# Correspondence

## A Genetic Algorithm with Disruptive Selection

Ting Kuo and Shu-Yuen Hwang

***Abstract*—Genetic algorithms are a class of adaptive search techniques based on the principles of population genetics. The metaphor underlying genetic algorithms is that of natural evolution. Applying the "survival-of-the-fittest" principle, traditional genetic algorithms allocate more trials to above-average schemata. However, increasing the sampling rate of schemata that are above average does not guarantee convergence to a global optimum; the global optimum could be a relatively isolated peak or located in schemata that have large variance in performance. In this paper we propose a novel selection method, *disruptive selection*. This method adopts a *nonmonotonic* fitness function that is quite different from traditional *monotonic* fitness functions. Unlike traditional genetic algorithms, this method favors both superior and inferior individuals. Experimental results show that GA's using the proposed method easily find the optimal solution of a function that is hard for traditional GA's to optimize. We also present convergence analysis to estimate the occurrence ratio of the optima of a deceptive function after a certain number of generations of a genetic algorithm. Experimental results show that GA's using disruptive selection in some occasions find the optima more quickly and reliably than GA's using directional selection. These results suggest that disruptive selection can be useful in solving problems that have large variance within schemata and problems that are GA-deceptive.**

### NOMENCLATURE

| | |
|---|---|
| $\mathbf{abs(M)}$ | Absolute matrix of a matrix $\mathbf{M}$. |
| $d(H)$ | Defining length of hyperplane $H$ (the length of the smallest segment containing all the defining loci of $H$). |
| $\mathbf{diag\,(V)}$ | Diagonal matrix of a vector $\mathbf{V}$. |
| $\mathbf{O^g}$ | Occurrence matrix that represents the probability that each kind of string occurs at generation $g$. |
| $\mathbf{F}$ | Function vector that represents the function values for each kind of string. |
| $\overline{f}(t)$ | Average value of the objective function of the individuals in the population $P(t)$. |
| $H$ | Hyperplane of the search space. |
| $H(t)$ | Set of individuals that are in the population $P(t)$ and are instances of hyperplane $H$. |
| $H_i \leq_{D,t} H_j$ | $H_i$ is *dominated* by $H_j$ in the population $P(t)$. |
| $H_i \geq_{MWE,t} H_j$ | $H_i$ is more worth exploring than $H_j$ in the population $P(t)$. |
| $H_i \geq_{R,t} H_j$ | $H_i$ is more *remarkable* than $H_j$ in the population $P(t)$. |
| $\mathbf{I}$ | Identity column vector with dimension $(l+1)\times 1$. |
| $L$ | Length of the bit string. |
| $m[H(t)]$ | Number of individuals in the set $H(t)$. |
| $M[H(t)]$ | Expected number of individuals in the set $H(t)$. |
| $o(H)$ | Order of hyperplane $H$. |
| $p_c$ | Crossover rate. |
| $p_m$ | Mutation rate. |
| $P(t)$ | Population at time $t$. |
| $\mathbf{S^g}$ | Selection vector that represents theprobability that each kind of string is selected at generation $g$. |
| $\mathbf{T^{p_m}}$ | Transition matrix that represents the probability of being mutated under the mutation rate $p_m$. |
| $tsr(x, t)$ | Expected number of offspring reproduced by individual $x$ at time $t$. |
| $tsr[H(t)]$ | Growth rate of the set $H(t)$ without the effect of crossover and mutation. |
| $u(x)$ | Fitness of the individual $x$. |
| $\overline{u}[H(t)]$ | Average fitness of the individuals in the set $H(t)$. |
| $\overline{u}(t)$ | Average fitness of the individuals in the population $P(t)$. |

Manuscript received July 14, 1993; revised February 13, 1994, and January 19, 1995. This work was supported by the National Science Council, Republic of China, under Grant NSC 84-2213-E-009-012.

The authors are with the Institute of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu 30050, Taiwan, ROC.

Publisher Item Identifier S 1083-4419(96)02300-X.

### I. INTRODUCTION

Genetic algorithms (GA's) have been applied in many diverse areas, such as function optimization [7], the traveling salesman problem [11], [14], scheduling [5], [30], neural network design [19], [27], system identification [23], vision [4], control [22], and machine learning [8], [9], [16]. Goldberg's book [12] provides a detailed review of these applications. The fundamental theory of genetic algorithms was presented in Holland's pioneering book [20].

