Genetic Algorithms, Problem Difficulty, and the Modality of Fitness Landscapes

Jeffrey Horn

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

IlliGAL Report No. 95004
July 1995

(internet email) jeffhorn@uiuc.edu
(phone) 217/333-2346
(fax) 217/244-5705
(www) http://GAL4.GE.UIUC.EDU/illigal.home.html

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

GENETIC ALGORITHMS, PROBLEM DIFFICULTY, AND THE MODALITY OF FITNESS LANDSCAPES

BY

JEFFREY HORN

B.A., Cornell University, 1985

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1995

Urbana, Illinois

Abstract

We generally assume that the modality (i.e., number of local optima) of a fitness landscape is related to the difficulty of finding the best point on that landscape by evolutionary computation (e.g., hillclimbers and genetic algorithms (GAs)). This thesis first examines the limits of modality by constructing a unimodal function and a maximally multimodal function. At such extremes our intuition breaks down. A fitness landscape consisting entirely of a single hill leading to the global optimum proves to be harder for hillclimbers than for GA crossover. A provably maximally multimodal function, in which half the points in the search space are local optima, can be easier than the unimodal, single hill problem for both hillclimbers and GAs. Exploring the more realistic intermediate range between the extremes of modality, local optima are constructed with varying degrees of "attraction" to evolutionary algorithms. To construct such optima, it is necessary first to define attraction for an algorithm. Most work on optima and their basins of attraction has focused on hills and hillclimbers, while some research has explored attraction for the GA's crossover operator. This thesis extends some of the latter results by defining and implementing *maximal* partial deception in problems with k arbitrarily placed global optima. This allows the creation of functions, such as the *minimum distance* function f_{mdG}, with k isolated global optima and multiple local optima attractive to both crossover and hillclimbers. This function appears to be a powerful new tool for generalizing deception and relating hillclimbers (and Hamming space) to GAs and crossover. Experiments using variations on f_{mdG} demonstrate four quasi-separable dimensions of GA problem difficulty: *multimodality*, solution *isolation* and *signal*, and *misleadingness* (*deception*). Finally, this thesis addresses the appropriateness of Sewall Wright's fitness landscape to the study of evolutionary computation, demonstrating that the metaphor can help us understand and predict the performance of our algorithms, including both hillclimbers and recombinative GAs.

For Gabriele, who cared as much about this thesis as I did.

Acknowledgments

First I must thank my thesis advisor, David E. Goldberg. In addition to his expertise, oversight, and encouragement, he also contributed the original, inspired ideas of exponentially long hillclimbing paths, bipolar deception, and massive multimodality. I also thank my coworkers and coauthors on much of the long path problem and multimodal deception efforts: David E. Goldberg and Kalyanmoy Deb. I am grateful for the reviews, comments, and suggestions of fellow IlliGAL graduate students Georges R. Harik, Kaitlin Sherwood, and Hillol Kargupta, and "off-site" colleagues Joseph C. Culberson and Terry Jones. I received support on a more general level from all the staff at the IlliGAL. On a more practical level, I gratefully acknowledge support provided by the U.S. National Aeronautics and Space Administration (NASA) under Contract NGT-50873, administered by the Johnson Space Center as part of the NASA Graduate Student Researchers Program. I also received support from the U.S. Army under Contract DASG60-90-C-0153, and from the U.S. Air Force Office of Scientific Research (AFOSR) under Grant F49620-94-1-0103. Finally, I thank my mother, Anna Horn, for treating higher education as compulsory, and my father, Ralph Horn, for conveying to me his enthusiasm and respect for graduate education.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Genetic algorithms (GAs) are robust adaptive systems that have been applied successfully to hard optimization problems, both artificial and real world (Goldberg, 1994). Yet GAs do fail. When and why? In this thesis, I choose to look at a few key characteristics of the *fitness landscape*[1] that a GA explores, and examine the relationships of these characteristics to the successful performance of the GA. In particular, I am interested in the concept of *local optima* in the fitness landscape, and how they help determine the success or failure of GA optimization. I attempt to relate the number, location, and "attractiveness" of global and non-global local optima in the *Hamming space* (or *bit-mutation*) landscape to the relative performance of hillclimbers and GA crossover. In doing so, I address the controversial issue of the relevance of Sewall Wright's fitness landscape metaphor (Wright, 1988) to our understanding of GA behavior, showing that the metaphor is indeed extremely helpful (and therefore relevant) to both our developing GA theory and to our own intuitive understanding.

I also show how these measurements of local optima fit within the framework of GA problem difficulty outlined by Goldberg (1993):

- Isolation
- Misleadingness

[1]Each string s can be seen as a coordinate in the search space S of all possible strings $s \in S$, with a scalar fitness $f(s)$ assigned to each such point. Thus the combinatorial optimization problem at hand is often viewed as a landscape, with the "height" of the landscape at a point s being given by $f(s)$. The global optimum, that is the best point in the search space and the goal of the search, clearly must be a local optimum, since neighboring points (at some finite radius k bits) are inferior in fitness. Other local optima in the search can confound a search algorithm by appearing to be globally optimal (i.e., by being the best found so far).

- Multimodality
- Noise
- Crosstalk

Isolation, misleadingness, and multimodality in particular are characteristics of the fitness landscape directly related to local optima. The last of these, multimodality, receives the greatest attention in this thesis, as it has not been extensively studied on its own. I examine just what relationship, if any, exists between the *modality* (the number of local optima) of the fitness landscape and hillclimbing or GA performance on that landscape. I examine the extremes of modality: unimodality (Chapter 3) and maximum modality (Chapter 4). At such extremes I find some interesting, counter-intuitive results. I then probe some of the vast middle ground of intermediate modality (Chapter 5), in which I generalize the work on deception and partial deception to define basins of attraction for GA crossover. I go on, in Chapter 5, to show the crucial relationships among the three aspects of problem difficulty (isolation, misleadingness, and multimodality), as they are all present in problems of intermediate modality and intermediate difficulty for GA crossover-based search.

Finally, this work partly addresses the robustness of hillclimbing versus that of GA crossover, an important open issue given the recent trend toward "hillclimbing GAs" (that is, GA variants with a greater role for local search[2]). On the problems presented in this thesis, the various local searchers appear to be more sensitive to the number, location, and "size" of local optima than is GA crossover.

[2](i.e., hillclimbers, mutation, and other algorithms exploitive of local information in the landscape.)

Chapter 2

Background

This section contains a brief introduction to genetic algorithms, general definitions of landscapes and local optima, and a short discussion of the relevance of "objective", operator-independent fitness landscapes to our understanding of evolutionary algorithms. Additional background, such as more specific terminology, can be found in Chapters 3, 4, and 5, in the more detailed discussions of the major topics of this thesis. In particular, the relationships of this thesis' results to previous published work is established within each of the chapters on minimum, maximum, and intermediate modality as appropriate. Finally, I should mention that most of the major results of this thesis have been, or are about to be, published in (Horn, Goldberg, & Deb, 1994) and (Horn & Goldberg, in press). I therefore omit from the remainder of this thesis the numerous possible citations to those two sources.

2.1 Genetic Algorithms

In this section I briefly discuss the nature and basic operation of the simple GA. The reader is referred to (Goldberg, 1989a) for an in-depth introduction to GAs and a thorough grounding in basic GA theory.

Throughout the history of GAs, since the 1975 publication of the first edition of John H. Holland's book (Holland, 1992), these algorithms have been applied to combinatorial optimization problems[1]. Such problems are defined by a vector of discrete decision variables $\vec{v}$ and

[1] Including continuous numerical optimization problems that have been mapped into combinatorial problems via discretization of the continuous decision variables.

a scalar objective function $f(\vec{v}) \rightarrow \Re$ to be optimized (i.e., minimized or maximized). The decision variables $\vec{v}$ are typically mapped to (encoded as) a bit string s consisting of ℓ binary variables (bits). The objective function $f(\vec{v})$ is usually scaled so as to be non-negative, perhaps inverted (if originally a minimization problem) to ensure a *direct* relationship between the objective function value and the desirability of the solution, and then used as the GA's *fitness function*. Thus each possible solution s is a particular setting of the decision variables corresponding to a particular string s which has a corresponding fitness $f(s)$ to be maximized or minimized.

One major application of GAs is to "black-box" function optimization. We assume a function $f(\vec{x}) \rightarrow \Re$ mapping a vector of decision variable settings to a real-valued (scalar) objective value (figure of merit). This objective function is to be optimized (minimized or maximized), but we know little or nothing else about f. To apply the GA to the problem, we first develop an encoding of the decision variables as a (generally binary) string (chromosome). The fixed length (ℓ-bit) binary encoding means that any of the 2^ℓ possible chromosomes (bit settings) represents a candidate solution to the problem, and can be assigned a *fitness* (usually some direct function of the objective function f). The GA maintains a fixed size N population of chromosomes, to which it applies standard, basic GA operators such as selection, crossover, and mutation.

2.1.1 GA Operators

The basic, well-known operators of the simple GA include *selection* (e.g., deterministic binary tournament selection), *crossover* (e.g., two-point), and *mutation* (e.g., bit-wise). More complicated, specialized and advanced operators include *niching* (e.g., fitness sharing) and *reordering* (e.g., inversion). I briefly describe examples of each of the three basic operators below.

Binary tournament selection operates on a population of individuals by selecting two individuals from the current population at random and comparing their fitnesses in a *binary tournament*. The individual with higher fitness is copied into the new population (i.e., the next generation). This process is repeated N times to create a new generation.

Two-point crossover operates on a pair of individuals (parents), call them **A** and **B**, replacing them by a pair of their offspring, call these **a** and **b**. The offspring are created by selectively copying bits of each parent into the offspring as follows. Two different *cutting points* are chosen at random from among the ℓ possible cutting points (between bit positions). The bits from parent **A** that are between the two chosen crossover points are copied over the corresponding bits from parent **B**, thus creating child **a**. Child **b** is created by the complementary process of copying the bits between the crossover points from parent **B** over the corresponding bits in parent **A**. The two children then replace the two parents in the population. Crossover happens with a probability p_c. Thus with probably $1 - p_c$ the crossover will not take place and the two children will be exact copies of their parents. Since I focus in this thesis on the processing of building blocks, I choose $p_c = 1.0$ (i.e., crossover always takes place) in all of the GA experiments reported herein.

Bit-wise mutation acts on a single individual, by flipping each of the ℓ bits independently with a probability p_m. This probability is usually kept very low, to minimize disruption of converged genes and to reduce the "randomness" and noise of search. Again in order to focus on the processing of building blocks, I assume no mutation ($p_m = 0$) in all of the GA experiments.

2.1.2 The Main Loop

I assume a *generational* GA, in which the entire current population (generation) of N individuals is replaced by a new population (next generation) created by applying the three operators described above to the current population. The initial population (generation 0) consists of N randomly generated individuals. Each individual is then evaluated for fitness, as explained above. With fitnesses assigned, the three operators can be applied to create the next generation. Following (Goldberg, 1989a), we hold $N/2$ pairs of binary tournaments. Each pair of tournaments thus selects two individuals. We apply two-point crossover to these two parents, and (optionally) apply bit-wise mutation to each of the resulting offspring. The resulting (possibly mutated) offspring are then placed in the new population. The sequential application of these three operators is repeated for a total of $N/2$ (assuming even N) times, generating a new population of size N (generation 1). The new population replaces the old population, and we repeat the process on the new population. Thus, the population of generation $t+1$ replaces the population of generation t.

This loop is repeated until some stopping criterion is reached, such as finding an individual with fitness within some small range δ of the ideal, or until the population has converged to near or complete uniformity (i.e., it is composed entirely, or nearly so, of copies of a single individual/solution). In the experiments of this thesis, the stopping criterion is uniform population convergence, as (indirectly) measured by the fitness difference between the best and worst members of the current population. When this difference is zero, we assume that the population is uniform[2].

2.2 Definitions: Landscapes and Optima

I make use of the paradigm of a **fitness landscape** (Wright, 1988), which consists of a search space, a metric, and scalar fitness function $f(s)$ defined over elements s of the search space S. Assuming the goal is to maximize fitness, we can imagine the globally best solutions (the global optima, or "globals") as "peaks" in the search space. For the purposes of this thesis, I define local optimality as follows. First assume a real-valued, scalar fitness function $f(s)$ over fixed length ℓ-bit binary strings s, $f(s) \in \Re$. Without loss of generality, I assume f is to be maximized.

A **local optimum** in a discrete search space S is a point, or region, with fitness function value strictly greater than those of *all* of its nearest neighbors. By "region" I mean a set of interconnected points of equal fitness. That is, I treat as a single optimum the set of points related by the transitive closure of the nearest-neighbor relation such that all points are of equal fitness[3]. This definition allows me to include flat plateaus and ridges as single optima, and to treat a flat fitness function as having no local optima.

The "nearest neighbor" relation assumes a metric on the search space, call it d, where $d(s_1, s_2) \in \Re$ is the metric's distance between points s_1 and s_2. Then the nearest neighbors of a point s' are all points $s \in S, s \neq s'$ such that $d(s', s) \leq k$, for some neighborhood radius k. In this thesis I use only *Hamming distance* (number of bit positions at which two binary strings differ) as the metric, and assume $k = 1$. Thus a point s (or connected region of equal fitness) with fitness $f(s)$ greater than that of all its immediate neighbors (strings differing from s in only one bit position) is a local optimum.

[2] Note that a non-uniform population can have uniform fitness, specifically when multiple points in the search space map to the same fitness value (e.g., flat plateaus and ridges). But I believe that for the fitness functions used in this thesis, the chance that a large, evolved, uniform-fitness population contains enough useful diversity to improve its fitness is negligible.

[3] The requirement for equality could be relaxed to allow for fitnesses within some δ of the globally optimal fitness.

2.3 The Wright Fitness Landscape: Useful Tool or Misleading Metaphor?

This appealing metaphor of a *landscape* induced by a fitness function on a metric search space was introduced by Wright (1988). It is widely used in the literature to define such terms as *local optima*, as I do here in this thesis, and to discuss such issues as *isolated* global optima (Goldberg, 1993). But the metaphor has recently come under attack as inappropriate for visualizing and analyzing the behavior of any search algorithms besides the simplest, one-bit hillclimbers such as steepest or next ascent. For example, Jones (1995) argues strongly for a rigorous definition of operator-induced landscapes as directed graphs whose vertices represent states of the algorithm and whose edges are state transitions labeled by the probabilities of the algorithm's operators inducing that transition.

While Jones gives each operator (in a multi-operator algorithm like the GA) its own landscape (e.g., separate landscapes for crossover, mutation, and selection), others have advocated a single landscape in which each vertex is a possible population and transitions are labeled by the combined probability of all the operators acting together to cause the transition. Examples of such "combined-operator-landscape" approaches include various "exact" models of GA dynamics, such as population distribution vectors and Markov chains (e.g., Vose, 1993). Still others leave Sewall Wright's landscape untouched, but measure, analyze, and characterize such landscapes using operator-specific statistics (e.g., Manderick, de Weger, and Spiessens's (1991) fitness correlation coefficients).

Within the framework of "operator(s)-induced" landscapes, Sewall Wright's fitness landscape is induced by a one-bit hillclimber or a one-bit mutation algorithm and is often called a *point-mutation* landscape. Therefore, in the view of most researchers following such strict definitions of operator-induced landscapes, the point-mutation landscape is of little relevance to the study of GAs, evolution strategies (ES), or any other search algorithm incorporating other than one-bit-flip operators.

I believe that the operator-induced landscapes described above represent rigorous and valid analytical approaches to GA theory. But they by no means replace Sewall Wright's metaphor as an aid to understanding GAs. All these different landscapes are representative of different and complementary approaches. For example, the operator-induced landscape is difficult to visualize for complex operators like crossover[4], although once defined the behavior of the search algorithm might be easy to visualize (e.g., following the local gradient (Vose, 1993), or perhaps taking the highest-probability transition). On the other hand, the Wright-style fitness landscape is relatively easy to visualize, though it is not clear exactly how a GA "traverses" such a landscape with a population. In this thesis, I show that a number of characteristics of the "Wright fitness landscape" are highly relevant to GA search, although modality (at least by itself) is not one of them.

[4] For example, Jones (1995) associates each vertex in the crossover landscape with a *pair* of individuals. As another example, the mutation landscape, like the all-operator landscape, is completely interconnected, since mutation can, with finite probability, transit from any one state to any other.

Chapter 3

Minimum Modality Can be Hard: The Long Path Problem

At one extreme of the modality spectrum is unimodality. Unimodal functions have only one local optimum, which is therefore the global optimum. It is well-known that such problems can be hard when the optimum is isolated, with little or no information available elsewhere in the search space. Such isolated peaks on otherwise flat fitness landscapes have been called *needle-in-a-haystack* (NIAH) problems (Goldberg, 1989a), and are clearly solvable only by enumeration of the space. But if we decrease the isolation of the single optimum, increase the size of its basin of attraction, and add information to larger portions of the search space, intuition tells us that the search should become easier and shorter than enumeration. In particular, if we make the entire search space a single hill, where every point is on a path to the global optimum, even simple evolutionary algorithms, such as a one-bit hillclimber, should quickly optimize the function. But I show otherwise.

I present a class of problems designed to be difficult for randomized search procedures that exploit local search space information. In particular, these problems challenge hillclimbers and mutation algorithms. The problems are difficult not because they contain local optima that capture the attention of the search procedure. Indeed, these problems are unimodal and "easy" in the sense that the simplest hillclimber will always find the global optimum, no matter where in the space it starts searching. Rather, these problems are difficult for hillclimbers because the only "path" up the hill to the global optimum is very long and narrow. The length of the path grows exponentially with the size of the (binary) problem, ℓ.

At the beginning of this theses, I listed five dimensions of problem difficulty for a GA. Of those five, three are clearly challenges for hillclimbers as well:

- isolation or *needle-in-a-haystack* (NIAH),
- misleadingness, and
- multimodality.

I propose a fourth type of problem that specifically targets the hillclimber's use of local information. I construct a path that leads a hillclimber to the global optimum, but on a path that is so long that for large problems ($\ell \gg 20$) one simply cannot wait for the algorithm to terminate. One might sit and watch a hillclimber making constant progress for years on an 80-bit problem.

I am motivated to construct and analyze these long path problems by several observations:

1. Computational difficulty of an apparently easy problem
2. Utility in analyzing hillclimbers and mutation algorithms
3. Possible distinguishing problem for GAs versus hillclimbers
4. Possible distinguishing problem for crossover versus mutation

These problems represent a dimension of difficulty that has been less obvious than noise, deception, local optima, etc. Unimodal, and with every point on a path to the global optimum, these problems are still intractable for hillclimbers of bounded step size. They might also be intractable for various kinds of mutation algorithms, and for algorithms incorporating recombination, such as GAs with high crossover rates. If they do offer as stiff a challenge to recombinative algorithms as they do to hillclimbers (and apparently to mutation algorithms as well), then we have found yet another class of GA-difficulty. If, on the other hand, the recombinative GA can perform significantly better (faster) on these problems than local-search procedures, then we have found a class of functions that distinguish these two types of algorithms. From this we can learn something about their different strengths and weaknesses relative to each other (Hoffmeister & Bäck, 1990).

3.1 Additional Definitions: Paths, Hills, and Hillclimbers

Although the term **hillclimber** denotes a specific set of algorithms for some researchers, I will use the term more loosely here to describe *all* algorithms that emphasize exploration of a local neighborhood around the current solution. In particular, I assume that a *hillclimber* explores a neighborhood of radius k-bits much better than it explores outside that neighborhood. Therefore, the hillclimber is much more likely to take "steps" of size $\leq k$ than it is to take steps $> k$. Examples of such hillclimbers include steepest ascent (Wilson, 1991), next ascent (Jones & Rawlins, 1993), and random mutation (Mühlenbein, 1992; Forrest & Mitchell, 1993; Mitchell & Holland, 1993; Mitchell, Holland, & Forrest, 1994). GAs with high selective pressure, non-zero mutation rates, and low crossover rates also exhibit strong local hillclimbing.

A **path** P of step size k is a sequence of points p_i such that any two points adjacent on the path are at most k-bits apart, and any two points not adjacent on the path are more than k-bits apart:

$$\forall p_i, p_j \in P,\ d(p_i, p_j) \begin{cases} \leq & k, \quad \text{if } |i-j| = 1 \\ \\ > & k, \quad \text{otherwise.} \end{cases}$$

where $d(p_i, p_j)$ is the Hamming distance between the two points and p_i is the ith point on the path P. Thus, a hillclimbing algorithm that takes steps of size $\leq k$ would tend to follow the path without taking any "shortcuts". The step size here is important, because I want to lead the algorithm up the path, but to make the path wind through the search space as much as possible, I will need to fold that path back many times. Earlier portions of the path may thus pass quite closely to later portions, within $k+1$ bits, so I must assume that the algorithm has a very small, if not zero, probability of taking steps of size $> k$.

3.2 A Simple Construction: the Root2path

In this section we construct a long path that is not optimally long, but does have length growing exponentially in ℓ, the size (order, or dimension) of the problem, and is simple in its construction. Let us call it the *Root2path*. Here I choose the smallest step size $k = 1$ to illustrate the construction. Each point on the path must be exactly one bit different from the point behind it and the point ahead of it, while also being at least two bits away from any other point on the path.

3.2.1 Recursive Construction

The construction of the path is intuitive. If we have a Root2path of dimension ℓ, call it P_ℓ, we can basically double its length by moving up two dimensions to $\ell + 2$ as follows. Make two copies of P_ℓ, say copy00 and copy11. Prepend "00" to each point in copy00, and "11" to each point in copy11[1]. Now each point in copy00 is at least two bits different from all points in copy11. Also, copy00 and copy11 are both paths of step size one and of dimension $\ell + 2$. Furthermore, the endpoint of copy00 and the endpoint of copy11 differ only in their first two bit positions ("00" versus "11"). By adding a *bridge point* that is the same as the endpoint of copy00 but with "10" in the first two bit positions instead of "00", we can connect the end of copy00 and the end of copy11. Reversing the sequence of points in copy11, we concatenate copy00, the bridge point, and Reverse[copy11] to create the Root2path of dimension $\ell + 2$, call it $P_{\ell+2}$, and length essentially twice that of P_ℓ:

$$|P_{\ell+2}| = 2\,|P_\ell| + 1 \tag{3.1}$$

As for the dimensions (even ℓ) between the doublings, which are all of odd-order (odd ℓ), we can simply use the path P_ℓ by adding a "0" to each point in P_ℓ to create $P_{\ell+1}$. If $|P_\ell|$ is exponentional in ℓ, then $|P_{\ell+1}|$ is exponential in $\ell + 1$.

