Adaptive Niching Via Coevolutionary Sharing

David E. Goldberg
& Liwei Wang

IlliGAL Report No. 97007
August 1997

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue Urbana, IL 61801
Office: (217) 333-0897
Fax: (217) 244-5705

ABSTRACT

An adaptive niching scheme called coevolutionary shared niching (CSN) is proposed, implemented, analyzed and tested. The scheme overcomes the limitations of fixed sharing schemes by permitting the locations and radii of niches to adapt to complex landscapes, thereby permitting a better distribution of solutions in problems with many badly spaced optima. The scheme takes its inspiration from the model of *monopolistic competition* in economics and utilizes two populations, a population of businessmen and a population of customers, where the locations of the businessmen correspond to niche locations and the locations of customers correspond to solutions. Initial results on straightforward test functions validate the distributional effectiveness of the basic scheme, although tests on a massively multimodal function do not find the best niches in the allotted time. This result spurs the design of an *imprint* mechanism that turns the best customers into businessmen, thereby making better use of the search power of the large population of customers. Although additional testing is needed, coevolutionary sharing appears to be a powerful means of controlling the number, location, extent and distribution of solutions in complex landscapes.

1 INTRODUCTION

Genetic algorithms have proven useful in a variety of problems across a spectrum of disciplines; however, in functions characterized by multiple peaks—even when the best peaks are equally fit—a simple GA will tend to converge to a single solution. As a result, various mechanisms have been proposed to stably maintain a diverse population throughout the search, thereby allowing GAs to identify multiple optima reliably. Many of these methods work by encouraging artificial niche formation through sharing, crowding, or other modifications to the selection or replacement process, but most of these methods introduce one or more parameters that affect algorithm performance, parameters such as the sharing radius in fitness sharing or the crowding factor in crowding. These parameters cannot be set easily without prior knowledge of the fitness landscape. Worse than this, in many problems the uniform specification of niche size is inadequate to capture solutions of varying location and extent without also increasing the population size beyond reasonable bounds. As a result, there remains a need to develop niching methods that stably and economically find the best niches regardless of their spacing and extent.

In this paper, we design, implement, and test a niching method to do just this. Coevolutionary shared niching (CSN) permits the formation of an arbitrary, specified number of the most highly fit niches subject to a user-specified minimum spacing requirement. The technique is loosely inspired by the economic model of *monopolistic competition*, in which businessmen locate themselves among geographically distributed customers so as to maximize their profit. In a similar manner, CSN creates two populations—businessmen and customers—where individuals in each population seek to maximize their separate interests thereby creating appropriately spaced niches containing the most highly fit individuals. The customer population may be viewed as a modification to the original sharing scheme (Goldberg & Richardson, 1987; Deb & Goldberg, 1989), in which the sharing function and σ_{share} are replaced by requiring customers to share within the closest businessman's service area. The evolution of the businessman population is conducted in a way that promotes the independent establishment of the most highly fit regions or niches in the search space.

The paper starts by briefly reviewing the basics of sharing and by reviewing a number

of niching schemes related to the objectives of this paper. This is followed by a discussion of the need for a coevolutionary solution and by the loose inspiration of the method from the model of monopolistic competition in economics. The basic scheme, coevolutionary shared niche (CSN), is then described, analyzed, and tested. Results on simple functions appear to validate the design; however, results on a massively multimodal test function are shown to be lacking. As a result, the scheme is extended using an *imprint* operator that carries the best of one population over to the other, thereby improving the performance of the niche identification mechanism of the scheme. The paper concludes by recommending a number of extensions of the work herein.

2 SHARING AND SUCH

A large number of methods are now available to promote stable formation of multiple optima within a population-oriented search scheme, but a complete review of the literature is beyond the scope of this study. In this section, we consider fitness sharing and a number of other approaches of effective niching.

2.1 Fitness Sharing

The point of departure for this work is so-called *fitness sharing* (Goldberg & Richardson, 1987) in which an individual's fitness is derated by an amount related to the number of similar individuals in the population. Specifically, an individual's shared fitness f_i' is equal to its raw fitness f_i divided by its niche count m_i:

$$f_i' = \frac{f_i}{m_i} \tag{1}$$

The niche count is calculated as the sum of sharing function values, which themselves are a function of some distance metric between two solutions:

$$m_i = \sum_{j=1}^{n} sh(d_{ij}) \tag{2}$$

The most commonly-used sharing functions are of the form

$$sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\alpha} & \text{if } d < \sigma_{share} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and common distance metrics are the Euclidean distance in parameter space and the Hamming distance in string space.

One common criticism of this scheme is the need to specify a value for σ_{share}. Procedures for estimating this parameter have been given (Deb & Goldberg, 1989), but these assume a relatively uniform distribution of points. Moreover, the naive solution of revising such estimates to account for a certain level of non-uniformity is not entirely satisfactory, because the population sizes required to maintain a large number of largely unoccupied niches is large and for the most part unnecessary.