The principle behind genetic algorithms is essentially that of Darwinian natural selection. According to *Neo-Darwinism*, the process of evolution can be classified into three categories: *stabilizing selection*, *directional selection*, and *disruptive selection* [26]. Stabilizing selection is also called normalizing selection, since it tends to eliminate individuals with extreme values. Directional selection has the effect of either increasing or decreasing the mean value of the population. In contrast, disruptive selection tends to eliminate individuals with moderate values. We will give a formal definition of each of the three types of selection later. All traditional GA's can be viewed as a process of evolution based on directional selection. However, in some cases, a current worse solution may have a greater chance of "evolving" toward a better future solution. Why discard these worse solutions rather than trying to exploit them? This idea motivates our research.

The notation of *schema* must be introduced first so that we may understand how genetic algorithms can direct the search toward high fitness regions of the search space. A schema is a set of individuals in the search space. In most GA's, the individuals are represented by fixed-length binary strings. In the case of a binary string, a schema can be expressed as a pattern that is defined over the alphabet {0, 1, *} and describes a subset of strings with similarities at certain string positions. In the pattern of a schema, 1's and 0's are referred to as *defining bits*; the number of defining bits is called the *order* of that schema. The distance between the leftmost and rightmost defining bits is referred to as the *defining length* of a schema. For example, the order of **0*11**1 is 4, and its defining length is 6. A bit string $x$ is said to be an *instance* of a schema $s$ if $x$ has exactly the same

1083–4419/96$05.00 © 1996 IEEE

bit values in exactly the same locations that are defining bits in $s$. For example, 00011 and 00110 are both instances of schema 00*1*, but neither 10011 nor 00000 is an instance of schema 00*1*. Since every string of length $L$ is an instance of $2^L$ schemata, its fitness gives some information about those schemata. Thus, while explicitly sampling strings, the GA implicitly samples schemata. Schemata can be viewed as defining hyperplanes in the search space. Usually, the terms schema and hyperplane are used interchangeably.

Since GA's are not admissible (i.e., no guarantee for an optimal solution), it is important to reduce the possibility of some GA's missing the optimal solution. GA's may fail to locate the optima of a function for several reasons [25]:

1) The chosen embedding (i.e., choice of domain) may be inappropriate.
2) The problem is deceptive (i.e., tends to contain isolated optima: the best points tend to be surrounded by the worst) that is provably misleading for the simple three-operator genetic algorithms [12].
3) The problem is not deceptive, but average fitnesses of schemata cannot be reliably estimated because the sampling error is too large.
4) Average fitnesses of schemata can be reliably estimated, but crossover destroys individuals which represent schemata of high utility.

In this paper, we will propose a novel selection method and show that this new method can be helpful in the second and the third cases.

The paper is organized as follows. In Section II, we give a simple review of genetic algorithm. Different types of fitness functions are described in Section III. GA's that use disruptive selection to solve a nondeceptive but GA-hard function are described in Section IV. Then, in Section V, we derive a deterministic model for convergence analysis and demonstrate that GA's using disruptive selection are more reliable than GA's using directional selection in solving a deceptive function. Finally, in Section VI, we conclude the paper with a discussion of our results.

## II. Genetic Algorithm

### A. Background

One description of genetic algorithms is that they are iterative procedures maintaining a population of individuals that are candidate solutions to a specific problem. At each generation the individuals in the current population are rated for their effectiveness as solutions, and in line with these ratings, a new population of candidate solutions is formed using specific genetic operators [20].

The three primary genetic operators focused on by most researchers are selection, crossover, and mutation. These are described below.

