Finding Multiple Solutions In Problems Of Bounded Difficulty

Georges Harik
Department of Computer Science
University of Michigan
Department of General Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801-2996

IlliGAL Report No. 94002
May 1994

Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
117 Transportation Building
104 South Mathews Avenue
Urbana, IL 61801-2996

Finding multiple solutions in problems of bounded difficulty

G. Harik

29 March 1994

1 Introduction

There is, in the realm of optimization, quite often a need to identify several good solutions for a problem, as opposed to only one optimal solution. This need arises from several sources. Often, an optimization problem is a simplification of a real world problem that in part requires human or inquantifiable judgment. In these types of problems, an optimizer needs to suggest various possible alternatives that can later be judged by a human expert. In other situations, a better understanding of a search space is desired in terms of the location of its various optima.

Genetic Algorithms (GAs) can be and are being used as general optimization techniques for a number of problems in various domains. They often produce good solutions to these problems quickly, while requiring little or no problem specific information. GAs are thus useful in domains that are not well understood, as well as domains in which using a complete model of the underlying problem is computationally infeasible. A GA capable of finding multiple solutions in real world optimization problems would thus be a valuable asset.

A multimodal real world setting presents two challenges to the simple GA. First, real world optimization problems are often quite large in that they involve many variables. Recent evidence indicates that the success that simple GAs have achieved thus far does not scale well to the handling of larger problems. Second, a GA operating in a multimodal setting needs to locate several different good solutions to any given problem. In a purely mechanistic sense, the simple GA always converges to a single solution and thus is not appropriate for multimodal optimization.

These two challenges can and to some degree have been overcome. The messy GA algorithm (Goldberg, Deb, Kargupta & Harik, 1993) has shown significant promise in overcoming the simple GA's scaling limitations. Multimodal techniques such as crowding (De Jong, 1975) and fitness-sharing have long been available for use with the simple GA. However, the work needed to extend the simple GA to work in a multimodal real world setting has not yet been completed. Two issues remain to be resolved in this work. First, the multimodal techniques currently available have not been explored sufficiently as to their interaction with the GA's search paradigm. Second, the messy GA has yet to be combined with a capable multimodal technique for the creation of a multimodal messy GA.

The purpose of this document is to suggest research into the incorporation of the messy GA, into a capable multimodal GA technique for the purpose of developing a robust multimodal GA that will be effective working on real world problems. The development of such an algorithm will accrue the following final and intermediary benefits:

- A set of conditions representing the minimal capabilities that a multimodal GA must posses in order to be considered able to solve real world optimization problems. This set of conditions could then be used to judge any possible multimodal GA.
- The development of a multimodal GA capable of solving real world optimization problems, an invaluable tool in both the realms of artificial intelligence and engineering design.
- The development or choice of a multimodal technique whose operation does not hinder the multimodal GA's search engine.
- A better understanding of the effects of one or more multimodal techniques on the GA's search capability.

1.1 A road map to this document

The first section of this document explored in general terms, the aims of the work being proposed herein. The second section of this document seeks to substantiate the problems hinted at in this document's introduction by reviewing past research. Before presenting such evidence, this section presents a necessary introduction to GAs and the building block propagation theory of their operation. Following this general introduction is a definition of bounded difficulty within the GA's framework. This presentation continues with a look at Thierens & Goldberg's work. The content of this work highlights the fact that the simple GA does not scale well with regards to problem size, when working on problems of bounded difficulty. Two possible solutions to this problem, the inversion operator and the messy GA are then explored. The messy GA approach is seen to be the currently more promising candidate for a solution to this problem.

This section also presents the possible techniques available for use with the GA in finding multiple solutions to a given problem. Of the multimodal techniques available, fitness-sharing is highlighted as the best one to use in a multimodal GA. This section also illustrates the fact that little work has been done investigating the effects of fitness- sharing on the search capabilities of the GA.

The third section of this document begins where the other researchers leave off by defining the list of tasks that need to be completed in order to produce a multimodal GA capable of real world optimization. In tandem, these tasks should lead towards the development or choice of a multimodal GA technique, and the incorporation of that technique into a powerful search engine, the messy GA. This section of the document also reviews the work already begun in fulfillment of these tasks. This review includes the introduction of a new multimodal technique, restricted tournament selection (RTS), as well as preliminary results from the integration of this technique with the messy GA.

The fourth section of this document specifies, as accurately as is presently possible, the actions that are to be taken for the completion of the list of tasks presented in the third section. The fifth section of this document includes a summary of the material presented herein as well as conclusions that can be drawn from this document.

2 GA search and multimodality

Genetic algorithms are evolutionary techniques that work on populations of individuals. Individuals in a GA's population mate and reproduce as in nature. Different individuals are assigned reproduction rates proportional to their fitnesses, which are measures of how well adapted those individuals are to their environment. Individuals in a GA are usually represented by their genomes

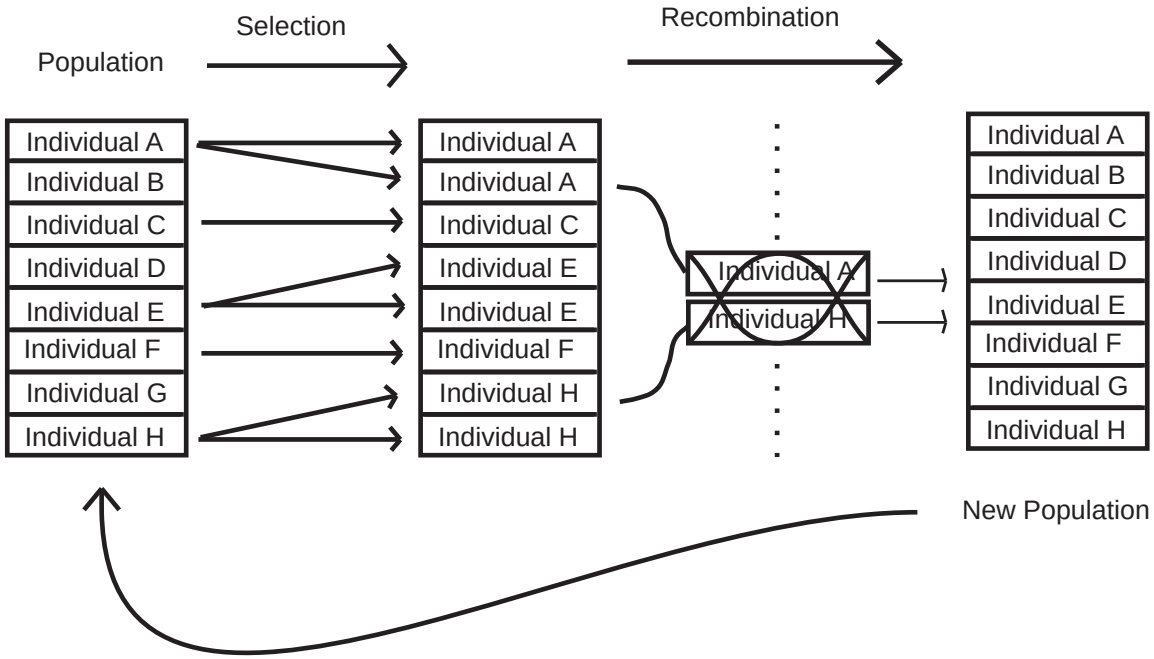


Figure 1: A GA at work

or genetic material. The GA then represents mating as a recombination of genetic material between two individuals. Additionally, other evolutionary elements such as mutation are sometimes thrown into this fold. Figure 1 shows a GA in operation.

The *population* of a GA is a multi-set P whose members are referred to as *individuals*. The cardinality of a GA's population is usually fixed, and referred to as the population's *size*. All individuals in the GA's population belong to a larger set of individuals S whose members include all the possible types of individuals within any particular framework. This set is sometimes referred to as the GA's *search space* since the GA essentially searches for optimal populations of individuals chosen from this set. The *fitness* of an individual is usually a real valued assignment $f : S \to R$. The action of *mating* is an assignment $c : S \times S \to S$. The action of *mutation* is an assignment $mu : S \to S$. The action of reproduction or *selection* is a map from a particular population P0 to another population P1 of the same size. This action usually results in an increase of the population's average fitness and is supposed to mirror the effects of natural selection on a population.

One major use for the Genetic Algorithm is as an optimization tool. The following mapping can be applied from real world optimization problems to GA components. Solutions to optimization problems are coded as the genetic material for individuals in a population. Individuals representing better solutions are awarded higher fitnesses, thus enabling them to reproduce more or faster than other individuals. The GA's incremental selection pressure, due to differing reproductive rates of individuals, allows the better solutions to take over the GA's population. During this process, pairs of solutions recombine through mating to form new solutions that are then reinserted into the population. The better solutions slowly begin contributing more genetic material to the formation of newer solutions. The progress of the GA on this problem then mirrors the progress of natural evolution on the genetic components of populations. Natural evolution induces the formation of individuals that are well-adapted to their environments. In similar vein, GAs work by evolving solutions that are well-adapted to their fitness-landscape. Since this fitness landscape usually offers

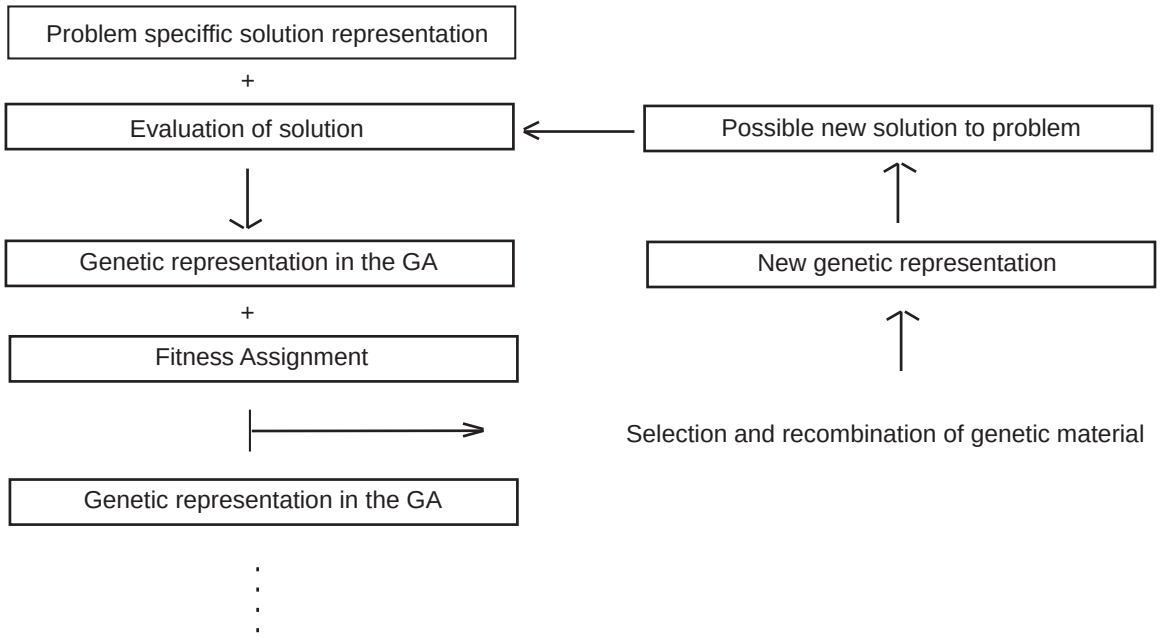


Figure 2: A GA mapping for any optimization problem

an indication of the 'goodness' of various solutions, GAs are often able to present good solutions to the problems presented to them. Figure 2 shows how any optimization problem can be stated in GA terms.

In a typical GA application, solutions to a certain problem, the GA's individuals, are coded as bit strings of a certain predetermined length. The bit string corresponding to any particular individual is that individual's *genetic material*. This predetermined number of bits is usually referred to as the problem's *length* or *size*. The search space of this GA is then seen to be the set of all strings of the proper length over the binary alphabet. That is $S = \{0,1\}^{length}$. An individual's fitness assignment, supplied by the end user, is usually a numerical measure of how well that individual solves the given problem. Under a widely used form of selection, *fitness-proportionate* selection, an individual's reproduction rate is directly proportional to its fitness score with the constant of proportionality being determined by the restriction that the population's size remain fixed. Mating between two individuals is embodied in a *crossover* operator that combines their attributes to form a new individual. The two individuals contributing the input of the crossover operator are referred to as the *parents* of the crossover. The resulting individual in a crossover operation is referred to as the *child* of the crossover. A commonly used crossover operator, *one-point crossover* works by selecting a random point along the parent bit strings, and then copying all the bits to the left of that point from one parent, and the bits to the right of that point from the other parent, to create the child. Mutation is usually represented as an operation in which one of the bits on a string is flipped or has its value changed. Figure 3 shows the application of one point crossover and mutation in a GA.

2.1 How do GA optimizers work?

In a GA, the fitter individuals slowly take over the population. During this time, new solutions are formed by recombining these increasingly fit individuals. The combination of fitness assignment

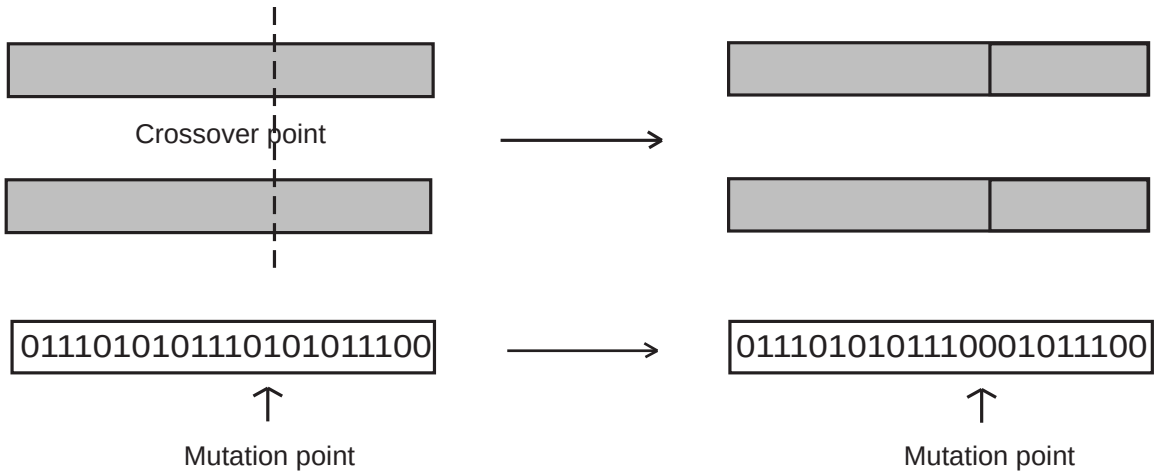


Figure 3: One point crossover and mutation

and reproduction bears a likeness to a stochastic parallel version of the test phase of a generate-test algorithm. In turn, the mating or crossover operator is the analogue of the generate phase of a generate-test algorithm. Having an entire population of individuals to work from allows the generate-test algorithm to stray from single-minded hill climbing. Furthermore, empirical and theoretical results indicate that the generate portion of this algorithm is not blind and in fact superior to the incremental changing of present solutions to form new solutions.

The crossover operator that a typical GA uses exploits information contained within substructures of the solutions the GA has evaluated thus far. The crossover operator serves a dual purpose in this scheme. It's first function is to continuously test the correlation of a collection of bits or *attributes*, with the assignment of a high fitness score. When the crossover operator preserves a group of parental bits in a child, it is implicitly testing the hypothesis that some subgroup of these bits is correlated with a high fitness score. This is done by allowing those bits to mix with a different set of assignments for the remaining bits, and then allowing that new individual to be acted on by selection. The second purpose of the crossover operator is to instantiate these highly fit subgroups of bits in the same individual. When the crossover operator combines a subset of one parent's bits with a subset of another parent's bits, the good collections of bits from both parents may then become synthesized in the child resulting from that crossover. The generate phase of this algorithm thus works under the assumption that certain groupings of attributes in individuals are somehow causally linked with the assignment of high fitness scores. A grouping of attributes is called a *building block*. Figure 4 shows how good building blocks can propagate and instantiate in one individual under the actions of selection and crossover.

While the crossover operator's job is to test for good building blocks for the problem, it is the selection phase's role to ensure that the seemingly better building blocks continually increase their representation in the population. Under fitness-proportionate selection, this part of a GA's operation is well understood and explained by Holland's schema theorem [1].

The discussion above identifies the two main steps in the GA's operation as:

- Determination and propagation of the good building blocks.

[1]The interested reader is referred to Holland (1973)

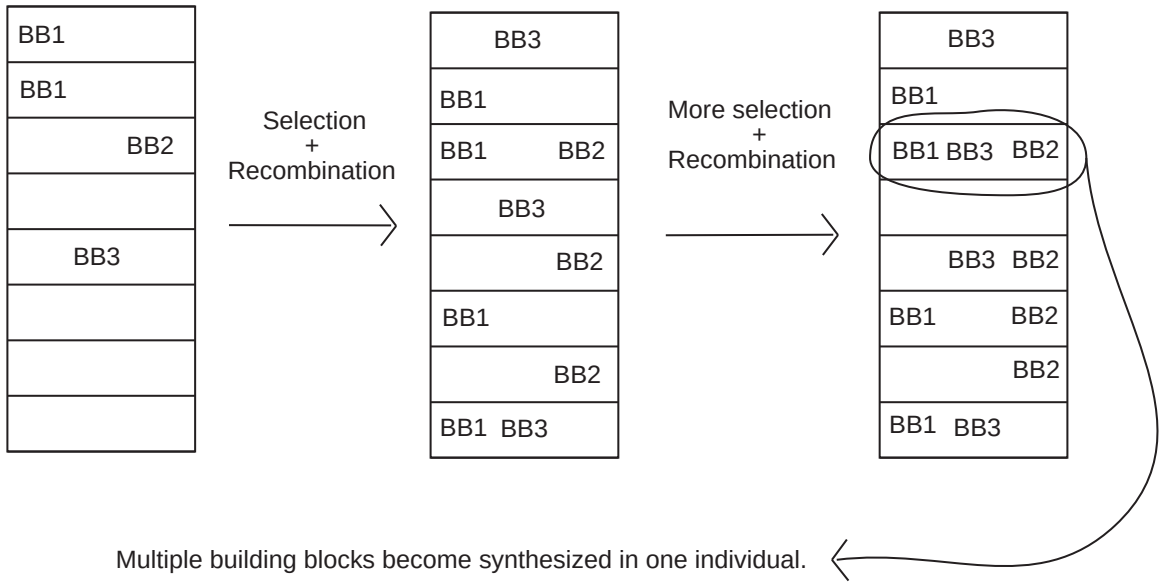


Figure 4: Building block propagation and synthesis

- Recombination of the good building blocks into one solution.

Currently, this identification is the most widely held view of how GAs work. This identification is referred to in this document as the *building block propagation* theory. In a GA application, these two steps do not take place separately. The processes that implement these two steps are applied concurrently. The interaction between these two steps is not simple. Often, care must be taken so as not to allow one of the steps to dominate the other. Following is a definition of bounded difficulty within the framework of this theory.