For the base case $\ell = 1$, we have only two points in the search space: 0 and 1. We put them both on the Root2path $P_1 = \{0, 1\}$, where 0 is the beginning and 1 is the end (i.e., the global optimum).

With every other incremental increase in dimension, we have an effective doubling of the path length. Solving the recurrence relation in Equation 3.1, with $|P_1| = |P_2| = 2$, we obtain the path length as a function of ℓ:

$$|P_\ell| = 3 * 2^{\lfloor(\ell-1)/2\rfloor} - 1. \tag{3.2}$$

[1]Thus the point "001" in P_3 becomes "00001" in copy00 and "11001" in copy11.

Path length increases in proportion to $2^{\ell/2}$ or $(\sqrt{2})^\ell$ and thus grows exponentially in ℓ with base ≈ 1.414, an ever-decreasing fraction of the total space 2^ℓ.

Since the path takes up only a small fraction of the search space for large ℓ, the entire Root2path approaches a Needle-in-a-Haystack (NIAH) problem as ℓ grows. We are interested in how long a hillclimber takes to climb a path, not how long it takes to find it. We therefore slope the remainder of the search space (i.e., all points not on the Root2path) towards the beginning of the path. The construction of the Root2path makes it easy to do this. Since the first point on the path is the all-zeroes point, we assign fitness values to all points *off* the path according to a function of *unitation*[2]. The fewer ones in a string, the higher its fitness. Thus most of the search space should lead the hillclimber to the all-zeroes point, in at most ℓ steps. Let us call this landscape feature the *nilness*[3] *slope*. Together, the path P_ℓ and the nilness slope form a single *hill*, making the search space unimodal. A deterministic hillclimber, started *anywhere* in the space, is guaranteed to find the global optimum.

To find the total number of "steps" from the bottom of the hill (the all-ones point) to the top (the global optimum), we add the "height" of the nilness slope, which is simply ℓ, to the path length (Equation 3.2), and subtract one for the all-zeroes point, which is on both the path and the slope:

$$\text{Hill-Height}(\ell) = 3 * 2^{\lfloor(\ell-1)/2\rfloor} + \ell - 2 \tag{3.3}$$

3.2.2 Linear Decoding

The recursive construction of the Root2path is illustrative, but inefficient for fitness evaluations in the simulations below. In Figure 3.1, I present pseudocode for a much more efficient *decoding algorithm*[4]. Note that the recursion in the pseudocode is (optimizable) tail recursion. Thus the decoding algorithm is linear in the problem length ℓ. Note that "rest-of-string" and "all-zeroes" are non-empty placeholders (e.g., "all-zeroes" must contain at least one "0"). Also, the function *PathPosition[str]* returns *false* if *str* is not on the path (i.e., it is on the nilness slope), and returns a non-negative (possibly zero) integer, which is the path position of *str*, otherwise.

[2] The unitation $u(s)$ of a string s is equal to the number of ones in s. For example, $u(\text{"0110110"}) = 4$.

[3] The nilness of a string s is simply $n(s) = \ell - u(s)$ (i.e., the number of zeroes in s).

[4] In the literature on coding theory, the development of such algorithms is an important followup to the existence proofs and constructions of long paths (Preparata & Niervergelt, 1974).

```
PathPosition[str] :=
  CASE
   (str == "0") RETURN 0;                          /* First step on path.           */
   (str == "1") RETURN 1;                          /* Second step on path.          */
   (str == "00|rest-of-string")                    /* On 1st half of path.          */
             RETURN PathPosition[rest-of-string];
   (str == "11|rest-of-string")                    /* On 2nd half of path.          */
             RETURN 3*2^Floor[(Length[str]-1)/2] - 2 -
                        PathPosition[rest-of-string];
   (str == "101" OR "1011|all-zeroes")             /* At bridge pt. (halfway)       */
             RETURN  3*2^(Floor[(Length(str)-1)/2] - 1) - 1;
   OTHERWISE    RETURN false;                      /* str is NOT ON PATH.           */

HillPosition[str] :=
  IF (PathPosition[str])                           /* If str is on path, return     */
                                                   /* path position plus problem    */
     THEN RETURN PathPosition[str] + Length[str];  /* length (slope).               */
     ELSE RETURN Nilness[str];                     /* Else return position on slope, */
                                                   /*   which is number of zeroes.  */

Fitness[str] :=
  IF (Odd[Length[str]])                            /* Check for odd or even         */
                                                   /* problem length.               */
     THEN RETURN HillPosition[str];                /* If odd, pt. could be on path. */
     ELSE IF (str == "0|rest-of-string")           /* If even, and leading zero,    */
                                                   /* point could be on path, so    */
           THEN RETURN HillPosition[rest-of-string] + 1;  /* remove zero and check. */
           ELSE RETURN Nilness[rest-of-string];    /* If leading one, then          */
                                                   /* off path, so return           */
                                                   /* position on nil. slope.       */
```

Figure 3.1: An efficient decoding algorithm for the Root2path function, $k = 1$. *Fitness* returns the problem value of a given string *str*.

The function *Fitness* can be used directly as the objective fitness function for optimization (maximization).

3.2.3 Example Paths

To help illustrate the recursive path construction and the exponential growth of path length, Table 3.1 presents some example paths for the six lowest order (ℓ) Root2paths. Each path is presented as an ordered sequence of strings (points) from first (all-zeroes) to last (the global optimum). Note how even-order ℓ paths are equivalent to the odd-order $\ell - 1$ paths but with a "0" prepended to all points.

Table 3.1: Example Root2paths for problem lengths $\ell = 1..6$.

$P_1 =$	$0, 1$
$P_2 =$	$00, 01$
$P_3 =$	$000, 001, 101, 111, 110$
$P_4 =$	$0000, 0001, 0101, 0111, 0110$
$P_5 =$	$00000, 00001, 00101, 00111, 00110, 10110, 11110, 11111, 11101, 11001, 11000$
$P_6 =$	$000000, 000001, 000101, 000111, 000110, 010110, 011110, 011111, 011101, 011001, 011000$

3.3 Visualization

Although simple to specify and construct, the Root2path possesses a complicated recursive structure and it exists in a high-dimensional (ℓ dimensional) space. In this section we try to visualize the Root2path using two different techniques to improve our intuitive understanding of the algorithm.

3.3.1 Visualization in Three Dimensions

We first try to visualize a two dimensional long path: Figure 3.2. Here the fitness is a function of the integers x and y: $f_{lp}(x, y)$, for the "long path" function. As with the "binary space" Root2path, the two dimensional spiral path is generated by induction on the size of the search space. Here s is the integer range over which x and y each vary. For the base case $s = 1$, the single point is the global optimum, with fitness 1; it is the only point on the path. Informally, the inductive step is to take a path of integer range (size) s, add a ring (or rather, square) of points around the outside, each with fitness 0, then add another ring of points and add them to the path, incrementing the fitness of every point on the size s path by the number of new path points added. This gives us a long path of size $s + 2$. In other words, every other increment of s adds another ring to the spiral and pushes up the old, inner spiral to maintain the slope up to the global at the center.

Thus the construction of the two dimensional spiral path has the same inductive form as those of binary-space long paths. However, in two dimensions we can actually put half the search space on the spiraling path to achieve path lengths of $O(|\text{search space}|)$. Again, we can separate non-adjacent points on the two dimensional spiral path by any number k of steps[5] by simply adding rings to the spiral every k increments of s instead of every other increment.

David H. Ackley describes such a visual image as Figure 3.2 in his 1987 dissertation (Ackley, 1987b):

> Although simple hillclimbing is guaranteed to work in unimodal spaces, it is worth pointing out that the path to the solution need not always be very direct. In Figure 1–4 [in (Ackley, 1987b)], for example, suppose there was a serpentine ridge in the landscape, spiraling around and around and up the side of the mountain. The hillclimber would get "caught" on the ridge, and the locally uphill path would lead it on a merry tour around the mountain. The hillclimber would gain altitude only very slowly. (pp 10–11)

3.3.2 Visualization in Two Dimensions (Binary Space)

Terry Jones (Jones, 1995; Jones & Forrest, in press) recently discovered that the original, binary space version of the Root2path, as constructed above, can be visualized in a two dimensional graph. Jones examines the use of *fitness distance correlation* (FDC) as a measure of problem difficulty, and applies it to a number of artificial problems, including the Root2path. In particular, he shows scatterplots of a sampling of points in the search space. Each plot shows distance (to the global optimum) on the x-axis, and fitness on the y-axis. Thus each sampled point is plotted as a coordinate pair: *(distance,fitness)*. For the Root2path, such a plot reveals the nilness slope quite clearly. If the on-path points are purposely sampled[6], the path itself is revealed[7].

Since the global optimum of the Root2path is only two bits different from the all-zeroes point (e.g., "1100000..."), a plot of the Root2path fitness function as a *function of unitation* (i.e., the number of "1" bits) is very similar to Jones' scatterplots. In Figure 3.3, I show six such plots, each with fitness $f_{\ell p}(s) \equiv \text{HillPosition}(s)$ plotted against unitation $u(s)$ for all 2^ℓ

[5] I assume the *metropolitan* or *city-block* metric for this 2-D space.

[6] Being a vanishingly small part of the search space, the path is not adequately represented under uniform random sampling of the entire search space.

[7] Jones found that FDC worked well predicting problem difficulty for many functions, but its accuracy on the Root2path was impeded by the small chance of randomly sampling the path. Forced adequate path sampling gave a much different FDC measure than random sampling. See (Jones, 1995; Jones & Forrest, in press) for discussion of FDC measures of the Root2path.

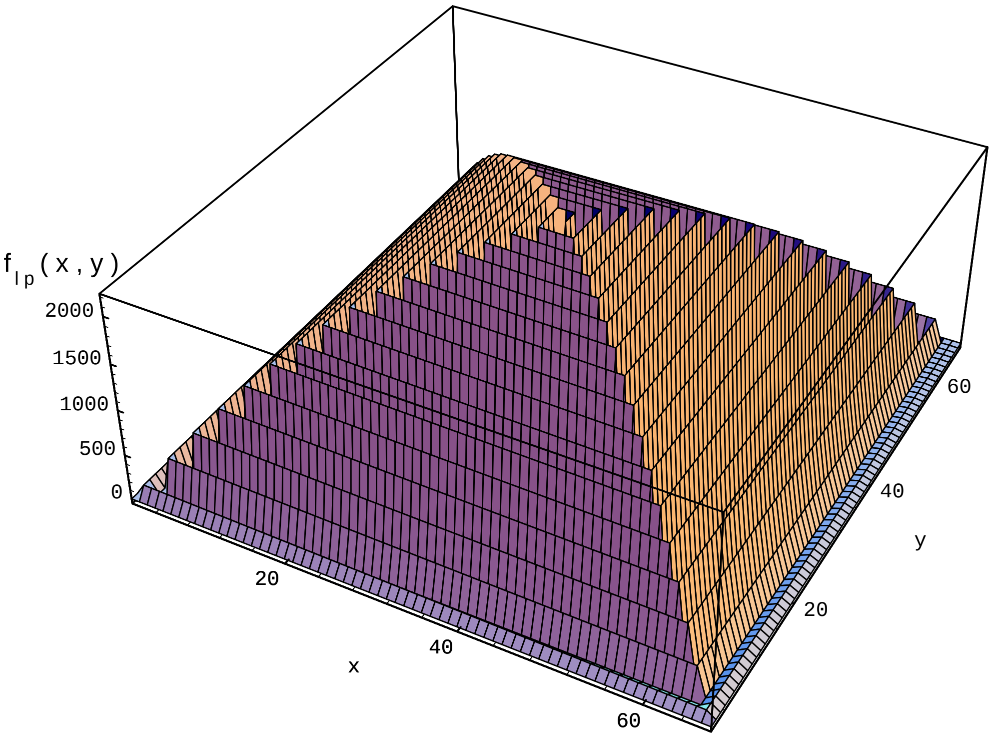

Figure 3.2: Two dimensional analog of the long path function $f_{\ell p}(x, y)$.

strings s of problem length ℓ, for $\ell = 1, 3, 5, 7, 9, 11$. (Remember that path length doubles for *odd* ℓ only, in our construction of the Root2path.) Here I have clearly labeled the nilness slopes and the paths themselves, connecting consecutive "levels" on the slope and consecutive steps on the path. The recursive structure of the path is quite clear for the larger problem sizes. Comparing a plot of an order ℓ path with that of an order $\ell + 2$ path shows how the latter is composed of two copies of the former "pivoting" about the central *bridge point* (labeled "**B**").

3.4 Simulation Results

In this section I demonstrate the predicted behavior of some simple hillclimbing algorithms on instances of the Root2path long path problem. As expected, one-bit hillclimbers take exponential time to climb the path, while $k > 1$-bit hillclimbers appear to take the shortcut and climb the path in linear time. I then compare the performance of the one-bit hillclimbers to that of a GA with no mutation (i.e., only crossover). The crossover-based GA finds the global optimum in many fewer function evaluations than the hillclimbers. Thus crossover can "climb" some hills faster than a hillclimber, a somewhat counter-intuitive result. Next I examine a single, partial "family tree" of a global optimum's ancestors in a single GA run, to try to get an intuitive, qualitative understanding of how crossover is succeeding on this function originally designed for hillclimbing. Further empirical results on GA crossover optimization of long path problems can be found at the end of the next chapter.

3.4.1 The Long Road for Hillclimbers

I have chosen to limit the testing of hillclimbers to five simple algorithms, analyzed in (Mühlenbein, 1992):

- Steepest ascent hillclimber (SAHC)
- Next ascent hillclimber (NAHC)
- Fixed rate mutation algorithm (Mut)
- Mutation with steepest ascent (Mut+SAHC)
- Mutation with next ascent (Mut+NAHC)

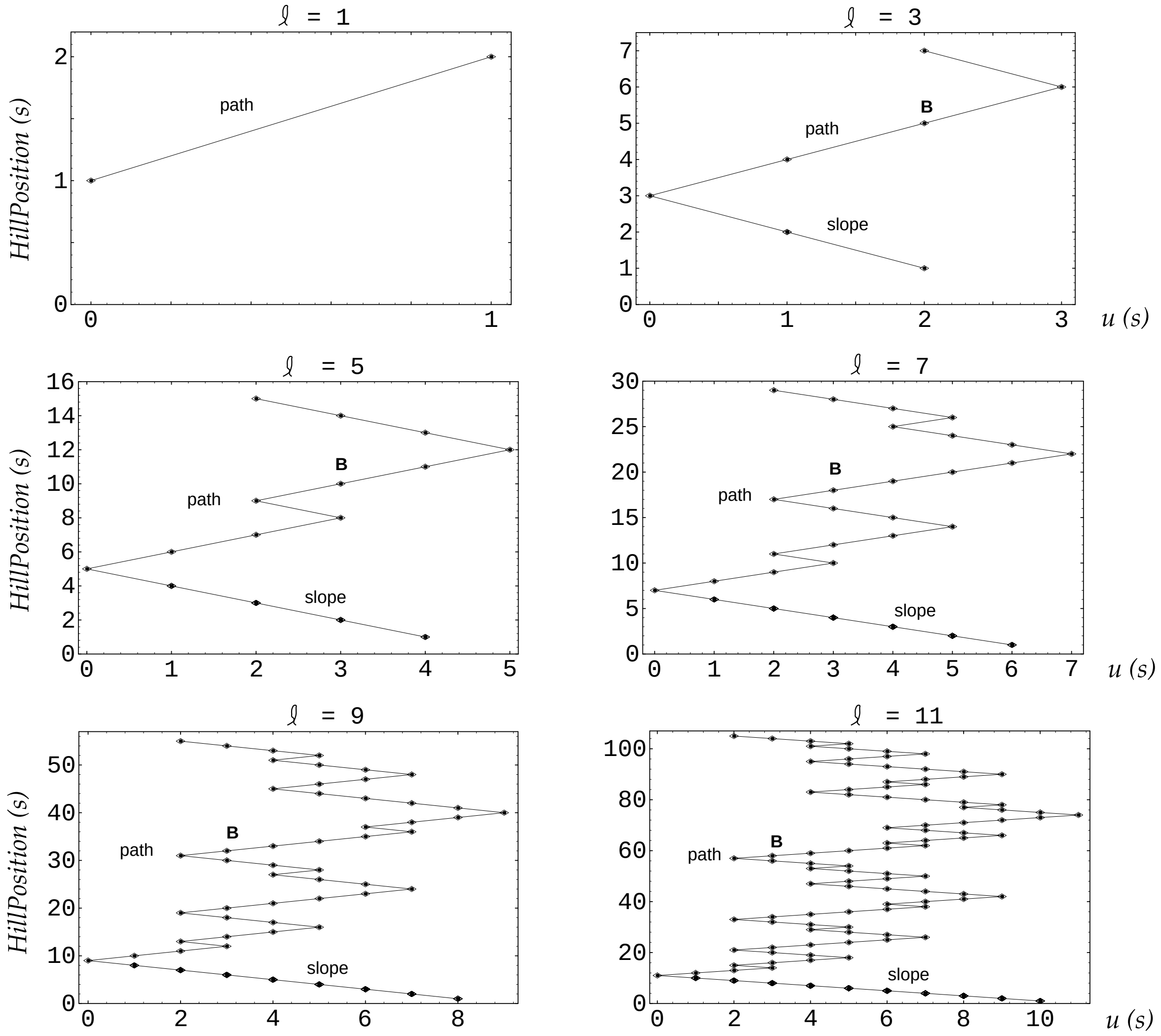

Figure 3.3: Scatterplots of $f_{\ell p}(s)$, with unitation $u(s)$ versus fitness HillPosition(s) ($\equiv f_{\ell p}(s)$) of string s. As Jones (1995) discovered, this kind of plot reveals the recursive structure of the Root2path. Note how the order ℓ path is made from two copies of the order $\ell - 2$ path pivoted around the bridge point **B**.

All five algorithms work with one point, s, at a time. The first point, s_0, is chosen randomly. Thereafter, s_{n+1} is found by looking at a neighborhood of s_n. With steepest ascent, all of s_n's neighbors are compared with s_n. The point within that neighborhood that has the highest fitness becomes s_{n+1}. If $s_{n+1} = s_n$, then steepest ascent has converged to a local (perhaps global) optimum. Next ascent is similar to steepest ascent, the only difference being that next ascent compares neighbors to s_n in some fixed order, taking the first neighbor with fitness greater than s_n to be s_{n+1}. In my runs, I assume SAHC and NAHC explore a neighborhood of radius one (step size $k = 1$).

Mutation (Mut) flips each bit in s_n with probability p_m, independently. The resulting string is compared with s_n. If its fitness is greater, the mutated string becomes s_{n+1}, otherwise s_n does. Mühlenbein (1992), and other researchers, have found that a bitwise mutation rate $p_m = 1/\ell$ is optimal for many classes of problems. The only mutation rate I use here is $1/\ell$.

The other two hillclimbing algorithms I run are combinations of mutation with steepest ascent (Mut+SAHC) and next ascent (Mut+NAHC). These combinations are implemented by simply mutating s_n, and allowing either next ascent or steepest ascent hillclimbing to explore the neighborhood around the mutated string. The resulting string, either the originally mutated string or a better neighbor, is then compared with s_n for the choice of s_{n+1}.

I test on Root2paths of dimension $\ell = 1$ to 20. I only consider (construct) paths of step size $k = 1$. In Figure 3.4, I plot the performance of the five hillclimbers. For each problem size ℓ, I run each algorithm at uniformly random starting points in the total search space. The plotted points are averages over five runs. I measure performance as the number of iterations (of the hillclimber's main update loop) required to reach the global optimum (i.e., the number of points in the sequence s_n). Note that this is less than the number of fitness evaluations used. Steepest ascent, for example, searches a neighborhood of radius one everytime it updates s_n; therefore its number of fitness evaluations is ℓ times the number of iterations.

As Figure 3.4 illustrates, at least three of the algorithms appear to perform superlinearly (perhaps exponentially) worse as ℓ increases. Mutation by itself tends to spend a long time finding the next step on the path[8]. Steepest and next ascent tend to follow the path step by step. Steepest ascent with mutation exhibits linear performance, however. The superiority of

[8] There exists a small chance that mutation by itself will flip two or more bits and take a big "shortcut", bypassing a significant portion of the path.

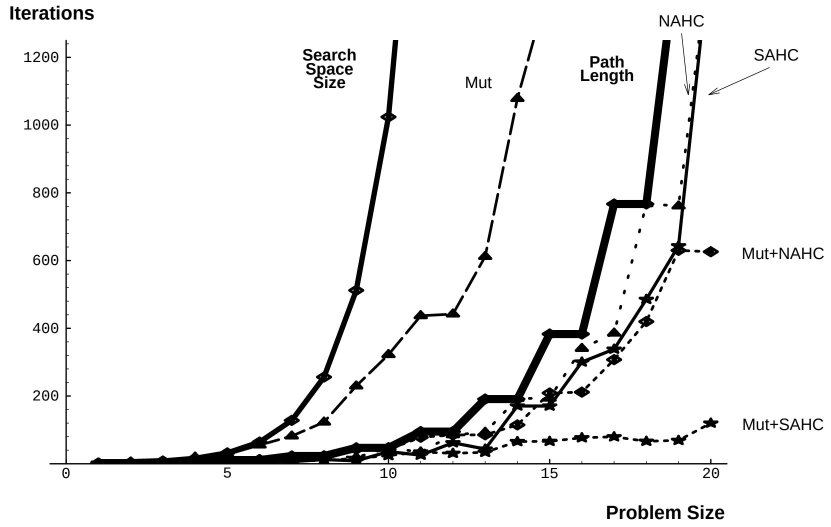

Figure 3.4: The performance of five hillclimbing algorithms as a function of the problem size ℓ. Performance (number of iterations of their main update loops) is averaged over five runs.

this hillclimbing variant is explained by its tendency to take steps of size two. Mutation with $p_m = 1/\ell$ takes a single-bit step, in expectation. Steepest ascent then explores the immediate neighborhood around the mutated point. Since the Root2path contains many shortcuts of step size two[9], steepest ascent with mutation is able to skip several large segments of the path.

