

ornl

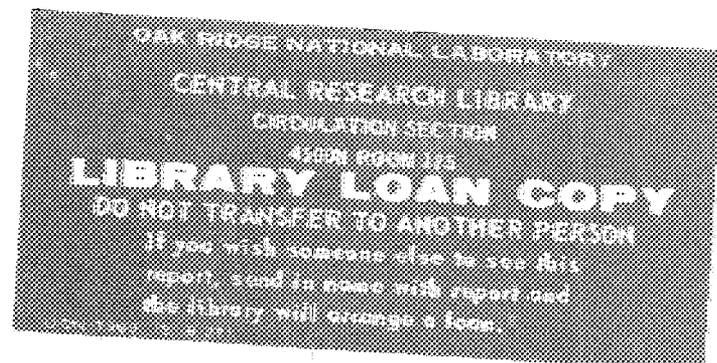
ORNL/TM-11140

**OAK RIDGE
NATIONAL
LABORATORY**

MARTIN MARIETTA

A FRAMEWORK FOR STUDYING GENETIC OPTIMIZATION OF COMPLEX SYSTEMS

Gunar Liepins



OPERATED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DCE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401. FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5286 Port Royal Rd., Springfield, VA 22101.
NTIS price codes—Printed Copy: A04 Microfiche A01

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-11140

Energy Division

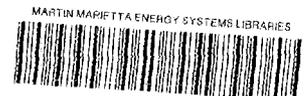
**A FRAMEWORK FOR STUDYING GENETIC OPTIMIZATION
OF COMPLEX SYSTEMS**

by

G. E. Liepins

Date Published - September 1989

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
operated by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400



3 4456 0327571 0

TABLE OF CONTENTS

ABSTRACT	vii
1. INTRODUCTION	1
2. GENETIC ALGORITHM BASICS	3
3. PERFORMANCE FACTORS	5
4. OPTIMIZATION DOMAIN	9
5. EXISTENCE RESULTS	13
6. POLYNOMIALS	23
7. CHANGE STRATEGIES	27
8. SUMMARY	33
9. REFERENCES	35

LIST OF TABLES

Table 1. A Fully Deceptive Function	16
Table 2. A Fully Easy Function (Transformed from a Fully Deceptive One)	17
Table 3. A Selection of Easy and Hard Functions	23
Table 4. A Hamming Cliff Problem	30
Table 5. One Choice for the Transformation T	30
Table 6. Typical Transformation of the Hamming Cliff Problem of Table 3.	31
Table 7. A Transformation as a Product of Transpositions	32
Table 8. Crossover of Transformations	32

ABSTRACT

Although functions of interest are generally defined either on some subset of the Cartesian product of the reals or some discrete set, genetic optimization only directly addresses functions defined on diadic groups. As a result the success of genetic optimization relative to the original domain is often dependent on the chosen embedding and representation. This paper analyzes "ga hard and ga easy" functions, provides a constructive procedure (that does not use Walsh transforms) to generate hard problems, proves that no single representation is optimal for all functions, and introduces two classes of transformations to change selected hard functions into easy ones. Two procedures for representational change are suggested.

1. INTRODUCTION

The genetic algorithm, a search technique motivated by the metaphor of biological evolution, strongly contrasts with classical optimization methods. The genetic algorithm can attempt to optimize complex, ill-structured systems; for example, multiobjective problems which are defined only by computer simulation models. To date, the genetic algorithm has been successfully applied to diverse problems including, for example, VLSI design (Cohon and Paris, 1987), network design (Davis and Coombs, 1987a, b, and Coombs and Davis, 1987), tuning of expert systems (Kuchinski, 1985), image registration (Grefenstette and Fitzpatrick, 1985), pipeline control (Goldberg, 1983, 1987a,b), information retrieval (Gordan, 1988), determination of internal parameters leading to nondominated solutions in large, complex scheduling codes (Hilliard et. al., 1988), and two person games (Axelrod, 1980a, b, 1987). Nonetheless, genetic algorithms have also had their setbacks and disappointments. They do not seem to have exhibited distinguished performance on traveling salesman problems (Liepins et al., 1989), their convergence properties remain poorly understood, and genetic algorithm research continues to be primarily empirical. This paper provides a framework for grappling with the disappointments and for studying genetic optimization of complex systems, with particular emphasis on the role of change of representation. The importance of a good choice of representation and the need for representational change arises not only in genetic algorithms, but also in neural nets where the hidden layers are thought to adapt the representation (Baum and Wilczek, 1988) and in artificial intelligence methods in general (Dietterich and Michalski, 1983; Rich, 1983). Amarel (1968), Korf (1980), and Subramanian (1989) have addressed representational issues.

The remainder of this paper first presents a brief overview of genetic algorithms: their historical development, the basic algorithm, and a brief discussion of factors that prevent genetic algorithms from converging to function optima. Attention is restricted to single objective, unconstrained problems. The role of embedding and representation is formally specified. Easy and hard functions are defined and procedures are specified to generate examples of such functions. A key result is that for any representation there exists a function and transformation such that the function in the original representation is hard (to optimize), but is easy in the transformed representation. "Easiness and hardness" are motivated in terms of polynomials, which in turn motivate a heuristic for transforming certain classes of hard functions into easy functions. The heuristic is illustrated with several small examples.

2. GENETIC ALGORITHM BASICS

Genetic algorithms are general purpose optimization algorithms (somewhat akin to simulated annealing in that sense) developed by Holland (1975) and based on ideas of Bledsoe (1961) and others. They were designed to search irregular, poorly characterized spaces. Holland's hopes were to develop powerful, broadly applicable techniques able to handle a wide variety of problems; perhaps not necessarily generating an optimal to any one problem, but rather providing a means to attack problems that are resistant to other known techniques. Holland was inspired by the example of population genetics and placed emphasis on crossover rather than mutation as the primary genetic operator. Thus, genetic search proceeds over a number of generations, with each generation represented by a population of chromosomes. A criteria of "survival of the fittest" provides the pressure for populations to develop increasingly fit individuals.

