# Fundamental Principles of Deception in Genetic Search

**L. Darrell Whitley**
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523
whitley@cs.colostate.edu

## Abstract

This paper presents several theorems concerning the nature of deception and the central role that deception plays in function optimization using genetic algorithms. A simple proof is offered which shows that the only problems which pose challenging optimization tasks are problems that involve some degree of deception and which result in conflicting k-arm bandit competitions between hyperplanes. The concept of a *deceptive attractor* is introduced and shown to be more general than the deceptive optimum found in the deceptive functions that have been constructed to date. Also introduced are the concepts of *fully deceptive problems* as well as less strict *consistently deceptive problems.* A proof is given showing that deceptive attractors must have a complementary bit pattern to that found in the binary representation of the global optimum if a function is to be either fully deceptive or consistently deceptive. Some empirical results are presented which demonstrate different methods of dealing with deception and poor linkage during genetic search.

**Keywords:** deception, tagged bits, GENITOR, uniform crossover.

## 1 Background

Based on earlier work by Bethke (1980), Goldberg (1987) introduced the concept of *deception* in order to better understand what kinds of situations are likely to create difficulty for a genetic algorithm when performing a function optimization task.

The fundamental theorem of genetic algorithms, the "schema" theory, attempts to explain the ability of genetic algorithms to search complex problem spaces effectively by showing that genetic algorithms change the sampling rates of hyperplanes in an n-dimensional hypercube corresponding to a binary encoding of the solution space (Holland 1975; Goldberg 1989a). Low level building blocks corresponding to large general hyperplanes in the space are recombined to drive the search toward more specific regions in hyperspace that display above average fitness. A problem is deceptive if certain hyperplanes guide the search toward some solution or genetic building block that is not globally competitive (Goldberg 1989a). Most earlier work in this area has concentrated on the construction of difficult or deceptive problems (Goldberg 1987, Goldberg, Korb and Deb 1989, Tanese 1989). More recently this work has expanded to to consider the relationship between deception, hyperplane variance, hyperplane dominance and genetic algorithms as dynamical systems (Grefenstette and Baker 1989, Liepins and Vose 1991, Whitley 1991, Goldberg and Rudnick 1991).

Hyperplanes are represented by bit combinations known as schemata. Consider an order-3 problem involving a search space encoded with 3 bits. This simple space can be represented by a cube, labeled so that adjacent corners in the cube differ by only a single bit. Using * as a don't care symbol, let schema 0** refer to a face in this cube. 0** is a schema of order-1, since it defines a partition of the search space occupied by all strings with a single bit (0) in the position indicated. The value of the hyperplane (or in this case, the simple plane) represented by 0** is denoted f(0**) and is calculated by averaging the fitness value of all the strings that match the template 0** (i.e., the average value of the 4 strings that form the corresponding face of the cube). The "order" of a hyperplane refers to the number of 0 or 1 bits specified in the schema representing that hyperplane. "Order" also relates information about the number of strings contained within the hyperplane represented: a hyperplane of order-1 contains 50% of the strings in the search space. Thus, the schema ***1****0*...* is of order-2 and the corresponding hyperplane contains (i.e., its schema matches) 25% of all the strings in the search space.

In the case of a fully deceptive order-3 function the hyperplane information represented by order-1 and order-2 schemata in the search space leads the search away from the global optimum, and instead directs the search toward what we will refer to as a *deceptive attractor*. Assuming the bits 111 represent the global optimum and the deceptive attractor is 000, full order-3 deception implies that the following relationships hold for the lower-order schemata:

```
f(0**) > f(1**)          f(00*) > f(11*), f(01*), f(10*)
f(*0*) > f(*1*)          f(0*0) > f(1*1), f(0*1), f(1*0)
f(**0) > f(**1)          f(*00) > f(*11), f(*01), f(*10)
```

where f(S) gives the fitness of either a string or the fitness of a schema, S, and is calculated by averaging all of the strings contained in the hyperplane represented by that schema. In other words, all lower-order hyperplane information associated with schemata whose presentation is made up of some subset of these 3 bits leads the search away from the global optimum of 111. A fully deceptive order-3 problem which satisfies the above inequalities has been defined by Goldberg, Korb and Deb (1989) where the bit strings have the following values.

| | | |
|---|---|---|
| Deceptive Function 1 | f(000) = 28 | f(001) = 26 |
| | f(010) = 22 | f(100) = 14 |
| | f(110) = 0 | f(011) = 0 |
| | f(101) = 0 | f(111) = 30 |

Part of the motivation for building deceptive problems is to better understand what kinds of conditions can inhibit genetic search from finding a globally optimal or near optimal solution. Problems that have the ability to seriously mislead a genetic search have been labeled *GA-hard* problems. As shown later in this paper, fully deceptive subproblems coupled with poor linkage can result in GA-hard problems.

## 2 Definitions

In attempting to reason about the nature of deception, it is necessary to distinguish different kinds of deceptive situations and to use terms with some precision. I have attempted to keep the definitions offered in this paper as consistent as possible with the existing literature.

First we need to refer to some concepts relating to genetic algorithms in general. In particular we wish to discuss hyperplane competitions, but certain competitions are particularly important. A *primary hyperplane competition* of order N involves the complete set of primary competitors, where the *primary competitors* in a hyperplane competition of order N are the set of $2^N$ hyperplanes having schemata with N bit values in the same locations. For example, *0**0, *0**1, *1**0 and *1**1 are competitors in a primary hyperplane competition of order 2.

The "global winner" of a primary hyperplane competition is that hyperplane which has the highest fitness value among a set of competitors, where the fitness value of a hyperplane is the average fitness of all strings that are contained in that hyperplane. This does not necessarily imply that this hyperplane correctly leads to or has bits consistent with the globally optimal solution.

Two more basic concepts are useful for discussing deception. Hyperplane X "contains" hyperplane Y when X is a lower order hyperplane and Y is a higher order hyperplane such that Y has exactly the same bit values in exactly the same locations as X, but also has additional bit values in positions occupied by don't care symbols (*) in X. For example, 0** contains 01*; Note that 0** also contains the same strings as 01* (i.e., 010 and 011) but contains other strings as well (i.e., 001 and 000). This leads to a more specific relationship between hyperplane competitions that is important to deception. Two primary hyperplane competitions of order N and order K (where $K < N$) are "relevant" to one another when collectively the hyperplane competitors of order K "contain" the hyperplane competitors of order N. Thus, the primary hyperplane competition between 11***, 10***, 01*** and 00*** involves a set of hyperplanes that collectively contains the primary hyperplane competition between 111**, 110**, 101**, 100**, 011**, 010**, 001** and 000**. Given some hyperplane competition of order N, "relevant" hyperplanes of order K (where $K < N$) are represented by schemata which have bit values (either 1 or 0) in some proper subset of the locations in which bits occur in the order N hyperplane competition.

