier systems. In classifier systems, a sub-goal reward mechanism called a bucket brigade passes reward through a network of rules like money passing through an economy. Booker suggests that appropriately sized subpopulations of rules can form in such systems if related rules are forced to share payments. This idea is sound and has been forcefully demonstrated in Wilson's recent work with boolean function learning (Wilson, 1986); however, it does not transfer well to function optimization, because unlike classifier systems, there is no general way in function optimization to determine which strings are related.

Shaffer (1984) has used separate, fixed size subpopulations in his study of vector evaluated genetic algorithms (VEGA). In this study, each component of the vector (each criterion or objective measure) is mapped to its own subpopulation where separate reproduction processes are carried out. The method has worked well in a number of trial functions; however, Shaffer has expressed some concern over the procedure's ability to handle middling nondominated individuals--individuals that may be Pareto optimal but are not extremal (or even near extremal) along any single dimension. Furthermore, although the study does use separate subpopulations, it is unclear how the same method might be applied to the more usual single-criterion optimization problem.

A direct exploration of biological niche theory in the context of genetic algorithms is contained in Perry's (1985) dissertation. In this work, Perry defines a genotype-to-phenotype mapping, a multiple-resource environment, and a special entity called an external schema. External schemata are special similarity templates defined by the simulation designer to characterize species membership. Unfortunately, the required intervention of an outside agent limits the practical use of this technique in artificial genetic search. Nonetheless, the reader interested in the connections between biological niche theory and GAs may be interested in this work.

Grosso (1985) also maintains a biological orientation in his study of explicit subpopulation formation and migration operators. Multiplicative, heterotic (problems with diploid structures where a heterozygote is more highly fit than the homozygote) objective functions are used in this study, and as such, the results are not directly applicable to most artificial genetic search; however, Grosso was able to show the advantage of intermediate migration rate values over either isolated subpopulations (no migration) and panmictic (completely mixed) subpopulations. This study suggests that the imposition of a geography within artificial genetic search may be another useful way of assisting the forming diverse subpopulations. Further studies are needed to determine how to do this in more general artificial genetic search applications.

Although he has not directly addressed niche and species, Mauldin (1984) has attempted to better maintain diversity in genetic algorithms through his uniqueness operator. The uniqueness operator arbitrarily returns diversity to a population whenever it is judged to be lacking. To implement uniqueness, Mauldin defines a uniqueness parameter k_u that may decrease with time (similar to the cooling of simulated annealing). He then requires that for insertion in a population, an offspring must be different than every population member at a minimum of k_u loci. If the offspring is not sufficiently different, it is mutated until it is. By itself, uniqueness is little more than a somewhat knowledgeable (albeit expensive) mutation operator. That it is useful in improving offline (convergence) performance is not unexpected. Grefenstette (1986) has recently supported the notion of fairly high mutation probabilities (p_m = 0.01 to 0.1) when convergence to the best is the main goal. It is interesting to note that uniqueness combined with De Jong's crowding scheme worked better than either operator by itself (Mauldin, 1984). This result suggests that maintaining diversity for its own sake is not the issue. Instead, we need to maintain appropriate diversity--diversity that in some way helps cause (or has helped cause) good strings. In the next section, we show how we can maintain appropriate diversity through the use of sharing functions.

Material protegido por derechos de autor

SHARING FUNCTIONS

In attempting to induce species, we must either directly or indirectly cause intraspecies sharing, but we are faced with two important questions: who should share, and how much should be shared? In natural systems, these two questions are answered implicitly through conflict for finite resources. Different species find different combinations of environmental factors--different niches--which are relatively uninteresting to other species. Individuals of the same species use those resources until there is conflict. At that point, they vie for the same turf, food, and other environmental resources, and the increased competition and conflict cause individuals of the same species to share with one another, not out of altruism, but because the resources they give up are not worth the cost of the fight. It might be possible to induce similar conflict for resources in genetic optimization. Unfortunately, in many optimization problems, there is no natural definition of a resource. As a result, we must invent some way of imposing niche and speciation on strings based on some measure of their distance from each other. We do just this with what we have called a <u>sharing function</u>.