2.2 Bounded difficulty and deception in GAs

Under the assumption that the building block propagation theory is essentially correct, it is clear that for an optimization problem to be difficult for a GA to solve, at least one of the following must occur:

- The building blocks forming the optimal solution must be difficult to propagate.
- The building blocks forming the optimal solution must be difficult to recombine.

Two building block concept definitions become necessary when considering the notion of bounded difficulty in GAs. The *order* of a building block is the number of bits that the building block is composed of. In a problem of length n, building blocks can be represented as strings of length n in the three-letter alphabet {0,1,*}. An individual is an instance of a building block if its genetic material matches that building block in each position, the letter * matching both 0s and 1s. In that representation, the order of a building block equals the number of *fixed elements* (0s or 1s) in the string representation of that building block. This string representation facilitates the definition of another building block concept. The *defining length* of a building block is the distance between its outermost fixed elements. Building blocks with small defining lengths are said to be *tightly linked*.

Thus the building block **0*1* is of order two and has a defining length of three. A problem whose good building blocks are tightly linked is said to have *tight linkage*.

If, to form an optimal solution in a particular problem, a necessary building block is of a large order, that building block might not be present in a randomly generated initial population. Therefore, that building block would not even begin propagating through the population if it is not composed of even smaller order building blocks. It is clear that for a GA to be able to solve a particular problem, there must exist small order, good building blocks that lead to an optimal solution to that problem. Thus, one common measure of problem difficulty has been the order of the largest good building block in a problem that must be found and cannot be formed through the recombination of smaller, good building blocks. The larger this order, the more difficult the problem in question is for a GA to solve. A problem whose measure of bounded difficulty is equal to n is said to be an *order n* problem. This measure of problem difficulty is closely related to the amount of nonlinearity present in a problem in its current GA representation.

The concept of deception as formulated by Goldberg (Goldberg, 1987) takes the notion of bounded difficulty to its extreme conclusion. An order n building block is said to be *deceptive* if it shares no fixed elements with all the good building blocks of smaller order than n. A problem is said to be *boundedly deceptive* up to or of order n if it is an order n problem and all its good order n building blocks are deceptive up to order n. Boundedly deceptive problems of order n are the most difficult order n problems that GAs can face. This is because the presence of deception misleads the inductive nature of GAs and causes them to make incorrect decisions if they do not propagate some good building blocks in their entirety. Thus, it is hypothethized that a GA technique that can, with no a priori problem-specific knowledge solve problems of bounded deception, is guaranteed to solve all other problems of the same level of bounded difficulty. This is why it has become standard in many studies to investigate the performance of various GA methodologies on boundedly deceptive problems.

Figure 5 shows a specific problem and the restrictions placed upon it by the requirement of bounded deception. In this figure the global optimum has ones at all its bit positions. This particular problem is deceptive of order three. The fourth line in this figure shows that all the building blocks of lower order than three are misleading. That is, on the average, when optimizing over any two bits in the genetic representation of a string, the GA is lead away from the global optimum. On the other hand, for this problem to be deceptive of order three, there must exist *some* order three building blocks that lead towards the global optimum. The right hand side of the last two rows shows the required order three building blocks leading towards the global optimum. The left hand side of the last two rows shows that not all order three building blocks need lead to the global optimum under the problem's order three requirement.

2.3 Crossover and the recombination of building blocks

Even if an encoding that bounds a problem's order is found, a GA can still fail to solve the given problem if it is not able to recombine the optimal solution's good building blocks. This type of failure is usually directly attributable to the type of crossover operator the GA uses.

One-point crossover favors small order tightly linked building blocks. Such building blocks are both preserved and recombined well using one point crossover[2]. On the other hand, using one-point crossover, building blocks with large defining lengths are often broken up and almost never recombined with other building blocks. Other crossover operators have been proposed to deal with this discrepancy and are kinder to building blocks with large defining lengths. Syswerda (1989)

[2]See the schema theorem for a proof of the first part of this statement

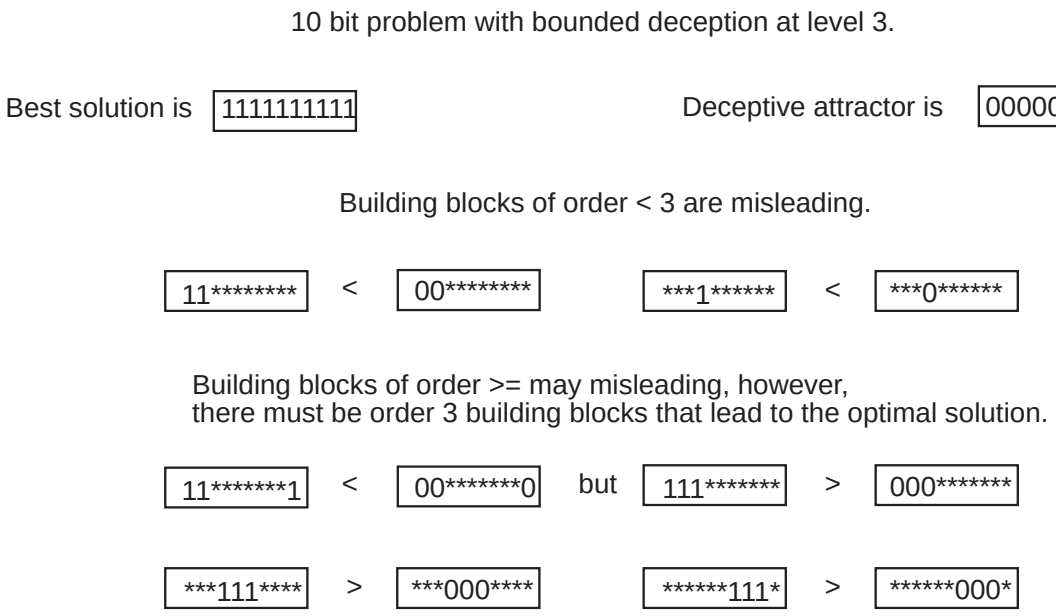


Figure 5: A boundedly deceptive problem and its schema averages

proposed the use of a *uniform crossover* in which each bit is taken at random from one of its two parents. Though this form of crossover has no bias against large defining lengths, it has a larger bias than one point crossover against medium order building blocks. Other forms of crossover have been presented; most fall into one of two camps: they either respect some form of locality or they don't. The former are similar to one-point crossover and the latter to uniform crossover.

The merits and detriments of these two types of recombination operators have been explored at length in both the empirical and theoretical arenas. Following is an examination of the studies undertaken on the various recombination operators.

2.4 Which crossover operator is best?

In the evaluation of various crossover operators, most studies have taken as their benchmark problems either problems of bounded difficulty or problems of bounded deception. One parameter unspecified by this problem choice is the defining lengths of the good building blocks in question. Most studies handle this in one of three ways. Some assume the shortest defining length possible for all building blocks. This is equivalent to the assumption that all the good building blocks are tightly linked. Some investigate various levels of separation for the bit components of good building blocks, while others assume a random distribution of these bits. Figure 6 shows the various assumptions on the defining lengths of the good building blocks that are compatible with bounded difficulty or deception. The black patches in this figure represent the bit positions that could be present in any one good building block under these three assumptions.

Crossover between chromosomes in natural systems is most similar to one-point crossover. Thus, one-point crossover has been the most widely used form of recombination in GAs. In the case where tight linkage is assumed, simple GAs with one point crossover have been empirically shown to solve boundedly difficult problems fairly rapidly (Goldberg et. al, 1991). This evidence seems to indicate that GAs solve these problems in $O(N \log N)$ time where N is the length of the problem.

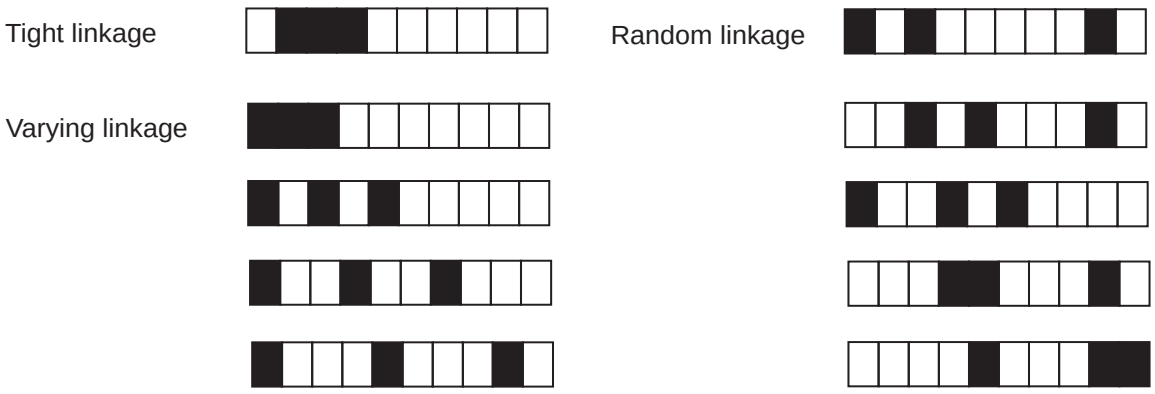


Figure 6: Various possible linkage assumptions

Empirical observations have also been made on similar problems where no such linkage assumption was adopted. One of the first such studies was undertaken by Frantz (1972). Frantz ran a simple GA on problems of bounded difficulty and varied the spacing of the bit components of the good building blocks throughout the coding string. He analyzed the effect of the different induced defining lengths on the search capabilities of the GA. His findings indicated that the GA's ability to locate optimal solutions was highly dependent on the defining length of the solution's building blocks. He found that problems with larger defining lengths often caused the GA to converge on local optima rather than finding the true optimum in these boundedly difficult problems. Other recent studies on similar problems have confirmed this result (Goldberg & Bridges, 1990). This is simply a consequence of the locality preserving nature of the one point crossover operator. In conclusion, it is seen that one point crossover and similar operators work very quickly when tight linkage is available but fail to work when the defining lengths of a problem's good building blocks are long. In fact, with no linkage information, one-point crossover closely resembles uniform crossover which is discussed below.

Grosso (1985) studied the ramifications of increasing the number of crossover points between strings in a GA. One-point crossover generalizes to any integer n-point crossover. Using more than one randomly chosen crossover point, the copying of bits from parents to child alternates the source parent every time a crossover point is encountered in a left to right scan beginning with a random parent. Grosso deduced that the more crossover points he used between pairs of individuals, the less his GA was able to deal with problems of a given order. Uniform crossover usually induces the equivalent of multiple crossover points between strings and thus Grosso's work foreshadowed the theoretical problems that uniform crossover was to face in dealing with boundedly deceptive problems.

Thierens and Goldberg (1993) studied the recombination times of uniform crossover in boundedly deceptive problems. Their work included a dimensional analysis comparing the takeover time of selection versus the building block recombination time of uniform crossover. If the GA converged before it was able to synthesize all the necessary building blocks together in one individual then it failed to solve the given problem. Thierens and Goldberg likened the GA's optimization to a race between the forces of selection and recombination. Using this analysis, they were able to determine parameter settings under which the GA was guaranteed to solve arbitrary boundedly deceptive problems using uniform crossover. Their results bore mixed news for uniform crossover. They verified both theoretically and empirically a GA using uniform crossover was able to deal with

such problems given appropriate parameter settings. However, their results indicated that the GA's required population size for use of uniform crossover would have to be exponential in the problem length attempted.

2.5 The simple GA's scaling problem

A GA working with uniform crossover is able to ignore the need for tight linkage, but at an unacceptable computational price. Similarly, one would expect that a GA using one point crossover without the assumption of tight linkage would suffer the same drawbacks. In fact, only a simple GA working with tight linkage seems to deliver on its promise of fast, accurate optimization. However, the assumption of tight linkage in any particular problem is unrealistic. The problems GAs have to tackle are complex and nonlinear in nature and often little linkage information is available in such problems before actual search begins.

In order that GAs reproduce, on larger problems, the success they have experienced thus far, a method must be presented using which the GA can extract at the appropriate time, the correct linkage information for the good building blocks at hand. Without such a method, the methodology by which GAs work is destined to exploit smaller and smaller building blocks as the size of the problems it is confronted with grows. This will lead to a point at which the GA does no better than simple statistical inference methods used separately or sequentially on the single component bits of a problem's encoding. With such a method, the GA can be made to work on arbitrarily large building blocks and will be as effective an optimization strategy as it has empirically promised to be thus far.

2.6 Inversion

Theoretical considerations thus imply that tight linkage is a prerequisite for fast problem-solving using GAs. However, the assumption of tight linkage in any particular problem is unrealistic. The problems GAs have to tackle are complex and nonlinear in nature. Quite often, no linkage information can be assumed in such problems before actual search begins. This conclusion agrees with the initial suspicions of the early GA researchers. In his dissertation, Bagley (1967) recognizes the importance of linkage information in the formation of good building blocks. He suggests the genetic operator of inversion as a solution to the lack of linkage information problem.

When a problem is encoded into a bit string, different variables or functional roles are often strewn about randomly in the encoding. A *functional role* for a bit position defines how the bit value of that position is to be interpreted in the specification of the problem solution. In simple GAs, the functional role of any particular bit position remains fixed throughout the GA's operation. A crossover operator that uses this fixed encoding is thus biased by this initial placement of functional roles and may be unable to recombine and synthesize certain sets of building blocks. The inversion operator reorders this placement of functional roles throughout the GA's run. Theoretically, this permits good building blocks to achieve tight linkage and subsequently recombine through crossover. Inversion works by randomly picking two points on the encoded string and reversing the order of the functional roles in between that pair of points. Since inversion alters the order of functional roles within a string, a bit's position alone no longer dictates the role it will play in the final solution specification that an individual represents. Inversion requires that additional information be kept by a GA to reform a standard fixed encoding for an individual in order that fitness evaluation take place. Using inversion, the encoded string is stored as a list of genes, each of which is a pair. The first element in the pair represents the functional role of the gene while the second element

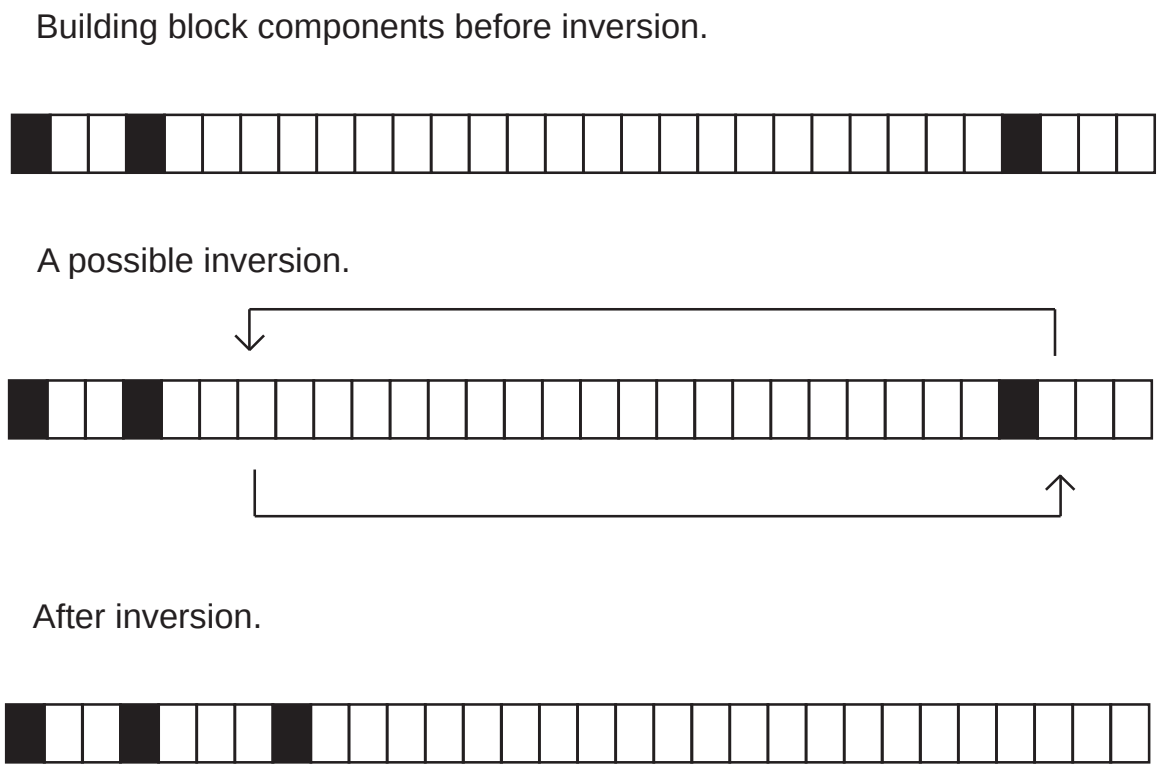


Figure 7: Inversion reordering a building block's components

represents the value of that functional role. Using this representation, functional roles may be placed arbitrarily close to each other on the encoded string.

For example, if in a particular 30-bit encoding for a problem, positions 1, 4 and 27 all being set at 1 form a good building block, then one-point crossover is very likely to break up this combination of alleles. However, inversion could then possibly remedy this problem by reordering the 1st, 4th and 27th functional roles to be closer together on the encoding string. Figure 7 shows how an inversion might be able to tackle this problem.

The addition of inversion adds a degree of complication to the crossover operator. This is because individuals with different encodings that recombine could miss certain functional roles for a full solution specification or could also contain ambiguous material in the form of more than one specification for a functional role. Thus, in the presence of inversion, the operation of crossover has to be modified. In his introduction of inversion, Bagley (1967) dealt with this problem by limiting crossover to individuals whose encoding order was largely similar. This lead to the crossover operator being applied at much lower rates than Bagley had expected and he concluded that inversion was not useful in this context. Frantz (1972) in his study of positional dependencies of building blocks also suggested inversion as a solution to the lack of linkage information problem. Learning from Bagley's work, he chose to expand the role of crossover with the use of inversion. He proposed multiple means of dealing with the negative interaction between crossover and inversion. He concluded that the best way to deal with this problem was to force one of the parents in a crossover to take the functional form of the other parent thus forcing encoding homology during crossover. That is, before crossover took place, the ordering of functional roles in one parent was changed so as to mimic the ordering in the second parent. This allowed Frantz to take full advantage of the crossover operator in his runs.

Unfortunately, his results also indicated that inversion was not capable of improving the search characteristics of the GA. Frantz did, however, suggest that perhaps inversion might be more useful in a longer time scale than that allowed by his runs.