Although there are many possible variants of the basic genetic algorithm, the fundamental underlying mechanism operates on a population of individuals (chromosomes), is relatively standard, and consists of three operations: (1) evaluation of individual fitness, (2) formation of a gene pool, and (3) recombination and mutation. The individuals resulting from these three operations form the next generation's population. The process is iterated until the system ceases to improve. (Usually at this time, the population has converged to a few well performing individuals.) Generally, each individual in the population is represented by a fixed length binary string which encodes values for variables. The population size remains fixed from generation to generation and is typically between 50 and 200 individuals. Individuals contribute to the gene pool in proportion to their relative fitness (evaluation on the function being optimized) that is, well performing individuals contribute multiple copies, and poorly performing

individuals contribute few (if any) copies. (Typically, each individual contributes its allotted integer number of copies. Whether or not an individual contributes an additional copy corresponding to the fractional part of its relative fitness is determined probabilistically -- see Baker 1987). The recombination operation is the crossover operator: The simplest variant selects two parents at random from the gene pool as well as a crossover position within the binary encoding. The parents exchange "tails" (the portion of the string to the right of the crossover point) to generate two offspring. The subsequent population consists of the offspring so generated. Sometimes it is useful to generate only a fraction of the subsequent population by crossover. In these cases, the offspring (probabilistically) replace the poorest performing individuals in the prior population. A thorough introduction and overview to genetic algorithms and classifier systems is provided in Goldberg (1989a) and public domain code is available from Grefenstette (1984).

3. PERFORMANCE FACTORS

A variety of factors affect genetic algorithm performance. Conceptually, the most straightforward are the various parameter settings for population size, crossover rate, and mutation rate. The most systematic study of appropriate choices for these parameters was undertaken by DeJong (1975). Among his results he demonstrated that for the nonsmooth and multi-modal functions in his experiment, genetic algorithms were at least competitive with PRAXIS (Brent, 1971) and the Fletcher-Powell algorithm (Fletcher and Powell, 1963); Fletcher, 1970; Huang, 1970). Grefenstette (1986) has investigated the use of a meta-level genetic algorithm for parameter settings. Goldberg (1988a) has investigated population sizing. Other investigators have investigated dynamically chosen crossover positions and rates. For a survey of these results see Liepins and Hilliard (1989).

The fundamental concept at the basis of most analysis of genetic algorithm performance is that of a schema (or hyperplane). Holland recognized that any algorithm that limits its information processing to one or two points is virtually doomed to failure on any space of sufficient cardinality. Fortunately, genetic algorithms can be shown to be more globally oriented. Consider a binary representation of length L . Each individual in a population belongs to 2^l schemata. (Each is an exemplar of 2^l schemata). For example, let "*" be a "don't care" symbol, that is "*" matches either 0 or 1. Then the individual 01101...111 belongs to the schema $0^*10^*...^*1^*$, the schema $^*1^*1^*...^*11$, and $2^l - 2$ other schemata.

In some sense, schema can be considered to represent the "direction" of the genetic search. Given two conflicting schemata (the "*", s occur in common locations and the schemata differ in at least one specific location), the genetic algorithm must determine how to allocate its search resources between the schemata. The genetic algorithm can be thought

to implicitly collect statistics about schemata utilities (fitness) in terms of the average fitness of their exemplars. Holland's (1975) results regarding allocation of trials to k-armed bandit problems suggest that so long as schemata utilities are correctly estimated, the genetic algorithm optimally allocates its resources. The selection of copies of individuals to the gene pool in direct proportion to their fitness biases the gene pool in the direction of the more favorably estimated schemata. In the early generations, a wide variety of schemata are represented in the gene pool, and the search is primarily a winnowing of low order schema (many "*"s -- high dimensional hyperplanes). Later generations show increased concurrence at individual bit positions, and the search proceeds among higher order schema.

In terms of schema-based analysis, genetic algorithms can fail to converge to function optima for at least one of three reasons:

1. Schemata utilities (the average fitness of the schema exemplars) cannot be reliably estimated by the genetic algorithm. In other words, the sampling error is too large.
2. Schemata utilities can be accurately estimated, but "point" in the "wrong direction."
3. Schemata utilities can be accurately estimated and "point" in the "right direction," but crossover destroys the individuals representing these schemata.

The third failure mode is partially addressed by the schema theorem (Holland, 1975) which provides a lower bound to the growth of the expected number of schema representatives: Let H be a schema. The defining length of $d(H)$ of H is distance between the first and last specific string position, for example $10^{*}1^{*}^{*}$ has defining length 4. The order $o(H)$ of H is the number of specific bits in H ; for example, the previous schemata H has order 3. Let $m(H,t)$ be the number of exemplars of schema H in the population at time t . Let $f(H)$ be the estimated schema fitness and f the average fitness of the population (at time t). Let p_c and p_m be the

crossover probability and mutation probability, respectively. Then $m(H,t+1)$ is at least the number of exemplars from H in the previous generation that survive crossover and mutation:

$$m(H,t+1) \geq m(H,t) \frac{f(H)}{f} [1-p_c \frac{d(H)}{L-1}] (1-p_m)^{o(H)}$$

Clearly, the schema theorem understates the number of expected representatives; it does not account for influx, which at times can be substantial (Goldberg 1989a).

The second failure mode has been investigated by Goldberg (1988b, 1989b), Goldberg and Bridges (1988), Bethke (1980) and Holland (1989). Goldberg (1989a) constructed a minimal deceptive problem (MDP) and solved a system of difference equations to show (somewhat unexpectedly) that the genetic algorithm was able to solve the problem except in certain configurations. Goldberg and Bridges (1988) considered idealized reordering operators on the minimal deceptive problem; their emphasis was on the prevention of schemata disruption. The Goldberg (1988b and 1989b) and Bethke (1980) studies of deceptiveness used Walsh transforms. (Walsh transforms can be viewed as the group characters to the domain of the function, the diadic group). Bethke's primary results were first, that functions whose k th derivatives are appropriately bounded by certain Walsh transforms would not result in deceptive schemata. Second, he was able to specify a construction that would result in partially deceptive schemata. Goldberg (1989b) constructed a function defined on bit strings of length three that is fully deceptive. He also showed that for linear functions, favorable schemata would not be disrupted by crossover. Bethke's (1980) and Goldberg's (1988b and 1989b) studies could be characterized as static analyses; analyses that do not reflect the changing genetic populations, and hence, do not address the first failure mode. Holland (1989) has introduced a dynamic counterpart to the Walsh transform to study deceptiveness as it evolves.

In contrast, the analysis of this study does not use Walsh transforms. Instead the analysis is done in the original space, rather than the transform space.

Although some of the results in Goldberg (1988a) could be extended to shed light on this issue, the first failure mode has gone virtually unstudied. In addition, representational changes provide no assistance in addressing this failure mode. One of the few potential recourses in this case is to increase the population size.