*Deception* implies that the global winner of some hyperplane competition of order N has a bit pattern that is different from the bit pattern of the global winner for some "relevant" lower order hyperplane competition. In other words, deception implies that these lower order competitions have their own "global winners" which do not have the same bit values as the global winner of the relevant hyperplane competition at order N.

A *deceptive problem* is any problem of a specific order N that involves deception in one or more relevant lower-order hyperplane competitions. Deception implies that there exists one or more relevant lower order hyperplane competitions that can potentially guide a genetic search away from the global winner of the hyperplane competition at order N. It does not imply the problem is fully deceptive, or that there is sufficient deception to misguide a genetic algorithm. This usage appears to be consistent with Goldberg's (1987) discussion of the minimal deceptive problem, which involves deception, but is not fully deceptive.

This usage is fairly broad; most problems may involve some degree of deception, since in general we would not expect that *all* lower level hyperplane competitions be consistent with relevant higher order hyperplane competitions. However, as will be shown in the next section, this particular definition separates optimization tasks into two extremely interesting classes: those that are not deceptive (and thus theoretically easy) and those which are deceptive.

A *fully deceptive problem* (or subproblem) of order-N is deceptive when all relevant lower-order hyperplanes lead toward a deceptive attractor. A *deceptive attractor* is a hyperplane of order N other than the true global winner which is supported by relevant lower level hyperplane competitions. For a fully deceptive problem, it can be shown that a deceptive attractor can only be that hyperplane represented by the schema which has a bit pattern that is the complement of the "global winner" of the hyperplane competition at order N (i.e., if the "global winner" of the hyperplane competition of order 3 is **1**1*1**, then the complement and potential deceptive attractor is **0**0*0**). A proof appears later in this paper. This usage of the term "fully deceptive" appears consistent with that found in Goldberg, Korb and Deb (1989).

It turns out that it is useful to distinguish different degrees of deception according to criteria other than order when describing and defining the deceptive attractor.

A *consistently deceptive problem* (or subproblem) of order-N is one in which none of the relevant lower-order hyperplanes lead toward the "global winner" of the primary hyperplane competition at order N we are interested in, but all of the relevant lower-order hyperplanes do not necessarily lead toward the deceptive attractor–except for the order-1 hyperplanes which (as will be shown) can only guide the search toward the "global winner" or the deceptive attractor of the hyperplane competition at order-N. It can be shown that choosing the deceptive attractor to be the complement of the global winner of the hyperplane competition at order N is also a *necessary* condition for the existence of a consistently deceptive problem because of the order-1 hyperplanes involved. Note that a fully deceptive problem is always a consistently deceptive problem, but a consistently deceptive problem is not necessarily a fully deceptive problem.

The term *deceptive function* will refer to a consistently deceptive problem where the number of bits used to encode the solution space corresponds to the order of the deception. A deceptive function is always consistently deceptive and may be fully deceptive. Thus, a consistently deceptive order-3 problem with a 3 bit encoding constitutes a deceptive function.

A *deceptive building block* of order N refers to a situation where some hyperplane H has a higher fitness value than its primary competitors, but all of the relevant lower order hyperplane competitions between schemata composed of subsets of bits in the same locations are misleading. A deceptive building block is similar to the concept of deceptive function, except the deceptive building block may correspond to a particular hyperplane competition that is part of a larger function optimization problem; thus the deceptive building block is in some sense a "subproblem" of some larger function. The concept of a "building block" in genetic algorithms is used to refer to schemata of low order (and often characterized by short defining length) and above average fitness. Building blocks are thought to be important to the process of genetic search. By concentrating on the hyperplane competitions involving building blocks (which should be well represented in sample populations and should be relatively stable under recombination) the genetic algorithm is able to generate good partial solutions and, through recombination, construct higher level schemata and complete strings. Empirical results (such as those presented later in this paper) show that if no building blocks are available, the ability of the genetic algorithm to produce global solutions is impaired. The deceptive building block is in some sense an "anti-building-block" since it represents a hyperplane competition that can mislead the genetic algorithm. A "deceptive building block" does not necessarily have a short defining length; in fact a deceptive building block with a long defining length is one kind of situation that makes the deception more problematic. However, the deceptive building block does involve competitions between hyperplanes with above average fitness.

A *deceptive hyperplane* is a hyperplane that incorrectly leads the search away from some more specific higher order hyperplane that in fact is superior to its competitors. It follows that the lower order hyperplanes which create a deceptive building block are "deceptive hyperplanes" *with respect to that particular building block.*

The notion of a deceptive hyperplane is defined with respect to a particular deceptive building block because the bits that make up the global winner of a hyperplane competition in a deceptive building block may not be the same bits that appear in another hyperplane competition or in the string that corresponds to the global solution of the problem. For example, in an order-4 fully deceptive function, the order-1 hyperplanes are deceptive hyperplanes with respect to the order-4 building block that constitutes the global solution, but the order-1 hyperplanes are not deceptive with respect to any of the order-2 or order-3 competitions, since these all consistently lead to the deceptive attractor. To define "deceptive hyperplanes" with respect to the global solution alone would be too restrictive since this would ignore the forces that drive genetic search. However the definition used here does not prevent us from referring to a hyperplane as being deceptive with respect to the global solution.

## 3 A Note On Convention

Throughout this paper the following convention will be used. *Let the bit "1" in any binary mapping be interpreted as indicating agreement with the bits in the optimal solution or global winner of a hyperplane competition rather than just a single binary string which maps to a particular value.* This interpretation generalizes the representation so that arguments concerning any specific set of strings hold for other arbitrary strings. This convention can also be operationalized. Liepins and Vose (1990) show that a simple translation function exists which will remap the entire binary space, thus making it possible to reassign the global optimum to an arbitrary bit string. Goldberg (1990) also notes that the function $(x \oplus c')$ translates the optimum to the "all-1s" position and translates all other strings to the correct corresponding positions in Hamming space, where x is each string in the binary encoding and c' is the complement of c, the location of the current optimum. The function $\oplus$ is a bitwise exclusive-or (bitwise addition modulo 2).

## 4 The Only Challenging Problems are Deceptive

As stated earlier, most problems may involve some deception, since in general we would not expect that *all* lower level hyperplane competitions will be consistent with relevant higher order hyperplane competitions. The following theorem supports the claim that the only challenging optimization tasks are problems involving some degree of deception.

**THEOREM 1:** *Given a fitness function for a problem representing some optimization task with a binary encoding of length L,* **if** *1) no deception occurs in any of the hyperplanes associated with that particular binary encoding and 2) the winners of the L order-1 hyperplanes can be correctly determined,* **then** *the global optimum of the function is determined by the one string contained in the intersection of the L order-1 hyperplane competition winners.*

**PROOF BY CONTRADICTION:**

Assume that deception does not occur in any hyperplane of the fitness function and we determine the correct winners of each of the order-1 hyperplanes. The intersection of the order-1 schemata will identify exactly one string as the candidate for the global optimum. If the candidate is not the global optimum, then at least one of the order-1 hyperplanes is deceptive, thus generating a contradiction.