This conclusion matches the theoretical results presented by Goldberg & Bridges (1990) on the performance of inversion on the separated minimally deceptive problem. The *minimally deceptive problem* is the smallest possible deceptive problem. It is composed of two bits where solving for the best average fitness separately in each bit leads away from the global optimum. When these two bits are placed far apart in a GA encoding, the GA is often lead to converge to a suboptimal problem solution. Goldberg and Bridges investigate a simplified form of the inversion mechanism that switches back and forth between a short and long defining length representation for a minimally deceptive building block in a coding string. Goldberg and Bridges show that the inversion operator helps the GA converge to the optimal solution in the minimally deceptive building block in conditions where a GA without inversion would not. Unfortunately, their results also indicate that the rate of application of inversion has to be very low so as not to undo the advantages it confers. A rate of inversion that is too high destroys the short lengthed structures it creates before they get a chance to gain a foothold in the current population. This result seems to imply that in real problems, the maximum applicable rate of inversion would be low enough so as to render it effectively useless.

The inversion operator searches the space of possible encodings for a problem given its underlying variables. Viewed in this light, inversion is seen to be a mutation of the current ordering. Unfortunately, the work done thus far seems to indicate that the use of inversion in this reordering problem is not likely to meet with success. Thus we are forced to examine other possible solutions to this problem. This particular search problem is not a simple one. The number of possible permutations in a problem of length n is $n!$. This is considerably larger than even our original solution space which has only 2^n solutions for a problem of length n. Furthermore, this space is not dense. Most random reorderings will place building block components fairly apart from each other. Additionally, since a particular ordering is good if it allows recombinations between different building blocks, the evaluation of a particular ordering over another takes time and effort. These combined facts rule out simple minded search algorithms for this problem.

Inversion introduces new patterns into the GA to be continuously tested by the GA's selection component. One possible explanation for its failure is that while a particular inversion strain is being tested, the GA is quickly converging to a specific solution and losing diversity at certain sets of functional roles. That loss of diversity later prevents it from finding the global optimum. Thus, inversion might work too slowly. However, as Goldberg & Bridges point out, raising the rate of inversion prevents even one ordering from being adequately tested. The only solution to this problem is to drastically reduce the selection pressure in the GA and thus give inversion adequate time to test different structures. It seems that this approach would lead to an insufferably slow GA which is not acceptable given the GA's nature as a heuristic algorithm. This hypothesis for inversion's failure seems to suggest another solution to the linkage problem. Perhaps operators are needed that search the ordering space quickly without destroying basic information present in one particular ordering or another. This observation suggests a crossover-like operator that works on the ordering space itself. This approach to the linkage problem seems plausible and perhaps deserves some investigation. Goldberg & Bridges (1990) come to a similar conclusion in their work. Yet another approach is to attempt to extract the good building blocks with their linkages intact from whole solutions. This approach has met with some initial success and has involved using overspecified and underspecified problem representations. This approach is embodied in the messy GA (mGA) developed by Goldberg, Korb & Deb (1989).

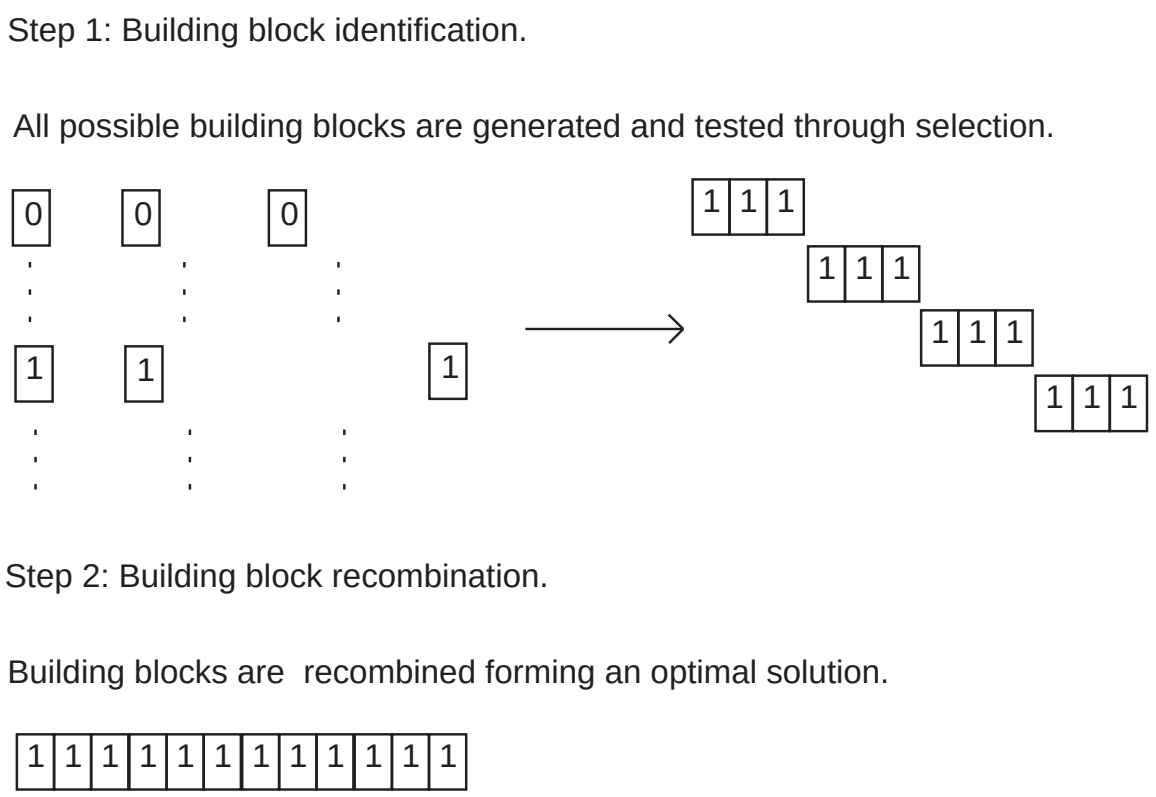


Figure 8: The original mGA formulation

2.7 The messy GA approach

The mGA seeks to replicate, on boundedly difficult problems, the efforts that a simple GA puts forth to solve simpler problems. In a simple GA working on a tightly linked problem, the selection process correctly identifies and replicates each of the good building blocks. During this time, the crossover operator, with its low level of building block disruption afforded it by the tight linkage, proceeds to instantiate all these building blocks in the same individual. The two main ingredients of this search are the identification of the good building blocks and their reformation in a single individual. In a problem with no linkage information, if these two steps are separated, the first step can be used to extract the correct linkage information for the problem. The separation of these two steps is precisely what the mGA achieves. Figure 8 shows how the original mGA worked.

In the mGA's initial formulation, the two main steps of building block identification and recombination operated as follows. All possible building blocks of order k were first generated. Through the action of selection, in the mGA's *primordial* phase, the best building blocks were identified and overtook part of the population. Then, in the recombination or *juxtapositional* phase, these building blocks were cut and spliced under the action of further selection until whole solutions were formed and the population converged. The cut and splice operators are specific to the mGA formulation and are supposed to mimic the effect of one-point crossover acting on tightly linked building blocks. These operators will be described shortly. To select between building blocks, the mGA needed some means of evaluating building blocks which are only partially specified solutions. The mGA also needed to restrict competition between building blocks so as to end up with enough functional roles to form at least one complete solution to the problem. Following is an example within whose

framework the mGA will be discussed.

One of the first problems that the mGA was tested on was a 30 bit problem with order three deception. It consisted of a concatenation of 10 deceptive order three subproblems in which all the subproblems' bits were maximally separated in the string encoding. The requirement of maximal separation implied that the subfunction bits were spread apart in the 30 bit encoding. The first subfunction was represented by bits 1,11 and 21; the second by bits 2,12 and 22 and so forth. Thus this problem had the worst linkage possible for a simple GA. Each subfunction of order three had as its optimum ones at all its positions. However, so as to fulfill the requirement of bounded deception, all the smaller order building blocks in each subfunction lead to the second best solution within that subfunction which contained zeroes at all its positions. The lack of tight linkage made this problem quite hard for a simple GA using one-point crossover. Thierens & Goldberg (1993) also showed this problem is difficult to solve using uniform crossover.

In this problem, bits 1,11 and 21 all set at one represent a good building block. In the selection phase of the mGA, the *primordial* phase, at level $k = 3$, this building block is represented by the list (1 1) (11 1) (21 1). In the primordial phase this building block and others compete against each other. As mentioned above, the mGA has to have a means of evaluating these partially specified structures. One way for the mGA to do this would be to average the fitness evaluations attained by randomly filling in the remaining 27 bits in the solution. However, as other building blocks would also have to be doing the same, this would introduce quite a bit of noise into the problem. Goldberg, Korb & Deb (1989) analyze this particular situation and find it to be generally infeasible. To reduce the noise of variation, the mGA could evaluate all the building blocks using the exact same set of random filling solutions for the rest of the problem. That is the mGA could pick a set of random solutions, and then evaluate all the building blocks using this same set and average out the fitness values obtained this way for each building block. This might be more feasible in terms of making less errors but would still be computationally prohibitive. Instead of evaluating all the building blocks using a large set of solutions, the original mGA developers evaluated all the building blocks using one particularly well chosen solution. This allowed them to evaluate all the possible building blocks with a minimum of both noise and computational expense[3].

This well chosen solution is what Goldberg, Korb & Deb (1989) call the competitive template. The mGA begins with a random individual as its template and operates at the $k = 1$ level. When the mGA is done at this level, its final solution is then optimal with respect to the information that can be attained using order one building blocks. The mGA's solution at the $k = 1$ level is then used as its template for the $k = 2$ level. Then the solution at the $k = 2$ level is used as the template for the $k = 3$ level and so on. Once k has equaled a problem's bounded level of deception, the mGA will have exploited the information contained in all the good building blocks whose order is less or equal to k. Thus the mGA will have solved the problem at hand. Figure 9 shows how the mGA with the competitive template formulation climbs the ladder of deception to solve any boundedly deceptive problem.

One thing not discussed thus far is how the mGA manages to maintain building blocks at all the functional roles for a problem in its selection phase. This problem is similar to an ecological niche problem in that multiple individuals, that could possibly compete without action on the mGA's part, have to coexist for a final solution to be constructed. The mGA enforces this coexistence by using a particular selection procedure, tournament selection, with thresholding which is a method of restricting competition to individuals containing similar genes or functional roles. *Tournament* selection works by pairing elements in the population together and forcing them to compete for a position in the next generation. This method of selection also allows, with a little more overhead,

[3]Note that this bears distant similarity to approximating integral values of functions using Gaussian Quadrature.

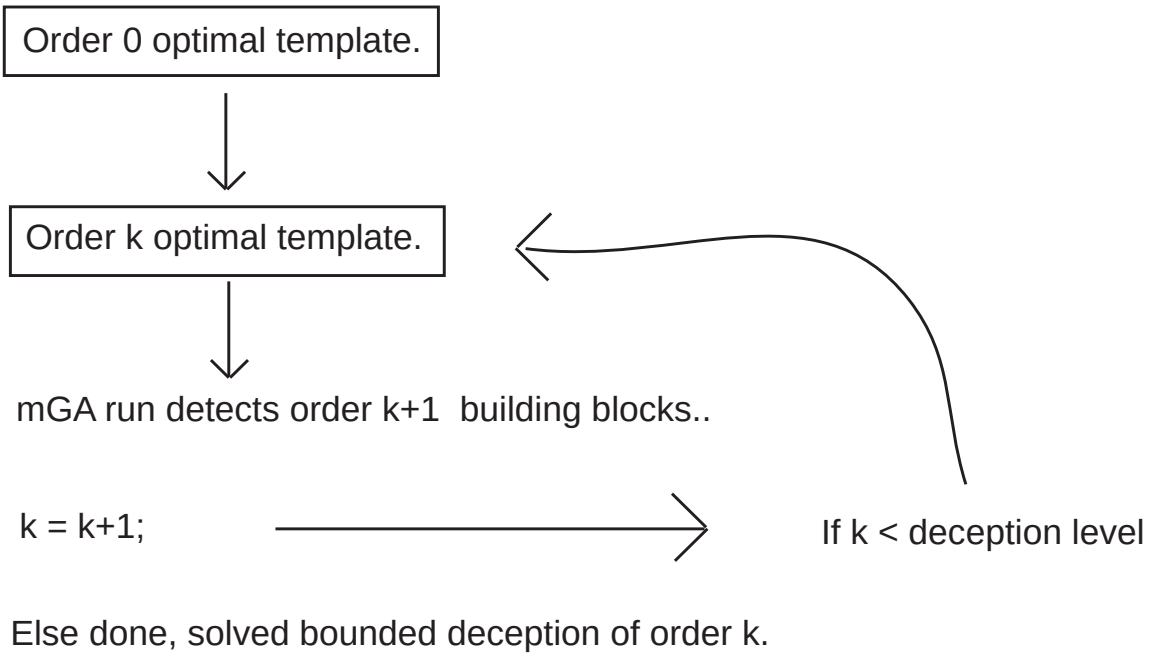


Figure 9: A competitive template allows the mGA to climb the ladder of deception

limiting competitions to elements that are alike under some distance measure. The *thresholding* method sets the number of genes or positions that two building blocks must have in common before being allowed to compete against each other. This restriction on competition allows selection to work ad infinitum and still not threaten the diversity of building blocks that is required to cover the entire coding string.

Once the mGA identifies the proper building blocks in its primordial phase, all that remains is their recombination. For this purpose, the mGA uses the cut and splice operators. The *cut* operator cuts an individual in two with probability $P_c(\lambda - 1)$ where λ is the number of genes in the individual, and the splice operator splices two strings with probability P_s. P_c and P_s are fixed search parameters. While working on building blocks, these operators bear a high resemblance to a one-point crossover operator working on a tightly linked problem. These operators introduce the possibility of overspecification at a certain gene or functional role. The mGA deals with this problem by only using the first specification encountered for a functional role in a left to right scan of the string representing an individual. This ensures that after a slice operation, at least one parent's building blocks will be fully present[4]. Figure 10 shows how the cut and splice operators work on building blocks.

Goldberg, Korb & Deb (1989,1990) empirically showed the mGA capable of solving a variety of boundedly deceptive problems of mixed size and scale. As expected, their mGA climbed the ladder of deception and overcame the hurdles placed in its path. One problem with the initial formulation of the mGA was that the primordial phase's computational time was polynomial in the number of decision variables, with the exponent being the problem's order, while its juxtapositional phase took roughly subquadratic time in the number of decision variables. Goldberg et. al. (1993) found the time complexity for the mGA's primordial phase unacceptable. This lead to the development

[4]Note that should the mGA choose randomly which positions to express this operation would start to resemble uniform crossover in some cases

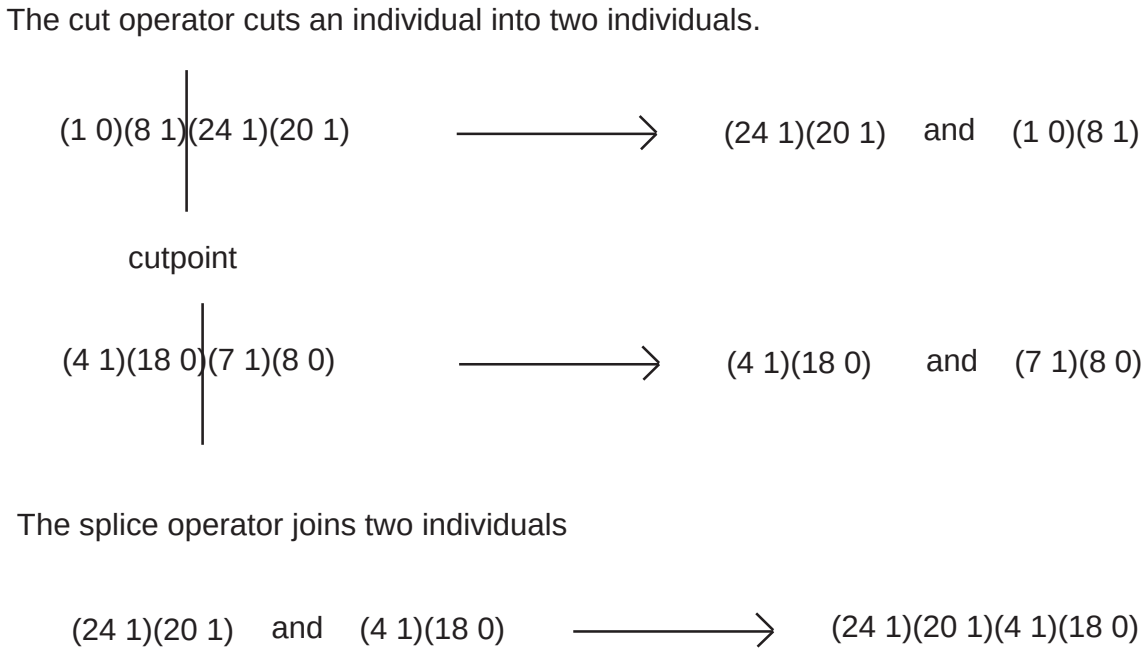


Figure 10: The cut and splice operators look like 1-point crossover

of the probabilistically complete initialization algorithm for building block identification.

The probabilistically complete initialization method does away with the explicit generation of all building blocks of a certain order k. Instead, enough full-length individuals are generated to ensure that multiple copies of all the requisite building blocks of level k are present within these full solutions. For example, eight randomly defined strings are expected to contain one copy of any given order three building block. Thus by generating 400 complete individuals, the mGA represents each order three building block approximately 50 times in the population. Using this form of initialization, the mGA has to extract the correct building blocks from these fully specified strings by a procedure of deletion and filtering. At each order, starting from full specification, selection is applied to the strings in the population for a certain number of generations and under the action of thresholding. This serves to enrich the population with the better building blocks. A random deletion procedure is then applied to each individual in the population. This reduces the effective length of all the population members. The mGA continues this process of selection and deletion until the population members are all of order k and it has found the correct building blocks for the problem. Then the regular juxtapositional phase takes over the mGA's operation. This form of initialization allows the mGA's primordial phase to operate in subquadratic time thus putting it in line with the juxtapositional phase. Goldberg et. al. (1993) show that this form of initialization performs well in the problems the original mGA was tested on and does so significantly faster than the original mGA. Currently, work is under way to identify the theoretical underpinnings of this method. Figure 11 shows how the new probabilistic initialization method identifies building blocks.