To force Mut+SAHC to stay on an exponentially long path, we clearly need paths of greater step size. A simple way of building such a path is to extend the construction of the Root2path as follows. For a step size of $k = 2$, we should double the path every *third* increment in dimension. Thus we prepend "111" and "000" to copies of P_ℓ to get $P_{\ell+3}$. We now have a path whose length grows in proportion to $2^{\ell/3}$. We could call these *CubeRoot2paths.* This can be generalized to step size k, to get paths that grow as $2^{\ell/(k+1)}$, exponential in ℓ for $k \ll \ell$.

3.4.2 Crossover's Success

It is not obvious that the long path problems are amenable to recombinative search. From its inductive construction, it is clear that the Root2path has structure. The same basic subse-

[9] For example, the beginning and end of the Root2path are only two bits apart!

Table 3.2: GA crossover versus hillclimbers: results from 10 runs of each algorithm.

PERFORMANCE ON **Root2path**, stepsize $k = 1$			
	Number of Functions Evaluations to Global Optimum mean (std. dev.)		
	problem size		
algorithm	$\ell = 29$	$\ell = 39$	$\ell = 49$
SAHC	1,425,005 (789)	>15,000,000 (0)	>40,000,000 (0)
NAHC	1,359,856 (247)	>15,000,000 (0)	>40,000,000 (0)
RMHC	1,382,242 (112,335)	>15,000,000 (0)	>40,000,000 (0)
GA (Xover only)	25,920 (5830)	75,150 (53,500)	151,740 (140,226)
Pop. size	4000	5000	6000
Height of Hill (path + slope)	49,179 steps	1,572,901 steps	50,331,695 steps

quences of steps are used over and over again on larger scales, resulting in fractal self-similarity. But I do not know if such structure induces building blocks exploitable by crossover (Holland, 1992), as I have not yet applied a schema analysis to these functions. However, certain substrings such as "11" and "1011" (in certain positions) are common to many points on the path. And the first results, summarized in Table 3.2, indicate that a GA with crossover alone is an effective strategy for climbing long hills in a short time.

In Table 3.2, I compare three hillclimbers to a GA on three different size Root2path problems (all with stepsize $k = 1$). The GA is a simple, *generational* GA, using single point crossover with probability $p_c = 0.9$, no mutation ($p_m = 0$), binary tournament selection, and the population sizes indicated in Table 3.2. Random mutation hill climbing (RMHC) is described in (Forrest & Mitchell, 1993; Mitchell & Holland, 1993; Mitchell, Holland, & Forrest, 1994). RMHC is like mutation-only (Mut) above, except that one and only one bit flip takes place. Thus RMHC starts with a random string s_0, and updates s_n by randomly flipping one bit in s_n to form s'_n. If s'_n is better than, or equal to s_n, then s'_n becomes s_{n+1}, otherwise s_n becomes s_{n+1}.

In (Forrest & Mitchell, 1993; Mitchell & Holland, 1993; Mitchell, Holland, & Forrest, 1994) the authors found that RMHC optimized Royal Road (RR) functions faster than SAHC, NAHC, and the GA. On the Root2path problems, however, the GA[10] seems to outperform the three hillclimbers, and RMHC apparently loses out to the *deterministic*[11] hillclimbers (due to the larger variance in its performance). Comparing the long path problems to the RR functions is not within the scope of this thesis[12]. But these early results pointing to superior performance by crossover might be of particular interest to those looking at when GAs outperform hillclimbers (Mitchell & Holland, 1993; Mitchell, Holland, & Forrest, 1994). One answer might be "on a hill" (at least, a certain kind of hill).

How is the GA finding the global optimum? Clearly it is not climbing the hill one step at a time, nor is it using mutation ($p_m = 0$). It is crossover (with selection) that is solving the problem, but it is not clear if crossover is recombining short, low-order, highly fit schemata (building blocks). The global optimum is only two bits different from the top of the "ZERO-MAX" slope, which crossover is known to easily "climb". One should therefore suspect that crossover is somehow accomplishing a two-bit "mutation via random crossover" to generate the global optimum.

I perform a quick check of these suspicions by looking at the "family history" of the first instance of the global optimum. Although a single run (starting with a single random initial population) cannot provide reliable statistics, it is nevertheless informative and revealing to trace the ancestry of an optimal string back through the early generations of a single run. Figure 3.5 is a partial trace of a successful GA run in which the first instance of the global optimum appears in generation 15. In this run problem length $\ell = 47$ bits, population size $N = 10000$, probability of single-point crossover is high ($p_c = 0.9$), and no mutation ($p_m = 0$) is used.

As Figure 3.5 illustrates, even a careful trace of the global optimum's ancestry does not completely clarify the situation. For example, one could interpret the mating at generation 14 to be an effective two-bit mutation of the "easy-to-find" all-zeroes string by a crossover with some (any) string with the appropriate two bits. On the other hand, one could argue that the

[10] For the GA I estimate the number of fitness evaluations (to find the global) as numgens $*$ popsize $* p_c$, where *numgens* is the number of generations until the global optimum **first** appears. If the GA prematurely converges (i.e., to a non-optimal point), it is restarted **without** resetting *numgens*. Thus *numgens* is cumulative over multiple (unsuccessful) GA runs.

[11] SAHC and NAHC are deterministic in the sense that they are guaranteed to find the global optimum of the Root2path within a maximum time. The GA and RMHC, on the other hand, are stochastic.

[12] It is interesting to note, however, that both landscapes are unimodal (by my definition of local optimality).

Figure 3.5: Partial family tree of the global optimum.

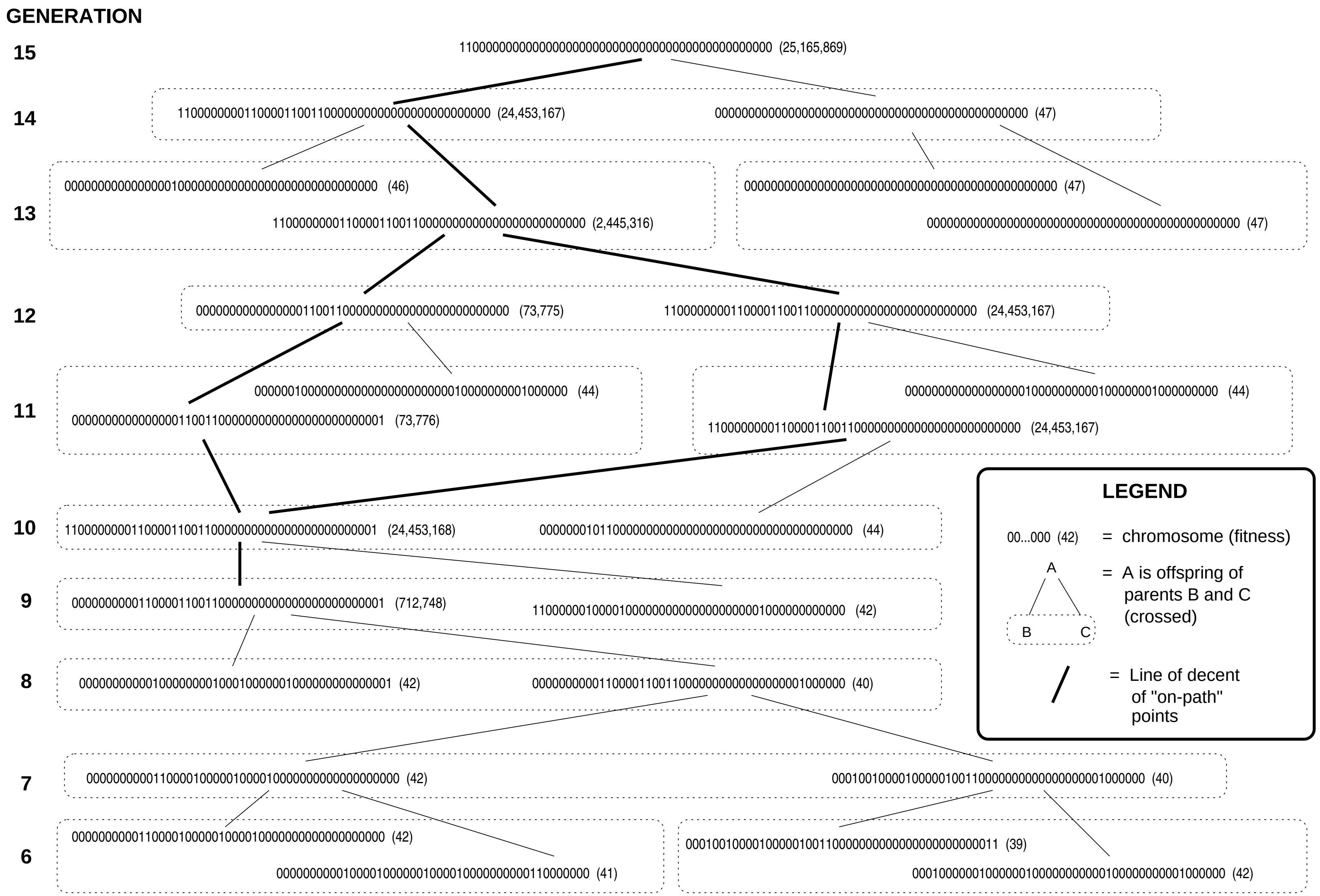

Figure 3.5: Partial family tree of the global optimum.

line of descent of high-fitness (on-path) ancestors (the bold lines in Figure 3.5), showing a series of "large jumps" on the path, indicates incremental improvement which in turn implies steady but rapid crossover "hillclimbing". Therefore, one can argue, it is not a single lucky "mutation" of the all-zeroes point that enables crossover to reach the end of the path, but rather a process of gradual improvement through the continuous processing of intermediate points on the path.

I suggest that what Figure 3.5 tells us above all is that thinking of crossover as a hillclimber or a mutator is difficult and easily misleading. With a relatively high crossover rate of 90%, and relatively high diversity (at least in these early generations, 0 to 15, of a size $10,000$ population), very few individuals survive a generation unchanged. Rather it is subsets of contiguous bits (i.e., schemata) that are surviving through lines of descent. For example, the pattern "0011001100" in bit positions 9 through 20, near the middle of the string, survive from their creation in the righthand individual in generation 8 through to the creation of the lefthand parent of the global optimum in generation 14. Indeed, this "building block" is passed through two separate lines of descent in generations 11 and 12. Similarly, zeroes tend to be good alleles both on the slope and on the path. The survival of large segments of contiguous zeroes allows the GA to "zero out" appropriate loci at intermediate points in the run, including the final step to the global optimum at generation 15.

Clearly crossover is solving this problem by processing schemata. I do not know if this is the most efficient method on this or any other long path problem. Indeed, crossover here seems to be exploiting the recursive structure of the Root2path. Such structure might or might not be present in other long paths[13]. But we have now have clear evidence that GA crossover is able to find the top of a hill much faster than some hillclimbers, apparently by processing schemata.

3.5 Extensions

The long paths problems are new to the GA literature. Many minor but interesting variations come to mind. For example, the Root2path is a "purely convex" function, meaning that the entire search space is a single hill, with gradients present at every point and in all directions (i.e., non-zero gradients exist between every pair of adjacent points). However, on the nilness slope are many points with identical fitness (i.e., all strings/points with identical unitation).

[13] The same basic structure should however be present in all Root2paths, including those of stepsize $k > 1$.

Thus there are pairs of (non-adjacent) points with no gradient. We could easily add gradients to such pairs (meaning that every point in the space would have a unique fitness) by adding a component function such as

$$f_{bcd}(s) = \frac{BCD(s)}{2^\ell},$$

where $BCD(s)$ is the *binary coded decimal* function, returning the integer from 0 to $2^\ell - 1$ decoded from the bit string s. The function $f_{bcd}(s)$ is uniquely valued for all strings s and ranges over $[0,1)$. Thus $f_{bcd}(s)$ added to the Root2path fitness would not change the fitness ordering of any path elements, nor would it induce any additional local optima. But it would mean that no two points in the space have the same value[14], an interesting (and in some situations important) search space characteristic.

In the remainder of this section I present what I believe are more significant potential results of the long path work. These ideas fall along two directions: construction of longer paths and rigorous analysis of expected performance.

3.5.1 Longer Paths

It is certainly possible to construct paths longer than the Root2path. A *Fibonacci path* of step size $k = 1$ is constructed inductively like the Root2path. The inductive step goes as follows. Given a Fibonacci path F_ℓ of dimension ℓ, and a Fibonacci path $F_{\ell+1}$ of dimension $\ell + 1$, we construct a path of dimension $\ell + 2$ by skipping a dimension and then doubling F_ℓ as with the Root2path. But rather than simply adding a single "bridge" point to connect the two copies of F_ℓ, we use the path $F_{\ell-1}$ to connect them. We know we can use $F_{\ell-1}$ to connect the two copies of F_ℓ since $F_{\ell+1}$ is composed of an F_ℓ path coupled with an $F_{\ell-1}$ path.

The formal construction and inductive proof of existence of the Fibonacci path will have to be postponed. The important result to mention here is that the Fibonacci path grows faster in length than the Root2path. The sequence of Fibonacci path lengths, obtained by incrementing ℓ, is the Fibonacci sequence, {1,2,3,5,8,13,21,34,55,...}, since:

$$|F_\ell| = |F_{\ell-1}| + |F_{\ell-2}|$$

[14] N. Radcliffe (personal communication, 1994) points out that $O(2^\ell)$ or $O(2^{\ell/2})$ different function values could easily exceed the floating point precision of many present day machines for even modest ℓ.

Solving the recurrence relation reveals exponential growth of $\approx 1.61803^{\ell}$, which has the golden ratio as its base. This base is larger than the base ≈ 1.414 in the exponential growth of the Root2path, but is still < 2. Thus even the Fibonacci paths will asymptotically approach zero percent of the search space.

The problem of finding a maximally long path with minimal separation has some history, and is known as the "snake-in-the-box" problem, or the design of difference-preserving codes, in the literature on coding theory and combinatorics (MacWilliams & Sloane, 1977; Preparata & Niervergelt, 1974). Maximizing the length of paths with k-bits of separation is an open problem, even for $k = 1$. However, upper bounds have been found that are $< O(2^{\ell})$. Thus the longest paths one can ever find[15] will be $O(2^{\ell/c})$ for some constant $c > 1$. For the Root2path, $c = 2$ for $k = 1$. For the Fibonacci path, $c = 1/(Log_2\Phi) \approx 1.44042$ when $k = 1$, where Φ is the golden ratio.

3.5.2 Analysis of Expected Performance

We need both empirical and analytical results for expected performance of various mutation algorithms and recombinative GAs. We could explore such apparent tradeoffs as step size k versus length of the path $|P|$. As k increases, $|P|$ decreases exponentially in k, but the number of fitness evaluations required to effectively search a neighborhood of radius k increases exponentially in k. A similar tradeoff involves the mutation rate p_m. As p_m increases, the mutation-only algorithm is more likely to take a larger shortcut across the path, but it is also less likely to find the next step on the path. These kinds of tradeoffs, involving parameters of the search space design and the algorithms themselves, are amenable to analysis of expectation.

3.6 Discussion

Long path problems are clearly and demonstrably difficult for local searchers (that is, algorithms that search small neighborhoods with high probability, and larger neighborhoods with vanishingly small probability). Such algorithms include hillclimbers and the GA's mutation op-

[15] It is easy to show that maximum path lengths must be $< 2^{\ell}$, at least for $k \geq 3$: divide the total volume of the search space by a volume $v(k)$ that is a lower bound on the number of off-path points that must "surround" each point on the path. This upper bound indicates that for $k \geq 3$, the optimal path length must approach zero exponentially fast as ℓ increases. This means that for $k \geq 3$, the best growth in path length for which we can hope is $O(x^{\ell})$, where $x < 2$.

erator. Surprisingly, some of these "intractable hills" can be solved efficiently by GA crossover. The fact that the GA solves problems specifically contrived for hillclimbers lends support to the intuition of GA robustness. Also able to solve long path problems of step size k are hillclimbers of step size $k' > k$. But long path problems of step size k' can be constructed to defeat such hillclimbers. The GA (with crossover) on the other hand *might* scale smoothly with increasing k. Such a result has implications for hybrid algorithms[16] that perform hillclimbing during or after regular GA search: the addition of hillclimbing to the GA could make an "easy" problem intractable. Thus the long path problems reveal another dimension of problem difficulty for evolutionary computation; a dimension along which we can characterize, measure, and predict the performance of our algorithms.

Although GA crossover can outperform some simple hillclimbers on some long path problems, fairly large population sizes are apparently required for crossover to succeed. I have shown reliable performance (in converging to the global optimum) only with population sizes of $N = 4000, 5000,$ and 6000, on Root2path problem lengths of $\ell = 29, 39,$ and 49 respectively. As I show in the next section on "maximum modality", the GA does not optimize the long path problems reliably with small population sizes (e.g., $N = 300$).

The long path function is interesting also because it points out that the modality of a search space, if measured solely as the number of local optima, is at best a first order estimate of the amenability of the space to search by hillclimbers, mutation, or by evolutionary search algorithms in general.

[16] For example, Radcliffe and Surry's (1994) *memetic algorithm*. See (Radcliffe & Surry, 1994) for a short summary of such hybrid GA/hillclimber algorithms.

Chapter 4

Maximum Modality Can Be Easy

At the other extreme of the modality spectrum, what is the maximum number of local optima possible in a binary problem of size ℓ-bits? we assume minimum radius (one-bit) peaks, so that our requirement for local optimality is minimal. That is, a point is a local optimum if and only if all adjacent points have inferior fitness values. We can calculate an upper bound on the number of such optima.

4.1 Calculating an Upper Bound

Let p be the number of local "peaks". In an ℓ-bit problem, each peak must have exactly ℓ immediate neighbors that are not local optima (call them "nonoptima"). Thus the number of adjacent pairs of optimum-nonoptimum is $p * \ell$. That is, for a problem to have p local optima it must have $p * \ell$ different pairings of optima and adjacent nonoptima. There are exactly $2^\ell - p$ nonoptima. Each of these nonoptima can have at most ℓ adjacent optima. Thus an *upper* bound on the number of optimum-nonoptimum pairings is $(2^\ell - p) * \ell$. We can increase p until the number of optimum-nonoptimum pairs equals the maximum: $p * \ell = (2^\ell - p) * \ell$. We derive $p = 2^{\ell-1}$, which is half the size of the search space[1]. This is an upper bound on the number of local optima.

[1] In other words, the number of optima cannot exceed the number of nonoptima.

4.2 Construction of f_{mm}

Using the concept of unitation, we construct a function with $2^{\ell-1}$ local optima, thus showing that the upper bound is indeed the exact maximum number of local optima. Ackley (1985, 1987a, 1987b) constructs such a "fine-grained local maxima" function, naming it the *porcupine function.* Although he neither claims nor shows that the porcupine function has the *maximum* number of local optima, Ackley does state that it has "an exponential number of local maxima" (Ackley, 1987a, 1987b)[2]. Here we construct a strictly non-negative variant of the porcupine function and prove that it does indeed contain $2^{\ell-1}$ local optima.

A *maximally multimodal* function f_{mm} of unitation assigns a high fitness to all bit strings of odd unitation, for example, and a low fitness to all strings of even unitation:

$$f_{mm}(s) = \begin{cases} 1 & \text{if } \mathrm{Odd}(u(s)) \\ 0 & \text{otherwise} \end{cases}$$

Since all strings of odd unitation are separated from each other by strings of even unitation, odd unitation strings are indeed local optima. Strings of odd unitation occupy exactly half the search space[3].

A maximally multimodal function[4] has the maximum number of possible "attractors" on which a search algorithm might get stuck. But of course each optimum's basin of attraction is at a minimum size. To illustrate that massive multimodality *by itself* does not imply difficulty for GAs or hillclimbers in general, we can add a gentle slope to the function that leads quickly

---

[2] By this he must mean that the number of local optima grows exponentially in problem size ℓ.

[3] It is interesting to note that maximally multimodal functions, in which half the search space are local optima, do not have to be functions of unitation but must have all of their local optima be of the same parity of unitation (i.e., either odd or even). This result can be shown by induction: Choose a local optimum. It has either odd or even unitation. The nearest neighbors are not local optima and are all of the opposite parity of unitation from the chosen optimum. Their nearest neighbors in turn must be local optima (since the function maximizes the number of optimum-nonoptimum pairings) and must be of the same parity of unitation as the first chosen optimum, and so on.

[4] The function f_{mm} is similar to the "maximally rugged" NK-landscapes of Kauffman (1989). In the general NK-landscapes, the bit-wise (or "loci") contributions to fitness are (random) functions depending on K other bits (out of the total N $(= \ell)$ bits in the problem). The parameter K provides a tunable ruggedness to the resulting landscape. At one extreme, $K = N - 1$, the function has approximately $\frac{2^N}{N+1}$ local optima (Manderick, de Weger, & Spiessens, 1991). Note that the individual contributions of each bit of f_{mm} are also dependent on all N $(= \ell)$ bits, as is true for any function of unitation (parity). However, unlike NK-landscapes, the loci contributions of f_{mm} are uniform, not random.

to the global optimum (all-ones in this case)[5], as Ackley did in his porcupine function (Ackley, 1985, 1987a, 1987b):

$$f_{mm,easy}(s) = u(s) + 2 * f_{mm}(s). \tag{4.1}$$

The function $f_{mm,easy}(s)$ resembles a one-max function "with bumps".

4.3 Expected Performance for Hillclimbers

A hillclimber with a non-zero probability p_{up2} of taking a step of size two or more bits uphill will climb to the global optimum in at most ℓ/p_{up2} steps[6]. For any p_{up2} that decreases no faster than linearly in ℓ, and in particular for a constant p_{up2}, the hillclimber will climb the hill in expected $O(\ell)$ steps[7]

4.4 Schema Analysis

A GA is also unlikely to have difficulty in quickly optimizing this function. In addition to being amenable to search by the GA's mutation operator, the function appears easy for crossover when we apply a static analysis of schema partitions. In every schema partition, the schema containing the global optimum (the all-ones schema) must have a higher average fitness than all other schemata competing in that partition. To see why the all-ones schema always wins, we first calculate the schema average fitness for any schema as a function of the schema's unitation (number of ones in the defined bits). In a partition of order o, the fitness of a schema $\hat{s}$ is equal to the unitation of the schema ($= u(\hat{s})$) plus the average unitation of the $(\ell - o)$ undefined bit positions ($= (\ell - o)/2$), plus twice the average contribution of $f_{mm}(s)$, the parity of unitation

[5] Here we assume that ℓ is odd. For even ℓ, we should change $f_{mm}(s)$ to favor even unitation if we want the all-ones string to be the global optimum.