On the other hand, representational changes can help address both the second and third failure modes. Historically, representational changes (through inversion) were entertained primarily to help prevent disruption due to crossover. Early studies of inversion were done by Bagley (1967), Cavicchio (1970) and Frantz (1972). Holland (1975) discusses inversion as a basic genetic operator. More recently, Whitley (1987) reported encouraging results with the use of the inversion operator. Shaefer (1985 and 1987) has broken with tradition and has incorporated central dynamic control of embedding through change of resolution, centering and windowing. The techniques for representational change advocated in this paper are thought to be new and the emphasis on representational change to transform hard functions into easy functions is also thought to be new.

4. OPTIMIZATION DOMAIN

In spite of genetic algorithm success with a variety of problems, understanding of their success and limitations remains primarily at the empirical level. A failure to clearly delineate and separate distinct problem components may be partially to blame for the slow theoretical progress in this area. Conventionally, genetic algorithms operate directly on functions defined on the diadic group, the n -fold Cartesian product of the integers modulo two. Empirically, it has been found that genetic algorithms operate most successfully on bit strings of modest length, say no longer than length 30-50. Rarely, however, is the original function of interest defined on this space. Instead, the original function is defined on some (subset of) Cartesian product of the reals, or on a discrete space. It becomes necessary to select a sample of points from the original domain and map them into the diadic group.¹ This is illustrated in Figure 1 where the image of the embedding d is the sample set of points selected, and the map i is the representation of the sample set in the diadic group. Generally, the map d is injective and i is bijective. Let f be the original function to be optimized, f_d its restriction to D , and f_i its representation in $\prod\{0,1\}$. These are defined so that the commutativity relations

$$f(d(x)) = f_d(x) = f_i(i(x))$$

obtain. Let f^* be the optimum of the original function f on the domain S . The objective of genetic optimization is to find a near optimum f_i^* to the function f_i defined on $\prod\{0,1\}$ such that f^* and f_i^* differ ($|f^* - f_i^*|$) only by an acceptable amount.

¹A function defined on a domain of interest with cardinality not a power of 2, can be extended to a domain of the required cardinality. Set $f(x)=L$, where L is some suboptimal value.

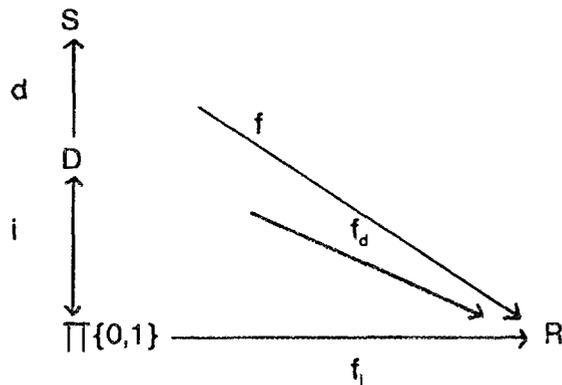


Figure 1. Explicit Embedding and Representation of a Function to be Optimized by Genetic Methods

For practical purposes, the original function domain can be considered to be one of the following six types:

1. R^n for n large, say $n > 50$.
2. A compact subset of R^n for n large.
3. R^n for n small, say $n < 50$.
4. A compact subset of R^n for n small.
5. A discrete set G with large cardinality, say $|G| > 2^{50}$.
6. A discrete set G with small cardinality.

Unless the dimensionality can be reduced, cases 1 and 2 cannot be handled by conventional genetic algorithms on present day serial computing machines; the bit string representations of points are overly long. Genetic approaches to problems on these domains almost certainly will need to be based on variants of some of the real valued genetic operators such as "creep" developed by Davis and Coombs (1987a,b) and Coombs and Davis (1987). Without certain smoothness assumptions or bounding of the location of the optimum (optima), case 3. is

intractable also. For purposes of this characterization, the domain of case 4. can be considered to be a compact subset of the reals. Shaefer (1985 and 1987) has developed adaptive embeddings that dynamically modify centering and resolution for compact subsets of the reals. Case 5. is an extremely difficult case for which the author knows of no relevant results or research. Case 6. is the fundamental case that needs to be understood. In this case the domain S can be identified with the set D (the embedding d is trivially a bijection), and the issue is to determine an appropriate representation i that leads to a genetic solution.

5. EXISTENCE RESULTS

Presented in this section are examples that essentially show the following: For a discrete space D (of small cardinality) there exist functions that cannot be (genetically) optimized in any representation. Other functions can be optimized in some representations and not others. Finally, a construction is given that suggests that for any representation i , a function f can be constructed such that its representation f_i cannot be optimized, but some other representation f_j can. These results underscore the need to choose a representation specific to the function being optimized.

Because of the stochastic nature of the genetic algorithm and many possible variants of its internal mechanisms, precise statements addressing convergence must explicitly specify the mechanisms and be given in probabilistic terms. For purposes of this exposition, such rigor will be foregone, and convergence will be loosely defined to mean "convergence most of the time with reasonable amount of work" (reasonable population sizes and a reasonable number of generations).

Example 1. Consider a function f which is identically constant at all points except one, which is the optimum. For sufficiently large cardinality spaces, and any representation, the function f cannot be optimized in a polynomial number (in the string length L) of genetic operations (crossover and mutation). This observation is intuitively plausible since the maximum number of distinct strings that will be observed after polynomially many genetic operations remains polynomial, and the total number of strings is exponential. (Any search of such a function is essentially a random search.) The observation is empirically true whenever the bit strings are of nontrivial length. (This example can be generalized. The weakest

generalization is that a function f is not genetically optimizable if it is constant off a "small" set, where "small" is a function of population size and string length.)

This example also serves to highlight the difference between estimable information, and perfect information. The function f with a single spike at the origin has identical, positive Walsh coefficients at all points. For competitive schemata, positive Walsh coefficients indicate that schemata with zeros have higher utility than their competitors (Bethke, 1980). Thus, f is a "theoretically easy" function, but one for which empirical estimates of schema utility generally fail to approximate the theoretical utility. Unless explicitly stated otherwise, the remainder of this paper implicitly assumes that genetic sampling produces adequate estimates.