The mGA is the only viable algorithm developed thus far that can quickly and reliably solve boundedly deceptive problems without the need for a priori linkage assumptions. Furthermore, it solves these problems in time that is subquadratic in the number of variables a problem contains. It thus offers the possibility of a computational advantage over the simple GA in scaling to larger problems. That fact provides the incentive to include the powerful mGA search engine with a multi-

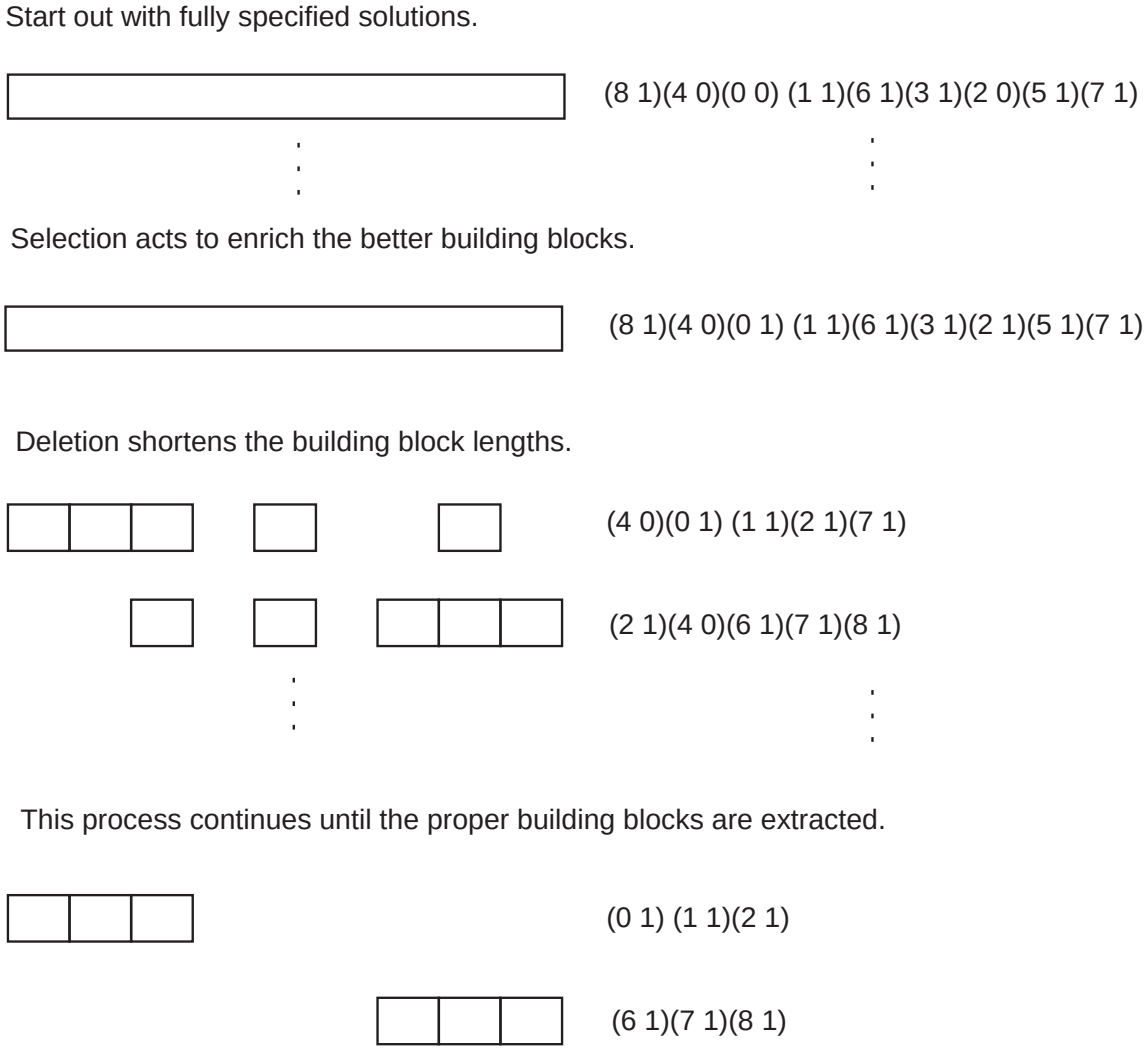


Figure 11: The probabilistic initialization method at work

modal technique in the formation of a robust real world GA optimizer. Following is an investigation of the multimodality techniques that could possibly be used in such an optimizer.

2.8 The preservation of diversity

Real world problems often require multiple solutions. In a mechanistic sense, this immediately implies that the simple genetic algorithm will not provide solutions that are adequate for these types of problems. Many modifications to the simple GA have been proposed to deal with these sorts of problems. These methods are sometimes referred to as *niching* techniques, because of their proximity in spirit to the formation of ecological niches in nature, using which multiple species can coexist.

Niching techniques belong to a general class of techniques that have been used in GAs to either preserve or inject diversity. The aim of these techniques has been two-fold. Their first aim is the location and preservation of multiple solutions in a given problem. Their second aim is to preserve or inject useful diversity in a GA in order to slow down the GA's premature convergence to a suboptimal solution. These techniques share a large degree of commonality. Following is an investigation of the relevant work done by past researchers in these areas.

Two general approaches to these techniques can be identified. The first approach focuses on changing the selective environment of certain sections of the search space so as to prevent the GA from quickly converging to those sections. This approach encompasses the methods of preselection, crowding and sharing. The second approach focuses on imposing certain geographies on the GA's population to prevent the rapid proliferation of certain highly fit individuals. This approach is embodied in the various island and isolation by distance parallel models used with the GA. Following is a discussion of the specific variations of these two approaches.

2.9 Diversity preservation methods using selection

Work on the preservation of diversity began with the advent of the GA. Cavicchio (1970) theorized that the children of a crossover were very likely to be similar to their parents. In order to preserve diversity for the purpose of preventing premature convergence, Cavicchio (1970) introduced various *preselection* schemes in which children replaced their own parents upon insertion into the new population. One of his preselection schemes met with some success. In that scheme, a child more fit than either of its parents replaced that parent.

In his thesis work, De Jong (1975) introduced a *crowding* mechanism in which newly created individuals in a population replaced similar individuals. Using crowding, only a fraction of the population, the *generation gap* G, would be replaced in a generational GA. When a new element was formed by crossover and mutation, a certain number of elements, the *crowding factor* CF, from the remainder of the population, were examined to find the individual closest to the created child. The measure of closeness in this situation was a hamming distance metric. The child then replaced this individual in the population. One of De Jong's test functions was multimodal. On that function, a simple GA with crowding performed better than a simple GA without crowding in terms of the speed of convergence to the best solution. This was a direct example of the second type of diversity maintenance, that for the purpose of better convergence. Though both preselection and crowding were developed as mechanisms to prevent the onset of premature convergence, they are clearly applicable in the realm of multimodal function optimization. A later study by Mahfoud (1992) investigated this facet of both algorithms.

A few GA applications not directly related to function optimization have been modified to

use problem specific techniques for the preservation of diversity. In his study of machine learning with classifier systems, Booker (1982) introduced a natural way of niching by limiting the number of resources an individual classifier may draw by matching a particular example. He did this by forcing related classifiers to share payment for matching the same individual. Later, Wilson (1986) utilized this method in the realm of boolean function learning. These works bear a slight resemblance to the fitness-sharing algorithm later developed by Goldberg et. al. (1987) for the purpose of multimodal problem solving. Goldberg (1983) successfully used a GA with crowding as the main search engine for a classifier system to be used in pipeline control. It is not clear that work done on GAs with classifier systems extends directly into the realm of multimodal function optimization.

Perry (1984) in his dissertation investigated the takeover of certain artificially created niches by differing species. He created multiple contexts in which the fitness function varied according with the matching of certain externally defined building blocks. He allowed individuals to migrate between these different contexts. In each of these contexts, the principle of competitive exclusion ensured that only one species survived. However, the existence of the multiple contexts themselves allowed a sort of niching to occur and allowed multiple solutions to coexist in his populations. This technique, because it makes use of the existence of explicitly different fitness-producing contexts, is not immediately applicable in the domain of function optimization.

Schaffer's VEGA (1984) was one of the first GAs to be used in the realm of multicriteria optimization. Such a space contains multiple optimal solutions that are inherently incomparable, representing a pareto optimal front. Schaffer tackled this problem by switching the emphasis of his selection from one criterion to another continuously throughout his GA's run. Schaffer's results in this case were not general as his GA ran for only a few generations and was only attempted on one test problem. Later work (Horn & Nafpliotis,1993;Fonseca & Fleming,1993) proposed yet another alternative to this form of optimization. The techniques used in work on these kinds of problems depend on the inherent incomparability of solutions in this sort of a setting. These techniques are not applicable in the realm of single-criterion optimization.

Mauldin (1984) attempted to reduce the GA's probability of premature convergence by continuously injecting diversity into his populations. This involved requiring that newly created elements differ by at least k bits from all current population members before being inserted into the population. An individual that did not match this criterion was then mutated until it did. At times, this was quite prohibitive computationally. Though Mauldin's results on De Jong's five test functions were positive, they do not strongly enough endorse this method which violates the spirit of the building block propagation theory by advocating mutation on such a large scale.

Baker (1985) suggested that a more controlled selection process than fitness-proportionate selection would often prevent a GA from converging prematurely. As it turns out, fitness-proportionate selection alone often leads the GA to such undesirable behavior. Thus were developed many fitness-scaling techniques and selection methods such as tournament selection which are more invariant as to the exact assignment of fitness scores. Though controlled selection of this sort is often necessary to allow adequate time for the evaluation and recombination of building blocks, it was not intended as, nor is it, a sufficient multimodal technique.

Until 1987, little work had been done in the area of preservation of diversity with regards to the location of multiple solutions using GAs. It was at this time that researchers working on GAs started to pay attention to these issues. The next few years saw the development of the fitness-sharing algorithm as well as a couple of investigations into the merits of preselection and crowding in this area. Since these methods represent the currently used GA techniques for the preservation of diversity, the following section presents a more detailed examination of this work.

2.10 Fitness sharing and crowding

Goldberg & Richardson's (1987) *fitness-sharing* algorithm was one of the first attempts to deal directly with the location and preservation of multiple solutions in a GA. The fitness-sharing algorithm took a cue from nature, restricting the unlimited growth of one type of individual by making each individual in the population share its fitness assignment with nearby elements in the population. This made explicit the limitation of there being a finite amount of resources exploitable by any particular species in a natural environment. Their analysis of this approach based on the two-armed bandit analogy indicated that this algorithm would lead to stable subpopulations of individuals forming around each 'niche' that was to be preserved provided that the niches were properly separated.

The fitness-sharing algorithm involves dividing an individual's fitness score by the number of other individuals sharing its niche. An individual's *niche* is defined as the collection of population elements within a certain distance of that individual according to some distance measure. The fitness-sharing algorithm operates on the following principle. As one niche begins to gain more and more members, the average fitness of its members begins to decline due to the sharing penalty imposed on each of them. This in turn allows the other niches in the population to increase in size. This imposed penalty enforces an equilibrium between the representations of the different niches. Within each niche, since each individual's fitness is penalized an equal fraction of its value, the best individual will still take over that niche. Thus this algorithm causes the best individual in each niche to win out within its niche while still maintaining multiple niches in the population. In their initial formulation, Goldberg & Richardson tested this fitness-sharing algorithm out on a number of different multimodal landscapes including problems with evenly and unevenly spaced peaks and peaks of equal and unequal height in Euclidean space. The fitness-sharing algorithm was able to maintain solutions at most of the higher valued peaks in all cases tested. Figure 12 shows how fitness sharing forces an individual to share its fitness with all other individuals in its niche.

The fitness-sharing algorithm requires both a distance metric over the problem space and a parameter σ_{share}. The σ_{share} parameter defines for each individual the maximum distance over which it has to share its fitness with other population members. Later work on fitness-sharing by Deb & Goldberg (1989) studied this parameter and involved a direct comparison between fitness-sharing and the crowding algorithm developed by De Jong (1975). Deb & Goldberg showed that one possible way to set σ_{share} would be to divide the search space into a number of equal-sized hyperspheres equal to the number of sought out optima. The σ_{share} parameter could then be set to equal the radius of each of these hyperspheres. This formulation was the one used in their initial successful runs using fitness sharing. Deb & Goldberg also tested De Jong's crowding algorithm on the same functions they had initially tested fitness-sharing on. They found that crowding was unable to maintain more than 2 peaks on all of the above problems and was generally inferior to the fitness-sharing algorithm.

A later study that highlighted both the strengths and the weaknesses of the fitness-sharing algorithm was undertaken by Goldberg, Deb & Horn (1992). They manufactured a large multimodal problem and attempted to solve it using the fitness-sharing algorithm. The problem they tackled was a 30 bit problem consisting of a concatenation of 5 subproblems. Each of the subproblems had 2 global optima and 20 local optima. The concatenated problem thus had 32 global optima nestled among over 5 million local optima. The sheer size of this problem posed a challenge for any optimization algorithm. Using their traditional computation for σ_{share}, Goldberg, Deb & Horn were unable to find and maintain all the global solutions to this problem. The global solutions in this problem were spaced so closely to each other that they were forced into competition by the traditional σ_{share} calculation. Goldberg et. al. then tried reducing σ_{share} and found that their

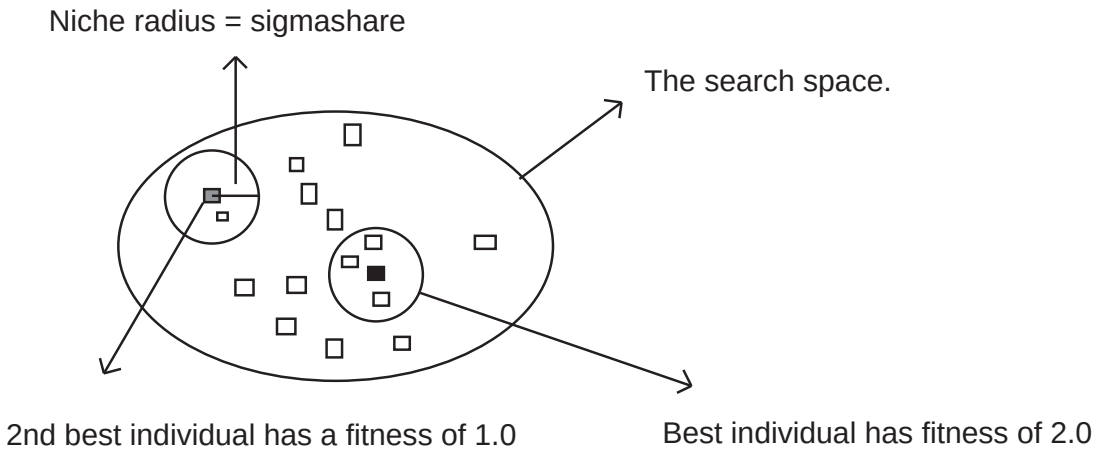


Figure 12: Fitness sharing causes resource division within a niche

algorithm got stuck finding many of the local optima in this problem. This was again to be expected given the large number of local optima in this problem. Their final solution was to use a small σ_{share} and raise their fitness function assignment to an exponent so as to emphasize the better solutions in the search space. This final approach allowed their algorithm to find and almost indefinitely maintain all 32 global solutions in this problem.

All of the previous problems that the fitness-sharing algorithm had been tried on had their solutions well-separated in the solution space. This massively-multimodal problem had its optima relatively close to each other. However, using exponential scaling of the fitness function, fitness-sharing was still able to find and maintain those optima. These two sets of problems help define the strengths of the fitness-sharing algorithm. The fitness-sharing algorithm seems able to maintain multiple optima in a problem space provided those optima are either well-separated in the solution space or do not differ much in their fitness assignment. These efforts point out yet another strength of the fitness-sharing algorithm. It's steady state solution in this case seems rather stable and preserves all the global optima for a long period of time. A later study by Horn (1993) predicts this form of stability theoretically in the two niche case.

The drawbacks of using fitness-sharing seem to be as follows. Its ability to maintain multiple solutions can fail if the optima in a problem are too closely spaced in the solution space and their fitness assignments are not close. Additionally, its calculation of niche size for each population element is reasonably time consuming though with sampling this problem might be ameliorated. Currently, one of the bigger drawbacks to using fitness-sharing is that its effects on the GA's search process are not well understood. A GA with fitness-sharing searches its solutions space in a very different way that a simple GA alone. In a simple GA, the good building blocks receive an exponentially increasing allocation of trials throughout the GA's life. In a GA with fitness-sharing, the passing of each generation causes an immediate jump of all niches towards equilibrium. If the niche radius is not large enough, this could easily cause stagnation in the early stages of the GA since each individual would be in its own niche and essentially no exploitation of information would occur. This interconnection between the user's specification of a problem and the GA's operation does not seem to contribute to the GA's overall robustness. The fact that this algorithm operates differently for different order preserving scalings of a problem, such as exponential scaling of the fitness function, also does not seem desirable. In tandem, these two factors complicate the inclusion of fitness-sharing into a GA using a selection method other than fitness-proportionate selection.

Some claim that the requirement that fitness-sharing need specify a parameter such as σ_{share} is an innate weakness of the algorithm. When looking for multiple solutions in a problem, a strict specification of requirements is needed, otherwise almost any solution will do. The specification of σ_{share} is merely one way of indicating how different solutions must be before they are considered 'different' solutions for the user. Thus the claim that this is a drawback to using the fitness-sharing algorithm seems unjustified.

Initial research by Deb & Goldberg (1989) indicated that crowding was unable to compete as a multimodal algorithm with fitness-sharing. Mahfoud's research backed up this initial indication. Mahfoud (1992) set out to test various forms of crowding and preselection on a number different multimodal functions. His conclusions on the crowding algorithm matched those of Deb & Goldberg. Crowding was unable to maintain more than two solutions on any of the problems Mahfoud had tried out. However, one of the forms of crowding that Mahfoud developed seemed somewhat capable of maintaining multiple peaks in all of his test functions. He used a steady-state algorithm that matched the child of a crossover with the closest parent in that crossover. Then the child would be accepted into the population if its fitness was higher than its parent's fitness. This algorithm was based on the assumption that the child of a crossover would always be similar to its closest

parent and thus this like replaces like strategy would preserve diversity in the population. This was an assumption like that of the preselection algorithm upon which it was based. This algorithm has a rapid implementation but its behavior is not predictable in amenable terms. Using it, one cannot guarantee how many solutions it will find or what the spacing of those solutions will be. This algorithm also seems to have trouble maintaining solutions that can cross with other solutions and thus reach higher peaks. In conclusion, it seems that to this date, fitness-sharing is the only algorithm of its kind that is a viable choice when one seeks multiple solutions to a problem using GAs.

2.10.1 Other related work

Smith, Forrest & Perelson's work (1993), though not in the realm of function optimization, bore some similarity to the multimodal techniques investigated above. Their work involved finding an optimal population of individuals to cover a fixed set of test examples. Each individual in their population represented an antibody covering some part of the test examples which represented antigens.

The fitness function they employed worked as follows. All of the individuals initially had their fitness assignments set at zero. An antigen from the test examples was then selected at random. A certain number N of randomly chosen individuals from the population were then observed and the individual best matching the antigen had a fixed amount added to its fitness [5]. This process was repeated several times in-between applications of fitness-proportionate selection. The effect of this process was similar to that of fitness-sharing, in that highly similar individuals would often share the payment provided by any number of antigens that they matched particularly well. This experiment mimicked Perry's work in that each of the antigens given provided a separate source of fitness availability from which the population members could draw. In accordance with expectation, this allowed their populations to maintain diverse elements that each matched either one or a many of the given antigens. Additionally, their N parameter was found to be proportional to the number of different solutions maintained at the end of their runs. Thus they could specify at the beginning of their runs whether they wanted to obtain one very good antibody or any particular number of more specific antibodies that together matched the set of antigens. Though this work is not immediately applicable to function optimization, it seems close in nature to an algorithm that might work in that domain [6].