**QED**

While this theorem makes some idealistic assumptions, it can be operationalized. If no deception occurs, then we can solve a problem in the following fashion. For each bit location we can directly solve a 2-arm bandit problem involving the corresponding two schemata *...*0*...* and *...*1*...*. Alternatively, the competition can be solved using statistical methods. The use of statistical methods potentially provides a means for dealing with the issue of noisy sampling–and thus the problem of determining the appropriate hyperplane competition winners. If no deception is present, such methods will determine (to some desired level of reliability) the probable value of the bits that belong in the corresponding position in the binary string representing the global solution to the problem. Certain difficulties can of course be encountered: what if two competing order-1 hyperplanes have equal or near equal value? Note that if there is no deception, all subpartitions of the order-1 hyperplanes must lead toward the global solution (to do otherwise would imply deception). Thus, if we can solve *any* of the order-1 hyperplanes, we can use the solution to narrow the search space. If there is no deception, it does not matter in what sequence the order-1 hyperplanes are solved. This means that we can break ties and resolve ambiguous competitions if the winners of competing order-1 hyperplanes can be chosen in any partition of the reduced search space. While it may not be possible to always operationalize this idea, it is often possible: we have already found the globally optimal solution to several test problems that have appeared in the genetic algorithm literature by solving the order-1 hyperplanes using statistical methods (Das and Whitley 1991).

This fundamental theory of deception seems almost trivial. Researchers who work with genetic algorithms have long known that hyperplanes at numerous levels play an important role in genetic search. However, this knowledge has not been directly related to deception and to what that implies about Holland's (1975) analogy to the 2-armed or k-armed (multi-armed) bandit problem.

The 2-armed bandit analogy for the order-1 hyperplanes goes as follows. Assume we have a slot machine with 2 arms. Each arm has a different payoff with an associated mean and variance. The problem is to determine which arm has the higher expected payoff. We would like to sample the two arms, while at the same time minimizing our expected loss. To do this, we should allocate exponentially more trials to the observed best. In actual practice, this may involve some initial sampling to determine the observed best. If we are dealing with some order-1 hyperplane competition, then *1*** and *0***, for example, would represent the two arms. If we are dealing with hyperplanes greater than order-1, then this becomes a "k-arm" competition. What Holland was also able to show is that the genetic algorithm can potentially allocate trials to hyperplanes in a near optimal fashion, allocating exponentially more trials to the observed best in numerous hyperplane competitions. The fact that this happens simultaneously for many different hyperplane competitions of different orders is referred to as *implicit parallelism.*

Deception and the 2-armed bandit analogy directly relate to issues raised by Grefenstette and Baker (1989) concerning the role of implicit parallelism in genetic search. They point out that a genetic algorithm does not solve 2-armed (or k-armed) bandit competitions between hyperplanes because "the genetic algorithm does not perform uniform sampling from the hyperplanes in the population, at least not after the initial generation" (Grefenstette and Baker 1989: 24). As an example, they suggest the following assignment of fitness values for some optimization function:

$$f(x) = \begin{cases} 2 & \text{if } x \,\epsilon\, 111{*}...{*} \\ 1 & \text{if } x \,\epsilon\, 0{*}{*}{*}...{*} \\ 0 & \text{otherwise} \end{cases}$$

A uniform sample of strings will indicate $f(0*...*) > f(1*...*)$. However, as genetic search proceeds, the schema 111*...* dominates over its hyperplane competitors and thus, 1*...* will come to be represented in a greater proportion of the strings in the population than 0*...* because the population is no longer uniformly sampling the universe of all strings. Grefenstette and Baker suggest that this represents a flaw in the 2-armed bandit analogy because the competition between 0*...* and 1*...* is not solved correctly. They also point out that the genetic algorithm is not implicitly solving all possible hyperplane competitions correctly, which does in fact show that *implicit parallelism* does not act in a ubiquitous fashion. However, John Holland (personal communication) never intended that implicit parallelism should imply that *all* hyperplane competitions are simultaneously solved through genetic search, only that many competitions are occurring simultaneously.

Having established that the only challenging problems are deceptive problems, there is additional cause for being cautious when characterizing the nature of implicit parallelism. Grefenstette and Baker's point about the limitations of implicit parallelism has serious implications. To see why this is true, first note that Grefenstette and Baker's problem is a deceptive problem. The hyperplane 0*...* is deceptive with respect to the hyperplane competition involving 111*...*. It is not a fully deceptive problem (or subproblem): $f(*1*...*) > f(*0*...*)$ and $f(**1*...*) > f(**0*...*)$. Furthermore, the problem is not GA-hard since, as Grefenstette and Baker correctly point out, a genetic algorithm will quickly converge to strings that reside in the hyperplane represented by 111*...* (Das and Whitley 1991).

The occurrence of a deceptive problem will always produce at least two primary hyperplane competitions (which are analogous to two different k-arm bandit problems), which have solutions that involve different bit patterns. If a genetic algorithm effectively and consistently allocates exponentially more reproductive trials to the true global winner in one of these two hyperplane competitions, such that the schema receiving exponentially more trials becomes ubiquitous in the population *then the genetic algorithm will fail to correctly solve the alternate hyperplane competition.* The genetic algorithm cannot continue to allocate exponentially more trials to the observed best (assuming the observed best is in fact the global winner) in both hyperplane competitions, since the two global winners are incompatible and only one can become ubiquitous in the population.

**CONJECTURE 1:** *Assuming that sampling noise does not corrupt hyperplane evaluations, the 2-armed (or k-armed) bandit analogy fails to apply to a particular hyperplane competition if and only if deception is involved: a genetic algorithm can only correctly solve one of two competitions that involve a set of hyperplane competitors of order K and some relevant hyperplane competition of order N, where $K < N$ and the global winner of the competition at order K is deceptive with respect to the global winner of the order N competition.*

This conjecture is motivated by the following observations. If no deception occurs in a function optimization task, then implicit parallelism has an opportunity to solve all of the hyperplane competitions correctly and the 2-arm bandit analogy potentially holds. On the other hand, deception will alway create conflicting hyperplane competitions whose global winners have conflicting bit patterns. It may be possible that conflicting winners can be represented in the population in such a way that conflicting patterns can coexist. Such an arrangement, however, would appear to be inconsistent with a critical part of the 2-armed bandit analogy which indicates that the observed best should be allocated an exponential number of reproductive trials. It seems likely that solving all hyperplane competitions correctly would only be reasonable if no deception exists, and as shown, such problems could theoretically be solved without a genetic algorithm. It is disturbing that the same deception that makes global optimization non-trivial also works to undermine implicit parallelism in exactly those hyperplane competitions that would appear to be most critical. The impact of deception on critical hyperplane competitions strikes at the very heart of the theoretical foundations of genetic algorithm.