2.2 Other Approaches

Some recent work aimed at speeding up sharing may have inadvertently attacked the problem of nonuniform niches. Specifically Yin and Germay (Yin & Germay, 1993) proposed alternating execution of a clustering algorithm and a shared GA. They claimed that the algorithm was qualitatively indistinguishable from a shared GA, but that the complexity is reduced from $O(n^2)$, where n is the population size to $O(nq)$, where q is the number of niches. As it turns out, the work is less interesting for the complexity reduction than it is for the use of a clustering algorithm. As has been pointed out elsewhere (Oei, Goldberg, & Chang, 1991), shared nichers can work in $O(nq)$ time when shared fitness values are sampled in a subpopulation of $O(q)$ size. The more interesting contribution is the use of a clustering algorithm, which permits the formation of non-uniformly sized niches. Unfortunately, the emphasis on efficiency masked this primary contribution; moreover, the test cases used were too easy to test the limits of nicher performance.

Perhaps the work closest to that herein is the dynamic niche sharing of Miller and Shaw (Miller & Shaw, 1996). In that study, they define a fixed number of dynamic niches with fixed radii and niche centers determined by a full population sort. For those individuals not in a niche regular fixed sharing is used. The authors claim certain advantages over other nichers on limited testing, but the primary weakness of the scheme is the use of fixed sharing outside the dynamic niches. One of the innovations in the present study will be to abandon fixed niche radii altogether and to think of the spacing parameter as a minimum distance between niche locations.

Others have attempted to design more competent nichers without sharing. Culberson's gene invariance GA (GIGA) (Lewchuk & Culberson, 1991; Culberson, 1993) and Mahfoud's deterministic crowding scheme (Mahfoud & Goldberg, 1992) improve on Cavacchio's preselection scheme to provide effective crowding pressure. Harik (Harik, 1995) develops a form of restricted tournament selection to do likewise. Harik and Mahfoud have demonstrated their schemes' ability to find optima in a massively multimodal test function, and Culberson's scheme is also likely to work on similarly difficult test functions. Fixed sharing has also been shown to work in such problems, but only through the addition of fairly severe fitness scaling (Goldberg, Deb, & Horn, 1992). One objective in this paper is to remove the need for artificial scaling mechanisms in sharing and permit adaptation of niche size and location within the population.

3 A COEVOLUTIONARY SOLUTION

In this section, we present our method of coevolutionary shared niching, which maintains and locates diverse niches by coevolving two populations, each with somewhat different objectives. We start by reviewing the primary problem to be solved. We continue by seeking a solution inspired by an economic model. We then apply the lessons of this model to the design of a more effective nicher.

3.1 The Primary Problem

A primary difficulty of implementing a general purpose niching mechanism is that the extent and location of a niche within a multimodal domain are not known a priori. Fixed niching radii can be made small enough to separate the closest niches, but in general this leads to unacceptably large population sizes and a large number of largely unimportant niches. The

other sharing-based techniques reviewed in the previous section have not been demonstrated to overcome this problem. Moreover, these techniques utilize serial algorithms such as sorts and clustering algorithms with the GA, thereby leaving bottlenecks that may prevent serious parallel computing in the future. In this study, we ask whether more evolutionarily inspired methods can locate non-uniformly spaced optima and still preserve the essential parallelism of an evolutionary algorithm.

3.2 An Answer from Monopolistic Competition

The answer to this question is in the affirmative and comes to us from a model in economics known as the model of *monopolistic competition* (Tullock, 1967). Suppose there is a community of *consumers* who live along a single road with a known density function. A group of independent and competing *businessmen* are contemplating the construction of a number of bagel shops. Once shops are located, customers will choose to go to the store that minimizes their bagel bill, including bagel and transport costs. With that knowledge, the bagel businessmen will space themselves along the road to capture maximum bagel profit. With uniform consumer density, equilibrium is reached when the stores are evenly spaced; with nonuniform density, businesses are more densely packed in more densely populated regions. Thus, because of the separate *interests* of consumers and businessmen, we see that there is a natural tendency for businesses to locate where they will do the customers the most good.

3.3 The Lesson for GAs: Separated Interests

This sounds like the ticket for obtaining properly spaced niches in a generalized niching scheme. Let us imagine the creation of two populations with separate interests: (1) a population of businessmen who serve (2) a population of customers. We ask whether it is possible to obtain well spaced multiple optima naturally without imposing separated clustering techniques or similar paraphernalia, and with careful thought about the roles of customers and businessmen together with careful design of their respective fitness functions we will see that it is.

In our scheme, the customer population may be thought of as the usual population of candidate solutions, and as such it is responsible for using the various genetic operators to search for more highly fit regions. Moreover, the customer niching scheme is not radically different from that with which we are already familiar. On the other hand, the coevolution of a businessman population is new, it receives the lion's share of our design effort, but at the outset we can say that the businessmen will "locate" themselves not geographically, but rather in solution space. That is, the businessmen will be coded with the same strings as the customers. In addition, because the businessmen are trying to maximize some "profit" measure, by analogy with the model of monopolistic competition they will "locate" themselves to obtain largest payoff, thereby placing niches appropriately in response to the landscape. With this basic concept, we turn to the details of designing appropriate fitness functions for customers and businessmen and coordinating the activity of the two populations.