Beasly, Bull & Martin (1993) presented a technique that in sequential fashion finds multiple solutions to any given problem. Their iterated approach involved finding a peak in a multimodal landscape and subsequently penalizing in a manner similar to fitness-sharing the elements of each future run that came close to the found peak [7]. They reported some success using this method on the same initial functions that fitness-sharing was tested on. Beasly et al. claimed this method has a lower time complexity than that of fitness sharing alone. It seems likely that in all its iterations, this algorithm repeatedly loses important information that might lead it to one of the optima. This algorithm, and the assumptions upon which it is based, have not been tested enough to draw conclusions upon its operation.

Beasly, Bull & Martin's time complexity analysis depends critically on two conjectures. The first is that if a problem of given difficulty requires a population size of N in order to solve it: a fitness-sharing based algorithm will find p solutions in that problem using a population size of Np

[5] How this match was computed is irrelevant here

[6] In fact, see Restricted Tournament Selection later on in this paper

[7] This 'closeness' was again defined by a σ_{share} like parameter

and operating for $O(\log N)$ generations. The second conjecture is that their iterated algorithm will successively find new solutions on each iteration with a population size of N scaled for the problem difficulty and in $O(\log N)$ generations. The first conjecture has been shown to be empirically accurate in one setting, that of the massively multimodal problem. However, the second conjecture seems overly optimistic and has yet to even be tested in an empirical setting. Until this work is done, no conclusion can be drawn from their work.

2.11 Diversity preservation methods using spatial separation

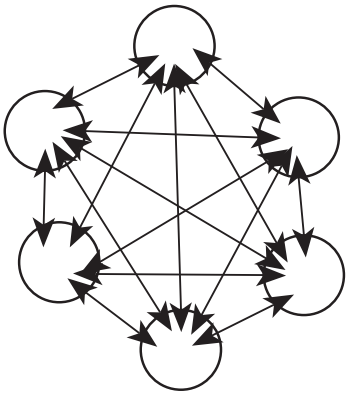
Much of the work done on GAs with spatial separation was inspired by the Shifting Balance theory of evolution posed by Sewall Wright in the 1940s (Wright, S., 1968). This theory attempted to explain the process of evolution on the genetic composition of individuals in a population. Wright suggested that populations were actually divided into small subpopulations or demes. He said that each of these demes would contain a "... wide random variability of gene frequencies ... leading to unique combinations of gene frequencies in each of innumerable different local populations ... With simultaneous stochastic variability of all nearly neutral pairs of alleles, some are likely to cross saddles (against weak selection) leading to superior selective peaks." This hypothesis seems to indicate that Wright believed that local subpopulations would allow evolution to overcome deceptive and nonlinear effects between genes.

Another biological theory adopted by people who do work on parallel GAs is the theory of punctuated equilibria (Eldrege, 1972). This "theory is based on two principles: allopatric speciation and stasis. Allopatric speciation involves the rapid evolution of new species after being geographically separated. ... Stasis, or stability, of a species is simply the notion of lack of change. It implies that after equilibria is reached in an environment, there is very little drift away from the genetic composition of a species." [8] In practice, this theory boils down to yet another justification for using parallel subpopulations with GAs. The difference between this theory and that of Sewall Wright's as applicable to GAs is as follows. Both theories advocate the use of parallel isolated subpopulations with communication. The theory of punctuated equilibria advocates this simply because it itself is an evolutionary theory. On the other hand, Sewall Wright's theory also gives some sort of explanation as to why good results could be expected by following its lead in GAs.

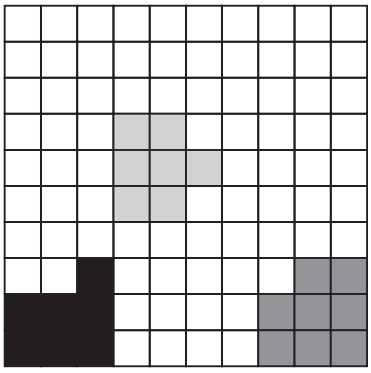
Methods using spatial separation to preserve diversity generally employ one of the following two approaches. The first approach is based on stopping a certain individual's growth once it has taken over some fraction of the population and letting the other individuals in the populations grow for some time before allowing the first individual to mix freely with them. This is usually accomplished by separating the population into different islands and allowing a fixed rate of migration between the islands. Once an individual then has taken over an entire island, it can no longer grow its representation in the population until it is allowed to migrate. During this time the other islands go about their evolutionary path unhindered by this highly fit individual that might very well extinguish other groups of solutions if it is allowed to compete with them. The second approach is to induce some sort of local geometry on the population itself and force local competition and mating within this structure. This form of local mating scheme arrangement again enforces a sort of buffering between multiple solutions and allows them for some time to grow concurrently. Within these two frameworks, we will discuss many of the algorithms that have thus far been introduced that use a form or other of spatial separation. Figure 13 shows the two different techniques used in spatial separation methods.

Among the first GAs using a form of spatial structure was the one studied by Grosso (1985)

[8] Quote from paper by Cohoon et. al.(1987)



In an island model, selection occurs inside the subpopulations, then migration interchanges individuals between subpopulations.



Different individuals are prevented from competing in local distance models by being geographically incrementally separated. The ratio of surface area to volume of an n-dimensional space limits the growth of particular individuals.

Figure 13: Two different spatial separation GA techniques

in his doctoral dissertation. Grosso used an island model spatial structure with varying levels of migration. His aim was to test out Sewall Wright's shifting balance theory. The test function that Grosso studied was one dependent on a diploid genic set. That is, each individual had two chromosomes, one from each parent. Grosso used a multiplicative fitness function with two possible alleles at each gene, one recessive and one dominant. Each diploid gene was individually judged and then the multiple of all the genes' individual fitness functions would be the final fitness of the individual. Grosso set the optimal solution at each diploid gene to be that with one dominant allele and one recessive allele. Grosso did this as it was well known that fixation quickly occurred at all gene positions whose optimal solution is either fully dominant or fully recessive. This type of problem can be related to one in which deception is present by mapping each diploid gene in the problem to a pair of regular genes, which would then be deceptive. Grosso wanted to test whether there were statistically discernible performance differences between a spatially structured GA and panmictic GA on this test problem. He found that there were no such differences and could not confirm Wright's theory.

Cohoon, Hedge, Martin & Richards (1987) proposed a GA based on the idea of punctuated equilibria and proceeded to test it out on a combinatorial problem. Their system was basically an island model with interconnections that minorly limited the migration patterns between the different subpopulations. Their algorithm provided superlinear speedup in that particular problem over a regular GA but they did no analysis past their empirical runs. Pettey, Luse and Greffenstette (1989) also implemented a parallel island model GA with migration. They tested their GA out on four function from De Jong's GA function testbed. They found mixed results in that their algorithm sometimes outperformed an undivided population and sometimes did not. Tanese (1989) implemented an island model parallel GA and compared its performance on functions using finite-order Walsh coefficients. She found that the parallel GA implementation afforded near linear speedup with the number of processors on these problems. However, as she herself stated, it is not clear as to whether these types of problems present enough structure so as to base any form of conclusions on them.

Gorges-Shleuter (1989) introduced a localized spatially separated model in which individuals are placed on some sort of topological space [9] and only allowed to compete and mate locally. This approach showed good results on a test of the traveling salesman problem. A number of other people have implemented parallel GAs based on one of these two architectures and claimed their merits based on empirical evidence. This includes work by Collins & Jefferson (1991); Yuval Davidor (1991); Muhlenbein, Schomisch & Born (1991) and others. Some claim that these parallel GAs preserve enough diversity in their populations to the point where they could be used to find multiple solutions in optimization problems. This claim has yet to be backed by any testing in this area. These claims also seem to be attributable to wishful thinking. The spatially structured GAs proposed so far do not take into account any sort of topology in the solution space itself. Thus it is unlikely that they would be able to come up with solutions that are meaningfully different in many kinds of problems. The diversity attained in parallelizing GAs may provide for better algorithms but does not seem to provide the capability to tackle multimodal problems.

2.12 A final note on multimodality

In summary, it seems that the spatial separation based techniques have not as of yet produced an algorithm capable of multimodal optimization. Of the selection based techniques, only fitness-sharing seems a likely candidate for such an algorithm. However, as discussed above, even fitness-

[9] A ring in this case

sharing deserves closer investigation before its adoption as a multimodal technique. It effects on the GA's search procedure are not yet well understood and might interact badly with a search engine like the mGA.

3 The work done so far

This document has identified two problems with the extension of the simple GA to deal with real world multimodal problems. The developed GA must scale well to the handling of larger problems and the multimodal technique it utilizes must not interfere with its search operation. Following is a list of tasks, that if accomplished, will allow the development of such a GA:

- The development of a set of requirements to determine whether a multimodal technique interferes with the simple GA's search engine.
- The development of a set of optimization problems that test the above requirements.
- The study of how well one or more multimodal techniques fit the requirements of the first task leading up to the choice of a multimodal technique for inclusion into the mGA.
- The incorporation of the chosen multimodal technique into the mGA.
- A study of how well the developed multimodal mGA fits the requirements presented in the first task.
- The solving of a real world problem using the multimodal mGA.

Before beginning work on this document, I had approached this problem in the following way. Initial, simple minded attempts to combine fitness-sharing with the mGA had failed because of an incompatibility between the two methods. This lead me to develop a new multimodal algorithm, restricted tournament selection (RTS), based on the principle of localized competition.

Afterwards, in a pilot incorporation of the RTS algorithm into the mGA, I managed to find and maintain all 32 solutions to the massively multimodal problem. Unfortunately, the simple incorporation that I used does not seem a candidate for a general solution to this problem. Additionally, further research I have undertaken has indicated that many simple incorporations of multimodal techniques into the mGA are not possible. Specifically, it seems that an mGA using only one template, cannot hope to solve a certain form of multimodal problem which seems easy for the simple GA.

Following is an exploration of the work I have done so far in this research. This includes a description of RTS and a number of theoretical and empirical results incurred for that algorithm. This also includes a description of the pilot incorporation of RTS into the mGA. Also is presented the explanation of why the use of a single template does not seem possible in the incorporation of multimodal technique into the mGA.

3.1 What is RTS?

Tournament selection is, as the name suggests, one way to select the collection of individuals out of one generation that are to survive to the next generation. In regular tournament selection, at each generation tournaments are held between pairs of individuals chosen at random from the population. The winners of these tournaments are then inserted into the next generation. This

makes perfect sense when solving unimodal problems. The winners of the tournaments are deemed to be the better or more fit individuals and they get to move on to be exploited more by the genetic algorithm. However, this selection method tends to make less sense if multiple solutions are desired.

The inherent problem in the multimodal situation for tournament selection can be viewed in two ways. Algorithmically, the takeover time for the best individual under tournament selection is logarithmic in the population size (Goldberg, D.E., & Deb,K. 1991). Therefore, all but one of the niches in a population will tend to disappear relatively rapidly. Another way to view this problem is that regular tournament selection compares apples and oranges. To maintain relatively different solutions, niches should be created that allow these solutions to peacefully coexist. This implies that members of one niche should be prevented from competing with members of another niche. Yet that competition which should be forbidden is precisely what is occurring in regular tournament selection.

To solve this problem, I propose a modification in the regular tournament selection routine that favors placing neighboring solutions in tournaments. I will henceforth refer to this modified algorithm as similarity based restricted tournament selection or RTS. RTS initially selects two elements at random, A and B, from the population and performs crossover and mutation on these two elements resulting in two new elements, A' and B'. A' and B' are then to be placed into the population as in a steady state GA. In a *steady state* GA the elements of the population are not completely replaced every generation. Rather, as pairs of elements are generated by crossover and mutation, they are continuously added to the population and thus immediately available themselves for crossover and mutation. This scheme allows the GA to choose which present elements each inserted pair of elements will replace. For each of A' and B' RTS scans w (*windowsize*) more members of the population and picks the individuals that most closely resemble A' or B' from those w elements. Let these elements be called A" and B". A' then competes with A" and if A' wins, it is allowed to enter the population. A similar competition occurs between B' and B". This kind of tournament will restrict an entering element of the population from competing with others that are too different from it[10].

This modification has the same flavor as that of crowding. As a niche fills up, more and more competitions for positions will be among members of the same niche, allowing other niches to flourish at the same time. However, in contrast to crowding, newly created elements in the population are not blindly inserted into the population replacing nearby members. Instead, they are forced into competitions with nearby members and have to win their competitions before being allowed to take over their competitors' positions. This method also lacks the selection pressure present in the initial stage of the crowding method in that the initial individuals are picked randomly from the population for the purpose of competition and mating. However, Mahfoud (1992) has shown that with reasonable crowding factors, this selection pressure is minimal.

3.2 How Does RTS Preserve Niches?

The basic preservation power of RTS comes from the fact that newly generated solutions are forced to compete with like solutions before being allowed to enter a population. Let's consider a situation where a GA has found and placed an equal number of points at a number of optimal peaks throughout the search space. For simplicity, we assume that all their function values are optimal over the space. This need not be so. In fact they need only form an optimal set[11]. Let the number of peaks thus found be s and the window size be w. Let $w > s : w = cs, c > 1$. When a new

[10]This will be more precisely defined later.

[11]The definition of an optimal set will come at a later section.

individual (A) is formed, to enter the population, it has to compete with the closest individual out of a set of w elements chosen from the population. Assume that A is closest to one of the optimal solutions found by the GA so far, call that solution N0. Since the optimal solutions found so far have the maximum function values found so far, the fitness of A is less than that of N0[12]. Thus it can be assumed that if a copy of N0 is present among the w individuals picked in the window, A will be forced to compete against N0 and A will not enter into the new population. By these assumptions:

> N0 occupies a proportion equal to $1/s$ of the population
> P(no copies of N0 present among w elements without replacement) ¡ P(no copies of N0 present among w elements with replacement) = $(1-(1/s))^w = (1-(1/s))^{cs} < e^{-c}$

This implies that the probability of replacing some element out the optimal population with A is very small. In fact it is exponentially decreasing with the exponent being the number of multiples that the window size is of the number of peaks that are to be maintained. If the window size is a large enough multiple of the number of peaks to be preserved, then the error should be sufficiently small that the underlying probability distribution of the population should not change much during one generation. In a steady state GA, the passing of a generation is identified by the processing of a number of individuals equal to the GA's population size. This would imply that the total proportional error during that generation can then be bounded from above by e^{-c}.

3.3 A Large Class Of Sets Preserved By RTS

For a set of solutions to be maintained by RTS, each newly formed individual's fitness has to be less than the fitness of the optimum closest to it. This is a much less stringent requirement than having all the peaks be maximal over the whole space. It in fact requires only that the peaks be optimal over the part of the space that they are closest to.

More formally, RTS preserves a large class of sets of solutions defined as follows. Let Z be the search space at hand. Let $d : Z \times Z \rightarrow R$ be a distance measure[13] over Z. Let $S = \{S0..SN\}$ be a set of solutions in P(Z). S is an optimal set of solutions under d if $\forall x$ in Z $\exists i : \forall j \; d(x, Si) \leq d(x, Sj) \Rightarrow f(x) \leq f(Si)$. In short, the fitness value of any solution has to be less than the fitness value of the solution in the optimal set it is closest to in the given distance measure. If d is a metric over the space Z then each of the solutions in S is forced to be a local optimum. To see this for $S0$, let $d0$ be the minimum distance from $S0$ to any other solution in S. Any element within B($S0$,$d0$)[14] has to have a fitness less than $S0$. Thus $S0$ is a locally optimal point in the space. Without a loss of generality so are all the other points in S.

This particular class of sets has an interesting interpretation. Each particular solution is forced not only to be a local solution but also optimal over a particular part of the space. Thus the points in a optimal set work together to cover the space. One simple way to view this is that the points in an optimal set are peaks of high fitness separated by valleys of low fitness. Thus an optimal set covers the main features of a search space.

[12] In the restricted case of having only an optimal set this still applies since N0 is the closest solution to A in the optimal set.

[13] We would probably use a distance metric here. The author is not quite sure what would happen if distance metric axioms are violated.

[14] An open ball centered at $S0$ of radius $d0$.

3.4 How Are Competitions Restricted Using RTS?

To assist in understanding how RTS limits the tournament competitions in a GA we define a function F on pairs of elements from the population. F (x,y) = proportion of the population that is nearer to x than y[15]. When a new individual A is created using our genetic operators, it has to enter a competition before being allowed into the current population. Thus, it will have to compete with some individual chosen from the population. We can thus view F(A, chosen element) as a random variable X defined over [0,1]. To extract the important elements of this analysis, we assume that the population is large enough that X can be viewed as a continuous random variable over [0,1]. We then wish to characterize X's probability density function. Although regular tournament selection does not work with quite the same mechanics, we can view it as randomly picking an element's competitor from the population at hand. Thus the pdf of F using regular tournament selection is constant over [0,1]. We now wish to investigate the probability density functions of X under RTS with different values of the window size w.

To pick a competitor for an element of the population A, we randomly select w elements from the population and pick the element that is closest to A. For each element picked out of those w elements, we can define a random variable $X(i)$ as F (A,ith element picked). We note that as each element is picked randomly, the pdf of $X(i)$ is thus a constant (=1) over [0,1]. We are then to choose the element out of these w elements that is closest to A. This corresponds to the element with the smallest r.v. $X(i)$ since the r.v. $X(i)$ equals the proportion of the population that is closer to A than the ith element picked. This is quickly seen to be the first order minimum statistic on w elements from a uniform distribution on [0,1]. That random variable's probability density function can be calculated as follows :

First look at X's distribution function, let j be : $0 < j < 1$
P(X≤j) = P(at least one of X(w) ≤ j) = 1 - P(All of the X(w) > j) = $1 - (1-j)^w$
Therefore the p.d.f. of X is equal to $dP(X \leq j)/dj = w(1-j)^{w-1}$
Verifying this is indeed a pdf, $\int_0^1 w(1-j)^{w-1} = |_0^1 - (1-j)^w = 1$

Qualitatively, we can say that RTS changes the underlying joint distribution of the elements picked together for competition. Quantitatively, by observing the form of the density function itself, we can say even more. We can calculate the probability that an individual A will have chosen for competition an element that is within the 1/wth closest elements of the population to it. This is simply equal to $1 - (1 - 1/w)^w$. This number is bounded from below by $1 - e^{-1}$. Thus at least a constant proportion equal to $1 - e^{-1}$ of the mates picked for competition with A will be proportionally within the closest $1/w$th of the population to A. Similarly, $1 - e^{-c}$ of the comparisons will be within the closest c/wth of the population to A. Thus we can conclude that this simple windowing mechanism is efficiently limiting comparisons to nearby members of the population thus enforcing the principle of localized competition. Additionally, this analysis suggests a strong and direct correlation between the window size w and the number of niches we hope to maintain in the population. Thus we are assured that by having the window size be some constant multiple of the number of niches to be maintained most of the competitions that occur will be within the niches themselves.

[15]Note that this definition is not symmetric because we may be working in a multidimensional space.