A sharing function is nothing more than a way of determining the degradation of an individual's payoff due to a neighbor at some distance as measured in some similarity space. Mathematically, we introduce a convenient metric d over our decoded parameters x_i (the decoded parameters are themselves functions of the strings $x_i = x_i(s_i)$):

$$d_{ij} = d(x_i, x_j)$$

Alternatively, we may introduce a metric over the strings directly:

$$d_{ij} = d(s_i, s_j)$$

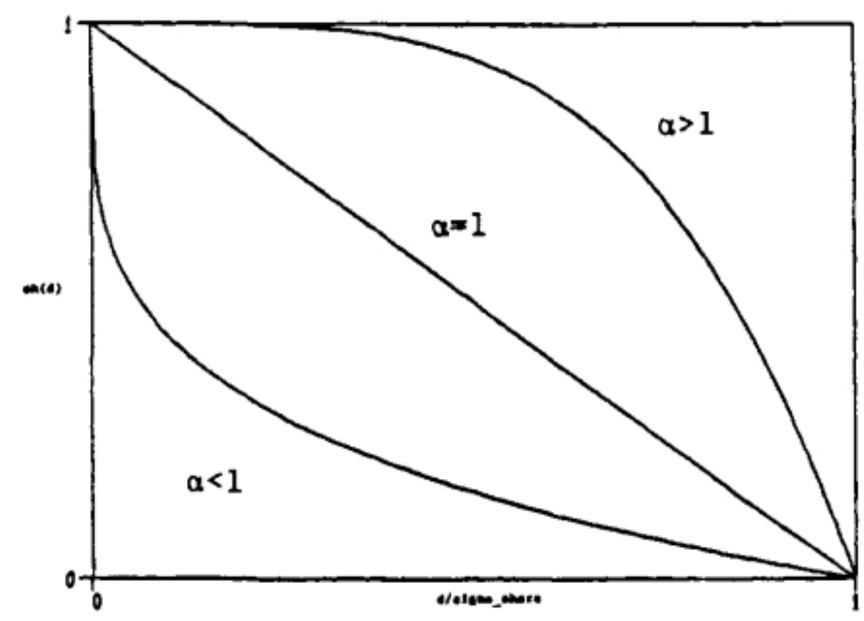
In this paper, we use a metric defined over the decoded parameters x_i (phenotypic sharing); later on, we briefly consider the use of metrics defined over the strings (genotypic sharing). However we choose a metric, we define a sharing function <u>sh</u> as a function of the metric value sh = sh(d) with the following three properties:

1. $0 \le sh(d) \le 1$, for all d
2. $sh(0) = 1$
3. $\lim_{d \to \infty} sh(d) = 0$

Many sharing functions are possible. Power law functions are convenient:

$$sh(d) = 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha}, \quad d < \sigma_{share}$$
$$= 0, \quad \text{otherwise}$$

In this equation, σ_{share} and α are constants, and Figure 4 displays power law sharing functions with α values equal to one, greater than one, and less than one.



<u>Figure 4</u>. Power law sharing functions sh=sh(d).

Once we have selected a metric and a sharing function, it is a simple matter to determine a string's realized or shared fitness. We define a string's shared fitness f' as its potential fitness divided by its niche count m_i':

$$f_i' = \frac{f_i}{m_i'}$$

The niche count m_i' for a particular string i is taken as the sum of all share function values taken over the entire population:

$$m_i' = \sum_{j=1}^{N} sh(d_{ij}) = \sum_{j=1}^{N} sh(d(x_i, x_j))$$

Note that the sum includes the string itself. Thus, if a string is all by itself in its own niche ($m_i' = 1$), it receives its full potential fitness value. Otherwise the sharing function derates fitness according to the number and closeness of neighboring points.