The empirical evidence suggests that the results of conflicting competitions can go either way. If the higher order schema representing the global winner in a fully deceptive building block has a long defining length, the lower order hyperplanes will typically win their k-armed bandit competitions and the higher order hyperplane will not. On the other hand, if the higher order schema representing the global winner of a fully deceptive building block has a tight linkage and is well represented in the chosen population size, the global winner of the higher order hyperplane competition typically wins its hyperplane competition and the deceptive lower order hyperplanes will not.

These observations lead to a more limited view of implicit parallelism. Many hyperplane competitions are being solved simultaneously so that the search is directed toward the global solution, but this need not imply that all hyperplane competitions are correctly processed by the genetic algorithm. Thus, we might expect that some deception need not be sufficient to mislead a genetic algorithm if the majority of the hyperplane competitions are resolved so as to lead toward global solutions. However, a much better understanding of implicit parallelism is needed to theoretically anchor genetic algorithms.

## 5 Constructing a Fully Deceptive Function

We now turn our attention to fully deceptive and consistently deceptive functions. Fully deceptive functions can be constructed using Walsh transforms (Goldberg 1989b, 1989c; c.f., Bethke 1980). However, we have found that arbitrary fully deceptive functions of any order greater than 2 can be constructed in the following fashion. (Liepins and Vose (1990) have independently developed a similar algorithm). Sort the binary strings in terms of their relative distance in Hamming space. When the strings are sorted in this fashion (with respect to their distance from the optimal solution or its complement) the distribution of strings into subgroups containing specific numbers of 1s and 0s is binomially distributed. For example, in an order-3 problem such a sorted sequence would be 111 followed by 011, 101 and 110, followed by 001, 010 and 100 and finally ending with 000. Note that there exists more than one such sequence: subgroupings having the same number of "1" bits such as 001, 010 and 100 can be arranged in any sequence. Once an ordering is chosen, number the strings 1 through N. String 1 will be the global optimum; string $N$ will be the deceptive attractor, the complement of string 1. Assign string 2 some positive base value $B$ (in the simplest case this can be zero). For any string $X$, where $X > 2$, the deceptive function $F_d$ assigns values as follows: $F_d(X) = F_d(X-1) + C$, where $C$ is any positive constant. Finally, the optimum is assigned a value: $F_d(1) = F_d(N) + C$.

The following fully deceptive order-4 function was constructed using the algorithm just outlined. In this case, the base value $B$ is 0 and the constant $C$ is 2. The problem is such that all lower order hyperplane information leads the search away from 1111 and towards 0000.

```
Deceptive Function 2

f(1111) = 30    f(0100) = 22    f(0110) = 14    f(1110) = 6
f(0000) = 28    f(1000) = 20    f(1001) = 12    f(1101) = 4
f(0001) = 26    f(0011) = 18    f(1010) = 10    f(1011) = 2
f(0010) = 24    f(0101) = 16    f(1100) = 8     f(0111) = 0
```

Note that every relevant schema of order 3 or less which is composed exclusively of 0s and *s is superior to all primary hyperplane competitors.

Deceptive Function 2, like all the artificial deceptive functions that have been constructed to date, has a deceptive attractor that is a local optimum in Hamming space since it is superior to any point adjacent to it in Hamming space. The reason that problems such as Deceptive Function 1 and Deceptive Function 2 are fully deceptive is clear: in Hamming space the deceptive optima have an "attracting basin" that covers the entire space except for the single spike that represents the true optimum. However, as will be shown, the deceptive attractors of fully deceptive functions need not always be local optima in Hamming space.

## 6 The Deceptive Attractor Theorem

The following theorem indicates that for any function mapping a set of strings to a set of values, the resulting assignment will not be fully deceptive at the particular order indicated unless there is a deceptive attractor which is the complement of the global optimum. However, as will be shown, this does not imply that the complement must have a high fitness value that is competitive with the global optimum. This theorem pertains not only to deceptive functions, but also "deceptive building blocks."

The term "attractor" suggests a dynamical system characterized by a stable attractor. The term "deceptive attractor" is used loosely here, but the evidence suggests that 1) if a problem is fully deceptive, 2) the deceptive attractor is the complement of the global optimum, and 3) recombination always disrupts the order-N information which identifies the global optimum, then the deceptive attractor will be a stable point for many populations that sufficiently sample the relevant deceptive hyperplanes at orders less than N. More specifically, if we define some idealized genetic algorithm where the search is dominated by the deceptive hyperplanes at order N-1 or less when search a fully deceptive problem of order N, then the deceptive attractor must be a stable point for any fully deceptive function: all of the order N-1 hyperplane competitions lead toward it (for a specific example, see Whitley 1991).

Nevertheless, the theorems that follows in the reminder of this paper do not rely on notions from dynamics systems, but rather are based on observations about the relationship of binary strings and hyperplanes in an N-dimensional hypercube. These definitions are therefore based on static relationships. If the genetic algorithm is modeled as a dynamical systems, these theorems may no longer hold, but they should at least offer a point of departure.

**THEOREM 2:** *In order for a function or building block of order N to be consistently deceptive in all relevant lower-order hyperplanes, the deceptive attractor must be the complement of the string which represents the global optimum in the deceptive function, or in the case of a deceptive building block, the deceptive attractor must be the complement of the schema representing the "global winner" of the relevant primary hyperplane competition at order N that is superior to all of its competitors.*

**PROOF BY CONTRADICTION:**

Assume a deceptive attractor of order N exists such that it shares at least one bit value in common with the "global winner" of the relevant hyperplane competition at order N. Choose one bit which has a value shared by the global winner and the deceptive attractor. Let G represent the bit that agrees with the global winner and let D represent the bit that agrees with the deceptive attractor. These are chosen so that these bits have the same value (0 or 1). The problem is not consistently deceptive at the specific order indicated unless $f(*...*G*...*) < f(*...*D*...*)$, but this is impossible, since the hyperplanes are represented by schemata that have the same bit and thus the hyperplanes have the same fitness.

Let C represent the complement of bits G and D (which are in fact the same bit). To be deceptive, it is necessary that $f(*...*G*...*) < f(*...*C*...*)$, but this also implies that $f(*...*D*...*) < f(*...*C*...*)$. Therefore, if the problem is deceptive in all hyperplanes, this hyperplane competition must be won by an order 1 schema containing the bit C and not D. Therefore the proposed deceptive problem with a deceptive attractor containing the bit D is not consistently deceptive in all hyperplanes at the particular order indicated.