[6] This is an upper bound on expectation: $E[t] \leq \ell/p_{up2}$, where t is the number of steps taken to reach the global. Note that these calculations assume nothing about the average number of function evaluations (or time involved) in taking a single step.

[7] A hillclimber with a significant p_{up2} is also expected to climb the Root2path in linear or constant time. However, I showed above how to construct the *CubeRoot2path*, a long path for hillclimbers of step size 2. A hillclimber with significant p_{up2} but near zero p_{up3} (probability of taking a step of size three or more bits) will take an expected time exponential in ℓ to climb such a path. In other words, the intractability of the long path problem class, and the ease of the maximally multimodal (easy) problem, (for hillclimbers of stepsize k) both scale with increasing k.

function, which will be $2 * 1/2$:

$$\bar{f}_{mm,easy}(\hat{s}) = u(\hat{s}) + \frac{(\ell - o)}{2} + 1. \tag{4.2}$$

The schemata with highest unitation will always be the winners of their partition competitions.

The function $f_{mm,easy}(s)$ is also amenable to Walsh analysis (Bethke, 1981; Goldberg, 1989b), since it can be written as a summation of just a few Walsh functions. Using the notation of Goldberg (1989b), we note that for $f_{mm,easy}(s)$ all the Walsh coefficients w_i of all orders i are zero, except for $w_0 = \ell/2 + 1$, $w_1' = -1/2$, and $w_\ell = -1$, where w_1' stands for all of the order-1 Walsh coefficients, which are identically valued in a function of unitation. The fitness of a schema $\hat{s}$ with order $o < \ell$ is simply expressed:

$$\begin{aligned} \bar{f}_{mm,easy}(\hat{s}) &= w_0 + u(\hat{s})(-w_1') + (o - u(\hat{s}))w_1' \\ &= \ell/2 + 1 + \frac{u(\hat{s})}{2} - \frac{(o - u(\hat{s}))}{2} \\ &= u(\hat{s}) + \frac{(\ell - o)}{2} + 1. \end{aligned} \tag{4.3}$$

Again it is clear that for all competition partitions the schemata with highest unitation will have the greatest schema average fitness, thus pointing the way to the global optimum (the all-ones point).

4.5 Visualization

Figure 4.1 plots an $\ell = 29$-bit instance of the maximally multimodal $f_{mm,easy}(s)$ as a function of unitation of s. One can clearly see the global optimum at $u(s) = 29$. The unitation function allows us to view the multimodal structure of $f_{mm,easy}(s)$, but it does not reveal the large numbers of optima, since data points for odd $u(s)$ contains $\binom{\ell}{u(s)}$ local optima. To depict a large field of many small optima, we again look to a three dimensional plot of a two dimensional analog of $f_{mm,easy}(s)$: $f_{mm,easy}(x, y)$.

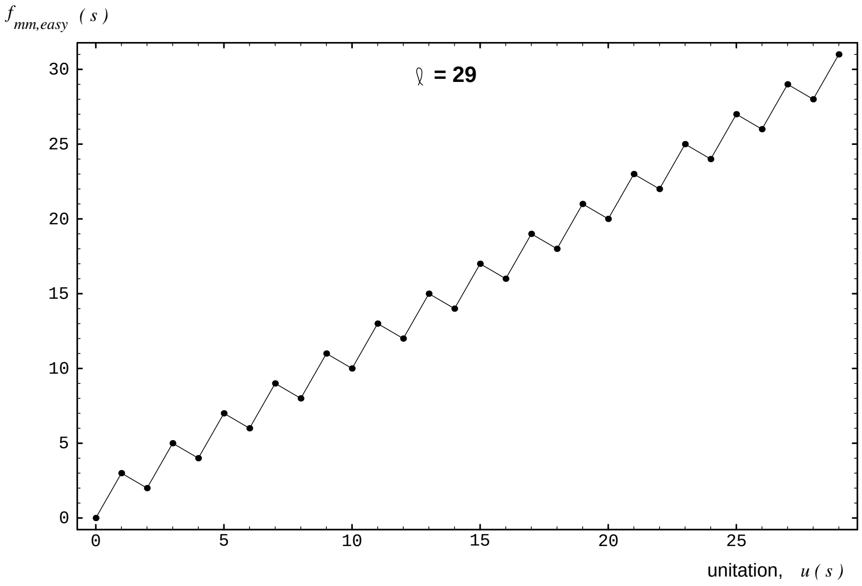

Figure 4.1: Two dimensional visualization of a $\ell = 29$ bit instance of the maximally multimodal function $f_{mm,easy}(s)$, plotted as a function of unitation $u(s)$.

Figure 4.2 is a visualization of a maximally multimodal function in three dimensions. Here the decision variables are the positive integers x and y, and the fitness function is

$$f_{mm,easy}(x,y) = (x+y) + \begin{cases} 10 & \text{if } \mathrm{Odd}(x+y) \\ 0 & \text{otherwise} \end{cases}$$

Just as in the case of binary strings, the 2-D formulation makes half the search space local optima while providing a constant gradient pointed straight at the global (for any $\geq$ 2-bit hillclimber) as well as plenty of schema information for crossover (at least for binary coded integers).

4.6 Simulation Results

Empirical results confirm the prediction that a crossover-based GA will perform quite well on $f_{mm,easy}(s)$. (Here I switch back to the ℓ-dimensional version of $f_{mm,easy}$ over binary strings s.) Table 4.1 summarizes a brief experiment in which I run a GA with crossover alone (probability of mutation $p_m = 0$) on three different sizes ($\ell = 29, 39$ and 49 bits) of both the long path problem

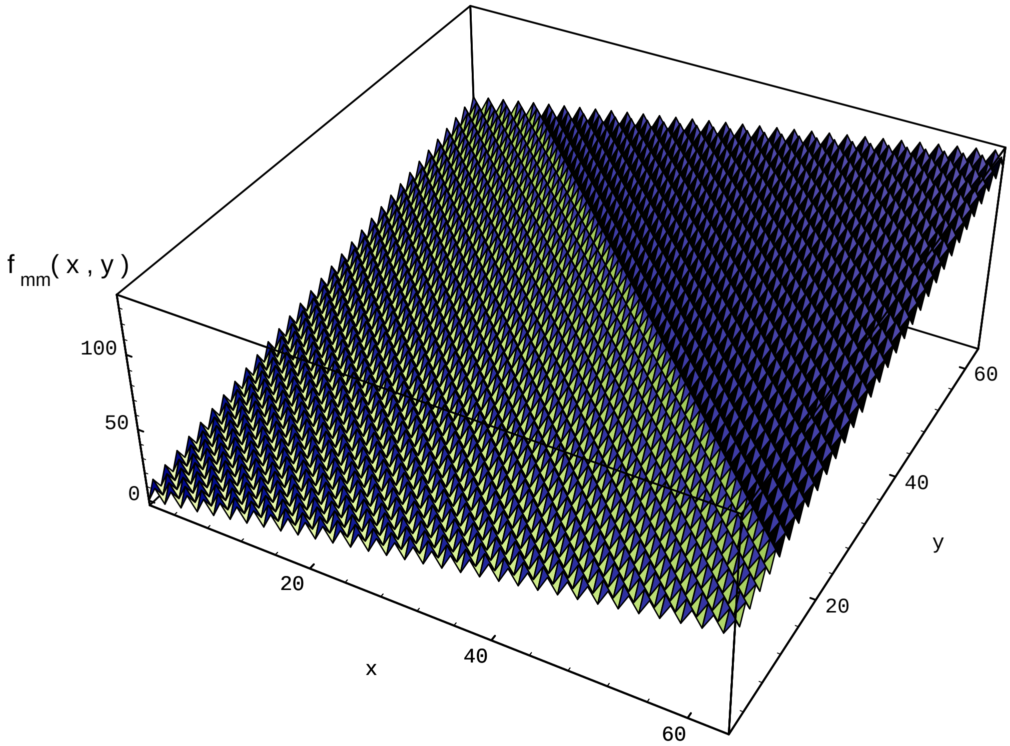

Figure 4.2: Two dimensional analog of the binary space maximally multimodal function.

Table 4.1: Unimodal versus Multimodal Problem Difficulty.

GA PERFORMANCE (**Long Path** vs. **Max. Multimodal**)			
	No. of Trials (of 40) Converging to Global		
problem type	**problem size**		
	$\ell = 29$	$\ell = 39$	$\ell = 49$
Long Path (Root2path, $k = 1$) f_{lp}	5	2	0
Maximally Multimodal $f_{mm,easy}$	40	39	39

$f_{lp}(s)$ and the maximally multimodal problem $f_{mm,easy}(s)$. The GA used in the experiment is a simple, generational GA with probability of (single point) crossover $p_c = 1.0$, (deterministic) binary tournament selection, and population size $N = 300$. For each problem type and size (e.g., 39-bit long path problem) I run 40 trials (i.e., 40 different random initial populations). The long path problems were all Root2paths with step size $k = 1$ as specified (by pseudocode) in the previous chapter. The maximally multimodal problem $f_{mm,easy}(s)$ used in the experiment is the same as that described in Equation 4.1 above. For each trial, the GA is run until the convergence criterion (uniform fitness of the population) was met. The final, converged population was then checked for the global optimum.

As Table 4.1 indicates, the GA performs much better on $f_{mm,easy}(s)$ than on $f_{lp}(s)$ for the three problem sizes shown. The GA with crossover alone cannot find reliably the end of the long path (i.e., the global optimum) for the larger problem sizes (i.e., $\ell = 39, 49$), except at higher population sizes (e.g., $N = 5000, 6000$) (see Table 3.2). The relative ease with which the GA solves $f_{mm,easy}(s)$ is consistent with the analysis above, with Ackley's reported observations of GA performance on his porcupine function (Ackley, 1985, 1987a, 1987b), and with our intuitions based on knowledge of the problem's construction. The relative performance of GA crossover is certainly not predicted, however, by a simple count of local optima in the search space.

4.7 Discussion

The maximally multimodal function is interesting if only because it points out that the modality of a search space, if measured solely as the number of local optima, is at best a first order estimate of GA difficulty (or difficulty for evolutionary search algorithms in general).

Chapter 5

Intermediate Modality: Local Optima and Their Basins of Attraction

The results of the previous two sections remind us that the fitness landscapes of interest to us (i.e., those that challenge the GA in a realistic and general manner) have intermediate modality. But it is not clear how to add modality to the long path problem, or to reduce the modality of the maximally multimodal function. For example, one might be tempted to generalize the work on maximum modality by maximizing the number of optima that are k or more bits apart (i.e., each optimum has a local neighborhood of radius $\geq (k-1)$ bits in which it is optimal). Unfortunately, calculating the maximum number of such optima for arbitrary k is an open problem in coding theory known as *sphere packing* (Harary, Hayes, & Wu, 1988; MacWilliams & Sloane, 1977). We can calculate an upper bound by simply dividing the search space size, 2^ℓ, by the hypervolume of the *nonoverlapping* local neighborhoods of the optima, $\sum_{r=0}^{\lceil k/2 \rceil -1} \binom{\ell}{r}$. But this results in a very loose upper bound, as evidenced by the simple case of $k=2$ bit separation: the upper bound yields 2^ℓ while we know that the actual maximum is half that[1].

[1] It is interesting to note that sphere packing with radius $\lceil k/2 \rceil - 1$ also provides an upper bound (within a factor of two) on the path length of $(k-1)$-step long path problems, since nonconsecutive steps on the path must be at least k bits apart. However, tighter upper bounds on $(k-1)$-step paths have been found (Preparata & Niervergelt, 1974).

5.1 Background

Although *maximizing* the number of local optima with neighborhoods of radius k is an open problem, work has proceeded along the lines of measuring and controlling the number of optima and their basins of attraction (or the "attractiveness" of the optima). Such work can be divided into two types: that which assumes hillclimbing attraction and that which assumes GA crossover attraction. That is, most papers analyzing local optima assume one type of algorithm or the other. I give some background on both approaches, but focus on GA crossover.

5.1.1 Hillclimbing Attraction

A number of recent papers define peaks (i.e., local optima together with their basins of attraction) in terms of hillclimbing. Goldberg (1991a) formally defines basins of attraction to a point x^* as all points x such that a given hillclimber has a non-zero probability of climbing to within some ϵ of x^* if started within some δ-neighborhood of x. Jones and Rawlins (1993) introduce *reverse hillclimbing* and *probabilistic ascent* as techniques for defining the basin of attraction of a particular local optimum in a fitness landscape for hillclimbers with known probabilities of ascent. Mahfoud (1993) analyzes the performance of multimodal GAs on multi-niche problems, where each niche is a local optimum with a basin of attraction defined by the probability of a hillclimber reaching the local optimum from a point in the basin.

5.1.2 GA Attraction

The literature on the attraction of peaks and regions in the landscape to the GA's crossover operator is largely based on Holland's *schema theorem* (Holland, 1992) and schema average fitness calculations (Bethke, 1981). Goldberg (1987, 1989a, 1989b, 1989c) and later others (Whitley, 1991; Homaifar, Qi, & Fost, 1991; Deb, Horn, & Goldberg, 1993) define and construct deceptive landscapes, in which the GA should be attracted to suboptimal local optima and led away from the global optimum. Schema analysis has also been used to construct "GA-easy" functions (Wilson, 1991; Deb & Goldberg, 1994) which have large basins of attraction for the global optimum[2]. More recently, Mitchell and Holland (1993; and with Forrest, 1994) have

[2] Like the maximally multimodal function, previous GA-easy functions contain local optima that can make the problem difficult for hillclimbing.

begun to weaken the GA-easy conditions by limiting the number and order of schema partitions leading toward the global[3].

I look to deception for guidelines on how to lead or mislead a GA both toward and away from multiple optima. Since a function with more than two global optima implies partial (and not full) deception[4], I first review and define partial deception.

5.1.3 Additional Definitions: Partial Deception

One must be careful in defining partial deception. It is easy to weaken the requirements for full deception such that a GA can easily find the global optimum of some functions meeting those requirements. I briefly present the definitions I use in this thesis.

A *deceptive attractor* (Whitley, 1991) is the suboptimal point D toward which a GA is (mis)led. In a particular schema partition, the schema containing the deceptive attractor is called the *deceptive schema*, while the schemata containing the global optima are called the *global schemata*. In general, a *deceptive partition* is a partition in which the deceptive schema has a high (schema average) fitness and/or all of the global schemata have low fitness. The rather vague concept of deceptive schemata "beating" global schemata, in terms of schema average fitness, has been interpreted in several different ways by different researchers. Here I list three specific definitions of interest (the order of labeling has only chronological meaning):

- **Type I deceptive** – global schemata lose to all other schemata (Bethke, 1981).
- **Type II deceptive** – deceptive schema wins over all other schemata (Goldberg, 1987).
- **Type III deceptive** – deceptive schema has higher fitness than all global schemata (Grefenstette, 1993).

It is not yet clear which, if any, of these definitions implies more difficulty for the GA on partially deceptive landscapes[5]. For fully deceptive functions, all partitions of order $< \ell$ are type II and

[3]It is interesting to note that according to my definition of local optimum, the Royal Road functions are unimodal.

[4]A function with more than one global optimum cannot be misleading in all ℓ of its order one partitions. Two (or more) distinct global optima must differ in at least one bit position. The order one schema partitions defined at each of those positions must "point to" at least one of the globals.

[5]I do note, however, that types I and II deception conditions imply type III.

III deceptive[6]. But in general (i.e., partial deception), deceptive and global schemata can be placed anywhere in the fitness ordering of a partition's schemata.

In the design of maximally misleading functions, our goal is to choose D and define the landscape such that the GA converges to D with high probability. I assume that I am given one or more globals (that is, their locations and perhaps their fitness value). In the case of a single global optimum g, the above definitions are sufficient to unambiguously identify a unique D, which is the complement of g (Whitley, 1991). I can then construct *fully deceptive* functions in which the schema containing D is the winner of *every* partition (i.e., types II and III deception) at every order up to the string length ℓ (Goldberg, 1989a, 1989b, 1991b). Full deception is clearly maximally misleading to a GA. It is also clearly bimodal, with local optima[7] at D and at g. To add more optima, and basins of GA attraction, I need to define partial deception.

Deception becomes a more practical tool of GA theory when it is embedded in fitness landscapes as *partial deception.* But at the moment, I can only define partial deception vaguely as less misleading than full deception and more misleading than GA-easy. Trying to order partially deceptive functions according to some scalar measure of deception is problematic. Goldberg recognizes the need to generalize full deception and does so by defining *order k full deception* (Goldberg, 1991b). Homaifar, Qi, and Fost (1991) call such limited deception *reduced order k deception.* Although it has gone by other names, this kind of partial deception is most widely known as *bounded deception*[8]. A problem has bounded deception of order k if and only if *all* partitions of order $(k-1)$ or less are deceptive[9]. Partitions of order $\geq k$ might or might not be deceptive. Thus for $k = \ell$, bounded deception is full deception. Goldberg, Deb, and Korb (1991) construct examples of boundedly deceptive problems by concatenating some number m

[6] Such partitions can also be considered type I deceptive (at least for some of the fully deceptive functions constructed in the literature) if I discount the global optimum from schema average fitness calculations, a modification suggested and justified later in this thesis.

[7] Whitley (1991) proves that D need not be a local optimum. But if it is not, then one of its immediate neighbors must be.

[8] In his 1981 dissertation (Bethke, 1981), Bethke constructed boundedly deceptive problems as examples of "...functions which will fool the genetic algorithm into exploring the wrong regions of the search space." He did not however identify the *order* of deception nor the idea of "boundedness" in general, as Goldberg, et. al. later did (Goldberg, Deb, & Korb, 1991).

[9] Deceptive partition here means that the winner of a partition is the schema containing D, (i.e., type II deception).

of fully deceptive subfunctions of length ℓ_s to get an $(m * \ell_s)$-bit function of order ℓ_s bounded deception[10]. Such functions have 2^m local optima, one of which is global[11].

The order k of bounded deception establishes a partial order over functions. I can safely say that a function of order k bounded deception is at least as difficult for a GA as a function of order $k' < k$ bounded deception, all other problem dimensions (e.g., noise, crosstalk) being roughly equal.

Other attempts to define, quantify, or order the degree of partial deception have resulted in unnecessarily weakening the requirements for deception. For example, Grefenstette (1993) describes an $\ell = 20$-bit function in which all partitions defined over the first ten bits lead toward the deceptive attractor (i.e., type II deception) and all partitions defined over the last ten bits lead toward the global optimum. "Despite this high level of Deception [*sic*]," this function was easily optimized by a GA with population size 200. However, as Goldberg points out with his analysis of a similarly constructed partially deceptive function (Goldberg, 1991b), we do not want to call such problems highly deceptive. Any function with low order partitions that lead toward the global optimum (i.e., any function with building blocks) is amenable to GA search. Adding additional misleading bits does not necessarily make the problem any more deceptive, let alone "arbitrarily more Deceptive..." (Grefenstette, 1993), because the number of low order building blocks remains undiminished. A function of low order k bounded deception is probably more difficult for a GA than Grefenstette's function with some arbitrarily

[10] It is easy to show that concatenating m fully deceptive subfunctions of order k yields a bounded, order-k deceptive problem. By "concatenating", I mean here that the resulting order mk-bit problem is any linear combination of the subfunctions; that is, the overall fitness of the "full" problem is some weighted sum of the subfunction fitnesses. Then any schema in the full problem must have a schema average fitness equal to the sum of the average fitnesses of schema in the subfunctions. That is, schema average fitnesses are linear combinations of "sub-schema" average fitnesses. Then any order $< k$ partition in the full (order mk) problem, must be deceptive, since all k loci must be chosen from "within" one or more deceptive subfunctions. In addition, many partitions of order $\geq k$ are also deceptive, since more than k loci can be chosen from within one or more of the m subfunctions. The only non-deceptive partitions are those of order $\geq k$ that include all k bits from one (or more) subfunctions. These results apply even for concatenations of varying length subfunctions (i.e., different orders k_i for each of the m subfunctions). As long as all m subfunctions are fully deceptive, and as long as the concatenation is some linear combination, then the concatenated function will be at least order k_{min} boundedly deceptive, where k_{min} is the order (length) of the smallest subfunction.

[11] Here again, note the correspondence to Kauffman's (1989) NK-landscapes: the loci (bit-wise) contributions to fitness in Goldberg, Deb, and Korb's (1991) order k boundedly deceptive problems are all dependent on exactly $k-1$ other loci (bits), with $k-1$ corresponding to Kauffman's K. The resulting function is of "intermediate" ruggedness (modality), somewhere between unimodal at $K = 0$ and maximum ruggedness at $K = N - 1 = \ell - 1$.

large number of misleading bits, even though the boundedly deceptive problem will have fewer deceptive partitions overall[12].

The above example points out the importance of both number and *order* of deceptive partitions. As discussed above, order k bounded deception only allows comparisons between functions with the same number of deceptive partitions, $\binom{\ell}{o}$, at each order $o < \min(k_1, k_2)$, where k_i are the orders of bounded deception for two functions. Goldberg, Deb, and Horn find another way to construct and order partially deceptive functions without losing essential misleadingness (Goldberg, Deb, & Horn, 1992; Deb, Horn, & Goldberg, 1993). They constrain a "bipolar deceptive function" to be a function of *folded unitation* $u_{fld}(s)$, which is simply the absolute difference between the number of ones and the number of zeros: $u_{fld}(s) = |u(s) - (\ell - u(s))| = |2\ u(s) - \ell|$. This leads to a symmetric function of unitation, $f_{bip}(u_{fld}(s))$. Enforcing full deception in the composite function $f_{bip} \circ u_{fld}$ leads to two globals (all-ones and all-zeros) in the unfolded binary space of full strings (i.e., Hamming space), and $\binom{\ell}{\ell/2}$ deceptive attractors (all points consisting of half ones and half zeros, for even ℓ). The symmetry of the function, induced by the folding, means that the single global optimum in the folded unitation space corresponds to two (complementary) global optima in Hamming space. The single deceptive attractor in the folded unitation space maps to many deceptive attractors in Hamming space.

Goldberg, Deb, and Horn show that the full deception enforced on the folded unitation function leads to partitions in the unfolded function in which the schemata containing the *most* deceptive attractors wins (in the sense of type II deception). This occurs in all partitions where the schemata containing the globals (the global schemata) can be *distinguished* from the schemata containing the deceptive attractors. Thus at order one, where the two schemata ...#1#... and ...#0#... compete, there is no possible distinction between the globals and the deceptive optima, and thus no preference between the schemata (i.e., they have equal fitness). But at order two, and above, the deceptive attractors can be distinguished, hence ...#01#... and ...#10#... beat ...#11#... and ...#00#... in all order two partitions.