Material protegido por derechos de autor

RESULTS

We evaluate the use of sharing functions through computational experiments on two multimodal problems. The first function is a periodic function with five peaks of equal magnitude:

$$f_1(x) = \sin^6(5.1\pi x+.5)$$

We compare the performance of a simple genetic algorithm with stochastic remainder selection, crossover, no mutation, and no sharing to the performance of the same GA with sharing. We use a triangular sharing function (α=1) with σ_{share} = 0.1; the metric d is taken as the absolute value of the difference between the string x values. Length 30 binary strings are decoded as unsigned binary integers and normalized by the constant 2^{30}-1. Genetic algorithm parameters have been held constant across all runs as follows:

Probability of mutation p_m = 0.0

Probability of crossover p_c = 0.8

Population size = 50

Maximum number of generations = 100

We seek methods that maintain appropriate diversity without introducing arbitrary diversity through mutation or other means. Therefore, we have set the mutation probability to zero to put the sharing function technique to its most stringent test: if appropriate diversity can be maintained without mutation, we should expected similar or better (off-line) performance when mutation is present.

Runs with and without sharing are started from the same population generated uniformly at random. After 50 generations the run without sharing has lost all points at two peaks as shown in Figure 5. By contrast, the run with sharing has stable subpopulations roughly equal in size at all five peaks as shown in Figure 6. Similar graphs at generation 100 are shown in Figures 7 and 8. Note how the run without sharing (Figure 7) has completely converged to a single peak even though there is no selective advantage for any peak. By contrast, the run with sharing remains committed to stable subpopulations clustered about each peak. This latter result is especially remarkable considering that no mutation has been used. With no mutation, once an allele is lost at a particular locus, there is no way to get it back. The existence of stable subpopulations about each peak shows how the sharing function maintains appropriate diversity--the necessary, sometimes competing building blocks--required to exploit all five peaks.

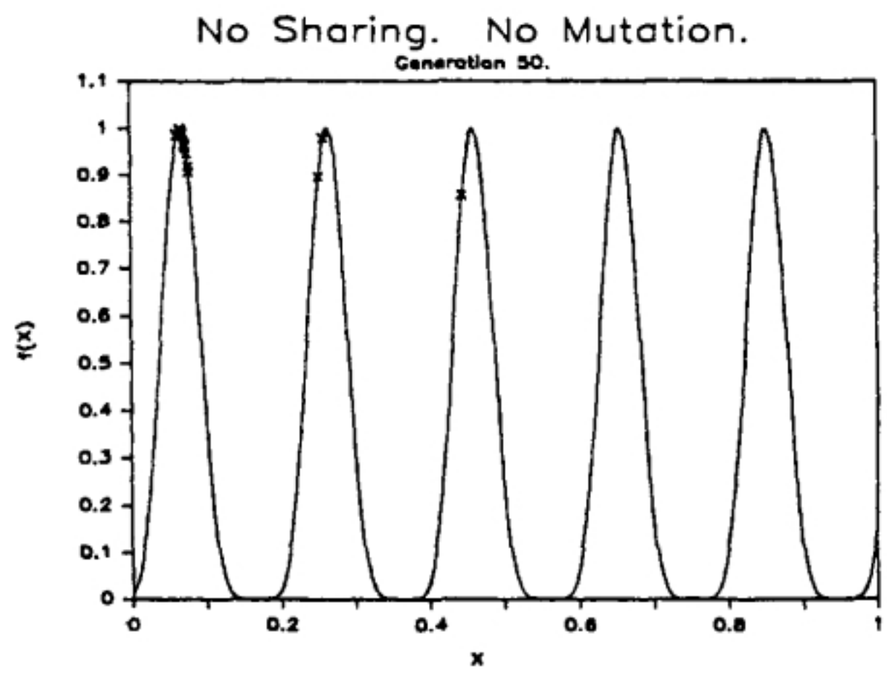


Figure 5. GA without sharing concentrates points at three peaks after 50 generations on function f_1 (equal peaks).