3.4 Customer Fitness

Before defining the customer fitness transformation, we need a number of definitions. Consider a customer population of ℓ-bit binary strings C of size n_c, a businessman population of ℓ-bit binary strings B of size n_b, and define the distance between a customer c and a

businessman b to be the Hamming distance of string c and b, denoted as $|c - b|$. We say customer c is *served by* or *belongs to* business b if b is the nearest businessman to c: that is, $\min_{b \in B} |c - b|$. The set of customers a businessman b serves at generation t is called the *customer set* of b, denoted as $C_{b,t}$. The number of customers a businessman b serves at generation t denoted as $m_{b,t}$ is the cardinality of $C_{b,t}$ at generation t, $\|C_{b,t}\|$. Additionally, customers and businessmen have raw fitness values, which simply come from the standard fitness evaluation in the usual manner. With these preliminaries we may turn to defining a sharing-like transformation that achieves the desired objectives.

For the customer population, the fitness modification is like that of standard fitness sharing as described in an earlier section in that the shared fitness is calculated as the raw fitness divided by the niche count. In CSN, however, the niche count is not taken as the sum of the sharing function values. Instead, we simply say that an individual shares with all the customers served by his businessman:

$$f'(c) = \left. \frac{f(c)}{m_{b,t}} \right|_{c \in C_b} \tag{4}$$

In many respects this is much simpler and more natural an interpretation of the niche count and shared fitness than that used in the original scheme.

3.5 Businessman Fitness

Selecting the fitness transformation for the customers is fairly straightforward because it is a simple modification of the existing sharing scheme. We simply have individuals share within the service area of their businessman. On the other hand, many possibilities exist for choosing the businessman fitness transformation, so we must be clear regarding what the businessmen do, and how we would like them to be distributed.

The businessmen locate the niches, and together with the minimum distance rule, they determine the extent and distribution of the niches. Where would we like the businessmen to be located? To some extent it depends upon the problem, but in this paper we choose to place businessmen at the highest niches, subject to a minimum distance requirement between businessmen. This combination will give us most of what is desirable in a practical nicher. Should we have many equally fit peaks, the scheme will place its businessmen at or among them. Should we have unequally fit peaks, the scheme will place niches uniformly to cover the highest peaks at a spacing determined by the minimum distance d_{min}; thereafter it will allocate businessmen (and niches) to lower fitness peaks.

One rule that will drive the combined scheme toward this fairly useful type of behavior is as follows. Simply define the transformed fitness of the businessman, $\phi(b)$, as the sum of the raw fitness values of his customers:

$$\phi(b) = \sum_{c \in C_{b,t}} f(c) \tag{5}$$

To see that this works, consider the fitness value of businessman i. Recognizing that the businessman is simply the sum of the raw fitnesses of his customers, we may write immediately that the businessman fitness is simply the niche count of his service area times the average raw fitness of his customers:

$$\phi_i = m_i \bar{f}_i \tag{6}$$

Now, consider that an evolutionary scheme will tend to prefer businessman i to businessman j when the fitness of i exceeds that of j, that is when $\phi_i > \phi_j$. Substituting equation 6, this occurs when

$$\bar{f}_i > \frac{m_j}{m_i}\bar{f}_j \tag{7}$$

For any reasonable customer nicher (for example, the one proposed herein), the number of individuals allocated to a low peak j should be less than that allocated to a higher one: $m_i > m_j$. Thus, we have shown that the businessman fitness transformation proposed will prefer higher fit peaks always. Moreover, this immediately suggests the need to impose a minimum distance between businessman as a side constraint. Without such a constraint, the scheme will always go for only the highest peaks or a delta neighborhood around them. We will demonstrate this behavior in a moment, but first we must describe the overall algorithm.

3.6 Simple CSN

In our first version of CSN, called simple CSN, we use a fairly standard simple selectorecombinative GA for the customer population and a selectomutational GA for the businessman population. The rationale for this decision initially was that the choice of the businessmen is less sensitive than the choice of the customers. After all, the underlying customer scheme is still basically a sharing-based nicher and it should cluster individuals no worse than ordinary sharing, even with a fairly poor distribution of businessmen. Of course, all bets are off if the businessmen population itself becomes insufficiently diverse, so we examine the details of the customer and businessman processing in more detail.

First, imagine the processing of the customer population given a fixed set of businessmen. All customers are compared to all businessmen and each customer is assigned to the closest businessman. The businessmen customer counts are used as niche counts, and each individual's shared fitness is calculated in the usual manner. Thereafter proportionate selection and genetic operators are used to compute a new customer generation. In this study, stochastic universal selection (Baker, 1987) and single-point crossover are used in the customer calculations throughout.