Example 2. Consider the minimal deceptive function (MDP), type II (Goldberg, 1989a). The simplest version of this problem is a two bit problem: $f(00)=0.9$, $f(01)=0.8$, $f(10)=0.6$, and $f(11)=1.0$. For certain initial populations $\text{pop}(0)$ (with the four strings unequally represented, but each represented with some positive frequency), the optimum (11) is unstable in the sense that the probability that the string (11) is represented in future populations goes to zero as the population index j goes to infinity. In other words, for sufficiently large j , the probability that the string (11) will be represented in $\text{pop}(j)$ is arbitrarily small. In fact, empirical studies suggest that this loss of the optimal string occurs as early as the tenth generation (Goldberg 1989a). However, a simple affine transformation T will transform the MDP type II into a stable problem: Let M be a 2×2 zero-one invertible matrix with determinant minus one defined by

$$M = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Define T by $T(x) = [(11) + Mx]$, all operations modulo 2. Let $g(x) = f(T(x))$. Then $g(\cdot)$ can be determined to be given as $g(00) = 1$, $g(01) = 0.9$, $g(10) = 0.8$, and $g(11) = 0.6$. (For example, $g(01) = f([(11) + M(01)]) = f([(11) + (11)]) = f(00) = 0.9$.)

Although this problem has been explicitly discussed from the perspective of instability, an extension of the analysis to schemata suggests that affine transformations can transform certain non-optimizable functions into optimizable functions.

Example 3. A Hamming cliff problem is one for which the optimum (optima) is (are) approached from one side by an increasing number of ones, and from the other side by a decreasing number of ones. Formally, let the distance measure $|\cdot|_H$ be the Hamming distance. For a Hamming cliff problem there exist two near optima x_1 and x_2 such that $|x_1 - x_2|_H$ is large. An illustration is the function f defined by $f(000) = f(001) = f(010) = f(101) = f(110) = f(111) = 0$, and $f(011) = f(100) = 1$. Hamming cliff problems often seem to be optimizable by genetic algorithms, but the populations are frequently unstable insofar as the relative frequency of any string varies appreciably across the populations and that crossing two "complementary" optima yields nonoptimal points (even in later populations, unless genetic drift has taken place). Again, a simple transformation serves to transform the illustrated order three Hamming cliff problem into a stable problem. Let M be a zero-one, 3×3 matrix with determinant one defined by

$$M = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

Then the function g defined by $g(x) = f(M(x))$ (all operations modulo 2) is stable and can be seen to be defined by $g(011) = g(111) = 1$, $g(x) = 0$ otherwise.

To this point, it has been shown that certain functions cannot be optimized under any representation, that affine transformations can transform certain unstable problems into stable problems. The suggestion has been made that these same affine transformations suffice to transform certain nonoptimizable functions into optimizable ones. The following example will lead to a proof that no representation is adequate for all optimizable functions. That is, for every representation i , there exists a function f_a and a representation j such that f_i is not genetically optimizable, but f_j is, where $f_i(i(x)) = f_j(j(x)) = f_a(x)$.

Example 4. The function f defined in Table 1 below is a fully deceptive function (Goldberg, 1989b) in the sense that schemata (of order one or two) containing the point (000) have higher utility than their competitors, but (111) is the optimal. (A formal definition of fully deceptive is given in definition 1, below.)

j	f(j)
000	13
001	11
010	7
011	-15
100	-1
101	-15
110	-15
111	15

Table 1. A Fully Deceptive Function

Let M be the zero-one, 3x3 matrix with determinant one defined by

$$M = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

Then the function g defined by $g(x) = f [(111) + Mx]$ can be shown to generate Table 2, and is (strictly) fully easy in the sense of definition 1. below.

j	$g(j)$
000	15
001	13
010	11
011	-15
100	7
101	-15
110	-1
111	-15

Table 2. A Fully Easy Function (Transformed from a Fully Deceptive One)

Definition 1. In accordance with Goldberg (1989b) define a function f of binary bit strings of length L to be fully deceptive if and only if the following conditions obtain. The function has a single optimum x^* whose complement is x^c , (for example, the complement of 001 is 110). Let s_1 and s_2 be two competing schemata in the sense that s_1 and s_2 have the same fixed bits and differ in the value of some (one or more) fixed bit. Assume that $0 < o(s_1) = o(s_2) < L$ and that x^c is an element of s_1 , but not s_2 . Then the utility $u(s_1)$ of s_1 exceeds the utility $u(s_2)$ of s_2 . Define a function to be strictly fully deceptive if and only if the following conditions obtain. Let s_1 and s_2 be two competing schemata with $0 < o(S_1) = o(S_2) < L$. Let J index the positions in which the schemata differ and let $s_{2i} = x^*_i$ for $i \in J$. Then $u(s_1) > u(s_2)$. Replace x^c by x^* to define fully easy and strictly fully easy.

These definitions allow the formalization of the results of Example 4: Functions exist which are fully deceptive in one representation and fully easy in another. This is precisely the content of Lemma 1., below.

Lemma 1. Consider a diadic group of order $8=2^3$. For any representation i , there exists a function f_i and a transformation T such that f_i is fully deceptive, but f_i defined by $f_i(x) = f_i(T(x))$ is fully easy.

Proof. Let functions f_i and f_j be the fully deceptive and fully easy functions f and g , respectively, of Example 4.

In general, fully deceptive functions are easy to construct $L \geq 3$. Let the order $O(x)$ of a bit string x be the number of "1's" in the string. Define a function f as follows:

$$f(x) = \begin{cases} 1 & \text{if } o(x) = L \\ 1 - 1/2L & \text{if } O(x) = 0 \text{ and} \\ (n-1)/L & \text{if } O(x) = L-n, n = 0, L. \end{cases}$$

It is easy to see that f is fully deceptive. Let S_1 and S_2 be two competing schemata with $0 < (S_1) = O(S_2) < L$. Let $(0...0)$ be an exemplar of S_1 . If $(1...1)$ is not an exemplar of S_2 , then clearly $u(S_1) > u(S_2)$. Otherwise, there are distinct instantiations t_1, t_2 (assignment of distinct values to the "don't cares") such that $t_1(S_1) = (0...)$ and $t_2(S_2) = (1...1)$. By construction, it follows that $f(t_1(S_1)) + f(t_2(S_1)) > f(t_1(S_2)) + f(t_2(S_2))$.

At this time, it is not known whether strictly fully deceptive functions exist. They do not exist for $L < 4$. Rather than pursue these details further, it is instructive to view functions as polynomials. For any function f on the diadic group, the coefficients of the corresponding polynomial can be recursively calculated. For example, let $x = (x_1, x_2, x_3)$ be an arbitrary binary string of length three. Then the function f of Example 4. can be written as

$$f(x) = 13 - 2x_1 - 6x_2 - 14x_3 - 20x_1x_2 - 12x_1x_3 - 8x_2x_3 + 64x_1x_2x_3.$$

Its transform g becomes

$$g(x) = 15 - 2x_1 - 4x_2 - 8x_3 - 16x_1x_2 - 16x_1x_3 - 2x_2x_3 + 18x_1x_2x_3.$$

It is not hard to see why the function f is deceptive; all coefficients other than the last "point to" fewer zeros in the solution rather than more. The perspective of functions as polynomials makes it easy to specify "nearly deceptive" functions.