**QED**

Expressed simply, the relevant order 1 hyperplane competitions can only agree with the global winner at order N or its complement. Therefore, to be fully or consistently deceptive, the only hyperplane that can function as a deceptive attractor is the complement of the global winner of the relevant hyperplane competition at order N. Again, this assumes a static view of the hyperplane relations. When a dynamical point of view is taken the situation becomes much more complex, but certain observation can be made. First, the theorem has stronger implications for fully deceptive functions, since all lower order hyperplanes lead to the same point; this should make it possible to make stronger claims about the status of the deceptive attractor as a true attractor in a dynamical system. Also, note that there will exist other attractors if the problem is not fully or consistently deceptive. There will also be other attractors if the lower order hyperplanes do not dominate the search (due to sampling bias or preservation of higher order schemata); in fact the global solution is one such competing attractor. Nevertheless, in a fully or consistently deceptive problem the order-1 hyperplanes must direct the search toward the deceptive attractor and this will have an impact on the other relevant hyperplane competitions. This leads to the following corollary, again defined from a static perspective.

**COROLLARY 2.1:** *If the deceptive attractor at order N is not the complement of the global winner of the relevant hyperplane competition at order N, then the problem is neither fully nor consistently deceptive.*

This follows from the proof of the deceptive attractor theorem since that proof considers hyperplanes *...*G*...*, *...*D*...* and *...*C*...*. Since these represent some set of hyperplanes with a single bit specified, if *...*D*...* is in fact not a deceptive hyperplane, then the problem is neither fully deceptive nor consistently deceptive.

## 7 The Deceptive Attractor and Local Optima

The above argument shows that the deceptive attractor must be the complement of the "global winner" of the relevant hyperplane competition at order N. However, limiting our consideration to deceptive functions for a moment, *it does not follow that the complement of the global winner of the hyperplane competition at order N will necessarily have a high fitness value.* The hyperplanes in a fully deceptive function can lead toward a "deceptive attractor" that is not a local optimum in Hamming space. The assertion also applies to deceptive building blocks, since we can construct such a problem by concatenating several deceptive functions where the deceptive attractor is not the local optimum. Consider the following example:

```
Deceptive Function 3

f(1111) = 30    f(0100) = 27    f(0110) = 5    f(1110) = 0
f(0000) = 10    f(1000) = 28    f(1001) = 5    f(1101) = 0
f(0001) = 25    f(0011) = 5     f(1010) = 5    f(1011) = 0
f(0010) = 26    f(0101) = 5     f(1100) = 5    f(0111) = 0
```

This function was constructed so as to be easy to understand. It is a fully deceptive function but the deceptive attractor has a fitness value that is only 1/3 that of the global optimum. The deceptive attractor is clearly not a local optimum in Hamming space because it is weaker than any of the strings adjacent to it in Hamming space. The deceptive attractor at order N must be hidden by strong strings located adjacent to it in Hamming space if the problem is to be fully deceptive. It turns out this factor is not as critical to consistently deceptive problems. In any case, the example given above shows that the deceptive attractor itself need not be a strong competitive string itself, but it must be cloaked by strong strings adjacent to in Hamming space to hide its weakness. But we can also make some observations about the fitness of the deceptive attractor relative to the strings (or relevant schemata) near it in Hamming space.

**THEOREM 3:** *A deceptive attractor of order-N for a binary encoded problem cannot maintain full deception at order-N if it is weaker than any string or schema which differs from the deceptive attractor by exactly two bits.*

**PROOF:**

Consider a fully deceptive problem of order N. Let $Z$ be a binary string (or schema) with bit values of 0 representing the deceptive attractor and $f(Z)$ the value associated with the deceptive attractor. Let $X$ be the string (or schema) that differs from $Z$ by two bits (i.e., it has two 1 bits) and which also has a higher value than any other string that differs from the deceptive attractor by two bits. If it is shown that $f(Z)$

must be greater that *f(X)*, then *f(Z)* must be greater than the value of any string that differs by exactly 2 bits from the deceptive attractor.

Two other strings (or schemata) are relevant. *S1* and *S2* are two unique strings (or schemata) that differ by one bit from both the deceptive attractor and the string (schema) *X*. In other words, they both have a single 1 bit. Assign the names to these strings so that *S1* represents the string (or schema) that has a fitness value greater or equal to *S2*. This also implies a hyperplane competition between these four strings (or schemata) involving schemata of order N-1. Assume *S1* and *X* share a bit we call B1; Also, *S2* and *X* share a different bit in common we will call B2. String *S1* and *X* compete against *S2* and *Z* when we consider schemata of order N-1 that replaces the bit B2 by a don't care operator, *. The remainder of the proof involves only simple algebra. The deceptive attractor theorem and the definition of full deception implies:

$$f(Z) + f(S2) > f(S1) + f(X)$$

which implies

$$f(Z) > f(S1) - f(S2) + f(X).$$

Since $f(S1) \geq f(S2)$, let $f(S1) - f(S2) = K$, where $K$ is a non-negative integer. This implies:

$$f(Z) > f(X) + K, \quad \textbf{therefore } f(Z) > f(X).$$

**QED**

The obvious difficulty with maintaining full deception at order N occurs in the order N-1 hyperplanes involved. To make the proof more concrete, consider some order-4 deceptive function. Let 0000 be *Z,* let 1001 be *X,* and let 1000 and 0001 be the two uniquely defined strings that differ by one bit from both the deceptive attractor and the string 1001. Pick either string as having the higher fitness value; for illustration purposes assume $f(1000) > f(0001)$. This also implies a relevant hyperplane competition between schemata of order N-1. Full deception implies $f(000*) > f(100*)$, which implies

$$f(0000) + f(0001) > f(1000) + f(1001).$$

The reader can do the remaining math.

**COROLLARY 3.1:** *In a fully deceptive function, if the deceptive attractor does not reside at a local optimum in Hamming space, a local optimum must exist at some string that is adjacent to the deceptive attractor in Hamming space. In other words, either the deceptive attractor or some string that differs from the deceptive attractor by 1 bit must be a local optimum in Hamming space.*

This follows from Theorem 2. If the deceptive attractor must have a higher fitness value than any string that differs from the deceptive attractor by exactly 2 bits, then the deceptive attractor (and the strings that differ from the deceptive attractor by a single bit) are surrounded by points in Hamming space that are known to be weaker than the deceptive attractor. This implies that either the deceptive attractor is a local optimum, or one of the strings that differs from the deceptive attractor by a single bit is a local optimum in Hamming space.