1) *Selection (or Reproduction)*: The population of the next generation is first formed by using a probabilistic reproduction process. In general, there are two types of reproduction processes: generational reproduction and steady-state reproduction. Generational reproduction replaces the entire population with a new population. In contrast, steady-state [29], [31], reproduction replaces only a few individuals in a generation. Whichever type of reproduction is used, individuals with higher fitness usually have a greater chance of contributing to the generation of offspring. Several selection methods may be used to determine the fitness of an individual. Proportional selection [12], [20], and ranking [2] are the main selection schemes used in GA's. The resulting population is sometimes called the intermediate population. The intermediate population is then processed using crossover and mutation to form the next generation.
2) *Crossover*: A crossover operator manipulates a pair of individuals (called parents) to produce two new individuals (called offspring) by exchanging segments from the parents' coding. By exchanging information between two parents, the crossover operator provides a powerful exploration capability. A commonly used method for crossover is called one-point crossover. Assume that the individuals are represented as binary strings. In one-point crossover, a point, called the crossover point, is chosen at random and the segments to the right of this point are exchanged. For example, let $x_1 = 101010$ and $x_2 = 010100$, and suppose that the crossover point is between bits 4 and 5 (where the bits are numbered from left to right starting at 1). Then the offspring are $y_1 = 101000$ and $y_2 = 010110$. Several other types of crossover operators have been proposed, such as two-point crossover, multi-point crossover [7], uniform crossover [1], [29], and shuffle crossover [10].
3) *Mutation*: By modifying one or more of the gene values of an existing individual, mutation creates new individuals, increasing the variability of the population. The mutation operator ensures that the probability of reaching any point in the search space is never zero.

In this paper, we restrict our attention to the selection operator.

### B. Selection Strategies

The selection operator plays an important role in driving the search toward better individuals and maintaining high genotypic diversity in the population.

Grefenstette and Baker [17] noted that in selection strategies the selection phase can be divided into the selection algorithm and the sampling algorithm. The selection algorithm assigns to each individual $x$ a real number, called the *target sampling rate*, $tsr(x, t)$, to indicate the expected number of offspring reproduced by $x$ at time $t$. The sampling algorithm then reproduces, according to the target sampling rate, copies of individuals to form a new population. Most well-known selection algorithms use proportional selection, which can be described as

$$tsr(x, t) = \frac{u(x)}{\overline{u}(t)} \tag{1}$$

where $u$ is the fitness function and $\overline{u}(t)$ is the average fitness of the population $P(t)$. For each selection algorithm,

$$tsr[H(t)] = \sum_{x \in H(t)} \frac{tsr(x, t)}{m[H(t)]} \tag{2}$$

where $H$ is a hyperplane, $H(t)$ is the set of individuals that are in the population $P(t)$ and are instances of hyperplane $H$, $m[H(t)]$ is the number of individuals of the set $H(t)$, and $tsr[H(t)]$ is the growth rate of the set $H(t)$ without the effects of crossover and mutation.

Thus,

$$\begin{aligned} tsr[H(t)] &= \sum_{x \in H(t)} \frac{u(x)}{\overline{u}(t) m[H(t)]} \\ &= \frac{\overline{u}[H(t)]}{\overline{u}(t)} \end{aligned} \tag{3}$$

where $\overline{u}[H(t)]$ is the average fitness of the set $H(t)$. Several researchers have studied mechanisms that affect the selection bias. Grefenstette [15] showed the effect of different scaling mechanisms on selection pressure. Baker [3] and Schaffer [28] studied how different selection mechanisms bias selection. Whitley and Kauth [31] introduced a parameter for directly controlling selection pressure.

Goldberg and Deb's [13] study is an excellent reference on selection methods; it compares four selection schemes commonly used in modern genetic algorithms.

### C. Schema Processing

The *Schema Theorem* [20], [21], a well-known property of GA's, is formed from (3).

*Schema Theorem:* In a genetic algorithm using a proportional selection algorithm, a one-point crossover operator, and a mutation operator, for each hyperplane $H$ represented in $P(t)$ the following holds:

$$M[H(t+1)] \geq M[H(t)] \left\{ \frac{\overline{u}[H(t)]}{\overline{u}(t)} \right\} \cdot \left[ 1 - \frac{p_c d(H)}{L-1} \right] (1-p_m)^{o(H)}. \quad (4)$$

Here,

- $M[H(t+1)]$ is the expected number of individuals that will be the instances of hyperplane $H$ at time $t+1$ under the genetic algorithm, given that $M[H(t)]$ is the expected number of individuals at time $t$,
- $p_c$ is the crossover rate,
- $p_m$ is the mutation rate,
- $d(H)$ is the defining length of hyperplane $H$,
- $o(H)$ is the order of hyperplane $H$, and
- $L$ is the length of each string.