Definition 2. Define a function f given on the diadic group G of binary bit strings of length L to be k -point (strictly) fully deceptive if and only if there exists a fixed set S containing k points of G such that the function f is (strictly) fully deceptive whenever the calculation of the schemata utilities omits the k designated points. Define k -point (strictly) fully easy similarly.

Example 5. The function f defined in (1) below is 1-point strictly fully deceptive. The optimum for this function is at $(1\dots 1)$ and the single point $(1\dots 1)$ is the designated point relative to the calculation of the schemata utilities and 1-point deceptiveness.

$$f(x) = 0 - x_1 - \dots - x_L - x_1x_2 - \dots - x_{L-1}x_L - \dots - x_2x_3 \dots x_L + 2^L x_1 \dots x_L$$

To see this, let s_1 and s_2 be two competing schemata of the same order, $0 < o(s_1) = o(s_2) < L$, with the same fixed positions. For any fixed position, assume that schema s_1 has a "0" in that position whenever s_2 does, s_1 may have a "1" in that position only if schema s_2 does, and $s_1 \neq s_2$. Then (modulo the 1-point condition), schema s_1 has a higher (theoretical) utility than schema s_2 . This follows because the determination of the utility of schema s_2 includes more negative terms than the corresponding calculation for s_1 .

Clearly, the difference between fully deceptive and 1-point fully deceptive are minimal. In the first case, the optimum is thought to be unstable in the sense of Example 2. of this paper. Stability for 1-point fully deceptive functions has not been investigated, but they are less likely to be unstable.

Again, a simple transformation T serves to transform this 1-point fully deceptive function to a fully easy function. Set $T(0\dots 0)=(1\dots 1)$, $T(0\dots 01)=(0\dots 0)$, $T(1\dots 1)=(0\dots 01)$, with all other points invariant. The function g defined by $g(x)=f(T(x))$ is the following polynomial

$$g(x)=2-2x_1-x_2x_3-\dots-x_L -\dots-x_2x_3\dots x_L+(2^L-4)x_1\dots x_L.$$

A corresponding affine transformation (based on a zero-one matrix with determinant equal to 1) that maps $(0\dots 0)$ to $(1\dots 1)$ and $(0\dots 1)$ to $(0\dots 0)$ can be specified by setting

$$M = \begin{pmatrix} 1 & 0 & & 1 \\ 0 & 1 & 0 & 1 \\ & & & 1 \\ 0 & 0 & & 1 \end{pmatrix}$$

and defining $T(x)=(1\dots 1) + Mx$.

Example 5. is sufficiently general to prove Lemma 2. To generate a 1-point (strictly) fully deceptive function, set the constant term equal to zero, and all but the last coefficient to negative one. For strings of length L , set the last coefficient to 2^L . To transform the 1-point (strictly) fully deceptive function to a (strictly) fully easy function, invoke the transformation specified in Example 5.

Lemma 2. For a diadic group of order 2^L , $L > 2$ and any representation i , there exists a function f_i and a transformation T such that f_i is 1-point fully deceptive, but f_i defined by $f_i(x) = f_i(T(x))$ is fully easy.

The primary significance of Examples 1-5 and Lemmas 1-2 is first, that there exist classes of functions that cannot be optimized regardless of representation, and more importantly, that no one fixed representation is best for all problems; representations should be suitably chosen to match the corresponding functions to be optimized. How to choose an appropriate representation is the topic of the remainder of the paper.

6. POLYNOMIALS

To motivate techniques for representational change, it is once more instructive to look at polynomial functions. Rather informally, certain of these are grouped in Table 3. as "easy" or "hard."

Easy:

linear:	$a_0 + a_1x_1 + a_2x_2 + \dots + a_Lx_L$
monotone:	$13 - 2x_1 - 6x_2 - 14x_3 - 20x_1x_2 - 12x_1x_3 - 8x_2x_3 - 64x_1x_2x_3$
hard:	
delta:	$x_1x_2 \dots x_L$
fully deceptive:	$13 - 2x_1 - 6x_2 - 14x_3 - 20x_1x_2 - 12x_1x_3 - 8x_2x_3 + 64x_1x_2x_3$

Table 3. A Selection of Easy and Hard Functions

Linear polynomials are defined to be those with no cross terms. Monotone polynomials have all coefficients (other than constant) the same sign. In Lemma 3., linear polynomials are proved to be semi-fully easy. Lemma 4. provides the corresponding proof for monotone polynomials. Semi-fully easy is defined in the same way as fully easy, but with multiple optima allowed and strict dominance between competing pairs of schemata replaced by dominance (equality allowed).

Lemma 3. All linear polynomials are semi-fully easy. Proof. Without loss of generality, assume that the function is to be maximized, and the terms are ordered so that the first m coefficients are positive, the next n are negative, and the final $L-(m+n)$ are zero. Thus, the optima occur at $(1 \dots 1 \dots 0 \dots 0 \dots \# \dots \#)$. Let s_1 and s_2 be competing schemata with entries s_{1j} and with the same fixed positions j such that $s_{2j} = 1$ implies $s_{1j} = 1$ for the fixed positions $j \leq m$ and s_{2j}

$= 0$ implies $s_{1j} = 0$ for the fixed positions $m < j \leq n$. Moreover, assume that $s_{1j} \neq s_{2j}$ for some j such that $1 \leq j \leq m+n$. Then $u(s_1) > u(s_2)$: The instantiation of the "don't care" positions determines a one-to-one correspondence between strings x that match s_1 and strings y that match s_2 . For every such pair, $f(x) > f(y)$. In the case that $s_{1j} = s_{2j}$ for all $j \leq m+n$, $u(s_1) = u(s_2)$.

Lemma 4. All monotone polynomials are semi-fully easy. Proof. Without loss of generality, assume that all coefficients are non-negative, and every x_j participates in some polynomial term $x_1 \dots x_j \dots x_m$ with non-zero coefficient. Thus, the optimum is at $x = (1 \dots 1)$. Let s_1 and s_2 be competing schemata with entries s_{1j} and with the same fixed positions j such that $s_{2j} = 1$ implies $s_{1j} = 1$, and $s_1 \neq s_2$. Then $u(s_1) \geq u(s_2)$. Proof. Every term that appears in the calculation of the utility of s_2 appears also in the calculation of the utility of s_1 .