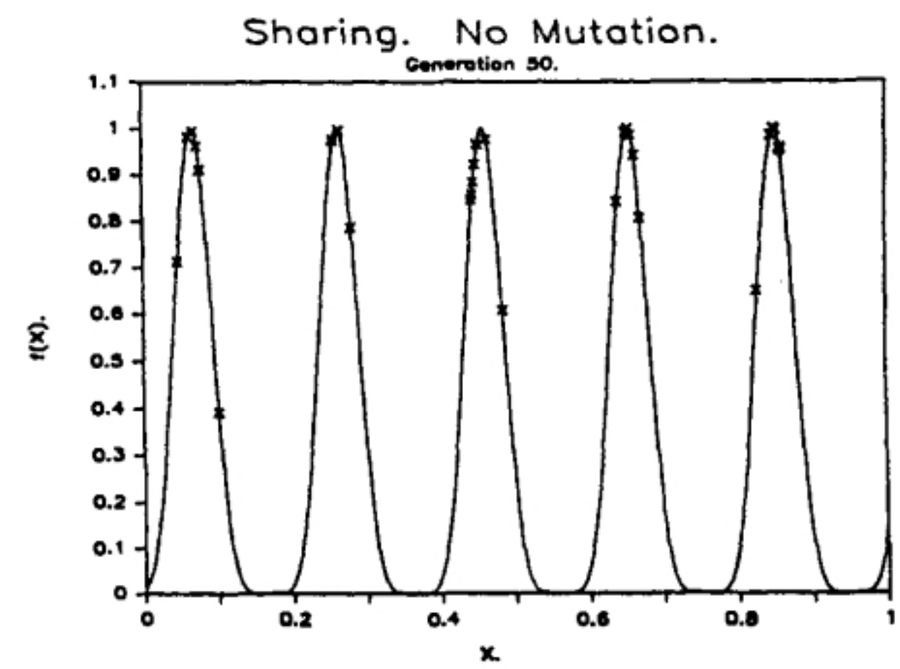


Figure 6. GA with sharing distributes points to all five peaks after 50 generations on function f_1 (equal peaks).

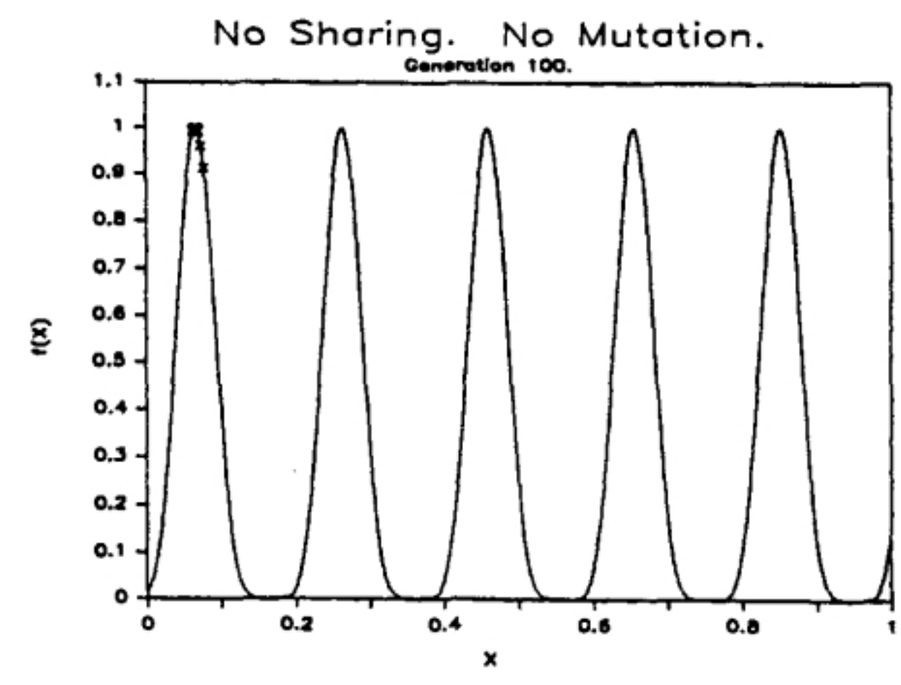


Figure 7. GA without sharing concentrates all points on a single peak after 100 generations on function f_1 (equal peaks).