Note that having two distinct globals precludes full deception and therefore requires some kind of partial deception at best (or worst!). By constraining the function to be of folded

[12] It is interesting to note that Grefenstette's (1993) function is not only unimodal by my definitions, but is also, like the long path problem, a single hill to the global optimum. All points in the search space lie on stepsize-1 paths to the global.

unitation, Goldberg, Deb, and Horn are able to make the overall function maximally deceptive by enforcing full deception in the folded function. They are then able to examine its implications in the unfolded space. The result is a function that has deceptive attractors located maximally far from *both* globals, and in which the deceptive schemata win in all partitions (type II), up to order ℓ, in which it is possible to distinguish globals from deceptives.

5.2 Maximal Bi-global Deception

I first generalize the work of Goldberg, Deb, and Horn on bipolar deceptive functions to the case of bi-global deceptive functions. I do not assume a function of unitation, folded or not, nor do I assume bipolarity (i.e., where the two globals are full complements of each other). I only assume two globals arbitrarily placed in the space. Rather than first choosing the deceptive attractor, as is usually done, I will instead try to *maximize deception* and see what deceptive attractor emerges. I use such terms as "maximal deception" loosely at first, and define them rigorously later.

Let the two arbitrarily chosen globals in an ℓ-bit problem be $G = \{g_1, g_2\}$, and let the distance between them be d_G bits. How does one choose a deceptive attractor D that maximizes deceptive partitions? An upper bound on the number of possibly deceptive partitions is the number of partitions in which we can *distinguish* D from the G. I shall call these simply *resolvable partitions*. We now try to place D so as to maximize the number of resolvable partitions at every order o.

First note that there are $(\ell - d_G)$ bits in which the two globals agree. Intuitively, the deceptive attractor should disagree with both globals in these bit positions to maximize the number of resolvable partitions. For each of the $(\ell - d_G)$ bits of agreement between the globals, setting the corresponding bit position in D to be the complement of the globals' bit setting gives one more bit position in which deceptive schemata can be distinguished from global schemata. Thus giving D the complement of the $(\ell - d_G)$ global bit settings only increases the number of resolvable partitions at every order. I therefore assume such complementary bit settings for the $(\ell - d_G)$ bit positions of D (in which $g_1$and g_2 agree) and next consider only how to set the d_G bit positions in which the globals disagree.

Let d_1^D be the Hamming distance from D to g_1. Then $d_G - d_1^D$ is the distance from D to g_2. At any order o schema partition there are $\binom{d_G}{o}$ partitions[13]. For exactly $\binom{d_1^D}{o}$ of these partitions, we cannot distinguish D from g_2, since all o defined bits are chosen from the d_1^D bits that distinguish D from g_1 and hence are positions at which D and g_2 are in agreement. Similarly, there are exactly $\binom{d_G-d_1^D}{o}$ partitions in which we cannot distinguish D from g_1. So the total number of resolvable partitions, $N_{rp}(o)$ at order o is

$$N_{rp}(o) = \binom{d_G}{o} - \left[\binom{d_1^D}{o} + \binom{d_G-d_1^D}{o}\right]. \tag{5.1}$$

To maximize N_{rp} we must minimize the sum of the two binomial coefficients in the square brackets above. Since $\binom{d_1^D}{o}$ is a strictly increasing function of d_1^D, and $\binom{d_G-d_1^D}{o}$ is strictly decreasing in d_1^D, the minimum of their sum will occur when $d_1^D = d_G - d_1^D \Rightarrow d_1^D = d_G/2$, for even d_G, and $d_1^D = \lfloor d_G/2 \rfloor$ or $\lceil d_G/2 \rceil$, for odd d_G. That is, I will have the maximum possible number of resolvable partitions when the deceptive attractor lies equally distant from g_1 and g_2, which means it is *maximally minimally distant* from G. Here I define *minimal distance* from a point D to a set G of k points as the minimum of the distances from D to each point $g_i \in G$, $1 \le i \le k$. The *maximally minimally distant point* from a set G is simply the point[14] in the space with the greatest minimal distance to G.

Note that we have maximized the resolvable partitions in the sense that we have the maximum number of resolvable partitions at every order. Assume that we can make all resolvable partitions deceptive, in at least one of the three senses defined earlier (I show that we can in the next section). I conjecture that a function f_1 with more deceptive partitions at every order $o < \ell$ than another function f_2 is more misleading to a GA, all other problem dimensions being roughly equal. Thus I am suggesting another relation that induces a partial ordering on the space of partially deceptive functions, just as Goldberg's bounded deception does. And just as bounded deception has full deception at the extreme, the partial ordering has a maximum for a

[13] I assume throughout this thesis that $\binom{n}{m} = 0$ for $m > n$.

[14] In the bi-global case there will always be $\binom{d_G}{d_G/2}$ such points for even d_G, and $2 * \binom{d_G}{\lfloor d_G/2 \rfloor}$ such points for odd d_G. For the purposes of this thesis, I assume that when the maximally minimal distance criterion yields a *set* of points, then we arbitrarily choose a single deceptive attractor from that set. We could also choose the entire set or some subset to be the *set* of attractors. The deception analyses and results would not change, and we would have multiple deceptive optima with increased "carrying capacity" for niched GAs (Goldberg, Deb, & Horn, 1992). But such considerations are beyond the scope of this thesis.

given placement of globals. A *maximally* deceptive function has no fewer deceptive partitions at each order than any other deceptive function possible given the set of globals. When the global set consists of only one global, the maximally deceptive function is a fully deceptive function. Otherwise, it is partially deceptive[15].

5.3 A Function to Meet the Bi-global Deceptive Conditions

In the section above I showed that to maximize the number of resolvable partitions at every order in a bi-global problem, we should choose the deceptive attractor to be the point that is maximally minimally distant from the two globals. Such a result is meaningless if we cannot define a function that is actually misleading to a GA in those resolvable partitions. In other words, is it possible to make all the resolvable partitions deceptive? Or does this lead to too many constraints? In this section I construct a function that satisfies the deceptive conditions in all of the resolvable partitions.

5.3.1 Construction

The use of maximal minimal distance to G in choosing the deceptive attractor suggests the use of such a function as the fitness function itself. That is, let the fitness of a point/string s be its Hamming distance to the nearest global. Let us call such a minimum distance function simply $f_{md}(s)$:

$$f_{md}(s) = \min_{\forall g \in G} H(s, g), \tag{5.2}$$

where $H(s, g)$ is the Hamming distance from a point s to the global g. At the globals themselves, of course, we substitute some globally optimal value f_{max} to get a function of minimum distance "plus globals":

$$f_{mdG}(s) = \begin{cases} f_{max} & \text{if } s \in G \\ f_{md}(s) & \text{otherwise,} \end{cases} \tag{5.3}$$

[15] I note that the bipolar deceptive function (Goldberg, Deb, & Horn, 1992; Deb, Horn, & Goldberg, 1993) is also a maximally deceptive function by my definition above. And a maximally deceptive function for two ($k = 2$) bipolar globals, with the further restriction to symmetric or *folded* unitation, is *bipolar deceptive* according to (Goldberg, Deb, & Horn, 1992) and (Deb, Horn, & Goldberg, 1993).

The min-dist function, with globals, $f_{mdG}(s)$, has some very interesting properties. Like trap and unitation functions, it is easy to define, is readily visualized, has mostly linear gradients, and is amenable to schema analysis. Below I present several additional observations of $f_{mdG}(s)$.

5.3.1.1 Observation: local optimality

The max-min-dist deceptive attractor is clearly a local optimum of $f_{mdG}(s)$ and indeed must be the global optimum of $f_{md}(s)$. This is true by definition of D as the point of *maximal* min-dist. To guarantee that f_{max} is globally optimal, we could set $f_{max} = f_{md}(D) + 1$. Or, to avoid having to find D, we can simply let $f_{max} = \ell + 1$, which is guaranteed to be global since the greatest Hamming distance in an ℓ-bit problem is ℓ.

5.3.1.2 Observation: generalizing trap functions

The function $f_{mdG}(s)$ reduces to a simple trap function when G contains a single global optimum. When G contains two global optima that are bit-wise complements of each other, $f_{mdG}(s)$ reduces to a *bipolar trap function.*

Ackley (1987a, 1987b) introduces trap functions with a single global optimum. Goldberg, Deb, and Clark (1992) construct and test fully deceptive trap functions. Deb and Goldberg (1993) loosen Ackley's (1987a, 1987b) definitions to create generalized (single global) trap functions and derive sufficient conditions (on their defined parameters of the function class) for full deception. Finally, Deb, Horn, and Goldberg (1993) generalize Deb and Goldberg's definitions to allow two complementary global optima, providing sufficient conditions for *bipolar deception* in trap functions with two such *bipolar* global optima.

If we set $f_{max} = f_{md}(D) + 1$ then the function

$$f_{mdG}^{trap}(s) = f_{mdG}(s) - 1 \tag{5.4}$$

might be called a *generalized, k-global deceptive trap function.* The function is zero-valued at all points adjacent to the global optima, as in many examples of trap functions.

With one global optimum, this function is an instance of a fully deceptive trap function. For example, with $\ell = 4$ and $G = \{1111\}$, we obtain exactly the fully deceptive trap subfunction used by Goldberg, Deb, and Clark (1992) to construct their problem f_4, as shown in Figure 5.1, top. Here $u(s)$ is the *unitation* (i.e., number of ones) of the given string s and $f(u(s)) = f^{trap}_{mdG}(s)$.

We can prove that $f^{trap}_{mdG}(s)$ is fully deceptive when G contains only one global optimum, since it satisfies Deb and Goldberg's (1993) condition for full deception:

$$\frac{a}{b} \geq \frac{2 - 1/(\ell - z)}{2 - 1/z}, \tag{5.5}$$

which is Equation 16 from (Deb & Goldberg, 1993). Here a is the (fitness) value of the deceptive optimum, b is the value of the global optimum, and z is the Hamming distance from the global optimum to the (unique) point at which the function is zero valued. In the case of $f^{trap}_{mdG}(s)$ with one global optimum, $a = \ell - 1$, $b = \ell$, and $z = 1$. Substituting these values into Equation 5.5 yields $\ell \geq 3$. So for problems of length three bits or more, $f^{trap}_{mdG}(s)$ is a fully deceptive trap function of one global.

Similarly, when G contains only two complementary strings, such as the all-zeroes and all-ones strings, $f^{trap}_{mdG}(s)$ reduces to a *bipolar* deceptive trap function (Goldberg, Deb, & Horn, 1992; Deb, Horn, & Goldberg, 1993). For example, with $G = \{000000, 111111\}$ in a six-bit problem, $f^{trap}_{mdG}(s)$, shown in Figure 5.1, is almost identical to the bipolar trap function used by Goldberg, Deb, and Horn (1992).

We can prove that $f^{trap}_{mdG}(s)$ is bipolar deceptive in the case of two bipolar globals. Assuming even ℓ, we find that $a = f^{trap}_{mdG}(D) = \ell/2$, $b = a+1 = \ell/2+1$, and $z = 1$ again. Substituting these into Deb, Horn, and Goldberg's (1993) condition for bipolar deception in a trap function[16]:

$$\frac{a}{b} > \frac{2 - 1/(\frac{\ell}{2} - z)}{2 - 1/z}, \tag{5.6}$$

Equation 5.6 simplifies to $\ell > 1 + \sqrt{5}$. Thus the bipolar deception condition is met by $f^{trap}_{mdG}(s)$ for problems of length four or more.

5.3.1.3 Observation: separability of isolation and misleadingness

A final observation is that $f_{mdG}(s)$ illustrates the separability of *misleadingness* and *isolation* as dimensions of problem difficulty. One can view the function $f_{md}(s)$ as providing the essential

[16] See their Equation 11, but note that they define ℓ as half the problem length, rather than the full problem length.

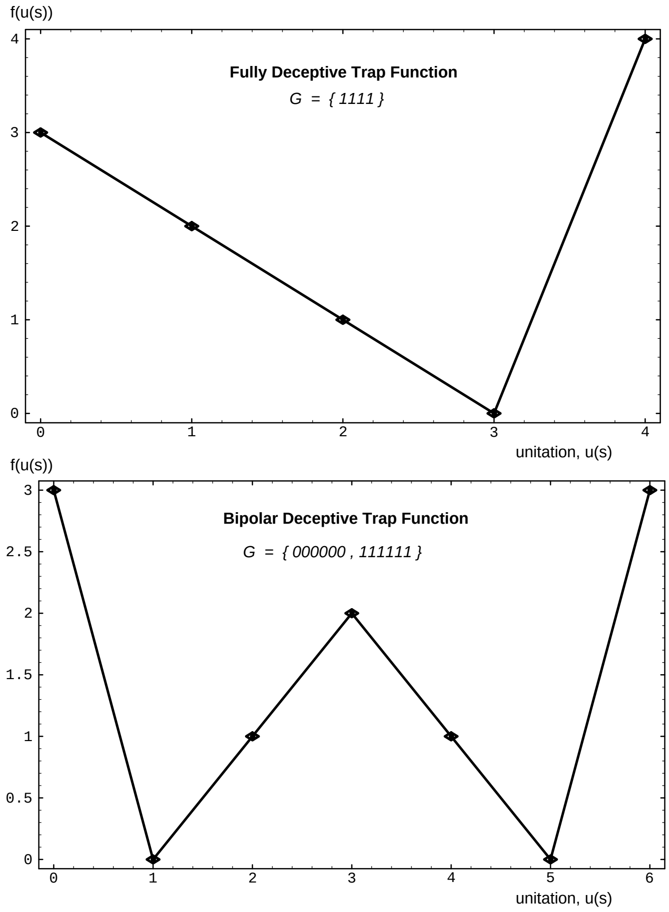

Figure 5.1: *Top:* With a single global optimum (e.g., $G = \{1111\}$), $f^{trap}_{mdG}(s)$ reduces to a fully deceptive trap function. Here $u(s)$ is the unitation (number of ones) in string s. *Bottom:* With two complementary (bipolar) global optima (e.g., $G = \{000000, 111111\}$), $f^{trap}_{mdG}(s)$ reduces to a bipolar deceptive trap function.

misleadingness of $f_{mdG}(s)$, while we could define a function

$$f_G(s) = \begin{cases} f_{max} & \text{if } s \in G \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

that provides the isolation of the globals. The function $f_{mdG}(s) = f_{md}(s) + f_G(s)$ is then literally the additive combination of misleadingness and solution isolation.

5.3.2 Schema Analysis

Next I perform a modified schema analysis of $f_{mdG}(s)$ looking for deceptive partitions. My modification to regular schema analysis is this: we ignore the global optima. That is, we assume their fitness is zero by simply analyzing $f_{md}(s)$ rather than $f_{mdG}(s)$. Adding the globally optimal values f_{max} to the schema fitness calculations would complicate them. The only point in doing so would be to find constraints on f_{max} in order to meet deception conditions on schema average fitnesses. We are not very interested in such constraints, since it really doesn't matter how large f_{max} is, as long as it is globally optimal. The value of the global optima do not enter into the GA's schema processing until a global is found, at which time the search is over and we are no longer interested in GA schema processing. We are modeling GA performance in the *search* for global optima; that is, during the generations preceding the discovery of a global.

We begin the schema analysis by assuming two globals g_1 and g_2. As before, we ignore the bit positions in which g_1 and g_2 agree, since considering the fitness contributions of these bits does not change the ranking of schemata within a competition partition. To see why this is so, remember that the fitness of a schema $\hat{s}$ of $f_{md}(s)$ is the average distance to the nearest global over all strings instantiating $\hat{s}$. Thus the bit positions at which g_1 and g_2 agree always add the same amount to the average fitness of each schema within a particular partition, regardless of the schema.

We assume ℓ bit positions in which g_1 and g_2 differ. We now calculate exact schema average fitnesses for any schema in an order o partition. Let $\hat{s}$ be a schema in the given order o partition, and let $d_1^{\hat{s}}$ be the distance from $\hat{s}$ to global g_1, making $(o - d_1^{\hat{s}})$ the distance from $\hat{s}$ to g_2. (I define the distance from a schema $\hat{s}$ to a point, such as global g_1, to be the number of bit positions, over the o defined bit positions in $\hat{s}$, in which $\hat{s}$ and g_1 differ. Note that $0 \leq d_1^{\hat{s}} \leq o$.)

To calculate the average fitness of $\hat{s}$ in $f_{md}(s)$, we add up the fitnesses of all strings contained in $\hat{s}$ and divide by the number of such strings:

$$\bar{f}_{md}(\hat{s}) = \frac{\sum_{s \in \hat{s}} f_{md}(s)}{2^{\ell - o}}. \tag{5.8}$$

We concentrate now on the sum in the numerator, since this will order the average fitnesses of schemata in the partition. The $2^{\ell-o}$ strings in the summation can be divided into two groups, those that are closest to g_1 and those that are closest to g_2. Let d be the number of bit positions from the $\ell - o$ undefined bit positions at which a particular string s disagrees with g_1. That is, the total distance from s to g_1 is $d + d_1^{\hat{s}}$. When this total distance is less than half of ℓ, then s is closer to g_1 than to g_2, and the fitness of s is its distance to g_1: $d + d_1^{\hat{s}}$. There are exactly $\binom{\ell-o}{d}$ strings in $\hat{s}$ that are d bits different from g_1 in the $\ell - o$ undefined bits of the partition. Furthermore, g_1 will be the closest global to s for $d = 0$ up to $d + d_1^{\hat{s}} = \lfloor \ell/2 \rfloor \Rightarrow d \leq \lfloor \ell/2 \rfloor - d_1^{\hat{s}}$. Thus the total contribution of strings near g_1 to the average fitness of $\hat{s}$ is

$$\sum_{d=0}^{\lfloor \ell/2 \rfloor - d_1^{\hat{s}}} (d + d_1^{\hat{s}}) \binom{\ell - o}{d}. \tag{5.9}$$

We calculate a similar sum for the remaining points, which are all closer to g_2. These points have fitness $(\ell - (d + d_1^{\hat{s}}))$, which is their distance to g_2, and they occur when $(\lfloor \ell/2 \rfloor - d_1^{\hat{s}} + 1) \leq d \leq (\ell - o)$. Adding these two sums together, we get the exact schema average fitness of any schema $\hat{s}$ in an order o partition of $f_{md}(s)$:

$$2^{\ell-o} \bar{f}_{md}(\hat{s}) = \sum_{d=0}^{\lfloor l/2 \rfloor - d_1^{\hat{s}}} (d + d_1^{\hat{s}}) \binom{\ell-o}{d} + \sum_{d=\lfloor \ell/2 \rfloor - d_1^{\hat{s}} + 1}^{\ell - o} (\ell - d - d_1^{\hat{s}}) \binom{\ell-o}{d}. \tag{5.10}$$

I can show that this sum increases as $d_1^{\hat{s}}$ approaches $d_1^{\hat{s}} = \lfloor o/2 \rfloor$ from $d_1^{\hat{s}} = 0$ and as it approaches $d_1^{\hat{s}} = \lceil o/2 \rceil$ from $d_1^{\hat{s}} = o$. Thus the more maximally minimally distant a *schema* is from the schemata containing the globals, the higher its average fitness. In particular, the maximum of $\bar{f}_{md}(\hat{s})$ occurs at $d_1^{\hat{s}} = \lceil o/2 \rceil$ and $d_1^{\hat{s}} = \lfloor o/2 \rfloor$.

This last result has several important implications for $f_{md}(s)$. First, it means that in all the resolvable partitions, the schema containing the deceptive attractor D has greater average fitness than the schemata containing the globals. Since $f_{md}(s)$ maximizes the number of resolvable partitions at every order, it also therefore maximizes the number of partitions where the global schemata lose to the deceptive schema (type III deceptive partitions). Second, in all but the order one and order ℓ partitions, the global schemata lose to all other schemata[17]. Thus $f_{md}(s)$ is maximally deceptive according to type I partition deception.

Third, the above result implies that the winning schema (i.e., superior to all others) in every partition is the schema that is maximally minimally different from the global schemata. If the winning schema contains the deceptive attractor D, the partition is type II deceptive. How many type II deceptive partitions can a deceptive attractor possibly have, at a given order o? The analysis is similar to the previous analysis. Let d_1^D be the distance from a deceptive attractor D to global optimum g_1 with ℓ bits of separation between g_1 and g_2. Assume an even partition order o. We can choose half the order o bit positions, from among the d_1^D bit positions in which D and g_1 differ, in exactly $\binom{d_1^D}{o/2}$ ways. We can choose the other $o/2$ bits, from among the $\ell - d_1^D$ bit positions that differ from g_2, in exactly $\binom{\ell-d_1^D}{o/2}$ ways. So there are $\binom{d_1^D}{o/2}\binom{\ell-d_1^D}{o/2}$ partitions of even order o in which the deceptive schema wins. Similarly, for odd-order o partitions this number is $\binom{d_1^D}{\lfloor o/2\rfloor}\binom{\ell-d_1^D}{\lceil o/2\rceil} + \binom{d_1^D}{\lceil o/2\rceil}\binom{\ell-d_1^D}{\lfloor o/2\rfloor}$. It is clear that the number of type II deceptive partitions is maximized when $d_1^D = \ell - d_1^D$, $\Rightarrow$ $d_1^D = \ell/2$ for even ℓ (and either of $\lfloor \ell/2\rfloor$ or $\lceil \ell/2\rceil$, for odd ℓ). Thus the choice of D as the deceptive attractor maximizes the number of type II deceptive partitions at every order. The function $f_{md}(s)$ is therefore maximally deceptive according to all three types of partition deception we defined.

5.4 Generalization to k Globals

I would like to immediately generalize these results to the case of k globals placed arbitrarily. That is, given any set G of k globals in an ℓ-bit problem, we should place the deceptive attractor(s) at those points maximally minimally distant from the entire set G in order to maximally mislead the GA. Unfortunately the analysis used above becomes much more complicated when applied to k globals. With more than two globals We lose the symmetry of bit position agreement/disagreement (where agreement with g_1 at a position means disagreement with g_2 at that position, and where being d_1^D bits away from g_1 means being $\ell - d_1^D$ bits away from g_2).

[17] At order one (for the ℓ positions in which the globals differ) both competing schemata are global schemata.