Now consider the processing of the businessman population. In simple CSN, each individual businessman is chosen in turn and a single mutation is selected randomly (without replacement) from among the ℓ mutation sites. The resulting individual is checked to see (1) if it is at least d_{min} from the other businessmen, and (2) whether it is an improvement over the original businessman. If these conditions are met, the candidate businessman replaces the original businessman. If not, the original mutation is restored and another mutation site is selected. This process is permitted to proceed a total of n_{limit} times ($n_{limit} \leq l$). If no candidate is found that meets criterion, the original businessman is retained and the process is repeated for the remaining businessmen in the population.

In this way, the businessman population is hoped to evolve toward the top n_b solutions that are at least d_{min} from one another. The replacement of parent by children imposes a niching-like effect that was first mentioned by Cavicchio (Cavicchio, 1970) and called preselection. The requirement of improvement is akin to the elitist recombination scheme discussed elsewhere (Thierens & Goldberg, 1994). We posit that this combination maintains sufficient diversity in the businessman population to promote adaptive niching in the customer population. In the next section, this hypothesis is checked by testing the ability of a population to achieve equilibrium from a badly unbalanced initial state.

3.7 Test of an Unbalanced Population

To test whether this combined mechanism is able to attain equilibrium in both the customer and businessman populations, we perform a test of unbalanced populations as follows. We define function F1 as

$$F1(x) = \sin^6(5\pi x)x \tag{8}$$

over the interval [0,0.4], as shown in figure 1. Over this interval the function has 2 peaks at approximately x = 0.1 and 0.3. A 30-bit string is decoded as a unsigned binary integer and mapped linearly to the interval [0.0,0.4]. To test the mechanism in the extreme, we initialize the population with 8 businessmen and 100 customers where we put all but one of the businessmen and customers on the second peak. Average results from 5 runs are shown in figure 1, showing generational time histories of the number of customers m_1^c and businessmen m_1^b at the first peak. As we see, the customer population quickly evens the score with the businessmen following closely behind. As expected the combination of elitism and like replacing like always gives a satisfactory return to equilibrium. Next, we test the mechanism in two relatively simple functions that are commonly used baseline tests in niching studies.

4 SIMPLE CSN ON EASY PROBLEMS

In this section, we present the results of running simple CSN against two relatively easy test functions, Goldberg and Richardson's (Goldberg & Richardson, 1987) test functions F1 and F2. These test functions have been used in many studies of niching and are a starting place for testing nichers and comparing them to earlier works. In the remainder of this section, those functions are described, and the results of the runs are presented.

4.1 Test Functions Employed

F1 is defined as before:

$$F1(x) = \sin^6(5\pi x) \tag{9}$$

except the range is extended to $0 \leq x \leq 1$. Maxima are located approximately at 0.1, 0.3, 0.5, 0.7, and 0.9. All maxima have a function value of 1.0.

F2 is defined as follows:

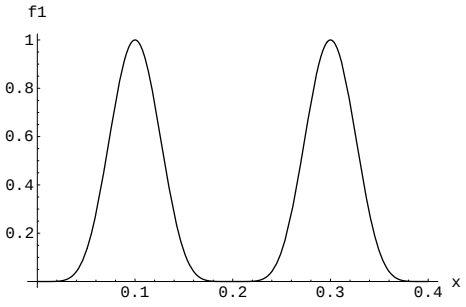
$$F2(x) = e^{-2(\ln 2)(\frac{x-0.1}{0.8})^2} \sin^6(5\pi x) \tag{10}$$

It is a oscillating function, having five equally spaced peaks of varying height in the interval [0,1], the peaks in descending order by peak function value are located at approximate x values 0.1, 0.3, 0.5, 0.7, and 0.9. Maxima have rounded values of 1.0, 0.917, 0.707, 0.459 and 0.250, respectively.

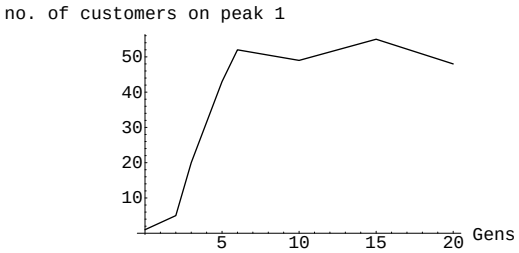
4.2 Experimental Setup

All GA variations use the same encoding for the F1 and F2 functions. Each genotype is a 30-bit string decoded as an unsigned binary integer. The genotype is then converted into a phenotype on the range $[0, 1]$ by dividing the resulting integer by $2^{30} - 1$.