3.5 Analysis Of RTS Absorbtion Times

Under selection acting alone, we can further study how a dominant individual begins to take over a population. Of particular interest is the time an individual takes over a proportion equal to $1/(w+1)$ of the population as well as the time it takes a dominant individual to take over one half of the population.

To study these times, we look closely at a population and determine the probability that the dominant individual in the population receives one more copy after one steady state reproduction. Let N be the size of our population and let C be the number of copies that a dominant individual has in the population at a particular time.

P (dominant member receives one more copy) =
P(dominant picked initially) * P (no copies of dominant in w choices | dominant picked)

P (dominant picked initially) = C/N
We use a binomial distribution[16] with w trials and a success rate $(N-C)/N$
P (no copies of dominant in w choices | dominant member picked initially) = $((N-C)/N)^w \Rightarrow$
P (dominant receives one more copy) = $(C/N)((N-C)/N)^w$

Thus the expected number of reproduction attempts before the dominant element gets one more member is $1/P$(dominant gets one more copy) which equals $(N/C)(N/N-C)^w$.

Now we can calculate the expected time before a dominant individual takes over a proportion equal to 1/(w+1) of the population. This is simply equal to $\sum_{I=1}^{I=N/(w+1)} N/I(N/(N-I))^w$ which turns out to equal $N^{w+1}\sum_{I=1}^{I=N/(w+1)} 1/I(N-I)^w$. We now note that over the range of the summation $1/I(N-I)^w$ is monotonic. Calculating its derivative and setting it equal to 0 we find.

$(N-I)^w + i(-w(N-I)^w - 1) = 0 \Rightarrow$
$(N-I) - Iw = 0 \Rightarrow I = N/(w+1)$

Thus the derivative changes only at $I = N/(w+1)$ and we can approximate the sum accurately by looking at $N^{w+1}\int_{I=1}^{I=N/(w+1)} 1/I(N-I)^w \; dI$. We see that over the range of the integration $N^w/e < (N-I)^w < N^w$. Thus we can bound the integral by $N * e\int_{I=1}^{I=N/(w+1)} 1/I \; dI$ which equals[17] $e * Nlog(N/(w+1))$. This gives us niche takeover time of the same order as $Nlog(N/(w+1))$ reproduction steps or $log(N/(w+1))$ generations. This implies that roughly before taking over 1/w+1 of the population the best individual receives no resistance from the windowing mechanism. Thus RTS still allows for the exploitation of information early on in the run in a manner similar to tournament selection with a tournament size of 2.

Now using the same technique we can show how RTS slows down the takeover of a fixed proportion of the population for a variable w. Let's look at the time it takes for an individual to take over one half of the population. As above, this is approximately[18] $N^w + 1\int_{I=1}^{I=N/2} 1/I(N-I)^w \; dI$. This integral can be evaluated by partial fraction expansion and is equal to

[16] This is an approximation that becomes more accurate as the population size get larger but is quite accurate in any case

[17] This factor of e in front is a rough estimate and can probably be reduced to something closer to 1. The intent of the author was to show that this factor was bounded by a constant number.

[18] Since the derivative of $1/I(N-I)^w$ changes only once, we are not missing much in the approximation by using the integral.

$NlogI - Nlog(N-I) + N^{w+1}\sum_{K=1}^{K=w-1} 1/KN^{w-k}(N-I)^k$

At I=N/2 the first two terms vanish and this evaluates to
$N^{w+1}\sum_{K=1}^{K=w-1} 1/KN^{w-k}(N/2)^k = N^{w+1}\sum_{K=1}^{K=w-1} 2^K/KN^w \geq N2^w/w$
At I=1 the first term vanishes and this approximately[19] evaluates to
$-NlogN + N^{w+1}\sum_{K=1}^{K=w-1} 1/KN^{w-k}(N-1)^k \approx -NlogN + N\sum_{K=1}^{K=w-1} 1/K \approx$
$Nlogw - NlogN \leq 0$[20]

The convergence time is seen to be greater than $N(2^w/w + log(N/w))$. Thus the number of generations needed for the best individual to take over half of the population is greater than $2^w/w + log(N/w)$. This calculation shows that increments in w radically affect how quickly an individual can take over a large part of the population. For example with w = 10, even with a small population of about 100, it would take on average more than 100 generations for an individual to take over half the population. Thus RTS provides the GA with much time to do its search before allowing a specific part of the search space to dominate all the other parts of the space.

3.6 The Effects Of RTS On Search

Frequently in the analysis of niching algorithms, the effect of an algorithm on the GA's search takes a backstage seat to the analysis of the algorithm's stability. However, GAs are primarily search mechanisms and proper steps must be taken to insure that whatever mechanism is added to a GA to allow it to solve multimodal problems does not impede its search strategy. What I have done above is to try to characterize the effects of RTS during a GA's run. In the beginning of a run, it allows rapid exploitation of schema information. RTS then slows down an individuals takeover of the population to the point where other niches are also allowed to form.

Within each niche, RTS acts like regular tournament selection pitting the individuals in the niche against each other. This implies that within each niche, RTS accrues the benefits provided by the use of tournament selection. RTS is thus a controlled selective algorithm that does not need a scaling mechanism as fitness-proportionate selection often does. Additionally, it does not depend on the actual fitness assignment scores but only on their rankings of individuals and is thus immune to the effects of order preserving scalings on the fitness-assignments of problems.

3.7 Empirical Testing of RTS

The above analysis indicated that RTS might be a useful tool in solving multimodal problems. It then remained to test RTS on a variety of problems that have been suggested before in the literature. Specifically, I tested RTS on the 5 functions used by Deb (1989). I also tested RTS on a massively multimodal problem (Goldberg et. al.,1992). Following are the function descriptions and the results of using RTS on those problems.

3.7.1 Parameter Setting And The Reporting Of Results

In each of the problems attempted the window size was set at four times the number of peaks to be found. The reported results show the number of individuals at each peak as a function of the number of generations the GA has run. An individual is considered to be resident at a peak if its

[19] Approximating N-1 by N.

[20] for any w < N which seems reasonable

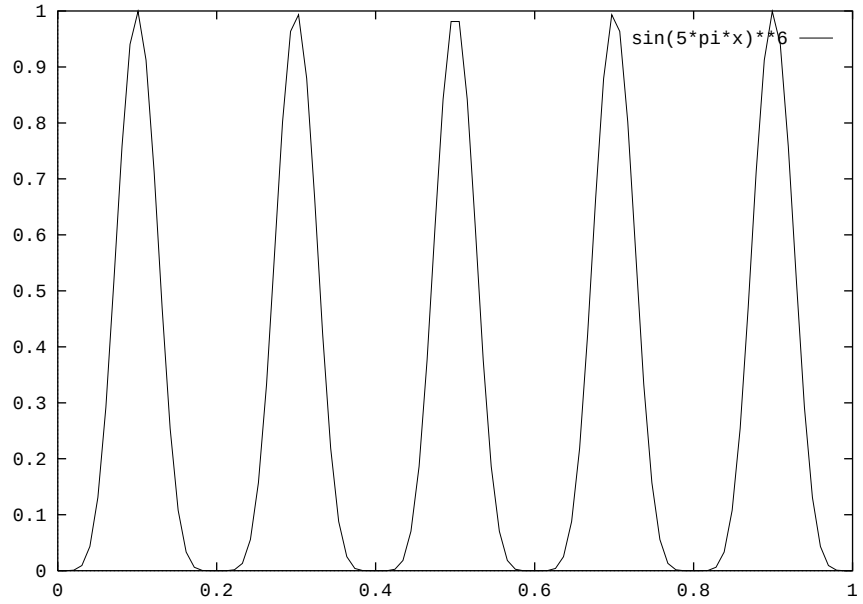


Figure 14: Function 1

fitness exceeds 99 percent of the maximum fitness at that peak. In most cases a population size of 200 was used. For real variables that were coded as bits, string lengths of 15 were used. A crossover rate of 0.4 was used in all the runs. In each of the multi-modal problems attempted, RTS was able to locate all of the peaks of the problem. Thus is each problem, we show the number of copies of individuals at all the peaks and how they are maintained for 100 generations.

3.7.2 Function 1

This is a function defined on [0,1] with five evenly spaced peaks of equal height. The function is defined by f(x) = $sin^6(5\pi x)$. The five peaks occur at x = 0.1, 0.3, 0.5, 0.7, 0.9 and all have a height of 1.0.

3.7.3 Function 2

This function is also defined on [0,1]. It has five evenly spaced peaks of unequal height. It is defined by f(x) = $e^{-2ln2((x-0.1)/0.8)^2}sin^6(5\pi x)$. The peaks occur at the same locations as Function 1 but have respective heights of approximately 1.0, 0.917, 0.707, 0.458 and 0.250.

3.7.4 Function 3

This function is also defined on [0,1]. Its five peaks are unevenly spaced over [0,1] but are of equal height. It is defined by f(x) = $sin^6(5\pi(x^{3/4} - 0.05))$. The five peaks each with a height of 1.0 occur at x = 0.246, 0.450, 0.681 and 0.934.

3.7.5 Function 4

This function defined on [0,1] has five peaks of unequal height unevenly spaced over [0,1]. It is defined by f(x) = $e^{-2ln2((x-0.1)/0.8)^2}sin^6(5\pi(x^{3/4} - 0.05))$.