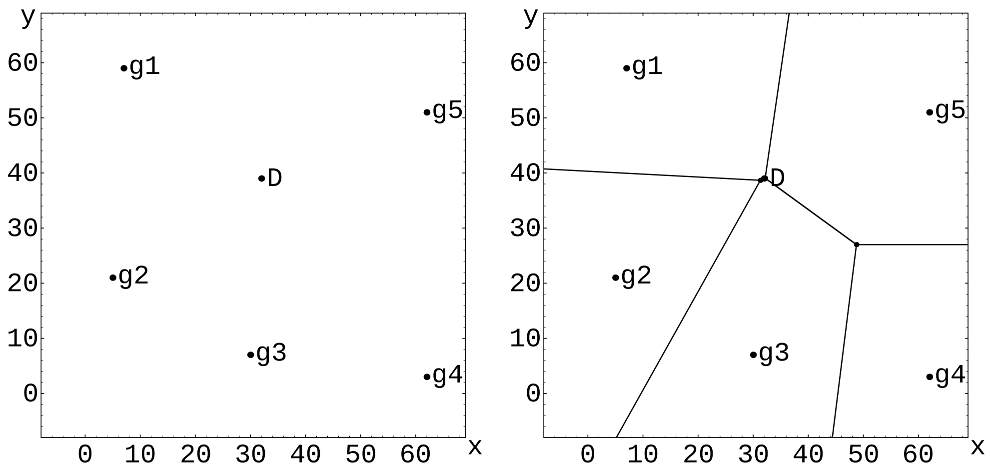

Figure 5.2: *Left:* A three dimensional visualization of a two dimensional problem with $k = 5$ globals at $G = \{(7,59),(5,21),(30,7),(62,3),(62,51)\}$, and a maximally minimally distant point at $D = (32,39)$. *Right:* The Voronoi diagram of the set G.

While I am continuing to explore the generalization to k globals, I can present some intriguing initial results.

5.4.1 Visualization in Three Dimensions

It is instructive to visualize the spatial relationship of the deceptive attractor D (the max-min-dist point) to the set G of k globals. Let us therefore return to the use of two dimensional analogues of the binary search spaces and functions. In Figure 5.2, I show $k = 5$ global optima located at grid positions $G = \{(7,59),(5,21),(30,7),(62,3),(62,51)\}$. Here we assume two integer-valued decision variables $\{x, y\}$ each taking values in the range (0..63). The maximally minimally distant point from G is $D = (32,39)$, as shown on the left of Figure 5.2. Interestingly, this point lies at the intersection of Voronoi lines in the Voronoi diagram of G, shown on the right of Figure 5.2. Since the lines in a Voronoi diagram divide the space into neighborhoods of elements of G and their nearest neighbors (Conway & Sloane, 1993), and thus lie on points equidistant from the nearest two globals, D must lie at an intersection of at least three Voronoi lines.

To better illustrate the relationship between the Voronoi diagram and the maximum minimum distance criterion, let us redefine f_{md} to be a function of integers x and y rather than of binary strings:

$$f_{md}(x,y) = \min_{\forall (x_g,y_g)\in G} \sqrt{(x_g - x)^2 + (y_g - y)^2}. \tag{5.11}$$

Noting that the maximum value (over the integer range (0..63)) is $f_{md}(D) \approx 32.0156$, we choose f_{max} to be 34:

$$f_{mdG}(x,y) = \begin{cases} 34 & \text{if } (x,y) \in G \\ f_{md}(x,y) & \text{otherwise.} \end{cases} \tag{5.12}$$

I plot $f_{mdG}(x,y)$, at the top of Figure 5.3, for the five globals. Note how the ridges correspond to the lines of the Voronoi diagram, and how each local optimum (except the globals) occurs at intersections of such lines. By definition of $f_{md}(x,y)$ and Voronoi diagrams, this must be the case. Thus the min-dist function $f_{md}(x,y)$ leads to an interesting, rugged, multimodal landscape with apparent basins of attraction for hillclimbers. To see that there are indeed basins of attraction for the local optima, I plot the local gradients of $f_{md}(x,y)$ at the bottom of Figure 5.3. As expected, these gradients point away from the nearest globals and towards the nearest ridge. We would expect a simple hillclimber to follow the local gradients up to the nearest ridge and thence to a local optimum. Finally note that the max-min-point D seems to have the largest basin of attraction (of all optima shown) for a simple hillclimber.

5.4.2 Schema Analysis

Again, to be of interest in the study of GA difficulty and multimodality, the min-dist function $f_{mdG}(s)$ with k globals must be misleading to the GA's crossover operator as well as to mutation. The schema analysis of $f_{mdG}(s)$ is complicated by the interactions of the multiple globals. I can however make a quick observation that indicates we are headed in the right direction (in order to make the GA head in the wrong direction!).

Assume any order o resolvable partition in an ℓ-bit problem with k globals. Let D be the choice of deceptive attractor. Let $\hat{s}_D$ be the schema containing the deceptive attractor and let $\hat{s}_i$ be the schema containing[18] global g_i, $1 \leq i \leq k$. Let us define $\vec{d^D} = \{d_1^D, d_2^D, ..., d_k^D\}$ to be the vector of distances from the deceptive schema $\hat{s}_D$ to each of the global schemata $\hat{s}_i$. Thus d_i^D, where $(0 \leq d_i^D \leq o)$, is the Hamming distance from $\hat{s}_D$ to $\hat{s}_i$, defined only over the o fixed bit positions. We also define similar distance vectors for each of the global schemata

[18] Some schemata may contain more than one global. This does not affect the analysis.

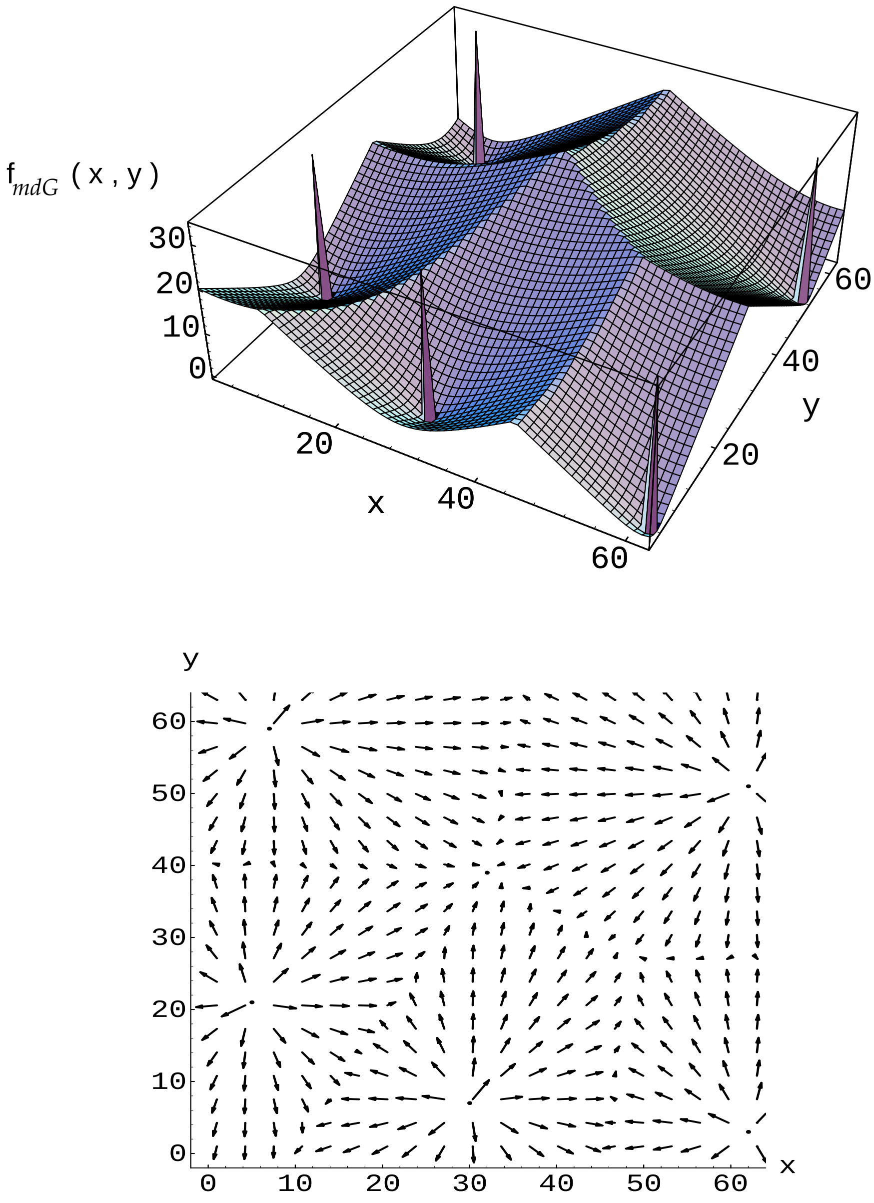

Figure 5.3: *Top:* A surface plot of $f_{mdG}(x, y)$. *Bottom:* This function is difficult for a hillclimber because all gradients lead away from the nearest global and, eventually, to a local optimum, as shown in this vector plot of $f_{mdG}(x, y)$.

$\hat{s}_i$: $\vec{d^i} = \{d_1^i, d_2^i, ..., d_k^i\}$, for $(i \in 1..k)$ where d_j^i is the Hamming distance $(0 \leq d_j^i \leq o)$ between $\hat{s}_i$ and $\hat{s}_j$. Finally, for each possible setting s of the $(\ell - o)$ bits undefined in this partition, we associate another vector of distances to the globals: $\vec{d_s} = \{d_1^s, d_2^s, ..., d_k^s\}$, where d_i^s is the number of bits $(0 \leq d_i^s \leq \ell - o)$, in which s differs from global g_i over the $(\ell - o)$ bits undefined by the partition (but defined by s).

To calculate the average fitness of a global schema $\hat{s}_i$, we generate the $2^{\ell-o}$ substrings s in the hyperplane defined by $\hat{s}_i$. For each s we calculate its distance vector $\vec{d_s}$ and add to it the distance vector $\vec{d^i}$ for $\hat{s}_i$. We then take the minimum component of the vector resulting from this summation as the distance to the nearest global from s. Summing the minimum components of these summed distance vectors over all $s \in \hat{s}_i$ and dividing by $2^{\ell-o}$ gives the average schema fitness[19] $\bar{f}_{md}(\hat{s}_i)$. We calculate $\bar{f}_{md}(\hat{s}_D)$ similarly, using $\vec{d_s}$ and $\vec{d^D}$.

Now if $\vec{d^D} \geq \vec{d^i}$ in all components (that is $d_j^D \geq d_j^i$, $\forall j \in 1..k$), for a particular global g_i, then clearly $\bar{f}_{md}(\hat{s}_D) > \bar{f}_{md}(\hat{s}_i)$, and the deceptive schema will beat the ith global schema. If $\vec{d^D}$ is superior (i.e., greater in all k components) to all k of the $\vec{d^i}$, then the deceptive schema will be superior to all the global schemata in that partition. Such a partition would thus be type III deceptive. To maximize the number of such partitions, we should try to increase the number of partitions in which the deceptive schema is as different as possible from all of the global schemata. Note that $\vec{d^D} \geq \vec{d^i}$ is only a *sufficient* condition for type III deception in a partition. Thus the number of partitions satisfying $\vec{d^D} \geq \vec{d^i}$ is only a lower bound on the number of type III deceptive partitions. However, this sufficient condition is met more often (i.e., at more partitions at every order) by the max-min-dist point D than by any other string. This in turn suggests that the max-min-dist point D will turn out to be the most attractive local optimum for GA crossover on the min-dist function with k globals.

5.4.3 Simulation Results

To verify that $f_{mdG}(s)$ is misleading to a GA, in the sense that the GA will tend to converge to a deceptive local optimum rather than a member of the global set G, I perform an experiment on a constructed function. I employ the methodology presented in (Goldberg, Deb, & Clark,

[19] Note that we again use $f_{md}(s)$ for schema fitness calculations, rather than $f_{mdG}(s)$, effectively ignoring the globals.

1992) and used again in (Goldberg, Deb, & Horn, 1992) to construct an order-ℓ_s boundedly deceptive problem.

5.4.3.1 Test problem f_{5md5G}

I first define a (partially) deceptive subfunction f_s of length ℓ_s bits. I then concatenate m identical copies of these subfunctions to form a length $\ell = m\ell_s$ problem f that is a linear combination of the m subfunctions. That is, the value of $f(s)$ is simply $\sum_{i=1}^{m} f_s(s_i)$, where s_i is the i^{th} substring of ℓ_s bits in string s. Thus the first subfunction is defined over the first ℓ_s bits, the second subfunction is defined over the next ℓ_s bits, and so on. Since each of the m subfunctions is defined over groups of adjacent bits, the problem assumes a *tight* ordering of loci in the encoding. Because the problem is a linear combination of the (partially) deceptive subfunctions, deceptive schema partitions are *additive* across subfunction boundaries. That is, any deceptive partition from one copy of f_s can be combined with any deceptive partitions from any other copies of f_s to form a higher order deceptive partition of the larger function f. This means that the essential misleadingness of f_s is preserved in f. However, the additive combination of the subfunctions also means that if the GA can solve *each* of the order-ℓ_s partitions corresponding to the f_s, then it can simply recombine those solutions (instances of the k subfunction global optima) through crossover to find one of the k^m global optima of f.

The subfunction $f_{md5G}(s_i)$ is a $k = 5$ global, $\ell_s = 10$-bit instance of the trap function version of $f_{mdG}^{trap}(s)$ given in Equation 5.4 (with one change: here I set f_{max} to $\ell_s = 10$, rather than to $f_{md5G}(D) + 1$). I choose the five globals arbitrarily:

$$G = \{0000000000, 0010110001, 1101110011, 0101010010, 1000101111\}.$$

This fully defines the $2^{10} = 1024$ values of $f_{md5G}(s_i)$:

$$f_{md5G}(s_i) = \begin{cases} 10 & \text{if } s_i \in \text{G} \\ f_{md}(s_i) - 1 & \text{otherwise.} \end{cases} \tag{5.13}$$

I concatenate $m = 5$ copies of $f_{md5G}(s_i)$ to create an $\ell = 50$-bit problem $f_{5md5G}(s)$. Since $f_{5md5G}(s) = \sum_{i=1}^{5} f_{md5G}(s_i)$, where s_i is the i^{th} group of 10 bits in s, $f_{5md5G}(s)$ has $k^m = 5^5 = 3125$ global optima with fitness $f_{5md5G}^{max} = 5 * f_{md5G}(g \in G) = 5 * 10 = 50$. Enumerating the

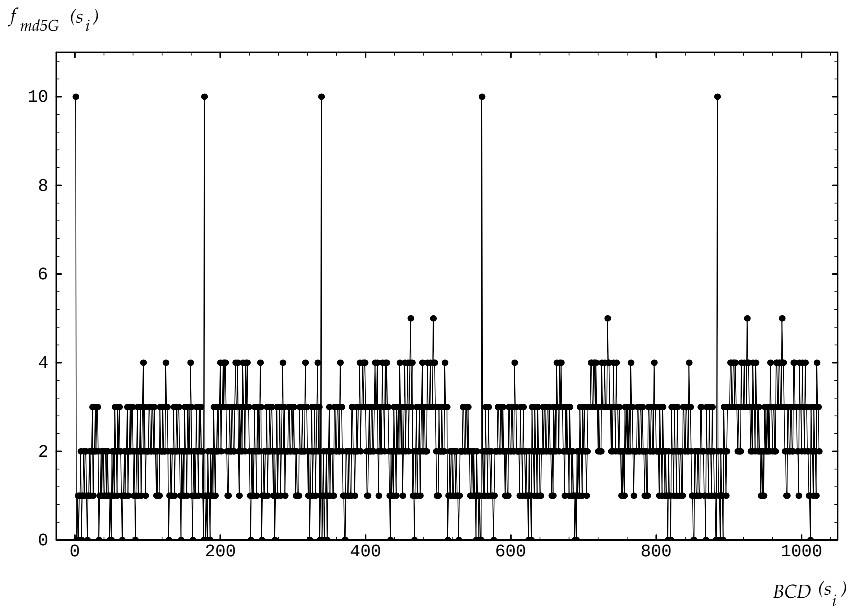

Figure 5.4: Two dimensional visualization of the 10-bit subfunction $f_{md5G}(s_i)$. The scatterplot places all 1024 possible strings s_i at coordinates $(BCD(s_i), f_{md5G}(s_i))$, where $BCD(s_i)$ is the *binary coded decimal* integer corresponding to substring s_i. This plot clearly shows the five global and five deceptive optima, although their "spacing" along the x-axis is not meaningful.

1024 values of $f_{md5G}(s_i)$, we find five deceptive local optima:

$$D = \{0111001101, 0111101100, 1011011100, 1110011100, 1111001100\},$$

each with fitness $f_{md5G}(d \in D) = 6 - 1 = 5$ since they are each exactly six bits different from the most similar global optimum.

Figure 5.4 is a two dimensional visualization of the subfunction $f_{md5G}(s_i)$. I plot all 1024 points in a scatterplot of $f_{md5G}(s_i)$ versus $BCD(s_i)$, where $BCD(s_i)$ is the *binary coded decimal* integer interpretation of the substring s_i. Thus BCD returns an integer from 0 to 1023. In Figure 5.4, one can clearly see the five global and five deceptive optima. Note that BCD is not used in our subfunction, so the spacing of points along the horizontal axis in Figure 5.4 is not meaningful.

The full function f_{5md5G} has $5^5 = 3125$ deceptive local optima of fitness $5 * 5 = 25$ (where all five copies of f_{md5G} are converged to members of D). It also has $\binom{5}{1} * 5 * 5^4 = 15,625$ local optima of fitness $10+4*5 = 30$ (where four copies of $f_{md5G}(s_i)$ have converged to members of D, and one has converged to a global optimum). In general $f_{5md5G}(s)$ has $\binom{5}{i}5^i5^{5-i}$ local optima, with fitness $10i + (5-i)5$, in which exactly i of the subfunctions are converged to members of G and the rest are converged to members of D. Thus $f_{5md5G}(s)$ has a total of 100,000 local optima, of which 3125 are global. Not only could we call $f_{5md5G}(s)$ *massively multimodal* (Goldberg, Horn, & Deb, 1992), but we could introduce the term *massively multiglobal* as well, since the number of global optima in functions like $f_{5md5G}(s)$ grows as k^m. Yet despite the large number of global optima, $f_{5md5G}(s)$ apparently is not as easy for the GA to solve as is a uniglobal function like $f_{mm,easy}(s)$.

5.4.3.2 Population sizing predictions of GA performance on f_{5md5G}

I run the same simple GA on f_{5md5G} as was described earlier in its application to $f_{mm,easy}$ and f_{lp} (Table 4.1): a generational GA with deterministic binary tournament selection, single point crossover with $p_c = 1.0$, and no mutation ($p_m = 0$). Here I run the GA at several different population sizes, using 40 different trials (different random initial populations) at each population size. For each trial I run the GA to convergence[20], and then measure the number of subfunctions optimized (i.e., number of subfunction global optima in an individual). Dividing the number of subfunctions optimized (zero to five) by five normalizes my convergence measure (zero to one).

I plot this convergence measure in Figure 5.5 for population sizes $N = 50$ to 1200, sampling every fifty increments in N (e.g., $N = 50, 100, 150, ..., 1200$). The plotted points track the mean convergence of each set of 40 trials, while the error bars extend one standard deviation above and one standard deviation below each plotted mean value[21]. Figure 5.5 illustrates the difficulty of the problem for the GA with inadequate population size. A brief examination of some of the final populations reveals that the GA usually converges to deceptive optima (members of D) when it fails to converge to a global optimum in a particular subfunction. With sufficient population sizes, however, the GA can overcome the partial deception of the $f_{md5G}(s_i)$ and reliably find a global optimum of $f_{5md5G}(s)$.

[20] My convergence criterion was uniform fitness in the current population (i.e., minimum fitness equals maximum fitness in the population).

[21] The numerical values of the means and standard deviations for Figure 5.5 are given in Figure A.1 of the appendix.

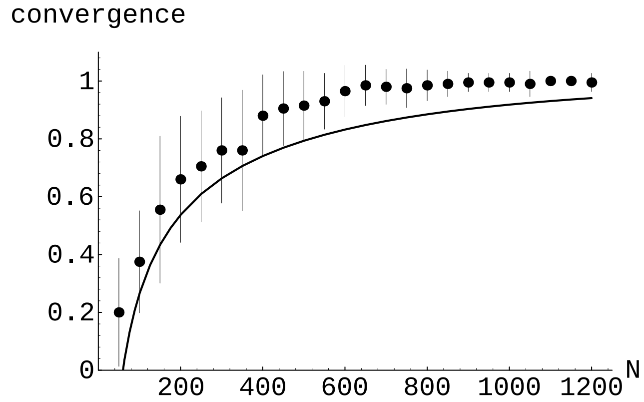

Figure 5.5: Predicted and actual GA performance on a boundedly, partially deceptive problem $f_{5md5G}(s)$. The plotted points are the average convergence values over 40 trials. Error bars extend one standard deviation above the mean, and one below. The solid line is the lower bound on expectation given by Goldberg, Deb, and Clark's (1992) population sizing equation. N is population size.

Clearly population sizing is critical to overcoming this type of bounded, maximal partial deception. I turn briefly to (Goldberg, Deb, & Clark, 1992) for some analytical guidance. The population sizing equations developed by Goldberg, Deb, and Clark (1992) can provide a lower bound on the expected performance of a simple GA on problems of bounded difficulty. Their framework is applied successfully to fully deceptive subfunctions in (Goldberg, Deb, & Clark, 1992), and to bipolar deceptive subfunctions in (Goldberg, Deb, & Horn, 1992). I apply it to the k-global, maximal partial deceptive subfunctions. The reader is referred to (Goldberg, Deb, & Clark, 1992) for guidance on applying the population sizing equations. Here I give only the problem specific parameters necessary to the equation.