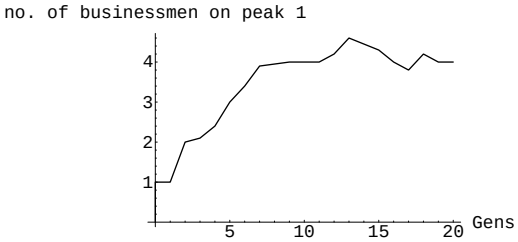
The customer population is initialized and fixed at a population size of 300. One-point crossover, CSN niching, and SUS selection are used to generate nonoverlapping populations



(a) $F1$

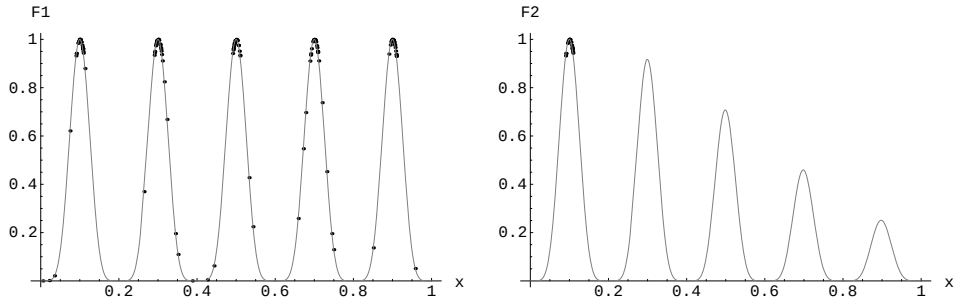


(b) m_1^c



(c) m_1^b

Figure 1: In the test of an unbalanced population, all but one individual in both the customer ($n_c = 100$) and businessman population ($n_c = 8$) are placed at the second peak. Despite this extreme unbalance the population is able to right itself consistently and repeatedly.



(a) $F1$ (b) $F2$

Figure 2: The distribution of customers is shown from a representative run of CSN against functions $F1$ and $F2$ at generation 100. As we expect, on $F1$ businessmen and customers array themselves uniformly on the 5 peaks, whereas on $F2$ all customers and businessmen are attracted to the highest peak.

as discussed earlier. Crossover probability is held at 1.0, and there is no mutation within the customer population.

The businessman population is processed with single-bit mutations and parental replacement conditioned on improvement and satisfaction of the d_{min} criterion as detailed before. The minimum distance parameter used initially was $d_{min} = 0$.

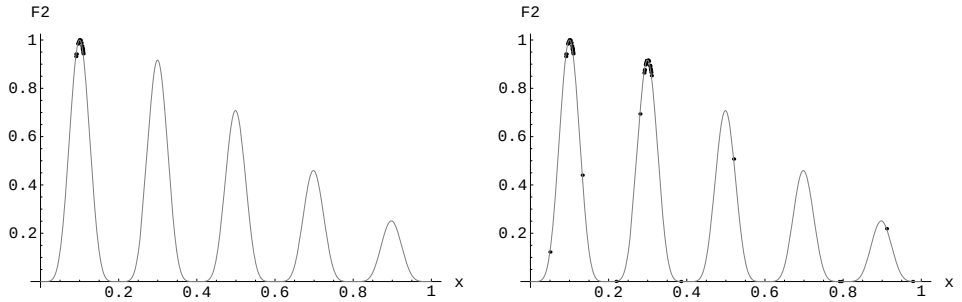
4.3 Results

Running simple CSN on $F1$ and $F2$, the results are shown in figure 2. As expected CSN maintains all five peaks in function $F1$. Moreover, examining the distribution of the 20 businessmen, we see roughly equal numbers of businessmen on each of the peaks.

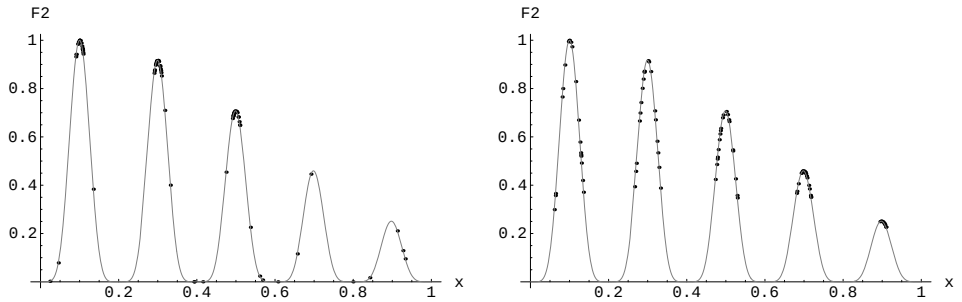
On F2, however, only individuals of the top peak are represented, and this would be both annoying and something of a mystery had we not analyzed the performance of the business population objective function earlier. Recall that businessmen are evaluated by the sum of raw fitness values in their service areas. This causes businessmen to drive toward the highest peaks. When peaks differ in elevation, there is room on the highest peak for many businessmen to coexist and service many customers each. This is precisely what happens in the runs on F2 with $d_{min} = 0$ specified.

Rerunning simple CSN for different values of d_{min}, we obtain the results shown in figure 3. As before, when $d_{min} = 0$, all businessmen converge to the first peak. When we raise d_{min} to 0.005, the second peak shows up. The third peak appears stably in the population when d_{min} is increased to 0.01. The customer population holds all five peaks when d_{min} equals 0.02.