The term $\{\overline{u}[H(t)]/\overline{u}(t)\}$ denotes the ratio of the observed average fitness of the hyperplane $H$ to the overall population average. This term determines the rate of change of $M[H(t)]$, subject to the "error" terms $\{1 - [p_c d(H)/L - 1]\}(1-p_m)^{o(H)}$. The term $M[H(t+1)]$ increases if $\overline{u}[H(t)]$ is above the average fitness of the population (when the error terms are small), and vice versa. The error terms denote the effects of breaking up instances of hyperplane $H$ caused by crossover and mutation. The term $\{1 - [p_c d(H)/L - 1]\}\{M[H(t)]\}$ specifies an upper bound on the *crossover loss*, the loss of instances of $H$ resulting from crosses that fall within the defining length $d(H)$ of $H$. The term $(1-p_m)^{o(H)}$ gives the proportion of instances of $H$ that escape a mutation at one of the $o(H)$ defining bits of $H$. We can say that the Schema Theorem expressed a reduced view of GA: only the effect of selection is emphasized, while the effects of crossover and mutation are presented as a negative role. In short, the Schema Theorem predicts changes in the expected number of individuals belonging to a hyperplane between two consecutive generations. Clearly, short, low-order, above-average schemata receive an exponentially increasing number of trials in subsequent generations. However, increasing the sampling rate of schemata that are above-average does not guarantee convergence to a global optimum.

## III. Monotonic Versus Nonmonotonic Fitness Functions

The fitness function determines the productivity of individuals in a population. Clearly, the Schema Theorem is based on the fitness function rather than the objective function.

In general, a fitness function can be described as

$$u(x) = g[f(x)] \quad (5)$$

where $f$ is the objective function and $u(x)$ is a nonnegative number. The function $g$ is often a linear transformation, such as

$$u(x) = af(x) + b \quad (6)$$

where $a$ is positive when maximizing $f$ and negative when minimizing $f$ and $b$ is used to ensure a nonnegative fitness. In order to avoid the rapid decline of selective pressure, many forms of dynamic fitness scaling have been suggested [12], [15]. For example, a *dynamic linear fitness function* has the form

$$u(x) = af(x) + b(t). \quad (7)$$

Two definitions concerning the selection strategy are described below [17].

*Definition 1:* A selection algorithm is *monotonic* if the following condition is satisfied:

$$tsr(x_i) \leq tsr(x_j) \leftrightarrow u(x_i) \leq u(x_j). \quad (8)$$

*Definition 2:* A fitness function is *monotonic* if the following condition is satisfied:

$$u(x_i) \leq u(x_j) \leftrightarrow \alpha f(x_i) \leq \alpha f(x_j). \quad (9)$$

Here (and hereafter) $\alpha = 1$ when maximizing $f$ and $\alpha = -1$ when minimizing $f$.

In this paper, we propose a *nonmonotonic* fitness function instead of a *monotonic* fitness function. A nonmonotonic fitness function is one for which (9) is not satisfied for some individuals in a population.

### A. Monotonic Fitness Functions

All traditional GA's use monotonic fitness functions. Monotonic fitness functions do not provide good performance for all types of problems. Grefenstette and Baker [17] have stated the following two theorems to describe the search behavior of genetic algorithms in terms of the objective function.

*Theorem 1:* In any GA that uses a proportional selection algorithm and a dynamic linear fitness function, for any pair of hyperplanes $H_i$, $H_j$ in the population $P(t)$, if the average value of the objective function over the set $H_i(t)$ is less than that over the set $H_j(t)$, then $H_i$ will receive fewer trials than $H_j$ does.

Although this theorem shows how to characterize the search behavior of a class of genetic algorithms in terms of the objective function, it still fails to cover many successful genetic algorithms, such as a genetic algorithm using linear rank selection [17].

*Definition 3:* $H_i$ is *dominated* by $H_j$ in $P(t)$ ($H_i \leq_{D,t} H_j$) if

$$\max\{\alpha f(x) | x \in H_i(t)\} \leq \min\{\alpha f(y) | y \in H_j(t)\}. \quad (10)$$

That is, every individual of $H_j(t)$ is at least as good as every individual of $H_i(t)$.