One can extract from (Goldberg, Deb, & Clark, 1992) the expected convergence (normalized) on a problem of bounded deception:

$$E[conv(N)] = 1 - \frac{\exp\left[g(N, d, \ell_s, \sigma_M^2)/2\right]}{\sqrt{2\pi g(N, d, \ell_s, \sigma_M^2)}}, \tag{5.14}$$

where

$$g(N, d, \ell_s, \sigma_M^2) = \frac{Nd^2}{2^{(\ell_s+1)}\sigma_M^2}, \tag{5.15}$$

N is population size, ℓ_s is the length of the subfunction, and d is the "signal" of the subfunction's global optima to be detected among the "noise" (variance) σ_M^2 of the entire function. For the function $f_{md5G}(s_i)$, $\ell_s = 10$, and the signal d is simply the difference between the global fitness and the fitness of the nearest competitor (the deceptive optima): $d = 10 - 5 = 5$. Finally, $\sigma_M^2 = (m-1)\sigma_m^2$, where m is the number of subfunctions and σ_m^2 is the variance in a single subfunction $f_{md5G}(s_i)$. For $f_{5md5G}(s)$, $m = 5$. One can approximate σ_m^2 by sampling, but with only 1024 values in $f_{md5G}(s_i)$ I calculate it exactly as $\sigma_m^2 = 1.31095$, so that $\sigma_M^2 = (5-1) * 1.31095 = 5.2438$.

Plugging the above calculated values into Equation 5.14, I plot the lower bound on expected convergence as the solid line in Figure 5.5. From this plot it appears that the population sizing equation provides an adequate lower bound on expected GA convergence on the multiglobal, multimodal, boundedly partially deceptive problem: f_{5md5G}.

5.4.3.3 Sources and degrees of difficulty

Clearly the function f_{5md5G} is of intermediate difficulty to the simple GA. One question raised by the plotted GA performance is how does this performance compare with performance on "harder" problems? In particular, f_{5md5G} was constructed according to the definitions and principles of maximal partial deception and bounded partial deception. Therefore f_{5md5G} *should* be less difficult than a function, say f_{5md4G}, with only four ($k = 4$) of the five globals in the set G. The function f_{5md4G} should in turn be less difficult than the function f_{5md3G} with another of the original five globals deleted from G ($k = 3$). And so on. The functions f_{5mdkG} should gradually and gracefully decrease in difficulty as more globals are added to G (i.e., for increasing k).

To test this hypothesis, I create four other functions, f_{5mdkG} for $k = 4..1$, by deleting globals from the right side of the list G, and then reconstructing f_{5mdkG} as five copies of the *redefined* maximally deceptive subfunction f_{mdkG} based on the remaining k globals. Thus for $k = 5$ we have the original[22] f_{5md5G} using

$$G = \{0000000000, 0010110001, 1101110011, 0101010010, 1000101111\}.$$

[22] Note one difference from the f_{md5G} used in the previous section: here I stick strictly to the multiglobal trap function definition of Equation 5.4. Thus I use $f_{max} = f_D + 1$ rather than $f_{max} = \ell_s = 10$. In the case of $k = 5$ (i.e., all five original globals), $f_D = 5$ so $f_{max} = 6$. To be clear about this different f_{max}, I call the newly defined function f_{mdkG}^{trap}.

And for $k = 4$, we have

$$G = \{0000000000, 0010110001, 1101110011, 0101010010\},$$

with $f_{max} = 6$ still. For $k = 3$,

$$G = \{0000000000, 0010110001, 1101110011\},$$

with $f_{max} = 6$ again. For $k = 2$,

$$G = \{0000000000, 0010110001\},$$

but this time $f_{max} = 8$. Finally for $k = 1$,

$$G = \{0000000000\},$$

and $f_{max} = 10$ (i.e., a fully deceptive trap function).

I perform the same experimental regime on each of these $k = 1..4$ versions of f_{5md5G} as I did on the original f_{5md5G}. That is, I run the simple GA forty times (i.e., forty different initial random populations) at each population size from $N = 50$ to 1200, in increments of 50 (i.e., $N = 50, 100, 150, 200, 250, ..., 1200$). And I use single point crossover with $p_c = 1.0$, no mutation ($p_m = 0$), and deterministic binary tournament selection in a generational (simple) GA. As before, each of the forty trials is run to convergence (using the same convergence criteria as before), and the mean convergence (i.e., percentage of building blocks found) is averaged over the forty trials. The means for each population size are plotted in Figure 5.6 for $k = 1..5$. The standard deviations can be found in the appendix, Figure A.2.

As Figure 5.6 illustrates, our expectations are confirmed. The function f^{trap}_{5mdkG} does appear to gracefully decline in difficulty as we go from full deception (at the subfunction level) at $k = 1$ to larger numbers of global optima[23]

[23] Note that the plotted performance for $k = 5$ (f^{trap}_{5md5G}) in Figure 5.6 is worse than that shown for f_{5md5G} (also $k = 5$) in Figure 5.5. This difference is due to the lower f_{max} used here (6 versus 10). The higher the f_{max} in the subfunctions the greater the subfunction signal in the signal-to-noise ratio in Equation 5.14.

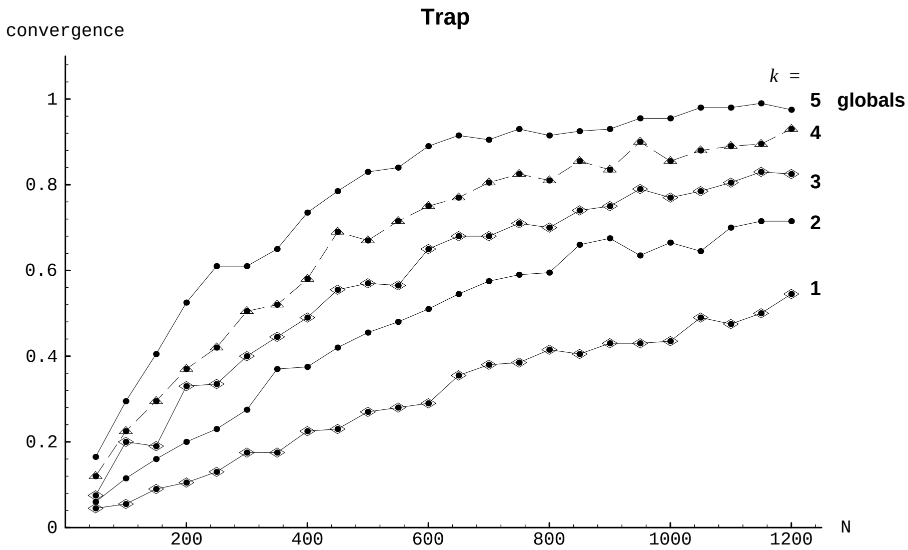

Figure 5.6: The multiglobal trap function f^{trap}_{mdkG} appears to gracefully decline in difficulty as we increase the number of globals from $k = 1$ (full deception) to $k = 5$.

But these comparative results immediately raise the question of what is the actual source of the increase/decrease in difficulty? Adding global optima to G has several related effects on f_{mdkG}, each of which might weaken the "hardness" of the function:

1. loss of deceptive (schema competition) partitions,

2. more copies of building blocks,

3. weaker deceptive optima (i.e., lower $f(D)$), and

4. smaller separation of G and D.

(1.) I conjecture that by adding randomly placed global optima, we "punch holes" in the hierarchies of deceptive partitions, thereby gradually reducing the "degree" or "amount" of deception (and therefore difficulty) of the problem. (2.) But as we add globals to G we also increase (linearly) the chances of randomly generating global (subfunction) optima in the initial population, and therefore increase the (expected) number of building blocks at generation 0. More copies of building blocks at the start implies greater chances of properly processing (i.e., isolating and mixing) them later in the run. (3.) In addition, by adding globals we (generally) decrease f_D, the fitness of the deceptive optima, which again increases the subfunction signal-to-noise ratio in Equation 5.14. (4.) Finally, adding globals also decreases the distance between global and deceptive optima, thereby increasing the probability of a lucky mutation or crossover operation "jumping" from D (or near D) to G.

I can begin to separate these sources of (the change in) problem difficulty by a series of carefully controlled experiments. In the first of these, I attempt to distinguish the role of three different facets of problem difficulty[24]:

- *isolation* (of global optima),
- *misleadingness* (of the search space), and
- *signal* (fitness difference between G and D).

[24] The first two aspects of problem difficulty, isolation and misleadingness, come directly from the original list of dimensions of problem difficulty in the introduction of this thesis and originally proposed by Goldberg (1993). The last item, signal, is critical to the population sizing of Equation 5.14 (Goldberg, Deb, & Clark, 1992), and can also be considered a component of misleadingness.

The simple, additive construction of subfunction f_{mdG} allows us to easily modify it to emphasize separately each of the aspects of difficulty above. Consider that $f_{mdG} = f_{md} + f_G$ consists of isolated global optima (the peaks given by f_G), the deceptive optima D which are the "global optima" of f_{md}, and the slopes defined by f_{md} which lead to D (misleadingness). Thus the original function f_{mdG} contains all three components of problem difficulty.

On the other hand, a "needle-in-a-haystack" (**NIAH**) version of f_{mdG} would consist of a flat (zero valued) landscape with k global peaks: $f^{NIAH}(s) = f_G(s)$. In the **NIAH** function, the globals are just as isolated as in f_{mdG}, since in both cases there is no information in the search space pointing GA search toward the globals. However, the **NIAH** landscape contains no misleadingness, since there is no information leading *away* from G either.

Intermediate between the original (**Trap**) version of f_{mdG} and the **NIAH** version is a function consisting only of the global and the deceptive optima. In other words, removing the "slopes" leading up to D (and down to G) that are defined by f_{md} leaves a NIAH-style landscape: flat (zero-valued) with spikes (peaks) at each of the members of G and D. (The fitnesses of these optima are the same as in f^{trap}_{mdG}: $f(G) = f_{max} = f(D)+1$, and $f(D) = f_{md}(D)-1$.) This **NoSlope** version of f_{mdG} has the isolation property of **Trap** and **NIAH**, But unlike **NIAH**, **NoSlope** (like **Trap**) has the reduced signal (of the global optima) induced by the competing optima D. Thus **NoSlope** should be intermediate in difficulty between **NIAH** and **Trap**. To sum up:

- **Trap** ($f^{trap}_{mdG} = f_{md} + f_G$) contains *isolation, reduced signal,* and *misleadingness*, while
- **NoSlope** ($f^{trap}_{mdG} = f_D + f_G$) contains *isolation,* and *reduced signal*, and
- **NIAH** ($f^{trap}_{mdG} = f_G$) contains *isolation*,

where f_D is similar to f_G but for deceptive optima (i.e., f_D is a spike function over D rather than G).

I rerun the experiments depicted in Figure 5.5 (i.e., same concatenation of five identical copies of the subfunction f_{mdG} at hand, same experimental regime, same GA parameter settings, but only on $k = 5$ globals, and not for $k < 5$). Figure 5.7 plots the results (means) of running the simple GA on all three versions of f_{mdG}. (The standard deviations of the plotted means are listed in Figure A.3 of the appendix.) The results confirm our intuition of relative performance. Over the range of population sizes shown here, the GA does indeed exhibit better performance

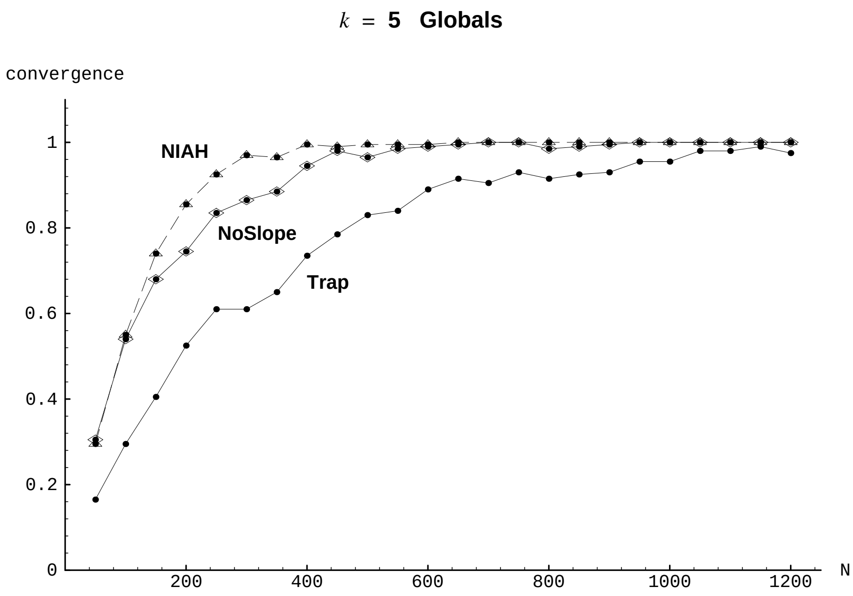

Figure 5.7: By selectively removing the gradient information (**NoSlope**), then the deceptive optima (**NIAH**), from the original **Trap** function f_{mdG}^{trap}, we can degrade the difficulty of the 50-bit, $k = 5$ global version of the problem.

on **NIAH** than it does when it has to contend with competing optima D (i.e., **NoSlope**). Apparently, the most difficult problem for the GA is the original **Trap** function, with its deceptive optima *and* misleading slopes (and deceptive schema partitions).

We can also look at the effects of adding and deleting globals from G (i.e., varying k) on the two new versions of f_{mdG}: **NIAH** and **NoSlope**. I rerun the experiment whose results are depicted in Figure 5.6, this time on the functions **NIAH** and **NoSlope**. Figure 5.8 presents the GA results on **NIAH** and **NoSlope**, with the results from Figure 5.6 (**Trap**) shown at the top for comparison[25]. In all three cases, the additional globals gradually erode the problem's difficulty. Also, note how the relative order of difficulty for the three versions of the function (i.e., **NIAH** $\rightarrow$ **NoSlope** $\rightarrow$ **Trap**, least to most difficult), seems to hold over the range of $k = 1..5$ globals.

[25] Again, the standard deviations for these runs have been placed in the appendix, in Figures A.4 and A.5.

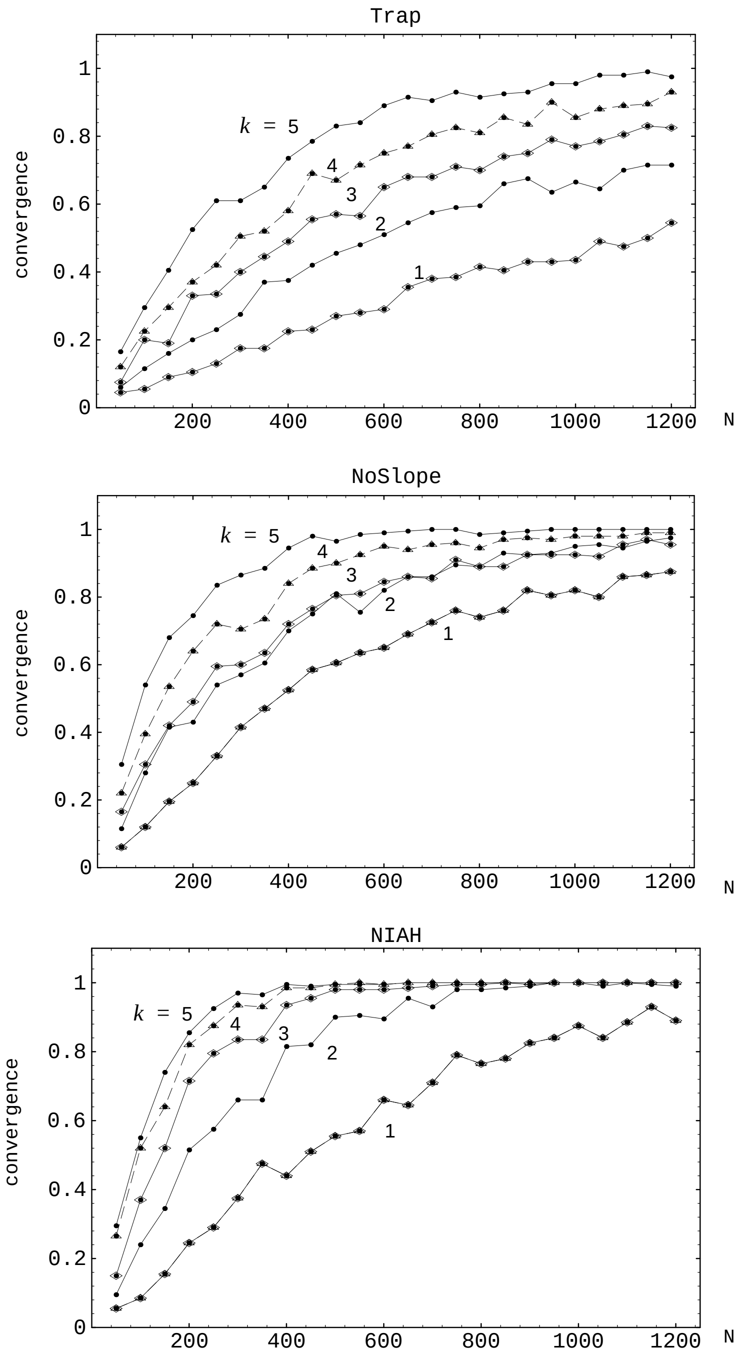

Figure 5.8: For all three versions of f_{mdG}, problem difficulty gracefully degrades as we add more globals (i.e., increase k).

Whether one views the added difficulty induced by the "slopes" as being due to the additional deceptive partitions or being due to "hillclimbing basins of attraction" depends on one's model of GA processing (i.e., whether the GA is solving schema partitions or climbing hills in the fitness landscape). Regardless of one's model of GA search, these experiments clearly indicate that each of the quasi-separable characteristics of fitness landscape optima (isolation, misleadingness, global-local fitness signal, and number k of globals) by themselves add to the overall difficulty of the problem for GA search.

5.4.4 Extensions

A number of other experiments come to mind that should be straightforward to conduct. For example, we could analyze the separation of G from D by moving the deceptive optima D around on the **NoSlope** function above. Care must be taken, however, to conduct *controlled* experiments, isolating only one aspect or characteristic of the landscape at a time.

Increasing the number of global optima k in the subfunctions, and increasing the number m of subfunctions, can lead to very high degrees of modality. For example, Goldberg, Deb, and Horn (1992) use five copies of their six bit bipolar deceptive function resulting in 32 global optima among over five million local optima. Five million is approximately 0.5% of the 2^{30} total size of the search space, and is thus within two orders of magnitude of the absolute maximum number of local optima (50%).

The difficulty of function $f_{mdG}(s)$ could be estimated by Jones' *fitness distance correlation* (FDC) measure (Jones, 1995; Jones & Forrest, in press) briefly mentioned earlier. It is clear from its definition that the component function f_{md} is maximally correlated with distance to the nearest global optimum. Therefore $f_{mdG}(s)$ should rate at the maximum (one) on the FDC scale of problem difficulty (which increases from -1 to 1).

The function $f_{mdG}(s)$, although general with respect to the number and placement of globals, can be generalized further. We could make each global more attractive by simply increasing the radii $r_{g,i}$ of each global g_i. Thus $f_{md}(s)$ would be defined as before *outside* of the $r_{g,i}$-bit neighborhoods of each global. Within each neighborhood, however, the $f_{mdG}(s)$ could be a plateau of optimal fitness, or a hill leading to the global[26].

[26] Deb and Goldberg (1993) define a linear slope leading from all points of distance a (equivalent to the radius r_g) from the global optimum, where the function is zero valued, to the global optimum itself, where the function

Finally, I note that the function f_{md} is not analytic at the globals nor at any of the points on the Voronoi lines. A differentiable version of f_{md} could allow additional analysis (e.g., of the function's gradients, schema average fitnesses, etc.). Selecting the minimum (or maximum) distance from a set of distances can be done using the *maximum Holder norm.* Using any lower order (i.e., finite) Holder norm in place of the maximum (or *infinite order*) Holder norm yields a function df_{md} that is differentiable everywhere and can be made arbitrarily close to f_{md} by increasing the order of the norm. I used such a technique to quickly and easily generate the gradient plot in Figure 5.3, bottom, taking the derivative of an approximation to $f_{mdG}(x, y)$ using a finite order Holder norm for $f_{md}(x, y)$.

is maximally valued. This function is their generalized (single global) trap function. My suggestion is to make the same generalization to the k-global trap functions.

Chapter 6

Conclusions

Modality by itself, if defined solely as the number of local optima, actually tells us little about the difficulty of searching a space. Unimodal landscapes can be difficult and maximally multimodal surfaces can be easy. But the presence of local optima *does* affect GA search. In particular, GA search is misled by *attractive* local optima. Thus it is the number *and* attractiveness of local optima, relative to the attractiveness of the global optima, that determine the degree of difficulty.

How do we rigorously define attractiveness? It is relatively easy to visualize local optima basins of attraction to hillclimbers on the fitness landscape of Sewall Wright. We can "see" the size, shape, and boundary of each such basin. On the other hand, current GA theory gives us a model of crossover-based search that defines attraction in terms of schema average fitnesses in hyperplane competitions; a paradigm difficult to visualize, but the only model of crossover search we have. Yet we "know" that hillclimbing basins of attraction affect crossover's search. And we know that local optima in the fitness landscape determine the winners of partition competitions (and vice versa).

In this thesis I have tried to relate the "hillclimbing-relevant" features of Wright's fitness landscape (e.g., hills, paths, slopes, optima, plateaus) to the "crossover-relevant" features of fitness functions (e.g., deceptive attractors, hyperplane competitions). That is, I have attempted to make connections between crossover's search of hyperplanes and the population's movement over the fitness landscape. In the process of relating GAs and hillclimbers via the fitness landscape, I have made some counter-intuitive observations:

- **Simple hillclimbing can be slow**, indeed, exponential in the problem length. That is, simply climbing to the nearest local optimum can be intractable. This is a new and surprising result with possible practical implications. The Root2path, the CubeRoot2path, and the Fibonacci path are all contrived problems, but other long paths might be more general. Even polynomially long paths could significantly change the expected running time of local search procedures. Thus hybrid GA-hillclimbers, in which offspring are "hillclimbed" to their nearest local optima before insertion into the new population, might do better to turn off local hillclimbing on problems suspected of harboring long paths.

- **GA crossover can outperform simple hillclimbing on simple hills.** There seems to be growing concern that GAs are too often outperformed by simple hillclimbers on many problems, including GA test suites and open problems. A common response to this concern is an effort to make the GA behave more like a hillclimber on hills. Of course, the problems on which the GA consistently outperforms hillclimbing are multimodal, with attractive local optima. But the long path problems show that even on a simple hill, GA crossover can beat hillclimbing. And so we must question any move to make GAs emulate hillclimbing, on any function.

- **Maximum modality is meaningless.** The maximum number of local optima is one half the search space. But at this extreme the local optima are minimally sized with minimal attraction. Such functions can be made relatively easy for GAs and local searchers.