An exact analysis of these results is beyond the scope of this study, but an approximate analysis can be obtained by thinking of level sets. At a given function cutoff value, f_{cut}, a line can be drawn across the function graph and the intervals with points lying at $f \geq f_{cut}$



(a) $d_{min} = 0$ (b) $d_{min} = 0.005$



(c) $d_{min} = 0.01$ (d) $d_{min} = 0.02$

Figure 3: The customer distribution is shown at generation 100 for representative runs of function $F2$ on simple CSN with varying settings of d_{min}. As expected, as d_{min} is increased more niches are allocated to lower fitness peaks.

can be identified. If we total the interval lengths and call the result L, we recognizing that with n_b businessmen that we should have something like $n_b \times d_{min} = L$. Checking this result in figure 3d, we draw a level curve just below the points on the last peak. Summing the length of the intervals yields $L = 0.36$, which is somewhat less than the predicted value of $20 \times 0.02 = 0.4$. Despite this inaccuracy, the approximate equation can be a useful tool in run sizing and interpretation of results.

5 HARD PROBLEMS AND IMPRINT CSN

In the previous section, simple CSN was shown to give careful control over the location and extent of niches through the coevolution of two populations and the introduction of a minimum distance parameter; however, the test functions used were fairly simple problems, and the procedure may fail if one of its links are weak. Recalling that a selectomutational GA was used to move the businessmen around, we might suspect that this could limit the algorithm's performance in problems where peaks are not easily hillclimbed. Therefore, in this section we test the simple CSN scheme against a known difficult massively multimodal function used as a test function elsewhere (Goldberg, Deb, & Horn, 1992). It proves to be inadequate to the task, and we propose an operator called *imprint* that carries good individuals over from the customer population to the businessman population, thereby tapping into the power of selection and recombination to overcome a hard problem.

5.1 A Worthy Challenger

A family of difficult massively multimodal test functions was developed by Deb, Horn, and Goldberg (Deb, Horn, & Goldberg, 1993) and one family member was used by Goldberg, Deb, and Horn (Goldberg, Deb, & Horn, 1992) to test sharing augmented by a fitness scaling mechanism. Defining the unitation variable u as $u(s) = \sum_{i=1}^{6} s_i$ over the bit variables s_i, a six-bit bipolar deceptive subfunction is defined below and depicted in figure 4:

$$f(s) = \begin{cases} 1 & \text{if } u(s) = 0, 6 \\ 0 & \text{if } u(s) = 1, 5 \\ 0.360384 & \text{if } u(s) = 2, 4 \\ 0.640576 & \text{if } u(s) = 3 \end{cases} \tag{11}$$

Further detail on the construction of the function is available elsewhere (Deb, Horn, & Goldberg, 1993). It is *deceptive* in the sense that low-order schemas lead to solutions maximally far from the global optima. In this paper as in Goldberg, Deb, and Horn (Goldberg, Deb, & Horn, 1992), five copies of the functions are summed together over successive sets of six bits giving a 30-bit function. A simple counting argument shows that each subfunction has 22 optima, resulting in a total of $22^5 \approx 5.15(10^6)$ optima of which only $2^5 = 32$ are global.

5.2 Experimental Setup and Results with Simple CSN

We use the simple CSN algorithm as described earlier, but we increase the businessman population to size $n_b = 35$ and the customer population to size $n_c = 2000$. The d_{min} parameter is set to 0 on all massively multimodal function runs. A tight coding has been assumed in that the six bits of each of the subfunctions are placed one right after another. This is tantamount to assuming that linkage information is known, not necessary, or can be

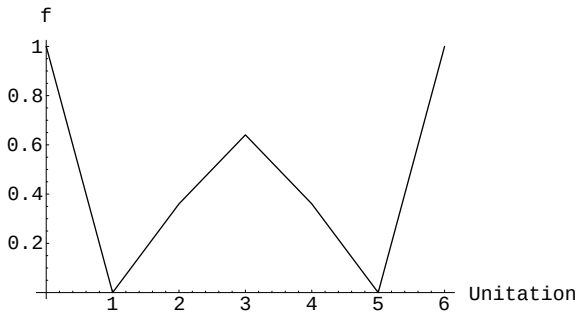


Figure 4: Five nonoverlapping copies of a six-bit bipolar deceptive function are summed to form a 30-bit massively multimodal test function.

discovered through the use of a non-traditional GA such as the fast messy GA (Goldberg, Deb, Kargupta, & Harik, 1993), the linkage learning GA (Harik, 1997), or the gene expression messy GA (Kargupta, 1996) Further description of these issues is beyond the scope of this paper. Suffice it to say, that our experimental design is a fair means of isolating the performance of the CSN mechanism, assuming the existence of a GA that is capable of rapidly finding single global solutions in hard problems on its own.

Figure 5 shows the result of simple CSN running against the massively multimodal function. The horizontal axis is the number of generations and the vertical axis is the number of peaks found. The graph shows that 17.6 out of 32 global optima are found, as averaged over five runs.

As expected, simple CSN fails largely because the businessman population is unable to hillclimb the difficult massively multimodal function reliably. On the other hand, the customer population contains all the building blocks necessary to generate the other optimal solutions, and crossover generates those solutions with high probability. This leads us to wonder whether there is any way to use the effectiveness of crossover in the customer population to influence the direction of the businessman population.