There exists at least one more class of polynomials that are trivially semi-fully easy, the class of linearly dominated polynomials.

Definition 3. For every variable x_j , let J be the set of (non-monomial) cross terms that include x_j as a factor. Let a_j be the coefficient associated with the singleton term x_j . By abuse of notation, let $\{a_i\}$ for $i \in J$ be the coefficients of the terms in J . Then the polynomial f is linearly dominated if and only if $|a_j| > |\sum_{i \in J} a_i|$.

Lemma 5. All linearly dominated polynomials are semi-fully easy.

Proof. Follows immediately from the semi-fully easiness of linear polynomials.

As can be seen from the transform g of the Goldberg function of Example 4., the linear, monotone, and linearly dominated polynomials do not exhaust the class of easy functions. Discouragingly, easy functions are not always trivial to recognize. Conversely, hard functions include at least those that resist accurate estimation of schemata utilities (delta or "needle-in-the-haystack" functions) as well as the fully deceptive, strictly fully deceptive, and 1-point fully

deceptive functions. Moreover, hard functions can differ only negligibly from easy functions; the Goldberg fully deceptive function differs from a monotone function in only one term -- a delta function. (It is known that this last observation is not generally true.) These observations should temper, but not extinguish hopes of developing useful techniques to adaptively select suitable function representations. Fully deceptive, strictly fully deceptive, and 1-point fully deceptive functions will often remain resistant to optimization, but these are unlikely to be the only difficult functions of interest.

7. CHANGE STRATEGIES

Since change of representation has historically yielded inconclusive results, why should the issue be revisited at this time? To answer the question, it is useful to more carefully consider previous implementations. Three factors are particularly pivotal. First, the class of transformations previously used consisted of permutations of the basis elements. (The central concern of earlier investigators was the third failure mode, disruption of schemata by crossover.) This corresponds to the subclass of affine transformations with zero translation and a zero-one matrix with precisely a single one in each row and column (permutation matrices). This class of transformations is insufficiently powerful to resolve the difficult problems discussed in this paper. Second, genetic populations generally were formed of individuals each expressed in their specific representation. With such a mixture, it is unlikely that the genetic algorithm can properly estimate schemata utilities. Third, representations underwent continual dynamic change, again confounding estimates of schema utility. The methods suggested here use a larger class of transformations. The first method uses either the affine transformations or permutations introduced earlier in this paper and would be implemented through central control; that is, there would be a nongenetic mechanism to determine change of representation. The second method is particularly suitable for implementation on a parallel processor, and would be implemented by a genetic algorithm at a meta-level. These two methods are complementary. The first could be used to identify a favorable starting representation. The latter could dynamically change representations.

To be implementable in practice, representational changes should be suggested by the currently available information about the function. Insofar as linear functions are strictly fully

easy, a plausible first heuristic for guiding change of representation is to select representations to linearize functions as much as possible.

Assume that the function f is to be maximized. Any function f can be written as a sum $f=f_L + f_N$ of linear and nonlinear terms, f_L and f_N , respectively, defined as follows. Consider bit strings of length L . Formally, let x_i be the bit string with a one in the i th position and zeros elsewhere. Every bit string can be uniquely written as a formal vector sum of distinct "basis" x_i 's, $x = \sum x_i$. For any bit string x , define $f_L(x)$ as $f(0\dots 0) + f(x)$ where the sum is over the unique set of basis elements determining x . Define $f_N(x) = f(x) - f_L(x)$. Given a current genetic population P , augment it by the basis elements and zero. Call this augmented set P' . One way to specify a representational change is to choose that transformation (from some specified class of transformations) that minimizes the nonlinear function component summed over the transformation of the elements of P' :

$$\text{minimize}_T \sum_{P'} f_N(Tx)$$

Two possible simple classes of transformations are the affine transformations and permutations of P' . It is not hard to show that an affine transformation can be written as the sum of a translation and multiplication by a zero-one matrix M with determinant of odd parity, $T(x)=y+Mx$. On the one hand, affine transformations are preferable to permutations first, because of their succinct representation, and second, because they affect individuals outside of P' . However, at this time, the second alternative seems more attractive since there is a straightforward heuristic that allows for the easy solution of the minimization.

Let P' be the current population augmented by the origin x_0 and the basis elements $\{x_i\}$, $i=1,\dots,L$. Without loss of generality, assume that the function f is to be maximized. Order the elements of P' in decreasing magnitude of the function values $f(x)$, say the order is

$f(y_1) \geq f(y_2) \geq \dots$ Order the basis elements x_i , $i=0, \dots, L$ and the origin x_0 so that x_0 is first and the remainder are in some arbitrary order, say x_1, x_2, \dots . If $f(y_1) > f(x_0)$, map y_1 to x_0 . Otherwise, if $f(x_1) < f(y_1) \leq f(x_2)$, and x_1 is the first basis element in the ordering that follows x_0 and has not been mapped into, then map y_1 to x_1 . Continue in this fashion to map elements y_i into basis elements as long as possible. Call this map T^{-1} . Define the transformation T as follows: If $x = T^{-1}y$, set $Tx = y$. If no element is mapped onto x by the transformation T^{-1} , complete the permutation T on P by extending it randomly (consistent with previous assignments).

The transformation T is the required permutation that minimizes the nonlinear component f_N summed over P' . This is easy to see because any permutation of P' leaves the sum

$$\sum_{P'} f(x) = \sum_{P'} f(T(x))$$

invariant, and (modulo a re-ordering of the basis elements) the construction heuristic construction maximizes

$$\sum_{P'} f_L(Tx).$$

Whether or not this change of representation aids in function optimization depends on the specific function and the evolving genetic populations. For the Goldberg fully deceptive function or the 1-point strictly fully deceptive problem illustrated earlier, no (practical) change of representation results in improvement unless the optimum is already present in the

population. More generally, whenever the current population is a (subset) of the zero and basis elements, this representational transformation will yield no change.

However, on other problems, such as the Hamming Cliff Problem illustrated in Table 4., the transformation is likely to produce improvements.

j	f(j)
000	0
001	0
010	1
011	2
100	2
101	1
110	0
111	0

Table 4. A Hamming Cliff Problem

For example, if the current population is $P = \{(011), (010), (110)\}$ then one choice for the transformation T is specified in Table 5. below, and the transformed function $g(x) = f(T(x))$ is given in Table 6.