We now look at a *consistently deceptive problem* that is not fully deceptive. While the requirements for full deception make it necessary that the deceptive attractor not have too low a fitness value, the order N-1 hyperplanes (e.g., f(000*) versus f(100*) in the above example) where deception breaks down first are not necessarily those that we would expect to be most important to genetic search. If we relax the requirements for deception at some order N, we can build problems which are consistently deceptive but not fully deceptive; of particular interest are problems that are consistently deceptive where many, but not all of the lower-order hyperplanes lead toward the deceptive attractor, but where none lead toward the global winner of the hyperplane competition at order N. Using the set of values given in Deceptive Function 3, if the value of the string "0000" is changed from 10 to 7, the problem is no longer fully deceptive because $f(000*) < f(100*)$, $f(000*) = f(010*)$ and finally $f(00*0) = f(10*0)$, but all other hyperplane information still leads toward 0000, the deceptive attractor. However in such cases it still makes sense to talk about order-4 deception that is less than fully deceptive, because the order-4 deceptive problem does not decomposed into an order-3 deceptive problem. In fact, several order-3 hyperplanes continue to be deceptive with respect to the correct order-4 building block. The hyperplanes *000 or 0*00 continue to have a higher fitness value than all of their competitors and therefore continue to support deception at the order-4 level. Even though the function is not fully deceptive since all of the lower order hyperplanes do not lead toward the deceptive attractor, the problem is nevertheless misleading in all hyperplanes. In none of the hyperplane competitions does a hyperplane win that directs the search toward the correct building block. We have reduced the degree of deception, but there are shades of gray between fully deceptive order-3 problems and fully deceptive order-4 problems. It is this kind of situation that motivated the distinction between fully deceptive problems and consistently deceptive problems. At this point there is no empirical evidence regarding the difficulty of consistently deceptive problems.

## 8 The Deceptive Attractor Remapping Strategies

What practical significance does all this have? Knowing that the deceptive attractor must be the complement of the global winner is useful if deception is occurring in a known or predictable location (ie: within a parameter setting); problem recodings that significantly change the Hamming distance between the global winner and the deceptive attractor of a hyperplane competition will reduce the degree of deception that currently exists. Small changes may not reduce the level of deception; for example, in Deceptive Function 3 reversing the mapping between 1000 and 0000 does not reduce the level of deception–in fact it strengthens the deception. But remapping many of the strings so that the "global winner" is moved closer in Hamming space to the deceptive attractor *and those strings that help to maintain the deception* will in many cases reduce the level of deception.

It would also seem possible that a remapping could create deception in other parts of the search space, although the possibility of this happening does not imply anything concerning the likelihood of such an event. The problem with recoding or remapping strategies is that knowing the exact location of the deception is not in general practical, since it involves finding a binary mask indicating the locations where a particular case of deception is occurring; *finding this binary mask involves a search space as large as the function optimization space.* It is also unclear if a single mask for deception would suffice: is it necessary to know how many distinct cases of deception exist and where they are located? The difficulty of locatings the bits that compose a deceptive building block problem also precludes such simple fixes for deception as inverting the final solution or inverting strings during search and evaluating their complements. This would appear to be bad news for simple remapping strategies, unless we can simply remap arbitrary portions of the encodings and hope that we somehow manage to remap part of the deception.

## 9 Deception and Linkage Problems

Deception is only part of the problem. The other factor that works to make a problem GA-hard involves the "linkage" between bits in a deceptive building block. Deception is a more serious problem if it occurs across some random (and therefore, unknown) combination of bits in the encoding. Having a poor linkage implies that the deceptive bits are highly separated on the encoding. Hyperplanes represented by bits that are distributed over a long encoding are inadequately sampled by the genetic algorithm because of a higher rate of disruption during recombination. Put more simply, if the deceptive bits are close together, then strings that sample some favorable combination of bits are more likely to pass on appropriate schemata to their offspring intact (because crossover disrupts critical schemata less often) and the genetic algorithm will not be misguided. Consider when deception occurs in a building block of 3 bits. We would expect the correct combination (111) to occur 12.5% of the time in the initial random population; if the critical bits tend to be passed on as a single block, then they will spread in the population and the deception that occurs in the lower order hyperplanes will not have a significant impact on search. However, if the same 3 bits are highly separated on the encoding, they are less likely to be inherited intact by offspring (due to the disruptive effects of crossover) and thus, the lower level hyperplane information becomes critical. When this happens, the genetic algorithm is likely to converge toward an incorrect solution.

Consider Deceptive Function 1. Also, assume the 3 bits that make up Deceptive Function 1 are highly separated in a binary encoding made up of several deceptive functions. As the competition between 000 (with an evaluation of 28) and 111 (with and evaluation of 30) heats up, there is not enough selective pressure to overcome the disruptive effects of recombination. Thus, the competition falls back to the level of the relevant lower order hyperplanes, and here 000 wins because of the deception built into the problem.

To build a suitable test function that is likely to be difficult for a genetic algorithm to solve, Goldberg, Korb and Deb (1989) defined a 30 bit function made up of 10 copies of a fully deceptive 3-bit function. The problem is difficult (or "ugly") when the bits are arranged so that each of the 3 bits of the subfunctions are uniformly and maximally distributed across the encoding. Thus each 3 bit subfunction, i, has bits located at positions i, i+10 and i+20. Goldberg et al. show that a standard genetic algorithm consistently converges to an incorrect solution on this problem, with each subfunction converging to 000 instead of 111. Again, this is consistent with the notion of hyperplane sampling, and biases known to exist in genetic algorithms against schemata with long defining lengths.

## 10 A Description of the Experiments

All of the following experiments used population sizes of 200 or 2000 as indicated. The experiments with population sizes of 200 used a total of 10,000 evaluations while the experiments with population sizes of 2000 used a total of 50,000 evaluations. These results do not necessarily represent our "best efforts" at solving these problems; rather, they are intended as comparative results. A rank based selection scheme was used with a linear selective bias of 1.5. Test problems include "ugly" versions of fully deceptive order-3 problems made up of Deceptive Function 1. A "random" version of the problem has also been defined by Goldberg, Korb and Deb (1989) so that the bits of the 10 subproblems are distributed randomly, such that some of the subproblems are highly distributed across the string of 30 bits, while other subproblems have bits that are closer together in the encoding. Since each test problem is actually composed of 10 fully deceptive subproblems, results are reported in terms of the number of subproblems correctly solved.

Additional tests were carried out comparing uniform crossover and 1-point crossover on an "ugly" fully deceptive order-4 problem. We constructed a 40 bit function composed of 10 4-bit fully deceptive subfunctions (specifically, Deceptive Function 2), with the 10 subfunctions distributed at positions i, i+10, i+20, i+30, for i = 1 to 10. No tests were done using a "random" arrangement of bits on this problem.

All of the reported results are based on 30 independent runs of the genetic algorithm. The genetic algorithm used was GENITOR which keeps copies of the best strings found so far. Because of this, GENITOR produces more selective pressure than would be the case with a standard genetic algorithm using a linear selective bias (Goldberg and Deb 1991). The additional selective pressure and conservation of previous "best" strings allows it to solve some of the deceptive subproblems.