Material protegido por derechos de autor

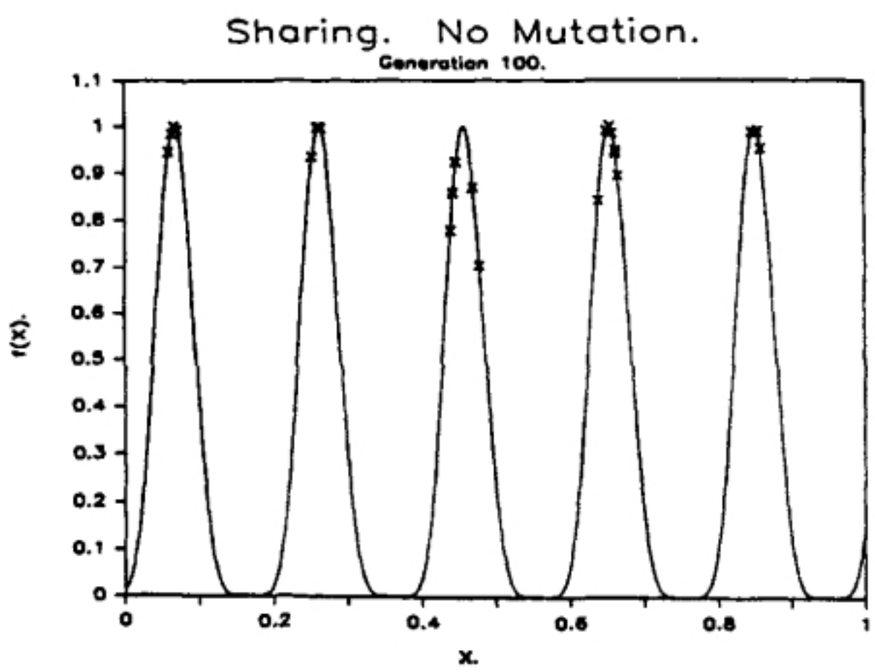


Figure 8. GA with sharing continues to distribute points among all five peaks after 100 generations on function f_1 (equal peaks).

The second test function $f_2(x)$ has five peaks with decreasing peak magnitude as given by the following equation:

$$f_2(x) = f_1(x) \cdot e^{\left[-4\ln 2 \frac{(x-0.0667)^2}{0.8^2} \right]}$$

We make comparisons of the two GAs, with and without sharing, using the same parameters and string coding as before. Specifically, we compare the two cases at generation 100. By that time, the genetic algorithm without sharing has allocated all of its trials to the highest peak as shown in Figure 9. By contrast, the GA with sharing forms stable clusters of points about four of the five highest peaks with cluster size roughly proportional to peak fitness as shown in Figure 10. This is the kind of performance we predicted earlier, except for the lack of strings at the lowest peak. To understand why this has occurred we briefly return to the theory of sharing presented earlier.

From our earlier discussion, we expect a stable equilibrium to form when the following equations hold true:

$$\frac{f_i}{m_i'} = \frac{f_{i+1}}{m_{i+1}'}$$

for all niches i, and

$$\sum_{i=1}^{M} m_i' = N$$

where M is the number of niches and N is the population size. It may be shown that this set of equations predicts proportions of niche members as the ratio of niche fitness to the total fitness. On function f_2, we expect a total number of trials to be allocated to the lowest peak as follows:

$$50*0.075/(0.075+0.22+0.5+0.85+1.0) = 1.42$$

In other words, we expect approximately one individual to remain at that peak, and we should not be surprised that no strings cluster there. If we want to maintain a subpopulation at such a low peak, our theory suggests that we need a larger population to overcome the unavoidable errors of stochastic sampling and selection.