In addition to revealing these "failed" links between landscape features (e.g., modality, simple paths) and GA (and HC) performance, I have also found some strong links:

- **Maximizing misleadingness (to GA crossover) appears to induce local optima D that are maximally distant from the global optima G.** In the past we have generally chosen the deceptive attractor D first, and then defined deceptive partitions in terms of D. Here we saw how maximizing the number of deceptive partitions of all orders leads to a unique choice of deceptive local optimum, since the number of deceptive partitions increases with distance to the set of globals.

- **Maximizing misleadingness seems to isolate the global optima**, even in the case of multiple (k) globals. That is, minimizing crossover's "basin of attraction" to G also minimizes the hillclimber's basin of attraction, resulting in a "spike" of high fitness at the globals.

- **Adding global optima gradually erodes the attraction of the deceptive optima.** Thus the presence of optima, even NIAHs, impinge on GA crossover search.

- **A highly if not maximally deceptive multiglobal function, f_{mdG}^{trap}, seems to relate the attraction of crossover to interesting geometric features of the landscape,** such as Voronoi diagrams and Delaunay triangularizations (Conway & Sloane, 1993), in addition to local optima, ridges, and the landscape's gradient field.

Finally, in addition to these links between GA crossover search and hillclimbing attributes of Wright fitness landscapes, the work on *intermediate modality* yields some insights and innovations to the study of deception in general:

- **Full deception and GA-easiness in uni- or bimodal landscapes tell us only about the extreme boundaries of GA success and failure.** But maximally deceptive multimodal functions allow us to embed deception in much more rugged and general landscapes, and to define arbitrarily sized and spaced local optima and their basins of attraction to a GA. These more realistic instances of deception illustrate the difficulty of generalizing simple approaches to solving *fully* deceptive problems, such as a complement operator (Grefenstette, 1993)[1].

- **Generalizing the definitions of full deception in order to characterize partial deception is tricky.** It is all too easy to lose the essential quality of misleadingness. I have introduced one general method of relaxing the deception conditions that partially orders problem spaces according to GA misleadingness, just as *bounded deception* has done before.

- **Three dimensions of GA problem difficulty (isolation, misleadingness, and multimodality) are quasi-separable but also closely related.** Previous GA theory work has concentrated on misleadingness, noise, and other aspects (from our original

[1] The complement of the deceptive attractor in a fully deceptive problem is indeed the global optimum, but this does not hold in the more general case of partial deception. For example, in a bipolar deceptive problem the complement of every deceptive optimum is another deceptive optimum.

breakdown) separately. Here we have considered modality separately, and found that it is not meaningful without discussing isolation and misleadingness. So this thesis, which began as a first attempt to tackle the *multimodality* dimension of problem difficulty, has grown to encompass, for the first time, the *relationships* among several dimensions of difficulty.

Lastly, I believe that the Wright fitness landscape, although most directly relevant to hill-climbers, is still appropriate and useful to GA theory. Its powerful images might sometimes mislead us in our thinking about GA search, but I think that more often the metaphor guides us down the right path.

Appendix A

Means and Standard Deviations from Chapter 5 Experiments

This appendix provides the statistics (means and standard deviations) used to generate the convergence graphs in Figures 5.5 through 5.8. These are the data derived from the experiments described in Chapter 5.

Statistics (mean,std.dev) for
"Predicted versus Actual" experiment (Figure 5.5)

Pop. Size		
N = 50	0.2	mean
	0.187	std. dev.
100	0.375	
	0.177	
150	0.555	
	0.254	
200	0.66	
	0.218	
250	0.705	
	0.192	
300	0.76	
	0.182	
350	0.76	
	0.209	
400	0.88	
	0.142	
450	0.905	
	0.128	
500	0.915	
	0.119	
550	0.93	
	0.0966	
600	0.965	
	0.0893	
650	0.985	
	0.07	
700	0.98	
	0.0608	
750	0.975	
	0.067	
800	0.985	
	0.0533	
850	0.99	
	0.0441	
900	0.995	
	0.0316	
950	0.995	
	0.0316	
1000	0.995	
	0.0316	
1050	0.99	
	0.0441	
1100	1.	
	0	
1150	1.	
	0	
1200	0.995	
	0.0316	

Figure A.1: Data from experiment shown in Figure 5.5.

Statistics (mean,std.dev) for
TRAP function experiment (Figure 5.6)

Pop. Size \ k =	5	4	3	2	1	globals
N = 50	0.165	0.12	0.075	0.06	0.045	mean
	0.175	0.126	0.108	0.0928	0.0846	std. dev.
100	0.295	0.225	0.2	0.115	0.055	
	0.187	0.188	0.157	0.142	0.111	
150	0.405	0.295	0.19	0.16	0.09	
	0.215	0.181	0.169	0.158	0.135	
200	0.525	0.37	0.33	0.2	0.105	
	0.234	0.224	0.238	0.175	0.157	
250	0.61	0.42	0.335	0.23	0.13	
	0.186	0.234	0.209	0.179	0.173	
300	0.61	0.505	0.4	0.275	0.175	
	0.186	0.207	0.217	0.161	0.171	
350	0.65	0.52	0.445	0.37	0.175	
	0.225	0.196	0.2	0.19	0.171	
400	0.735	0.58	0.49	0.375	0.225	
	0.172	0.226	0.212	0.203	0.182	
450	0.785	0.69	0.555	0.42	0.23	
	0.194	0.175	0.195	0.196	0.184	
500	0.83	0.67	0.57	0.455	0.27	
	0.154	0.215	0.205	0.187	0.179	
550	0.84	0.715	0.565	0.48	0.28	
	0.171	0.221	0.192	0.216	0.216	
600	0.89	0.75	0.65	0.51	0.29	
	0.157	0.174	0.148	0.222	0.181	
650	0.915	0.77	0.68	0.545	0.355	
	0.119	0.224	0.18	0.207	0.184	
700	0.905	0.805	0.68	0.575	0.38	
	0.136	0.147	0.168	0.208	0.23	
750	0.93	0.825	0.71	0.59	0.385	
	0.0966	0.158	0.202	0.226	0.214	
800	0.915	0.81	0.7	0.595	0.415	
	0.1	0.143	0.187	0.2	0.214	
850	0.925	0.855	0.74	0.66	0.405	
	0.108	0.157	0.171	0.241	0.224	
900	0.93	0.835	0.75	0.675	0.43	
	0.107	0.149	0.168	0.22	0.224	
950	0.955	0.9	0.79	0.635	0.43	
	0.0959	0.143	0.15	0.202	0.205	
1000	0.955	0.855	0.77	0.665	0.435	
	0.0846	0.143	0.184	0.205	0.23	
1050	0.98	0.88	0.785	0.645	0.49	
	0.0608	0.109	0.159	0.233	0.197	
1100	0.98	0.89	0.805	0.7	0.475	
	0.0758	0.135	0.154	0.187	0.206	
1150	0.99	0.895	0.83	0.715	0.5	
	0.0441	0.128	0.147	0.197	0.192	
1200	0.975	0.93	0.825	0.715	0.545	
	0.0809	0.124	0.193	0.207	0.187	

Figure A.2: Data from experiment shown in Figure 5.6.

Statistics (mean,std.dev) for
"k = 5 Globals" comparison experiment (Figure 5.7)

Pop. Size	NIAH	NOSLOPE	TRAP	
N = 50	0.295	0.305	0.165	mean
	0.297	0.272	0.175	std. dev.
100	0.55	0.54	0.295	
	0.289	0.223	0.187	
150	0.74	0.68	0.405	
	0.249	0.191	0.215	
200	0.855	0.745	0.525	
	0.136	0.212	0.234	
250	0.925	0.835	0.61	
	0.126	0.175	0.186	
300	0.97	0.865	0.61	
	0.0723	0.159	0.186	
350	0.965	0.885	0.65	
	0.0893	0.127	0.225	
400	0.995	0.945	0.735	
	0.0316	0.0904	0.172	
450	0.99	0.98	0.785	
	0.0441	0.0608	0.194	
500	0.995	0.965	0.83	
	0.0316	0.0893	0.154	
550	0.995	0.985	0.84	
	0.0316	0.0533	0.171	
600	0.995	0.99	0.89	
	0.0316	0.0441	0.157	
650	1.	0.995	0.915	
	0	0.0316	0.119	
700	1.	1.	0.905	
	0	0	0.136	
750	1.	1.	0.93	
	0	0	0.0966	
800	1.	0.985	0.915	
	0	0.0533	0.1	
850	1.	0.99	0.925	
	0	0.0441	0.108	
900	1.	0.995	0.93	
	0	0.0316	0.107	
950	1.	1.	0.955	
	0	0	0.0959	
1000	1.	1.	0.955	
	0	0	0.0846	
1050	1.	1.	0.98	
	0	0	0.0608	
1100	1.	1.	0.98	
	0	0	0.0758	
1150	1.	1.	0.99	
	0	0	0.0441	
1200	1.	1.	0.975	
	0	0	0.0809	

Figure A.3: Data from experiment shown in Figure 5.7.

Statistics (mean,std.dev) for
NOSLOPE function experiment (Figure 5.8, middle)

Pop. Size	k = 5	4	3	2	1	globals
N = 50	0.305	0.22	0.165	0.115	0.06	mean
	0.272	0.207	0.175	0.142	0.113	std. dev.
100	0.54	0.395	0.305	0.28	0.12	
	0.223	0.219	0.181	0.191	0.162	
150	0.68	0.535	0.42	0.415	0.195	
	0.191	0.219	0.226	0.219	0.215	
200	0.745	0.64	0.49	0.43	0.25	
	0.212	0.245	0.239	0.22	0.243	
250	0.835	0.72	0.595	0.54	0.33	
	0.175	0.186	0.2	0.182	0.22	
300	0.865	0.705	0.6	0.57	0.415	
	0.159	0.207	0.175	0.179	0.224	
350	0.885	0.735	0.635	0.605	0.47	
	0.127	0.183	0.175	0.166	0.246	
400	0.945	0.84	0.72	0.7	0.525	
	0.0904	0.177	0.186	0.235	0.225	
450	0.98	0.885	0.765	0.75	0.585	
	0.0608	0.135	0.175	0.168	0.25	
500	0.965	0.9	0.805	0.81	0.605	
	0.0893	0.111	0.178	0.186	0.2	
550	0.985	0.925	0.81	0.755	0.635	
	0.0533	0.108	0.15	0.189	0.197	
600	0.99	0.95	0.845	0.82	0.65	
	0.0441	0.0877	0.147	0.149	0.221	
650	0.995	0.94	0.86	0.86	0.69	
	0.0316	0.103	0.145	0.145	0.235	
700	1.	0.955	0.855	0.86	0.725	
	0	0.0959	0.143	0.13	0.179	
750	1.	0.96	0.91	0.895	0.76	
	0	0.081	0.119	0.128	0.204	
800	0.985	0.945	0.89	0.89	0.74	
	0.0533	0.101	0.135	0.143	0.198	
850	0.99	0.97	0.89	0.93	0.76	
	0.0441	0.0723	0.135	0.124	0.213	
900	0.995	0.975	0.925	0.925	0.82	
	0.0316	0.067	0.117	0.133	0.174	
950	1.	0.97	0.925	0.93	0.805	
	0	0.0723	0.117	0.14	0.205	
1000	1.	0.98	0.925	0.95	0.82	
	0	0.0608	0.141	0.0877	0.18	
1050	1.	0.98	0.92	0.955	0.8	
	0	0.0608	0.0992	0.0846	0.181	
1100	1.	0.98	0.955	0.945	0.86	
	0	0.0608	0.0959	0.101	0.165	
1150	1.	0.99	0.97	0.965	0.865	
	0	0.0441	0.0853	0.0893	0.166	
1200	1.	0.99	0.955	0.975	0.875	
	0	0.0441	0.106	0.067	0.174	

Figure A.4: Data from experiment shown in Figure 5.8 (middle).

Statistics (mean,std.dev) for
NIAH function experiment (Figure 5.8, bottom)

Pop. Size	k = 5	4	3	2	1	globals
N = 50	0.295	0.265	0.15	0.095	0.055	mean
	0.297	0.269	0.191	0.136	0.111	std. dev.
100	0.55	0.52	0.37	0.24	0.085	
	0.289	0.289	0.288	0.245	0.149	
150	0.74	0.64	0.52	0.345	0.155	
	0.249	0.273	0.303	0.268	0.215	
200	0.855	0.82	0.715	0.515	0.245	
	0.136	0.18	0.248	0.264	0.258	
250	0.925	0.875	0.795	0.575	0.29	
	0.126	0.133	0.172	0.218	0.256	
300	0.97	0.935	0.835	0.66	0.375	
	0.0723	0.123	0.156	0.218	0.257	
350	0.965	0.93	0.835	0.66	0.475	
	0.0893	0.107	0.149	0.193	0.319	
400	0.995	0.985	0.935	0.815	0.44	
	0.0316	0.0533	0.114	0.183	0.304	
450	0.99	0.985	0.955	0.82	0.51	
	0.0441	0.07	0.0959	0.216	0.31	
500	0.995	0.995	0.98	0.9	0.555	
	0.0316	0.0316	0.0608	0.197	0.285	
550	0.995	1.	0.98	0.905	0.57	
	0.0316	0	0.0608	0.143	0.258	
600	0.995	0.995	0.98	0.895	0.66	
	0.0316	0.0316	0.0608	0.12	0.269	
650	1.	1.	0.985	0.955	0.645	
	0	0	0.0533	0.0846	0.277	
700	1.	1.	0.99	0.93	0.71	
	0	0	0.0441	0.0966	0.248	
750	1.	1.	0.995	0.98	0.79	
	0	0	0.0316	0.0608	0.231	
800	1.	1.	0.995	0.98	0.765	
	0	0	0.0316	0.0758	0.252	
850	1.	1.	1.	0.985	0.78	
	0	0	0	0.0533	0.23	
900	1.	1.	0.995	0.99	0.825	
	0	0	0.0316	0.0441	0.218	
950	1.	1.	1.	1.	0.84	
	0	0	0	0	0.177	
1000	1.	1.	1.	1.	0.875	
	0	0	0	0	0.133	
1050	1.	1.	1.	0.99	0.84	
	0	0	0	0.0441	0.188	
1100	1.	1.	1.	1.	0.885	
	0	0	0	0	0.142	
1150	1.	1.	1.	0.995	0.93	
	0	0	0	0.0316	0.116	
1200	1.	1.	1.	0.99	0.89	
	0	0	0	0.0441	0.15	

Figure A.5: Data from experiment shown in Figure 5.8 (bottom).

References

Ackley, D. H. (1985). A connectionist algorithm for genetic search. In J. J. Grefenstette (Ed.), *Proceedings of an International Conference on Genetic Algorithms* (pp. 121–135). Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers.

Ackley, D. H. (1987a). An empirical study of bit vector function optimization. In L. Davis (Ed.), *Genetic algorithms and simulated annealing* (pp. 170–204). London: Pitman Publishing.

Ackley, D. H. (1987b). *A Connectionist Machine for Genetic Hillclimbing.* Boston, MA: Kluwer Academic Publishers.

Bethke, A. D. (1981). Genetic algorithms as function optimizers. (Doctoral dissertation, University of Michigan at Ann Arbor). *Dissertation Abstracts International*, **41**(9), 3503B. (University Microfilms No. 81-06101)

Conway, J. H., & Sloane, N. J. A. (1993). *Sphere packings, lattices, and groups* (2nd ed.). New York: Springer-Verlag.

Deb, K., & Goldberg, D. E. (1993). Analyzing deception in trap functions. In D. Whitley (Ed.), *Foundations of Genetic Algorithms 2* (pp. 93–108). San Mateo, CA: Morgan Kaufmann.

Deb, K., & Goldberg, D. E. (1994). Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, **10**, 385–408.

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

Forrest, S., & Mitchell, M. (1993). Relative building-block fitness and the building-block hypothesis. In L.D. Whitley (Ed.), *Foundations of Genetic Algorithms 2* (pp. 109–126). San Mateo, CA: Morgan Kaufmann.

Goldberg, D. E. (1987). Simple genetic algorithms and the minimal deceptive problem. In L. Davis (Ed.), *Genetic algorithms and simulated annealing* (pp. 74–88). London: Pitman Publishing.

Goldberg, D. E. (1989a). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

Goldberg, D. E. (1989b). Genetic algorithms and Walsh functions: part I, a gentle introduction. *Complex Systems*, **3**, 129–152.

Goldberg, D. E. (1989c). Genetic algorithms and Walsh functions: part II, deception and its analysis. *Complex Systems*, **3**, 153–171.

Goldberg, D. E. (1991a). Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, **5**, 139–168.

Goldberg, D. E. (1991b). Construction of high-order deceptive functions using low-order Walsh coefficients. *Annals of Mathematics and Artificial Intelligence*, **5**, 35–48.

Goldberg, D. E. (1993, February). Making genetic algorithms fly: a lesson from the Wright brothers. *Advanced Technology for Developers*, **2**, 1–8.

Goldberg, D. E. (1994). Genetic and evolutionary algorithms come of age. *Communications of the Association for Computing Machinery*, **37**(3), 113–119.

Goldberg, D. E., Deb, K., & Clark, J. H. (1992). Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, **6**, 333–362.

Goldberg, D. E., Deb, K., & Horn, J. (1992). Massive multimodality, deception, and genetic algorithms. In R. Männer, & B. Manderick (Ed.s), *Parallel Problem Solving from Nature, 2* (pp. 37–46). Amsterdam: North-Holland.

Goldberg, D. E., Deb, K., & Korb, B. (1991). Don't worry, be messy. In R. K. Belew, & L. B. Booker (Ed.s), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 24–30). San Mateo, CA: Morgan Kaufmann.

Grefenstette, J.J. (1993). Deception considered harmful. In L. D. Whitley (Ed.), *Foundations of Genetic Algorithms 2* (pp. 75–91). San Mateo, CA: Morgan Kaufmann.

Harary, F., Hayes, J. P., & Wu, H-J. (1988). A survey of the theory of hypercube graphs. *Computational Mathematical Applications*, **15**(4), 277–289.

Hoffmeister, F., & Bäck, T. (1990). *Genetic algorithms and evolutionary strategies: similarities and differences* (Technical Report "Grüne Reihe" No. 365). Dortmund, Germany: Department of Computer Science, University of Dortmund.

Holland, J. H. (1992). *Adaptation in natural and artificial systems* (2nd ed.). Cambridge, MA: The MIT Press.

Homaifar, A., Qi, X., & Fost, J. (1991). Analysis and design of a general GA deceptive problem. In R. K. Belew, & L. B. Booker (Ed.s), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 196–203). San Mateo, CA: Morgan Kaufmann.

Horn, J., Goldberg, D. E., & Deb, K. (1994). Long path problems. In Y. Davidor, H.-P. Schwefel, & R. Männer (Ed.s), *Lecture Notes in Computer Science: Vol. 866. Parallel Problem Solving From Nature – PPSN III* (pp. 149–158). Berlin: Springer-Verlag.

Horn, J. & Goldberg, D. E. (in press). Genetic algorithm difficulty and the modality of fitness landscapes. In L. D. Whitley, & M. D. Vose (Ed.s), *Foundations of Genetic Algorithms 3.* San Mateo, CA: Morgan Kaufmann.

Jones, T. (1995). *Evolutionary algorithms, fitness landscapes and search.* Unpublished doctoral dissertation, University of New Mexico, Albuquerque, NM.

Jones, T., & Forrest, S. (in press). Fitness distance correlation as a measure of problem difficulty. In L. Eshelman (Ed.), *Proceedings of the Sixth International Conference on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann.

Jones, T., & Rawlins, G. J. E. (1993). Reverse hillclimbing, genetic algorithms and the busy beaver problem. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 70–75). San Mateo, CA: Morgan Kaufmann.

Kauffman, S. A. (1989). Adaptation on rugged fitness landscapes. In D. L. Stein (Ed.), *Lectures in the Sciences of Complexity* (pp. 527–618). Reading, MA: Addison-Wesley.

MacWilliams, F. J., & Sloane, N. J. A. (1977). *The theory of error correcting codes.* Amsterdam: North-Holland.

Mahfoud, S. W. (1993). Simple analytical models for genetic algorithms for multimodal function optimization. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (p. 643). San Mateo, CA: Morgan Kaufmann.

Manderick, B., de Weger, M., & Spiessens, P. (1991). The genetic algorithm and the structure of the fitness landscape. In R. K. Belew, & L. B. Booker (Ed.s), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 143–150). San Mateo, CA: Morgan Kaufmann.

Mitchell, M., Forrest, S., & Holland, J. H. (1991). The royal road for genetic algorithms: fitness landscapes and GA performance. In J. Varela, & P. Bourgine (Ed.s), *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life* (pp. 245–254). Cambridge, MA: The MIT Press.

Mitchell, M., & Holland, J. H. (1993). When will a genetic algorithm outperform hill climbing? In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms* (p. 647). San Mateo, CA: Morgan Kaufmann.

Mitchell, M., Holland, J. H., & Forrest, S. (1994). When will a genetic algorithm outperform hill climbing? In J. Cowan, G. Tesauro, & J. Alspector (Ed.s), *Advances in Neural Information Processing Systems*, **6**. San Mateo, CA: Morgan Kaufmann.

Mühlenbein, H. (1992). How genetic algorithms really work, I. fundamentals. In: R. Männer, & B. Manderick (Ed.s), *Parallel Problem Solving From Nature, 2* (pp. 15–26). Amsterdam: North-Holland.

Preparata, F. P., & Niervergelt, J. (1974). Difference-preserving codes. *IEEE Transactions on Information Theory*, **IT-20**(5), 643–649.

Radcliffe, N. J., & Surry, P. D. (1994). Formal memetic algorithms. In T. C. Fogarty (Ed.), *Lecture Notes in Computer Science: Vol. 865. Evolutionary Computing* (pp. 1–16). Berlin: Springer-Verlag.

Vose, M. D. (1993). Modeling simple genetic algorithms. In L. D. Whitley (Ed.), *Foundations of Genetic Algorithms 2* (pp. 63–73). San Mateo, CA: Morgan Kaufmann.

Whitley, D. L. (1991). Fundamental principles of deception in genetic search. In G. J. E. Rawlins (Ed.), *Foundations of Genetic Algorithms* (pp. 221–241). San Mateo, CA: Morgan Kaufmann.

Wilson, S. W. (1991). GA-easy does not imply steepest-ascent optimizable. In R. K. Belew, & L. B. Booker (Ed.s), *Proceedings of the Fourth International Conference on Genetic Algorithms* (pp. 85–89). San Mateo, CA: Morgan Kaufmann.

Wright, S. (1988). Surfaces of selective value revisited. *American Naturalist*, **131**(1), 115–123.