	$\xrightarrow{T^{-1}}$	
P	$\left\{ \begin{array}{l} 001 \\ 010 \\ 110 \end{array} \right.$	000
		010
		110
		000
		011
		001
		100

Table 5. One Choice for the Transformation T

j	g(j)
000	2
001	0
010	1
011	0
100	2
101	1
110	0
111	0

Table 6. Typical Transformation of the Hamming Cliff Problem of Table 3.

The second method of representational change uses the genetic algorithm at two levels, one to search for representations, and one to optimize the function in question. This method is particularly suitable for implementation on multiprocessor computers. The details remain to be investigated, but the basic concept is simple. For a given function f , the suitability of a representation i can be evaluated as the expected population average fitness after some number of genetic generations. The expectation should be taken with regard to different random initial populations and random seeds. An estimate of the fitness of the representation might simply be the average population fitness for the one fixed initialization and randomization.

Choose a base representation and initial population. For each node of the multiprocessor, select (randomly or otherwise) a transformation and use the genetic algorithm to search for function optima under that transformation. After some number of generations, determine the fitness of the transformation in terms of the corresponding population fitness. Use these representation fitness estimates to drive a genetic search over transformations.

A hidden issue is how to apply crossover to transformations. Clearly, affine transformations can be concatenated into bit strings. However, ensuring that the crossover produces a matrix

with determinant of odd parity appears to be less than straightforward. An alternative would be to represent transformations as permutations. In this approach, consider each bit string to be a binary representation of an integer. Any transformation could be coded as the corresponding permutation of integers. Thus, the fundamental units of search at the representational level would be strings of transpositions, and crossover would be done with regard to these (variable length) strings. These concepts are illustrated in Tables 7 and 8. below.

bit string transformation

54:	(00110110)	T	(11100011)	:227
1:	(00000001)	→	(00100000)	: 32
32:	(00100000)		(00110110)	: 54
222:	(11100011)		(00000001)	: 1

product of transpositions

(54,227)(54,1)(54,32)

Table 7. A Transformation as a Product of Transpositions

parents	(451,46)(46,322)(322,67)		(67,81)
	(1,14)(14,6)		
children	(451,46)(46,322)(322,67)		
	(1,14)(14,6)(67,81)		

Table 8. Crossover of Transformations

8. SUMMARY

The roles of embedding and representation in genetic optimization have been made explicit. Three modes of genetic algorithm failure have been discussed: unreliable estimates of schemata utilities, misleading schemata utilities, and disruption due to crossover. Easy and hard functions have been constructed, and it has been shown that for any representation, there exists a function and a transformation such that the function is difficult in the original representation, and easy in the transformed. Possible reasons for the inconclusive results of previous investigations into representational change have been discussed, and several new approaches to dynamic representational change have been introduced.

9. REFERENCES

- Amarel, S. (1968). "On Representations of Reasoning about Actions," Machine Intelligence, Vol. 3.
- Axelrod, R. (1980a). "Effective Choice in the Prisoner's Dilemma," Journal of Conflict Resolution 24, 3-25.
- Axelrod, R. (1980b). "More Effective Choice in the Prisoner's Dilemma," Journal of Conflict Resolution 24, 379-403.
- Axelrod, R. (1987). "The Evolution of Strategies in the Iterated Prisoner's Dilemma," in L. D. Davis (ed), Genetic Algorithms and Simulated Annealing, Pitman, London, and Morgan Kaufmann Publishers, Inc., 32-41.
- Bagley, J. D. (1967). "The Behavior of Adaptive Systems Which Employ Genetic and Correlational Algorithms," Ph.D. Thesis, University of Michigan, Dissertation Abstracts International, 28 (12), 5106B. (University Microfilms No. 68-7556).
- Baker, J. E. (1987). "Reducing Bias and Inefficiency in the Selection Algorithm," in J. J. Grefenstette (ed), Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, 14-21.
- Baum, E., and F. Wilczek. (1988). "Supervised Learning of Probability Distributions by Neural Nets," Jet Propulsion Laboratory, Pasadena, Calif. (preprint).
- Bethke, J. D. (1980). "Genetic Algorithms as Function Optimizers," Ph.D. Thesis, Department of Computer and Communication Sciences, University of Michigan.
- Bledsoe, W. W. (1961). "The Use of Biological Concepts in the Analytical Study of Systems." Paper presented at ORSA-TIMS National Meeting, San Francisco.
- Brent, R. P. (1971). "Algorithms for Finding Zeros and Extrema of Functions Without Calculating Derivatives," Ph.D. Thesis, Computer Science Department, Stanford University.
- Cavicchio, D. J. (1970). "Adaptive Search Using Simulated Evolution." Ph.D. Thesis, University of Michigan, Ann Arbor.
- Cohon, J. P. and W. D. Paris. (1987). "Genetic Placement," IEEE Transactions on Computer - Aided Design of Integrated Circuits and Systems.
- Coombs, S. and L. Davis (1987). "Genetic Algorithms and Communication Link Speed Design: Constraints and Operators," in J. J. Grefenstette (ed), Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, 257-260.

- Davis, L. and S. Coombs (1987a). "Genetic Algorithms and Communication Link Speed Design: Theoretical Considerations," in J. J. Grefenstette (ed), Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, 252-256.
- Davis, L. and S. Coombs, (1987b). "Optimizing Network Link Sizes with Genetic Algorithms," Conference on Computer Simulation and Modeling, Tucson, Ariz.
- DeJong, K. A. (1975). "Analysis of the Behavior of a Class of Genetic Adaptive Systems," Ph.D. Thesis, Department of Computer and Communication Sciences, University of Michigan.
- Dietterich, T. G. and R. S. Michalski. (1983). "A Comparative Review of Selected Methods for Learning from Examples," in R. S. Michalski, Carbonell, and Mitchell (eds.), Machine Learning, Palo Alto, Calif. Tioga.
- Fletcher, R. and Powell, M. J. P. (1963). "A Rapidly Convergent Descent Method for Minimization," The Computer Journal, 7:149.
- Fletcher, R. (1970). "A New Approach to Variable Metric Algorithms." The Computer Journal, 13:317.
- Frantz, D. R. (1972). "Non-Linearities in Genetic Adaptive Search," Ph.D. Thesis, University of Michigan, Ann Arbor. Dissertation Abstracts International, 33(11), 5240B-5241B. (University Microfilms No. 73-11 1116).
- Goldberg, D. E. (1983). "Computer - Aided Gas Pipeline Operation Using Genetic Algorithms and Rule Learning," Ph.D. Thesis reprinted in 1984 as Technical Publication BER-84001SP by the College of Engineering, University of Alabama.
- Goldberg, D. E. (1987a). "Computer - Aided Pipeline Operation Using Genetic Algorithms and Rule Learning. Part I: Genetic Algorithms in Pipeline Optimization," Engineering with Computers 3, 35-45.
- Goldberg, D. E. (1987b). "Computer - Aided Pipeline Operation Using Genetic Algorithms and Rule Learning. Part II: Rule Learning Control of a Pipeline Under Normal and Abnormal Conditions," Engineering with Computers 3, 47-58.
- Goldberg, D. E. (1988a). Sizing Populations for Serial and Parallel Genetic Algorithms, The Clearinghouse for Genetic Algorithms," TCGA Report No. 88004.
- Goldberg, D. E. (1988b). Genetic Algorithms and Walsh Transforms: Part 1, A Gentle Introduction, The Clearinghouse for Genetic Algorithms, TCGA Report No. 88006.
- Goldberg, D. E. (1989a). Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley.