5.3 Imprint

The failure of the businessman population is largely caused by the inadequacy of the businessman search strategy. Waiting times for deceptive optima are unacceptably long under selection and mutation alone (Mühlenbein, 1992), and to some extent, it appears that we are tying our hands behind our back here. The recombinative customer population is fully capable of identifying the best building blocks even with only partially effective niching, and so we should expect relatively short times to discovery of good candidate solutions. Why can't we use that capability to extract and transfer candidates to the businessman population.

This is precisely the idea behind an operation we call *imprint*. Simply stated, a candidate

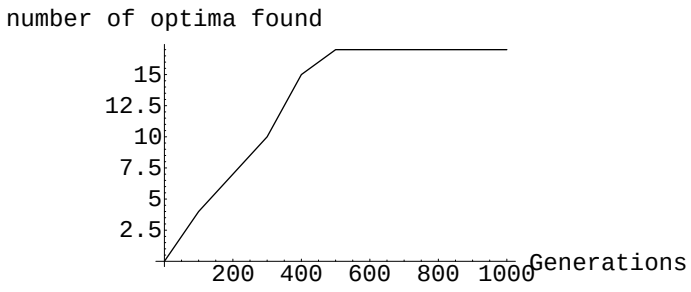


Figure 5: Simple CSN with mutation is unable to find all 32 peaks, as shown in the graph of the number of optima found in the customer population as a function of generations. Results are averaged over five runs.

businessman is chosen from among the best of the customer population. If this candidate meets all necessary requirements it is inserted into the businessman population.

5.4 How to Imprint

Instead of creating one businessman afresh each time, we would like to carry over better ones from the customer population, but we would like those individuals to meet the d_{min} requirement. This could be done using a sort as in Miller and Shaw (Miller & Shaw, 1996), but for efficiency reasons and for possible future parallelization we choose to do this as follows. For each businessman, we choose a random customer and check to see if it is an improvement over the businessman. If it is we check to see if it is d_{min} distant from the other businessmen. If so, the customer is copied into the businessman population, replacing the original businessman. If either of these tests fails, another customer is chosen randomly and the testing process continues up to a total of n_{limit} customers. If n_{limit} is chosen to be a multiple of the number of businessmen n_b, we should expect to see one or more individuals from a given businessman's niche.

Comparing the foregoing description to simple CSN, we note that we have simply replaced the mutation process with the imprint transfer mechanism. Next we test CSN with the imprint mechanism and compare it to simple CSN.

5.5 CSN with Imprint on the Massively Multimodal Function

We rerun the massively multimodal experiment using the same problem parameters, except that n_{limit} is set to $n_b = 35$. As shown in figure 6, the number of optima found by CSN with

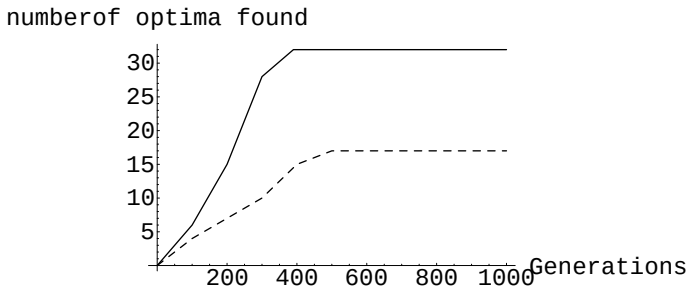


Figure 6: Simple CSN with mutation vs. CSN with imprint, against massively multimodal problem. CSN with imprint finds all 32 peaks.

imprint is 32 over the five runs. This was the desired result, and it demonstrates the ability of CSN with imprint to sort through complex landscape with many misleading optima.

In the next section, we consider a number of extensions of this work.

6 EXTENSIONS

This paper has proposed, designed, analyzed and tested a means of adaptive niching using coevolutionary sharing. The results suggest that the scheme has effective distribution and search capabilities, particularly when imprint is used to transfer the best solutions from the customer population to the businessman population. With proof of concept in the bag, a number of tests and extensions are recommended:

1. Test in badly non-uniform domains.
2. Consider certain efficiency improvements.
3. Consider in concert with linkage-friendly GAs.
4. Apply in multiobjective and real world problems.
5. Compare CSN to other effective niching scheme.

The reminder of this section considers each of these briefly.

This paper has considered the basic distributional and search properties of the CSN scheme through analysis and experimentation, but one of the advertised virtues of the scheme has not been put to a sufficiently difficult test. Specifically, coevolutionary shared niching should work exceedingly well in environments with badly non-uniform optima. Tests should be designed to demonstrate this point, but there is some evidence from the experiments on $F2$ that support the ability of the algorithm to find many niches in a small region

or a more distant region depending upon the d_{min} setting. We believe that other functions with badly spaced optima should yield to these controls equally well.