### 10.1 GENITOR With 1-point Crossover

The 1-point crossover operator was a reduced surrogate operator. No mutation was used in these experiments. These experiments provide a baseline for comparative purposes, since they represent the operation of the algorithm in its normal mode of operation. Results for this and other operators are given in Table 1.

### 10.2 GENITOR With Uniform Crossover

Uniform crossover involves choosing each bit independently from the two parents, producing a random assortment of bits from the two parents. The potential advantage is that now the location of the bits on the encoding is irrelevant; the "ugly" version of the problems is exactly the same as the "random" version of the problem. Thus, there is no bias against schemata with long defining lengths. On the other hand, the advantages of tight linkage (i.e., functionally related bits that are close in the encoding have a tight linkage) are lost, since they no longer can be inherited as a block. One might expect uniform crossover to be competitive on problems such as the "ugly" deceptive problem, since there is not a bias against schemata with long defining length. But when this is considered more carefully, it seems clear that uniform crossover breaks apart higher level schemata during recombination and thereby shifts the burden of search to the order-1 and order-2 schemata. If the problem is deceptive, this action will increase the probability that the deceptive hyperplanes will be able to mislead the genetic algorithm. This suggests that Syswerda's tests (1989) using uniform crossover were not sufficiently deceptive to make them challenging tests problems (Das and Whitley 1991).

### 10.3 Tagged Bits and Random Orderings

The difference between the results on the "ugly" and "random" versions of the fully deceptive order-3 problem clearly illustrates that the location of the bits composing the subproblems in the overall problem encoding is important. This implies that using "tags" to randomize the location of the specific bits in the overall encodings might be effective in improving the performance of the genetic algorithm. In the "random" version of the test function, there is one fixed random ordering that is used by all strings. When tags are used each string has its own individual ordering. Normally, the interpretation of a binary string is positionally dependent. If we tag the bits, for example, so that the eighth bit in some binary strings is denoted (8 0), then we know the eighth bit is a "0" regardless of its location. The binary 11011001 for example, could be represented using the following LISP-like notation

((1 1) (2 1) (3 0) (4 1) (5 1) (6 0) (7 0) (8 1))

This kind of scheme can of course be implemented any number of ways; the point is that the bits can now be allowed to move and different "linkages" can developed in different strings.

The key issues in the implementation used here is that in the initial population both the bit strings and the sequence of bit tags were randomly generated. This means that each string started with its own randomly generated set of tags. In order for recombination to occur, two parent strings must align so that they have the same sequence of bits. To accomplish this one parent is randomly chosen to be the "template" and the other parent is rearranged so that its sequence of bits match this template. Recombination is then applied using 1-point crossover. Also, note that the offspring inherits the same sequence of bit tags as the parent providing the template. Thus, as search proceeds, the distribution of bit tag sequences will not remain random.

Goldberg and Bridges (1990) have also considered the use of tagged bits; the main difference between the work presented here and previous work is that the genetic search presented here began with a random distribution of different bit tag sequences, whereas in Goldberg's studies (which were in part analytical), it was assumed that the population started with all strings having the same sequence and inversion was applied to strings in an effort to try to improve the linkage.

As is the case with uniform crossover, using a tagged bit representation means that there is no difference between the "ugly" and "random" versions of this problem. The tagged bit scheme consistently produced the best results.

### 10.4 A Distributed Genetic Algorithm

A distributed genetic algorithm was used that employed 10 subpopulations. Each subpopulation contained 200 strings and was allowed 5,000 evaluations; these restrictions made the total amount of work comparable to the runs using a single population of 2000 and a total of 50,000 evaluations. On the "ugly" fully deceptive order-3 problem it solved 55% of the subproblems correctly. In other tests we had allowed each subpopulation of 200 strings a total of 50,000 recombinations; given the additional recombinations it solved 100% of the subproblems. The success of the distributed algorithm lies in the fact that each subpopulation solves a different subset of the deceptive subproblems; migration of strings between subpopulations allows these different solutions to be recombined, thereby solving all of the subproblems. We should also note however, that the distributed algorithm used a mutation operator and the serial algorithm did not; this could also account for part of the results.

In related experiments we used 1, 2, 5, 10, 50 and 100 subpopulations with and without mutation. The total population size and number of recombinations was held constant in the various experiments; the single population was composed of 5000 strings and allowed 500,000 recombinations while the run with 100 subpopulations, where each subpopulation had 50 members (100 * 50 = 5000) and was allocated 5000 recombinations (100 * 5000). Both the runs with and without mutation showed the same trend: the more distributed the population, the better the results on ugly deceptive problems (Starkweather, Whitley and Mathias 1991).

## 11 Summary of Experimental Results

The following table summarizes the experimental results. The test problems are the ugly order 3 fully deceptive problem (Ugly,O-3), the random version of the same problem (Random,O-3), of the ugly order 4 fully deceptive problem (Ugly,O-4). The "1-point" versions are direct application of GENITOR, while the "Tagged" and "Uniform," versions should be self explanatory. The "Parallel" version used 10 subpopulations of 200 (*total population, 2000) and 5000 evaluations in subpopulation (*total evaluations, 50,000).

| Problem | Approach | Pop Size | Solved | Evals |
|---|---|---|---|---|
| Ugly,O-3 | 1-point | 200 | 27% | 10,000 |
| Ugly,O-3 | Uniform | 200 | 27% | 10,000 |
| Random,O-3 | 1-point | 200 | 46% | 10,000 |
| Ugly,O-3 | Tagged | 200 | 53% | 10,000 |
| Ugly,O-3 | Uniform | 2000 | 35% | 50,000 |
| Ugly,O-3 | 1-point | 2000 | 38% | 50,000 |
| Random,O-3 | 1-point | 2000 | 52% | 50,000 |
| Ugly,O-3 | Parallel | 2000* | 55% | 50,000* |
| Ugly,O-3 | Tagged | 2000 | 64% | 50,000 |
| Ugly,O-4 | Uniform | 200 | 3% | 10,000 |
| Ugly,O-4 | 1-point | 200 | 7% | 10,000 |
| Ugly,O-4 | Tagged | 200 | 16% | 10,000 |

TABLE 1: Summary of Results

While we have not done any statistical tests on this data, three hypotheses are suggested by these results. First, the use of "tagged bits" appears to help when the initial linkage represents a worse case situation. Second, uniform crossover does not outperform 1-point crossover in this domain and may be inferior. Third, a distributed algorithm produces considerable improvement in performance. However, one should be careful not to generalize from these results; these are highly artificial problems and may not be typical of other optimization problems.

## 12 Conclusions and Future Directions

The theoretical results and empirical tests offered in this paper are meant to 1) stimulate further debate on the issue of deception and its relationship to hyperplane sampling and implicit parallelism and 2) to establish a foundation for further research. The paper has probably raised more questions than have been answered.