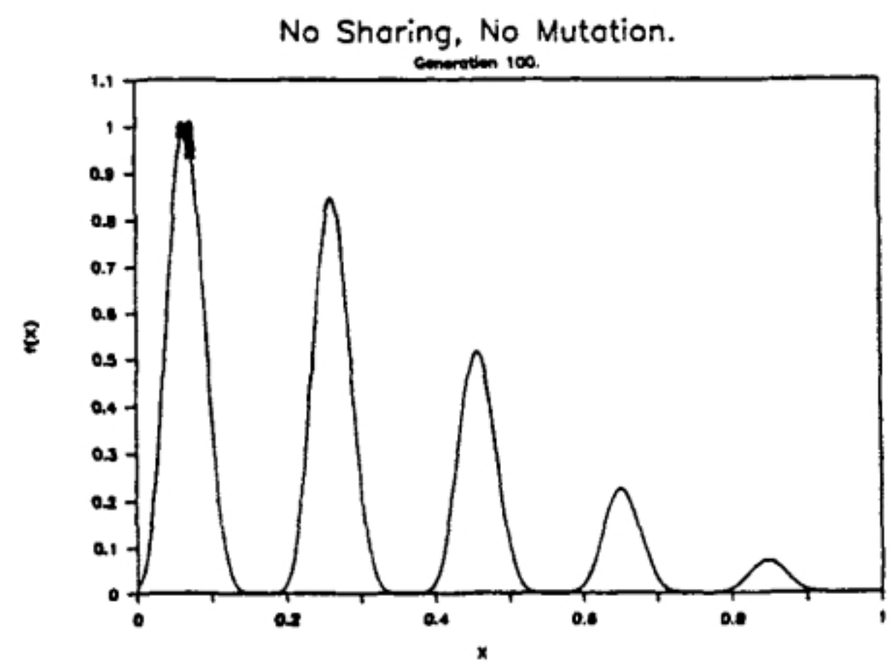


Figure 9. GA without sharing concentrates all points on highest peak (at generation 100) on function f_2 (decreasing peaks).

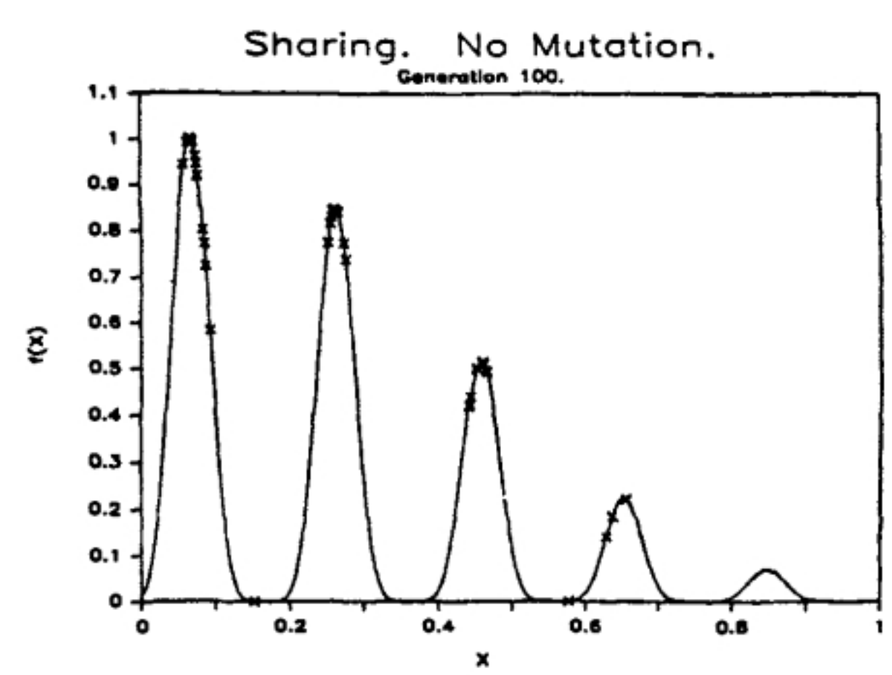


Figure 10. GA with sharing distributes points to all but lowest peak (at generation 100) on function f_2 (decreasing peaks). Lowest peak has expected population size of only one member, not enough to overcome selection and sampling errors.

Material protegido por derechos de autor

EXTENSIONS

The method of sharing functions is not limited to one-dimensional problems. Sharing functions may be evaluated using any reasonable metric that includes any number of problem parameters. Additionally, there is no reason to limit the method to phenotypic sharing (where the measures are calculated based on differences in the phenotype--the decoded problem parameters). As an alternative, sharing functions may be evaluated using the genotype (the string) directly. In this form, the Hamming distance (the number of different bits) may be an especially useful metric.