- Goldberg, D. E. (1989b). Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis, The Clearinghouse for Genetic Algorithms, TCGA Report No. 89001.
- Goldberg, D. E. and C. L. Bridges. (1988). An Analysis of a Reordering Operator on a GA-Hard Problem, The Clearinghouse for Genetic Algorithms, TCGA Report No. 88005.
- Gordon, M. (1988). "Probabilistic and Genetic Algorithms for Document Retrieval," Communications of the ACM, Vol. 31 (10), 1208-1218.
- Grefenstette, J. J. (1984). A User's Guide to Genesis, Tech Report CS-84-11, Computer Science Department, Vanderbilt University, Nashville, Tennessee.
- Grefenstette, J. J. and J. M. Fitzpatrick (1985). "Genetic Search with Approximate Function Evaluations," in Grefenstette (ed), Proceedings of an International Conference on Genetic Algorithms and Their Applications, 160-168.
- Grefenstette, J. J. (1986). "Optimization of Control Parameters for Genetic Algorithms." IEEE Transactions on Systems, Man, and Cybernetics, SMC-16(1), 122-128.
- Hilliard, M. R., G. E. Liepins, M. Palmer, and G. Rangarajan. (1988). "The Computer as a Partner in Algorithmic Design: Automated Discovery of Parameters for a Multi-Objective Scheduling Heuristic," Proceedings of Impact of Recent Computer Advances on Operations Research, Williamsburg, Virginia.
- Holland, J. H. (1975). Adaption in Natural and Artificial Systems. Ann Arbor: The University of Michigan Press.
- Holland, J. H. (1989). "Searching Nonlinear Functions for High Values." Applied Mathematics and Computation (to appear).
- Huang, H. Y. (1970). "Unified Approach to Quadratically Convergent Algorithms for Function Minimization," Journal of Optimization Theory and Application, 5:405.
- Korf, R. (1980). "Towards a Model of Representation Changes," AI Journal, Vol. 14, 41-78.
- Kuchinski, M. J. (1985). Battle Management Systems Control Rule Optimization Using Artificial Intelligence (Technical Report No. NSWC MP 84-329). Dahlgren, Virg., Naval Surface Weapons Center.
- Liepins, G. E., Mr. R. Hilliard, J. Richardson and M. Palmer. (1989). "Genetic Algorithm Applications to Set Covering and Traveling Salesman Problems," to appear in OR/AI: The Integration Problem Solving Strategies.
- Liepins, G. E. and M. R. Hilliard. (1989). "Genetic Algorithms: Foundations and Applications," to appear in Annals of Operations Research.
- Rich. E. (1983). Artificial Intelligence, McGraw-Hill.

Shaefer, C. G. (1985). "Directed Trees Method for Fitting a Potential Function," in J. J. Grefenstette (ed), Proceedings of an International Conference on Genetic Algorithms and Their Applications, Lawrence Erlbaum, Hillside, N. J., 207-225.

Shaefer, C. G. (1987). "The ARGOT Strategy: Adaptive Representation Genetic Optimizer Technique," in J. J. Grefenstette (ed), Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillside, N. J. 50-58.

Subramanian, D. (1989). "A Theory of Justified Reformulations," Ph.D. Thesis, Stanford University, Stanford, Calif.

Whitley. (1987). "Using Reproductive Evaluation to Improve Genetic Search and Heuristic Discovery," in Grefenstette (ed), Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillside, N. J.

INTERNAL DISTRIBUTION

- | | | | |
|------|-----------------|--------|----------------------------------|
| 1-5. | G. E. Liepins | 19. | V. Ng |
| 6-8. | A. M. Hutton | 20. | S. Purucker |
| 9. | V. D. Baxter | 21. | D. E. Reichle |
| 10. | D. Bjornschad | 22. | R. S. Solanki |
| 11. | C. S. Bowman | 23. | F. Southworth |
| 12. | J. Christian | 24. | B. E. Tonn |
| 13. | S. Das | 25. | ORNL Patent Office |
| 14. | P. F. Daugherty | 26. | Central Research Office |
| 15. | C. W. Hagan | 27. | Document Reference Section |
| 16. | M. R. Hilliard | 28-30. | Laboratory Records |
| 17. | E. Hillsman | 31. | Laboratory Records - Record Copy |
| 18. | R. B. Honea | | |

EXTERNAL DISTRIBUTION

32. Dr. Bruce G. Buchanan, Department of Computer Science, University of Pittsburgh, Room 318, Alumni Hall, Pittsburgh, PA 15260.
33. J. J. Cuttica, Vice President of Research and Development, Gas Research Institute, 8600 W. Bryn Mawr Avenue, Chicago, IL 60631.
34. D. E. Morrison, Professor of Sociology, Michigan State University, 201 Berkey Hall, East Lansing, MI 48824-1111.
35. R. L. Perrine, Professor, Engineering and Applied Sciences, Civil Engineering Department, Engineering I, Room 2066, University of California, Los Angeles, CA 90024.
36. Dr. Martin Williams, Professor, Department of Economics, Northern Illinois University, DeKalb, IL 60115.
37. Office of Assistant Manager for Energy Research and Development, DOE/ORO, P. O. Box 2001, Oak Ridge, TN 37831-8600.
- 38-48. Office of Scientific and Technical Information, U. S. Department of Energy, Post Office Box 62, Oak Ridge, TN 37831