There has been a great deal of debate recently in the genetic algorithm community about the problems we use as test functions. What does it really mean when one variant of genetic algorithm outperforms another on some test suite? The theoretical results presented in this paper concerning deception and the 2-armed bandit analogy strongly support a position which has been advocated by David Goldberg: test problems must be used that involve some degree of deception. This also places increased importance on work relating Walsh transforms to genetic algorithm and deception (Goldberg 1989b; 1989c). This does not imply, however, that the only reasonable test problems are "ugly" fully deceptive problems. These problems are highly artificial in two ways: 1) they decompose into subproblems that can be solved independently and 2) deception occurs in every position in every subproblem so that the complement of the "deceptive solution" is always the global solution. I would very strongly argue that methods which utilize information about *where* the deception exists will only solve these very artificial problems–however, in general this information is not available.

We have already obtained empirical evidence that simple problems which are not deceptive in the order-1 hyperplanes can be solved by solving the order-1 hyperplanes (Das and Whitley 1991). The methods appear to be faster than genetic search (since much less information is being processed) and robust even when given noisy hyperplane sampling. Theorem 1 on solving nondeceptive problem by solving the order-1 hyperplane competitions is somewhat idealistic (since it assumes no sampling noise and accurate determination of order-1 winners) but nevertheless points toward a workable method for deciding if problems are significantly deceptive. This potentially represents an important contribution to the field of genetic algorithms.

This paper has established a description of various kinds of deception. The concepts of "fully deceptive" and "consistently deceptive" also suggest a new need for empirical tests to evaluate the difficultly of problems displaying different degrees of deception. Are problems that are consistently deceptive as hard or nearly as hard as fully deceptive problems? Are fully deceptive problems as hard to solve if the deceptive attractor is not a local optimum in Hamming space and does not have a fitness value competitive with the correct building block? We are already working on the answers to these questions.

In general, the experiments described in this paper are only a first step toward arriving at a better understanding of deceptive problems. Our results suggest that the use of "tagged bits" might be helpful in solving the linkage problems. Other have argued against the use of "tagged bits" because finding the optimal sequence of bits while also finding the optimal bit values greatly increases the size of the search space (Liepins and Vose 1991; Goldberg and Bridges 1990). However, it may be that we do not need to find optimal sequences, but rather just avoid particularly "bad" or "ugly" sequences. On the other hand, these test problem started with the worst bit arrangement we could devise; on real problems where parameters are coded as a contiguous block of bits the use of tagged bits may actually make the initial arrangement worse, not better. Thus, I would argue that this issue is still open and no firm conclusions should be drawn at this time.

Approaches that have *not* been examined include Goldberg, Korb and Deb's (1989) "Messy GAs" which were designed in part to handle deceptive problems; also not examined are crowding techniques. Crowding techniques would seem like a natural solution to deception, since crowding forces the development of "niches" based on the similarity of the binary strings competing for space in a population. Given the results presented in this paper concerning the relationship of a deceptive attractor to the relevant global winner of a hyperplane competition, we might expect the global winner to occupy a different niche than that occupied by the deceptive attractor and the strings or schemata close to the deceptive attractor in Hamming space. In some sense the results using the distributed genetic algorithm also is produced by a kind of "niching," but one based implicitly on genetic drift and sampling errors rather than explicit forced niching.

**Acknowledgements**

My thanks to David Goldberg for insightful comments on an earlier draft; the remaining flaws are, of course, my own. The research was supported in part by a grant from the National Science Foundation, grant number IRI-9010546, and by a grant from the Colorado Institute of Artificial Intelligence (CIAI). CIAI is sponsored in part by the Colorado Advanced Technology Institute (CATI), an agency of the State of Colorado. CATI promotes advanced technology education and research at universities in Colorado for the purpose of economic development.

**References**

Bethke, A. (1980) Genetic Algorithms as Function Optimizers. Ph.D. Dissertation, Computer and Communication Sciences, University of Michigan, Ann Arbor.

Das, R. and Whitley, D. (1991) The Only Challenging Problems are Deceptive: Global Optimization by Solving Order-1 Hyperplanes. To appear: *Fourth International Conf. on Genetic Algorithms.*

Goldberg, D. (1987) Simple Genetic Algorithms and the Minimal, Deceptive Problem. In, *Genetic Algorithms and Simulated Annealing,* L. Davis, ed., pp: 74-88, Morgan Kaufmann, Pubs.

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

Goldberg, D. (1989b) Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction. *Complex Systems* 3:129-152.

Goldberg, D. (1989c) Genetic Algorithms and Walsh Functions: Part II, Deception and its Analysis. *Complex Systems* 3:153-171.

Goldberg, D. (1990) Construction of High-order Deceptive Functions Using Low-order Walsh Coefficients. IlliGAL Report No. 90002. Department of General Eng. Univ. of Illinois at Urbana-Champaign.

Goldberg, D. and Bridges, C. (1990) An Analysis of a Reordering Operator on a GA-Hard Problem. *Biological Cybernetics* 62: 397-405.

Goldberg, D. and Deb, K. (1991) A Comparative Analysis of Selection Schemes. *Foundation of Genetic Algorithms,* G. Rawlins, ed. Morgan Kaufmann.

Goldberg, D., Korb, B., and Deb, K. (1989) Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Systems* 4:415-444.

Goldberg, D. and Rudnick, M. (1991) Genetic Algorithms and the Variance of Fitness. IlliGAL Report No. 91001. Department of General Eng. Univ. of Illinois at Urbana-Champaign.

Grefenstette, J. and Baker, J. (1989) How genetic algorithms work: a critical look at implicit parallelism. *Proceeding of the Third International Conference on Genetic Algorithms, 1989.* Washington, D.C., Morgan Kaufmann, Publishers.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press.

Liepins, G. and Vose, M. (1990) Representation Issues in Genetic Optimization. *J. Experimental and Theoretical Art. Intell.* 2(1990)101-115.

Liepins, G. and Vose, M. (1991) Deceptiveness and Genetic Algorithm Dynamics. *Foundation of Genetic Algorithms,* G. Rawlins, ed. Morgan Kaufmann.

Starkweather, T. Whitley, D., and Mathias, K. (1991) Optimization Using Distributed Genetic Algorithms *Parallel Problem Solving from Nature,* Springer-Verlag, Publishers.

Syswerda, G. (1989) Uniform Crossover in Genetic Algorithms. *Proc. Third International Conf. on Genetic Algorithms,* Morgan Kaufmann, Publishers.

Tanese, R. (1989) Distributed Genetic Algorithms. *Proc. Third International Conf. on Genetic Algorithms,* Morgan Kaufmann, Publishers.

Whitley, D. (1991) Deception, Dominance and Implicit Parallelism in Genetic Search. Technical Report. Department of Computer Science, Colorado State University.