In many cases, there may be no need to perform the sharing calculations as precisely as has been implied by the above equations. With the full formulation, $\binom{N}{2}$ sharing function evaluations are required (N^2 if symmetry is not exploited) to calculate the niche count values m_i exactly. This level of computation may be reduced by taking a random sample of k (k<<N) share function values and extrapolating the mean. If the mean of the k share function evaluations for string i is μ_i and the population is of size N, then the following formula provides a reasonable way to estimate the niche count m_i':

$$m_i' = (N-1)\mu_i + 1$$

Although this approximate niche count method has not been tested, Monte Carlo sampling techniques have been adopted in at least one other GA study with success (Grefenstette & Fitzpatrick, 1985). There is every reason to suspect that cheaper, approximate niche count estimates may be used without excessive performance degradation.

CONCLUSION

In this paper, we have developed a method for improving the performance of genetic algorithms in multimodal function optimization problems. This method uses sharing functions to induce artificial analogs to the natural concepts of niche and species, thereby permitting the formation of stable, non-competing subpopulations of points surrounding important peaks in the search space. The method has been tested on two multimodal functions, one with peaks of equal size and one with peaks of decreasing size. In both cases, the genetic algorithm with sharing is able to maintain stable subpopulations about significant peaks while an identical GA without sharing is unable to maintain points at more than a single peak. Additionally, the GA with sharing is also able to maintain stable subpopulations of appropriate size: the number of points in each cluster is roughly proportional to the peak fitness value. This automatic allocation of resources in a reasonable fashion should not only be transferable to other multimodal optimization problems, it should help in maintaining appropriate diversity in genetic algorithms without resorting to mutation and mutation-like operators that unnecessarily degrade on-line performance. These proof-of-principle results should permit the extension of these methods to other larger and more complex problems of genetic optimization.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant MSM-8451610.

REFERENCES

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In J. J. Grefenstette (Ed.), Proceedings of an International Conference on Genetic Algorithms and Their Applications (pp. 101-111). Pittsburgh: Carnegie-Mellon University.

Booker, L. B. (1982). Intelligent behavior as an adaptation to the task environment. (Doctoral dissertation, Technical Report No. 243. Ann Arbor: University of Michigan, Logic of Computers Group). Dissertations Abstracts International, 43(2), 469B. (University Microfilms No. 8214966)

Cavicchio, D. J. (1970). Adaptive search using simulated evolution. Unpublished doctoral dissertation, University of Michigan, Ann Arbor.

De Jong, K. A. (1975). An analysis of the behavior of a class of genetic adaptive systems. (Doctoral dissertation, University of Michigan). Dissertation Abstracts International, 36(10), 5140B. (University Microfilms No. 76-9381)

Goldberg, D. E. (1983). Computer-aided gas pipeline operation using genetic algorithms and rule learning (Doctoral dissertation, University of Michigan). Dissertation Abstracts International, 44(10), 3174B. (University Microfilms No. 8402282)

Goldberg, D. E. (1986). Simple genetic algorithms and the minimal deceptive problem (TCGA Report No. 86003). Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms.

Goldberg, D. E., & Thomas, A. L. (1986). Genetic algorithms: A bibliography (TCGA Report No. 86001). Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics, SMC-16(1), 122-128.

Material protegido por derechos de autor

Grefenstette, J. J., & Fitzpatrick, J. M. (1985). Genetic search with approximate function evaluations. In J. J. Grefenstette (Ed.), Proceedings of an International Conference on Genetic Algorithms and Their Applications (pp. 112-120). Pittsburgh: Carnegie-Mellon University.

Grosso, P. B. (1985). Computer simulation of genetic adaptation: Parallel subcomponent interaction in a multilocus model. (Doctoral dissertation, University of Michigan, University Microfilms No. 8520908).

Holland, J. H. (1975). Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press.

Mauldin, M. L. (1984). Maintaining diversity in genetic search. Proceedings of the National Conference on Artificial Intelligence, 247-250.

Perry, Z. A. (1984). Experimental study of speciation in ecological niche theory using genetic algorithms. (Doctoral dissertation, University of Michigan). Dissertation Abstracts International, 45(12), 3870B. (University Microfilms No. 8502912)

Shaffer, J. D. (1984). Some experiments in machine learning using vector evaluated genetic algorithms. Unpublished doctoral dissertation, Vanderbilt University, Nashville.

Wilson, S. W. (1986). Classifier system learning of a boolean function (Research Memo RIS-27r). Cambridge: Rowland Institute for Science.

Material protegido por derechos de autor