The mechanism for imprint transfer should be improved along straightforward lines. At present, *random* individuals are chosen from the customer population for possible replacement of a businessman. Since a candidate must satisfy d_{min} and improvement requirements it would be useful to choose from a pool of candidate who are more likely to demonstrate improvement. Specifically, we propose flagging those customers who have fitness greater than the lowest fitness of the businessman population. Thereafter, imprint candidates are chosen from the flagged pool of customers instead of from the entire population. In this way, improvement to the business population should come more quickly. The factor n_{limit} should still be set to some multiple of n_b to help assure that at least one individual comes from the relevant niche. It might also be possible to gain additional efficiency by further restricting the search for a new businessman to those customers within a businessman's customer set. In this way, best candidates in the neighborhood would already be flagged and much random comparison could be eliminated.

Recall that competent GAs—GAs that solve hard problems, quickly, reliably, and accurately—were modeled in this study by using prior-coded tight linkage and single-point crossover. As mentioned earlier, there are now a number of GAs (Kargupta, 1996; Harik, 1997; Goldberg, Deb, Kargupta, & Harik, 1993) that effectively adapt the linkage without prior knowledge and these schemes should be tested in concert with imprint CSN.

Niching is becoming an important part of real applications and CSN should be "battle-tested" against real-world problems. A particularly important application area is in multiobjective optimization and CSN should be tested as part of one of the niching-sharing scheme currently in use (Srinivas & Deb, 1994; Fonseca & Fleming, 1993).

Additionally, a number of schemes have been put forward as the "best" niche, but their capability to adapt to varying niche location and extent, population sizing, and operational and implementation requirement have not been carefully compared. This work is beyond the scope of our proof-of-concept study, but these issues deserve theoretical and empirical investigation. At a minimum, coevolutionary shared niching, deterministic crowding, restricted tournament selection, dynamic sharing, and Yin and Germay's scheme should be examined along these lines.

7 CONCLUSIONS

This paper has designed, analyzed, implemented, and tested an adaptive niching scheme called coevolutionary shared niching in which two populations with separated interests locate stable subpopulations of the best solutions almost regardless of the solution spacing, extent, and modality.

Although additional testing is necessary, and has been recommended, this method promises to join an important and growing group of niching algorithms that promote the preservation of useful diversity in complex landscapes.

References

Baker, J. E. (1987). Reducing bias and ineffiency in the selection algorithm. *Proceedings of the Second International Conference on Genetic Algorithms*, 14–21.

Cavicchio, Jr., D. J. (1970). *Adaptive search using simulated evolution.* Unpublished doctoral dissertation, University of Michigan, Ann Arbor, MI. (University Microfilms No. 25-0199).

Culberson, J. C. (1993). Crossover versus mutation: Fueling the debate: TGA versus GIGA. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 632.

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., Horn, J., & Goldberg, D. E. (1993). Multimodal deceptive functions. *Complex Systems*, *7*(2), 131–153.

Fonseca, C. M., & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 416–423.

Goldberg, D. E., Deb, K., & Horn, J. (1992). Massive multimodality, deception, and genetic algorithms. *Parallel Problem Solving from Nature, 2*, 37–46.

Goldberg, D. E., Deb, K., Kargupta, H., & Harik, G. (1993). Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 56–64.

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.

Harik, G. R. (1995). Finding multimodal solutions using restricted tournament selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 24–31.

Harik, G. R. (1997). *Learning linkage to efficiently solve problems of bounded difficulty using genetic algorithms.* Ph.d. thesis., University of Michigan Ann Arbor. (Also IlliGAL Report No. 97005).

Kargupta, H. (1996). The gene expression messy genetic algorithm. *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation. Nagoya University, Japan.*, 631–636.

Lewchuk, M., & Culberson, J. C. (1991). *Genetic invariance: A new approach to genetic algorithms* (Tech. Rep. No. TR 91-03). Edmonton, Canada: University of Alberta, Department of Computing Science.

Mahfoud, S. W., & Goldberg, D. E. (1992). A genetic algorithm for parallel simulated annealing. *Parallel Problem Solving from Nature, 2*, 301–310.

Miller, B., & Shaw (1996). Genetic algorithms with dynamic niche sharing for multimodal function optimization. pp. 786–791.

Mühlenbein, H. (1992). How genetic algorithms really work: I.Mutation and Hillclimbing. *Parallel Problem Solving from Nature, 2*, 15–25.

Oei, C. K., Goldberg, D. E., & Chang, S.-J. (1991). *Tournament selection, niching, and the preservation of diversity* (IlliGAL Report No. 91011). Urbana, IL: University of Illinois at Urbana-Champaign.

Srinivas, M., & Deb, K. (1994). Multiobjective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation*, *2*(3), 217–249.

Thierens, D., & Goldberg, D. E. (1994). Elitist recombination: An integrated selection recombination GA. *Proceedings of the First IEEE Conference on Evolutionary Computation*, 508–512.

Tullock, G. (1967). *Toward a mathematics of politics.* Ann Arbor, MI: The University of Michigan Press.

Yin, X., & Germay, N. (1993). Improving genetic algorithms with sharing through cluster analysis. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 100.