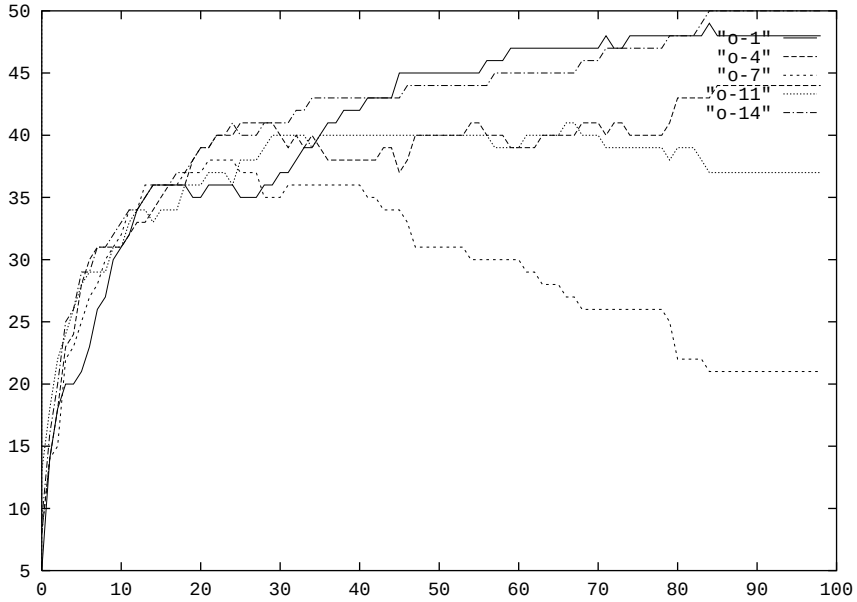


Figure 15: RTS on Function 1

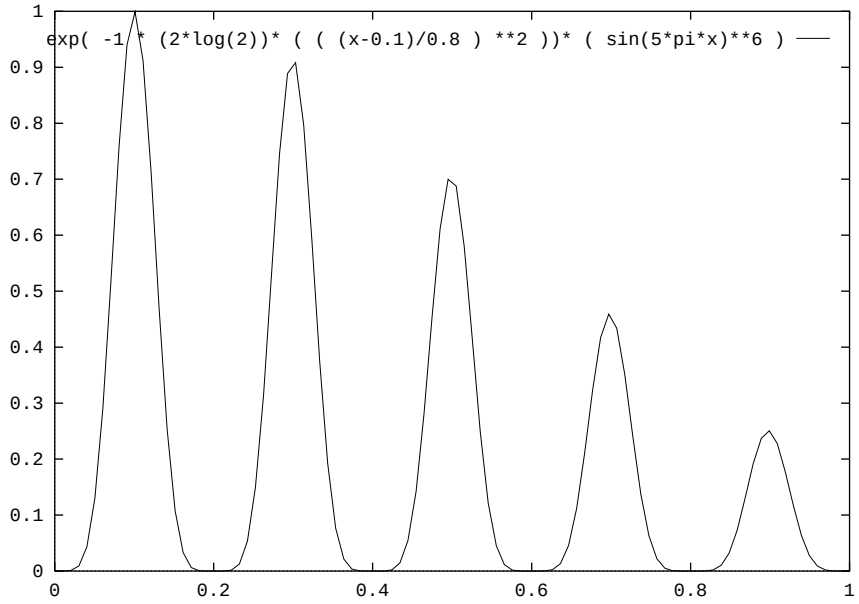


Figure 16: Function 2

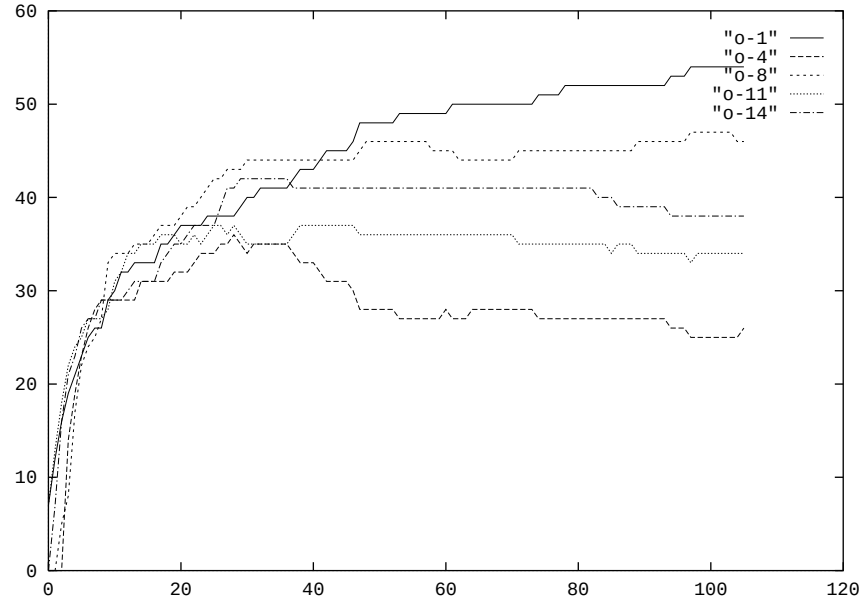


Figure 17: RTS on Function 2

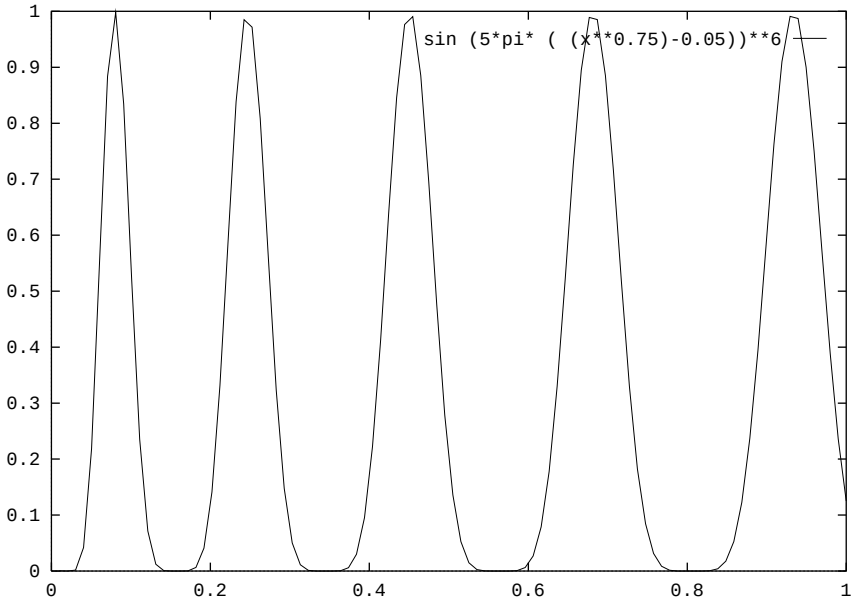


Figure 18: Function 3

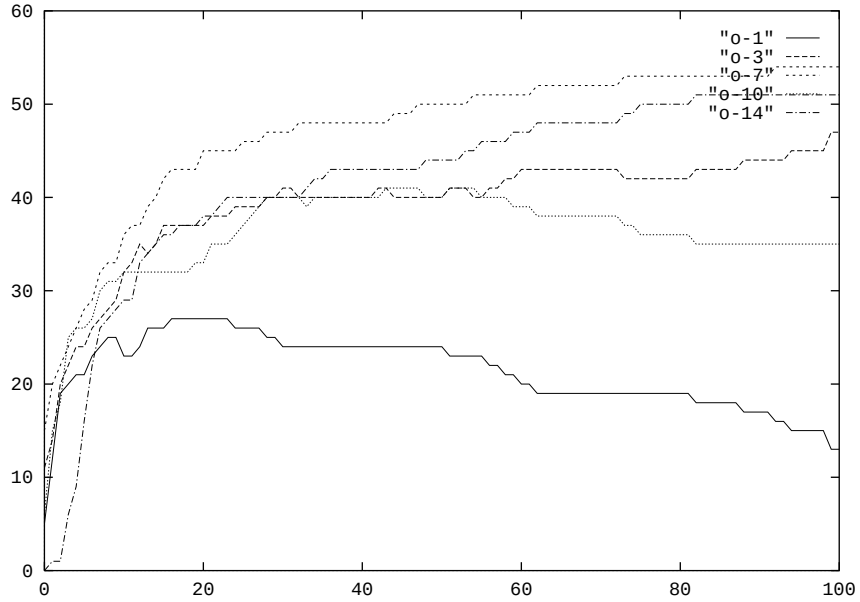


Figure 19: RTS on Function 3

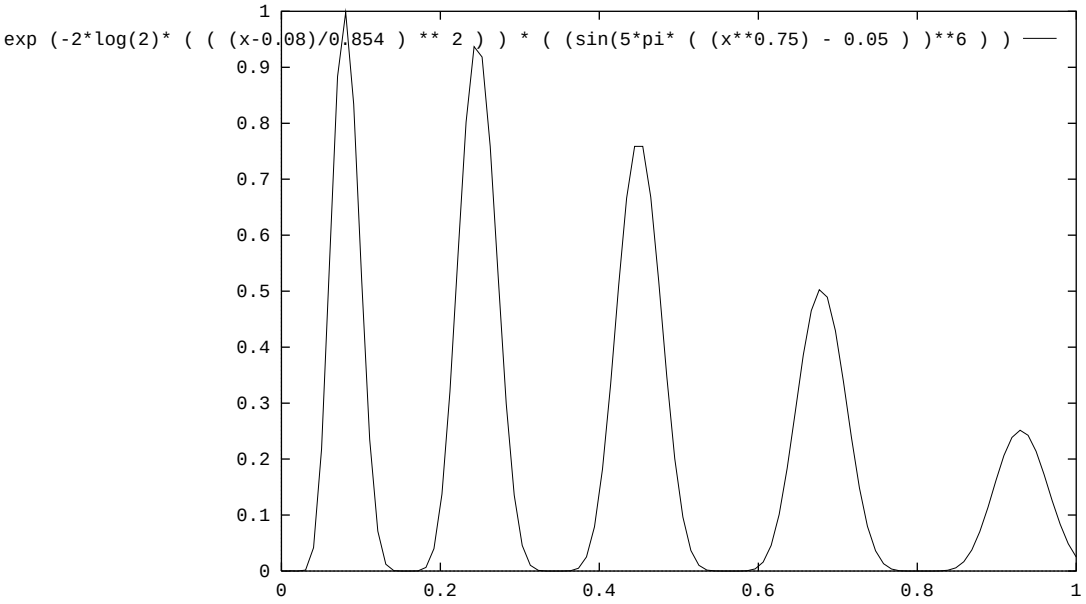


Figure 20: Function 4

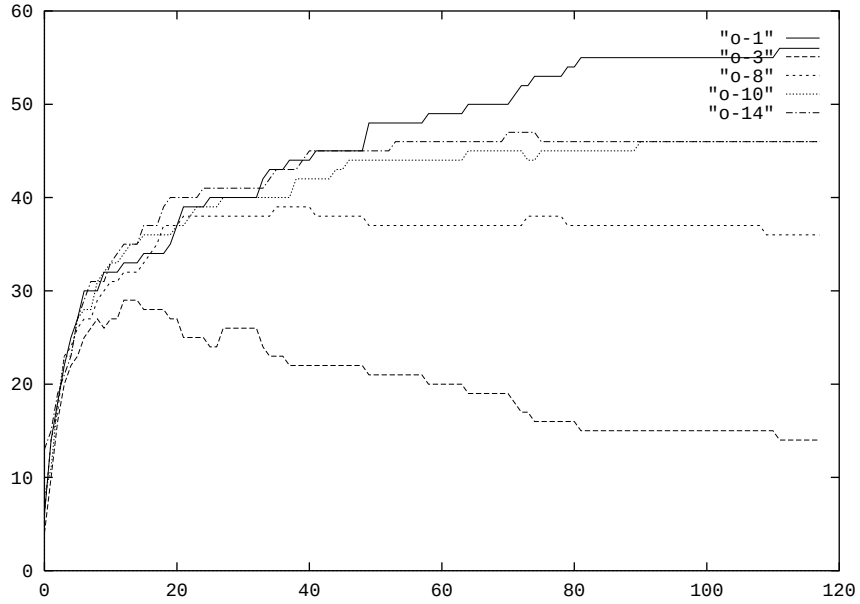


Figure 21: RTS on Function 4

3.7.6 Function 5

This is the two-dimensional function that is a variant of Himmelbau's function (Reklaitis et. al.,1983). Himmelbau's function is defined over [-6,6]×[-6,6] as f(x,y) = $(x^2+y-11)^2+(x+y^2-7)^2$. This is a minimization problem. The function values are multiplied by -1 to convert it into a maximization problem.

3.7.7 Function 6

This is the massively multimodal problem posed by Goldberg, Deb & Horn (1992) It is based on 6 bit subfunctions of unitation[21]. In this case, 6 bit bimodal deceptive functions are used and 5 of them are concatenated to form a 30 bit problem. This function has over a million local optima and 32 well spaced global optima. The population size used here was 1280 and the window size used was 32. An individual was considered a solution only if it was one of the 32 global optima.

3.8 Connections To Past Work

The work on RTS is related to the work on crowding done by De Jong (1975) and Mahfoud (1992) in that it too works by inducing a certain maximal capacity on the size of niches. However, as opposed to crowding, the selection pressure in RTS seems better suited to the exploitation of information present in schema samples. Also, in contrast to deterministic crowding RTS seems to take into account the fact that in Genetic Algorithms search, the population size is a limited resource. In deterministic crowding the stability of a maintained set of niches seems to be related to the effects of crossover on that set of niches. It is not clear whether this would be in general beneficial or detrimental to search in a variety of real world problems. On the other hand, the number of peaks that RTS attempts to search for and exploit information from, is directly related to the window size chosen for a particular application.

[21]Where the function value depends only on the number or 1s in a substring

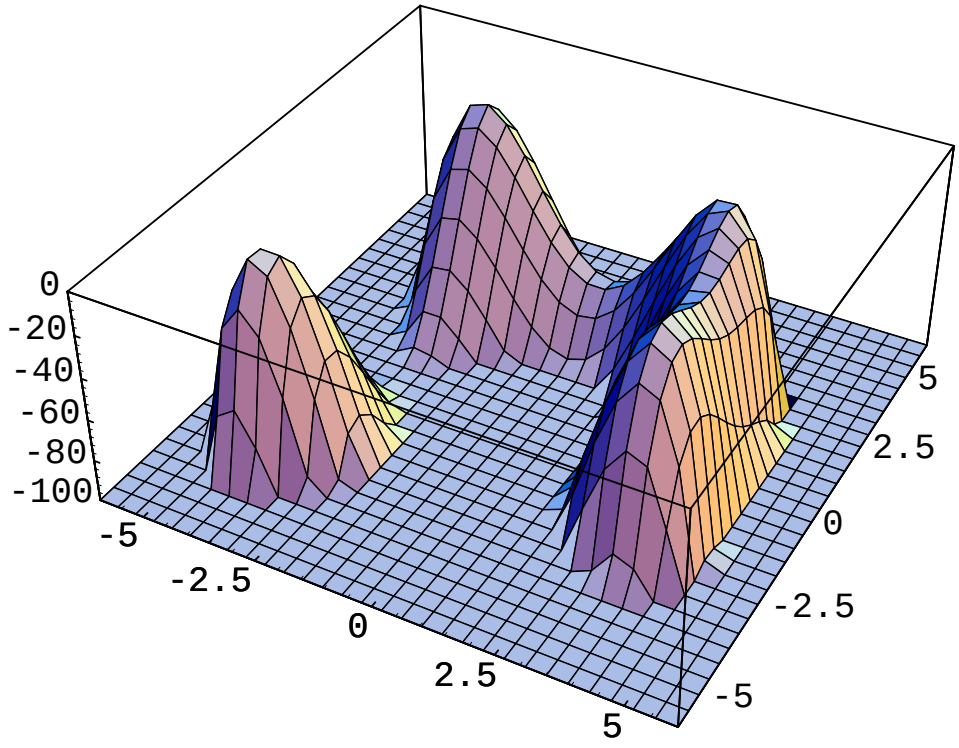


Figure 22: The peaks of Function 5

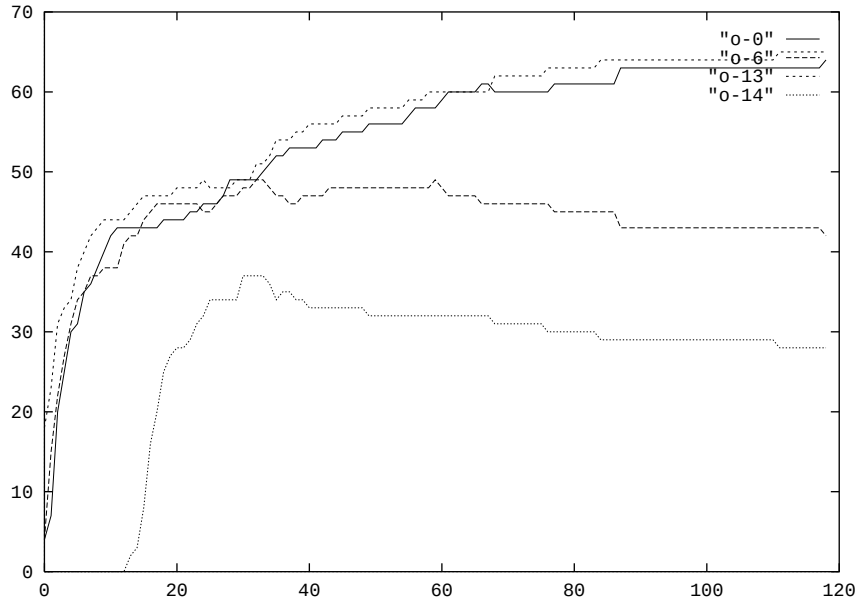


Figure 23: RTS on Function 5

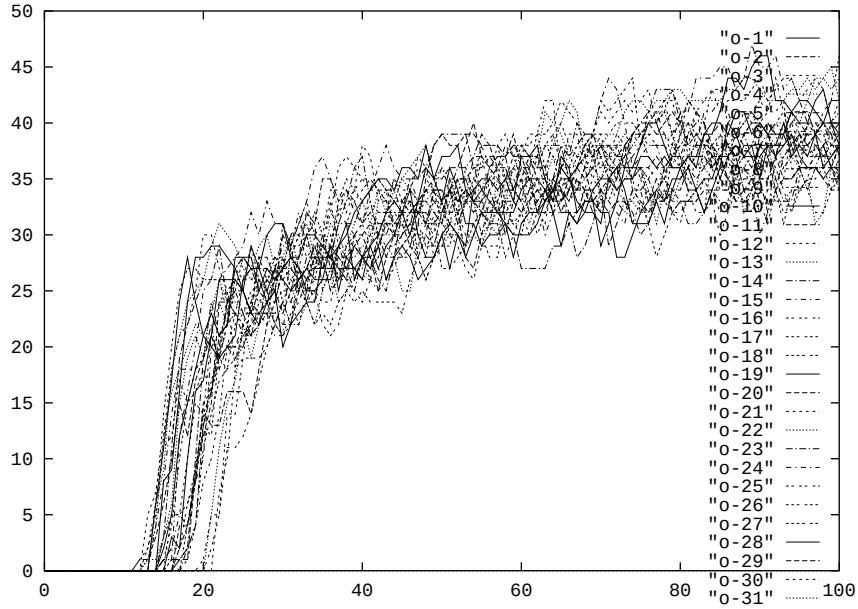


Figure 24: RTS on Function 6

RTS also bears a similarity to the immune system model (Smith, Forrest & Perelson, 1993) that uses GAs to evolve a cooperative set of antibodies to cover a number of antigens. That work was not directly applicable to multimodal function optimization because the antigens had to be separately provided throughout the search. However, if the antigens that are to be covered are taken at random from the current population, and the closest individual to cover them is then forced into a competition with that antigen, the result is an algorithm that is very similar to RTS. Though RTS was inspired by the paradigm of forcing local competitions to climb to local optima, it is interesting to note its similarities with a number of other algorithms.

3.9 Concluding remarks on RTS

RTS is a selection based diversity maintenance approach that implements the concept of localized competition in GAs. The discussion above presents the theory on why this technique should work on a number of problems as well as a class of problems that RTS is well-suited for. This theory has been shown correct in a number of empirical tests. The development of RTS has now reached a point where a comparison with other multimodal techniques is possible. Thus RTS, along with fitness-sharing represent the two possible techniques for use in a multimodal mGA.

3.10 Initial thoughts on the multimodal mGA

When I began work on developing a multimodal mGA, my initial hopes were that the mGA could be patched together with one of the above diversity preserving techniques and a multimodal mGA would result. However, there are certain problems with this approach to a multimodal mGA. A multimodal mGA has to preserve diversity along many different fronts. As with a regular mGA, when identifying the different building blocks of a solution, the multimodal mGA, henceforth referred to as the MmGA, has to maintain building blocks at all the positions or functional roles of the coding string. Additionally, within each set of functional roles, the MmGA has to preserve

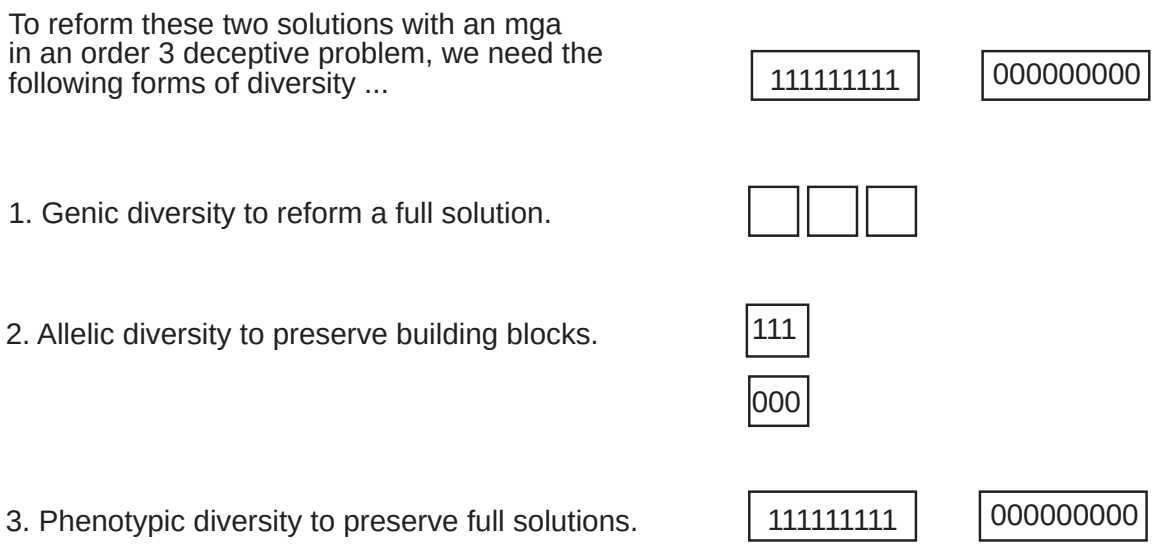


Figure 25: Different forms of diversity preservation required in an MmGA

various bit specification alternatives. Furthermore, when reforming entire solutions, some criterion must to be used to separate the different solutions gotten by the MmGA and preserve them long enough for the user to observe them. Unfortunately this additional criterion need bear no relation to the diversity requirements discussed above.

Figure 25 shows the different forms of diversity preservation required in a possible MmGA implementation. The problem presented in this figure has two global maxima. One maximum consists of ones at all its bit positions while the other consists of zeroes. The MmGA also has to preserve all the functional roles in a problem so as to be able to recombine these functional roles into a full solution specification (#1 in the figure). The MmGA has to preserve both alternatives at each set of functional roles so as to be able to form both global solutions (#2 in the figure). Finally, the MmGA has to preserve both final solutions in its population (#3 in the figure).

It may be possible to combine the first two forms of diversity preservation needed in an MmGA. This could be accomplished by combining the genic and allelic difference metrics into a genic-allelic distance metric over a building block space and then preserving diversity according to that metric. The mGA needs a high degree of selection pressure throughout its deletion phases. Thus without a diversity preserving technique that maintained with a high degree of probability all the separate niches, an MmGA would quickly converge to cover only a few of the required functional roles. This particularly stringent requirement essentially rules out using a form of crowding or a spatial separation technique. This is because the preservation capabilities of crowding are suspect and because spatial separation techniques cannot be guaranteed to look all the different niches in a space.

Fitness-sharing and RTS are thus the only proven techniques that remain viable in this situation among the different techniques investigated. However, the use of fitness-sharing in this situation presents certain technical difficulties. In the different reduction stages, the MmGA will have to process strings of different lengths. Some method must be specified to define the σ_{share} parameter in each and every one of these stages. Furthermore, simple minded use of fitness-sharing relies on the existence of a close relationship between the phenotypic distance measure used and the genic-allelic distance measure used by the fitness-sharing algorithm. A *phenotypic* distance measure is a

distance measure in the solution space. Otherwise, the MmGA may have to preserve building blocks that are close genotypically to each other to be able to reform phenotypically different solutions. This requirement denies us the use of fitness-sharing in such a situation.

3.11 Single template vs. multiple templates

Currently, the mGA uses a single template to evaluate building blocks. With an appropriate diversity-preserving algorithm to keep multiple different building blocks in the population, we might expect that an initial form of an MmGA could be built using this single template approach. I have been able to identify two problems with such an approach. The first problem I encountered was when implementing this particular form of an algorithm to solve the massively multimodal problem proposed by Goldberg, Deb & Horn. I was using one of the 32 global solutions as the mGA's template and RTS to attempt to preserve all the good building blocks in this problem. What I noted in that problem is that there was actually no selective pressure in that instance for occurrences of all the other good building blocks over selecting random allele-gene combinations from the template. Thus as opposed to the regular mGA where the good building blocks always presented a positive signal over the deceptive portions of the template, in the MmGA this was not necessarily the case. In fact in this particular problem, even the global building blocks did not provide a positive signal over the template. This problem will recur in all situations in which the template's fitness is equal to or greater than the fitnesses of the other solutions sought out. In a unimodal problem, this is never the case. A few solutions can be proposed to this problem such as awarding higher fitnesses to building blocks that do not match the template and other forms of modification of the underlying fitness function. However, my beginning investigation into this area seems to imply that these modifications are not in general feasible.

This conclusion is based on the idea that in multimodal problem spaces there may exist different contexts in which certain building blocks are detectable. Any particular template chosen by the MmGA would then lie within one of these contexts. Thus, the further along we are in the deletion steps of the MmGA, the more this template will bias building blocks towards the context in which it lies. I have demonstrated this form of context dependence in a simple setting with two contexts and two global solutions. This formulation is described in Appendix A. This form of interaction between the separate building blocks in a problem more or less precludes the use of a simple template scheme in this situation. It also indicates the need for template representation for several and possibly all of the multimodal solutions sought in a particular problem. This of course necessitates a change in the operation of the MmGA that has yet to be considered.

3.12 One success to build on

Using one particular algorithm, I was able to achieve success on the massively multimodal problem with a MmGA. I maintained a separate template for each individual in the population. In the selection phases of this algorithm I would then select a template at random from the population and force the competing building blocks to use this template thus forming two fully specified individuals which can then compete. Additionally, I entered these two resultant solutions back into the template population and thus, the template population itself was evolving concurrently with the building block population. The selection into the template population was then overseen by the RTS diversity preserving algorithm.

At the end of the deletion phase of the MmGA, this algorithm had provided and maintained all 32 solutions to the massively multimodal problem with no initial linkage assumptions. In contrast,

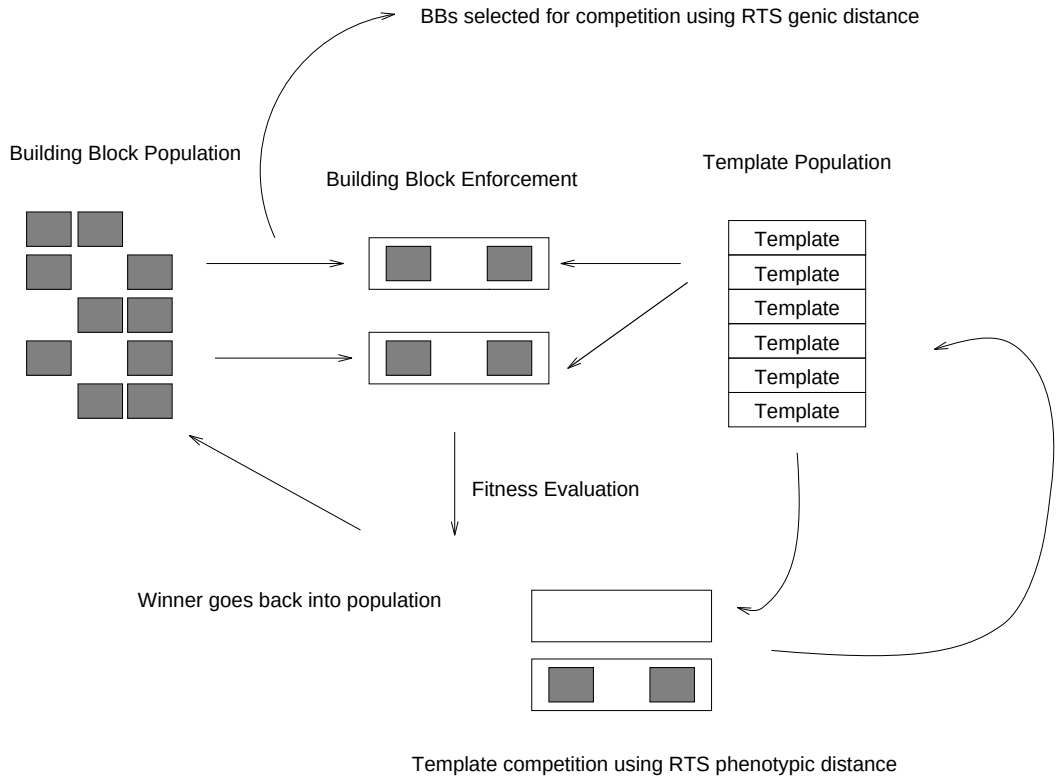


Figure 26: A possible MmGA algorithm

several runs of the simple GA with fitness-sharing but without the assumption of tight linkage had produced not even one of the 32 global solutions to the problem. This simplistic integration of RTS and the mGA had accomplished what the simple GA and fitness-sharing could not.

This method took advantage of several factors in isolating building blocks from particular solutions. It involved forcing pairs of building block individuals to use random templates in the selection phase. This form of selection naturally gave an advantage to individuals which contained good building blocks as those individuals were both more likely to transmit good information from their original template as well as not disrupt the information present in their newer template. Additionally, this form of exchange in the selection phase took advantage of the fact that although the massively multimodal problem has thirty-two solutions it really has only one context. That is, the good building blocks are good in all the different areas of the search space. Taking advantage of this fact allows the algorithm to select its template for selection at random from the population. Figure 26 shows the basics of how this algorithm worked.

Taking advantage of the fact that building block structures mix better with many templates is a logical thing to do. That is, after all, the basis for regular GAs on tightly linked structures. However, the assumption that multimodal problems have only one context is more dubious. As Appendix A shows, the fact that problems of bounded difficulty can be created that contain multiple contexts and multiple solutions is just an indication of how common and simplistic this form of problem may be. Thus, the second factor that this algorithm took advantage of in this particular situation may not be present in all of the problems we might want to tackle using a MmGA. There still exists the possibility that the means used in this algorithm would extend somehow to work well in a multiple context multimodal problem. The way to begin this extension is not yet clear.

4 What remains to be done?

As of this moment, two multimodal techniques, fitness-sharing and RTS have been identified as possibilities for incorporation into the mGA search engine. Further investigation of the merits of these two algorithms depends on the availability of a reasonable set of requirements for them to be tested against. What is needed here is an analogue to the notion of bounded difficulty that would be applicable in a multimodal setting. Thus one of the first tasks that remain to be completed for the fulfillment of this work is the definition of bounded difficulty in a multimodal setting. As the concept of bounded difficulty has created a standard for testing unimodal GAs, the multimodal equivalent to this concept should create a standard for the testing of multimodal techniques for GAs.

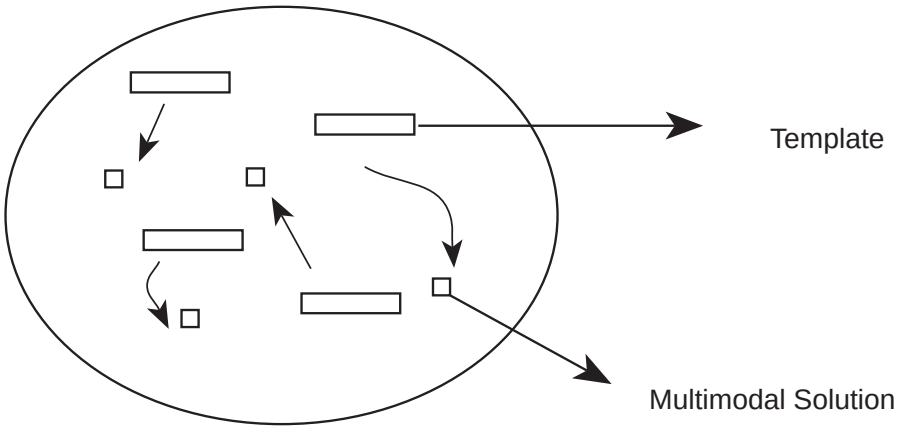
The next step along this line of development will be the evaluation of fitness-sharing and RTS, as well as other possible multimodal techniques. This evaluation will use as the basis for its judgment, the rate of success of these two algorithms on the boundedly difficult multimodal problems described above. However, as this evaluation of both techniques will take place within the framework of the simple GA, these tests will still work under the luxury of the tight linkage assumption. The result of this task should be the choice of a multimodal technique to integrate with the mGA as well as a better understanding of the effects of fitness-sharing and RTS on the GA's search mechanism.

The next phase in the development of the MmGA will then be the integration of a multimodal technique into the mGA. It seems highly likely that a reworking of the mGA's template scheme will be in order here. This will involve changing the way that building blocks are evaluated in the mGA, as well as formulating an algorithm for the generation and propagation of the multiple templates. At present, my instincts point to a formulation in which the template population co-evolves along with the building block population. It seems possible to slowly force the template population to reorganize itself around the various niches in a problem thus providing an implicit representation of all possible required contexts in that problem. This co-evolution would provide separate environments in which different building blocks could grow, thus removing the need to explicitly maintain diversity at the building block allelic level. The work that I had done on the massively multimodal problem involved the co-evolution of building block and template populations in a single context problem. Though that work was successful, I stated the difficulty in extending it to a multiple context problem. In addition, that work explicitly maintained allelic-genic diversity among the building blocks, and thus did not test my suspicion that with a multiple template scheme, allelic diversity would not have to be explicitly maintained with a multimodal technique. However, that work suggests that the co-evolution of solutions as contexts might well be possible. The co-evolution of templates and building blocks as presented in Figure 27 presents one possible solution to the problem of multiple contexts in multimodal problem solving.

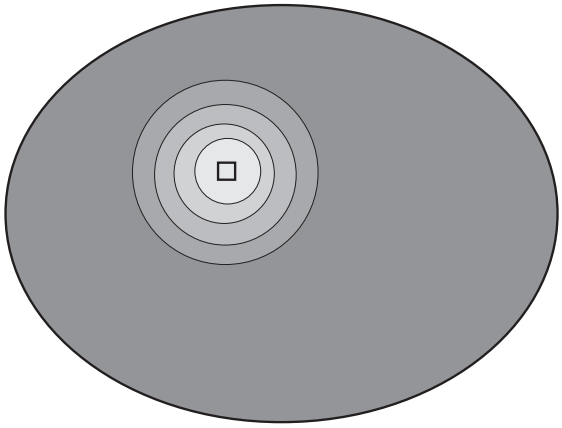
The next step past the development of a MmGA algorithm would be to provide empirical proof that it could solve boundedly difficult multimodal problems correctly and in reasonable time. This work will provide a minimum level of competence that can be expected when the MmGA is used. Beyond this point, the only remaining task is that this algorithm must prove itself capable in a challenging real world problem.

The final results of my work should be as follows:

- A definition of bounded difficulty for multimodal deceptive functions.
- An investigation of fitness-sharing and RTS working on problems of bounded difficulty.
- The integration of the appropriate multimodal technique into the mGA for the creation of a multimodal MmGA



Templates can be made to evolve towards different solutions.



The color here represents the signal strength of a building block over a particular template. It is hypothesized that signal strength increases upon proximity to the final solution.

When the template is near to a solution, that solution's building blocks should have a higher signal when imposed upon that template.

Figure 27: Template and building block co-evolution at work

- An investigation of the MmGA working on problems of bounded difficulty.
- A successful application of the MmGA in a real world environment.

5 Summary and Conclusion

This document has presented a justification for doing research on the incorporation of a multimodal technique into the mGA. It began by stating the need for a robust multimodal GA in many real world settings. It then demonstrated two possible problems with the extension of the current simple GA methodology to work in a real world multimodal setting. Simply put, the two problems were as follows: the simple GA does not scale well to the handling of larger problems; and the multimodal techniques currently available have not been analyzed as to their effects on the GA's search procedure. The document then detailed how the scaling problem of simple GAs has been addressed in the mGA's development. This action identified the two main tasks of this work: the analysis of various multimodal techniques for their effects on the GA's search procedure; and the incorporation of one of those techniques into the mGA. This document then presented a review of work begun in fulfillment of these tasks. This included the development of RTS and a pilot incorporation of RTS into the mGA. This document then concluded with a list of tasks to be undertaken for the completion of this work.

From a GA's standpoint, real world problems are difficult. Real world problems offer the challenge of having their building blocks randomly strewn throughout a coding string due to an almost arbitrary initial coding. They offer the challenge of nonlinear interactions between the subparts of these building blocks, thus leading a GA astray with their low order components. Often, their fitness functions have to be approximated leading to the need for multiple solutions in particular problems. Other times what is needed is not only a global optimum but a distribution of various optima in the space to enhance our understanding of the nature of these problems. In short, real world problems often combine two aspects of GA difficulty that have been handled separately but whose nonlinear interaction forces a combined approach when the two aspects are simultaneously present.

The mGA was developed as an approach to dealing with problematic interactions between groups of genic positions on coding strings. Various multimodal techniques have been developed to deal with problems in which multiple solutions are sought out. However, the interaction between this pair of roadblocks to GA robustness is complex. The possibility of a straightforward combination of the basic mGA with a multimodal technique seems unlikely. GA practitioners must be given adequate tools with which to approach their problems and thus further investigation into the combination of a modified mGA with a multimodal technique seems warranted and indeed necessary.

6 Acknowledgment of support

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

Appendix A - A Multiple Context Problem

My intention in this appendix is to show that there exist certain multimodal problems of bounded difficulty that cannot be solved with the mGA technique using a single template scheme. This implies that work done in extending the mGA to handle multimodal problems will have to include at the least a multiple template configuration so as to be able to handle these types of problems. The way I approach this is by designing a problem of bounded difficulty that has two global optima. I will show that any particular choice of one template will not be able to support the building blocks necessary to reform the two global solutions.

A.1 The problem definition

This is a simple six bit problem that is composed of two three bit subfunctions. The basic idea behind this formulation is that in each three bit subfunction, there will exist two good building blocks, one being 111 and the other 000. Left alone, this would lead to four global solutions. However, I will add a component of negative interaction between the two subfunctions. If the first three bit subfunction leans towards one of the global building blocks, then it will positively reinforce that particular global solution at the other 3 bit subfunction. That is if the first three bits were 110 then it would positively reinforce each 1 it saw in the next three bits. This leads to the problem having only two global solutions, 111111 and 000000.

The precise problem definition is as follows. Let p be the number of 1s in the first three bit subfunction and let q be the number of 1s in the second 3 bit subfunction. $p0 = 3-p$ and $q0 = 3-q$ then represent the number of 0s in those two subfunctions respectively. The fitness function will have four components. The first component, $C1$ is equal to 10 if $p = 0$ or $p = 3$ and otherwise equal to 0. The second component, $C2$, likewise is equal to 10 if $q = 0$ or $q = 3$ and 0 otherwise. These two components identify the building blocks 111 and 000 to be the two good building blocks in the problem. The next two components are the interaction components between the two building blocks. If $p > p0$ then $C3 = INTER * p * q$ else $C3 = INTER * p0 * q0$. Similarly, if $q > q0$ then $C4 = INTER * p * q$ otherwise $C4 = INTER * p0 * q0$. INTER is a parameter that controls the amount of interaction between the two component building blocks. The resulting fitness function is then $C1 + C2 + C3 + C4$.

A.2 Results and conclusions

Setting INTER=2, we see the following: The three bit building blocks 111 and 000 are still optimal at both subfunctions according to a schema average calculation. Thus their representation will begin by growing in the population. This results implies that the problem is of bounded difficulty no higher than order three. However, when we investigate how this function would work with a single template scheme, we observe the following surprising result: for any template that is used, if we rank the order three building blocks in one subfunction in terms of their fitnesses, one of the global building blocks 111 or 000 will have the highest fitness among that set; while the other will have the lowest fitness. Thus for any chosen template, given any reasonable selection algorithm on building blocks, we will lose one of the building blocks necessary to reform the two global solutions. This results seems to imply that there are multimodal problems of bounded difficulty that cannot be handled by a single template scheme.

My explanation for this phenomenon is as follows: a building block's signal varies depending on what region of the search space is sampled. When we pick a particular region to examine, it may

well be that not all the building blocks we are interested in can survive in that region and thus the choice of only one region limits us. Similarly, the choice of only one template forces the mGA to examine building blocks largely from the standpoint of that particular region and that will limit its applicability in certain applications.

The obvious approach to solve this problem is to increase the number of contexts within which building blocks are evaluated. However, just how many contexts are needed is not clear. One possible answer to this question is that the minimal number of contexts needed is equal to the number of solutions we wish to maintain. This comes from the natural assumption that any particular building block must have a high signal when imposed upon the locally optimal solution which it is to form. This assumption is what has inspired me to start work on the co-evolution of the template and building block population inside the mGA.

References

[1] J. D. Bagley. The behavior of adaptive systems which employ genetic and correlation algorithms. *Dissertation Abstracts International*, 28(12):5106B, 1967. (University Microfilms No. 68-7556).

[2] J. E. Baker. Adaptive selection methods for genetic algorithms. In J. J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 101–111. 1985.

[3] D. Beasly, D. R. Bull, and R. R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125, 1993.

[4] L. B. Booker. Intelligent behavior as an adaptation to the task environment. *Dissertations Abstracts International*, 43(2):469B, 1982. (University Microfilms No. 8214966).

[5] D. J. Cavicchio. Adaptive search using simulated evolution. (University Microfilms No. 25-0199), 1970.

[6] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. Richards. Punctuated equilibria: A parallel genetic algorithm. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154. 1987.

[7] R. J. Collins and D. R. Jefferson. Selection in massively parallel genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 249–256. Morgan Kaufmann, San Mateo, CA, 1991.

[8] Y. Davidor. A naturally occurring niche and species phenomenon: The model and first results. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 257–263. Morgan Kaufmann, San Mateo, CA, 1991.

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

[10] K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50. 1989.

[11] N. Eldrege. Punctuated equilibria: an alternative to phyletic gradualism. *Models of Paleobiology*, pages 82–115, 1972.

[12] C. M. Fonseca and J. Fleming. Genetic algorithms for multiobjective optimization, formulation, discussion and generalization. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423. 1993.

[13] D. R. Frantz. Non-linearities in genetic adaptive search. *Dissertation Abstracts International*, 33(11):5240B–5241B, 1972. (University Microfilms No. 73-11,116).

[14] D. E. Goldberg. Computer-aided gas pipeline operation using genetic algorithms and rule learning. *Dissertation Abstracts International*, 44(10):3174B, 1983. (University Microfilms No. 8402282).

[15] D. E. Goldberg. Simple genetic algorithms and the minimal, deceptive problem. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 74–88. Morgan Kaufmann, Los Altos, CA, 1987. (Also TCGA Report 86003).

[16] D. E. Goldberg and C. L. Bridges. An analysis of a reordering operator on a GA-hard problem. *Biological Cybernetics*, 62:397–405, 1990. (Also TCGA Report No. 88005).

[17] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93. 1991. (Also TCGA Report 90007).

[18] D. E. Goldberg, K. Deb, H. Kargupta, and G. Harik. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 56–64. 1993.

[19] D. E. Goldberg, K. Deb, and B. Korb. Messy genetic algorithms revisited: Studies in mixed size and scale. *Complex Systems*, 4:415–444, 1990.

[20] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989. (Also TCGA Report 89003).

[21] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. 1987.

[22] K. Goldberg, D. E. amd Deb and J. Horn. Massive multimodality, deveption, and genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 37–46. 1992.

[23] M. Gorges-Schleuter. Asparagos an aynchronous parallel genetic optimization strategy. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 422–427. 1989.

[24] P. B. Grosso. *Computer simulation of genetic adaptation: Parallel subcomponent interaction in a multilocus model.* (University Microfilms No. 8520908), 1985.

[25] J. H. Holland. Schemata and intrinsically parallel adaptation. *Proceedings of the NSF Workshop on Learning System Theory and its Applications*, pages 43–46, 1973.

[26] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[27] J. Horn. Finite markov chain analysis of genetic algorithms with niching. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 110–117. 1993.

[28] J. Horn and N. Nafpliotis. Multiobjective optimization using the niched pareto genetic algorithm. Technical report, Illinois Genetic Algorithms Laboratory, University of Illinois, 1993.

[29] S. W. Mahfoud. Crowding and preselection revisited. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 27–36. 1992.

[30] M. L. Mauldin. Maintaining diversity in genetic search. *Proceedings of the National Conference on Artificial Intelligence*, pages 247–250, 1984.

[31] H. Mühlenbein, M. Schomisch, and J. Born. The parellel genetic algorithm as function optimizer. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 271–278. Morgan Kaufmann, San Mateo, CA, 1991.

[32] Z. A. Perry. Experimental study of speciation in ecological niche theory using genetic algorithms. *Dissertation Abstracts International*, 45(12):3870B, 1984. (University Microfilms No. 8502912).

[33] C. C. Pettey and M. R. Leuze. A theoretical investigation of a parallel genetic algorithm. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 398–405. 1989.

[34] G. V. Rekalitis, A. Ravindran, and K.M. Ragshell. *Engineering optimization: Methods and applications.* New York: Wiley, 1983.

[35] J. D. Schaffer. Some experiments in machine learning using vector evaluated genetic algorithms. *Dissertation Abstracts International*, 46:2760B, 1984. (University Microfilms No. 85-22492).

[36] R. E. Smith, S. Forrest, and A. S. Perelson. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149, 1993.

[37] G. Syswerda. Uniform crossover in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 56–64. 1989.

[38] R. Tanese. Distributed genetic algorithms for function optimization. *Dissertation Abstracts International*, 50:5180B, 1989. (University Microfilms No. 90-01722).

[39] D. Thierens and D. E. Goldberg. Mixing in genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 38–45. 1993.

[40] S. W. Wilson. Classifier system learning of a Boolean function. Research Memo RIS-27r, The Rowland Institute for Science, Cambridge, MA, 1986.

[41] S. Wright. *Evolution and the genetics of populations: a treatise.* Chicago, University of Chicago Press, 1968.