*Theorem 2:* In any GA that uses a *monotonic* selection algorithm and a *monotonic* fitness function, for any pair of hyperplanes $H_i$, $H_j$ in the population $P(t)$, if $H_i$ is dominated by $H_j$ in $P(t)$, then $H_i$ will receive fewer trials than $H_j$ does.

Although Theorem 2 offers a description of the behavior of a larger class of genetic algorithms, it fails to distinguish the features of successful genetic algorithms from those of obviously degenerate search procedures, such as an algorithm that assigns every individual a target sampling rate of 1. In short, Theorems 1 and 2 do not provide, respectively, necessary and sufficient conditions for good performance in a genetic algorithm [17].

### B. Nonmonotonic Fitness Functions

Nonmonotonic fitness functions can extend the class of GA's. As suggested above, a worse solution also contains information that is useful for biasing the search. This idea is based on the following fact. Depending upon the distribution of the function values, the fitness function landscape can be more or less mountainous. It may have many peaks of high values beside steep cliffs that fall to deep gullies of very low values. On the other hand, the landscape may be a smoothly rolling one, with low hills and gentle valleys. In the former

case, a current worse solution, through the mutation operator, may have a greater chance of "evolving" toward a better future solution.

In order to exploit such current worse solutions, we define the following new fitness function.

*Definition 4:* A fitness function is called a *normalized-by-mean* fitness function if the following condition is satisfied:

$$u(x) = |f(x) - \overline{f}(t)|. \tag{11}$$

Here, $\overline{f}(t)$ is the average value of the objective function $f$ of the individuals in the population $P(t)$. Clearly, the normalized-by-mean fitness function is a type of *nonmonotonic* fitness function. We shall refer to a monotonic selection algorithm using the *normalized-by-mean* fitness function as *disruptive selection.*

Now, we can give a formal definition of directional selection, stabilizing selection, and disruptive selection as follows.

*Definition 5:* A selection algorithm is *directional* if it satisfies

$$tsr(x_i) \le tsr(x_j) \leftrightarrow \alpha f(x_i) \le \alpha f(x_j). \tag{12}$$

*Definition 6:* A selection algorithm is *stabilizing* if it satisfies

$$tsr(x_i) \le tsr(x_j) \leftrightarrow |f(x_i) - \overline{f}(t)| \ge |f(x_j) - \overline{f}(t)|. \tag{13}$$

*Definition 7:* A selection algorithm is *disruptive* if it satisfies

$$tsr(x_i) \le tsr(x_j) \leftrightarrow |f(x_i) - \overline{f}(t)| \le |f(x_j) - \overline{f}(t)|. \tag{14}$$

Next, we shall examine the schema processing under the effect of disruptive selection.

Since the sampling error is inevitable, standard GA's do not perform well in domains that have large variance within schemata. It is difficult to explicitly compute the observed variance of a schema that is represented in a population and then use this observed variance to estimate the real variance of that schema. Hence, we will try to use another statistic, a schema's observed deviation from the mean value of a population, to estimate the real variance of the schema. By using this statistic, we can determine the relationship between two schemata.

*Definition 8:* $H_i$ is more *remarkable* than $H_j$ in $P(t)$ ($H_i \ge_{R,t} H_j$) if

$$\frac{\sum_{x \in H_i(t)} |f(x) - \overline{f}(t)|}{m[H_i(t)]} \ge \frac{\sum_{y \in H_j(t)} |f(y) - \overline{f}(t)|}{m[H_j(t)]}. \tag{15}$$

That is, on average, $H_i$ has a larger deviation from $\overline{f}(t)$ than $H_j$ has. Since disruptive selection favors extreme (both superior and inferior) individuals, $H_i$ will receive more trials in subsequent generations.

We can now characterize the behavior of a class of genetic algorithms as follows.

*Theorem 3:* In any GA that uses a *monotonic* selection algorithm and the *normalized-by-mean* fitness function, for any pair of hyperplanes $H_i$, $H_j$ in the population $P(t)$,

$$H_i \ge_{R,t} H_j \rightarrow tsr[H_i(t)] \ge tsr[H_j(t)]. \tag{16}$$

*Proof:* $H_i \ge_{R,t} H_j$ implies

$$\frac{\sum_{x \in H_i(t)} |f(x) - \overline{f}(t)|}{m[H_i(t)]} \ge \frac{\sum_{y \in H_j(t)} |f(y) - \overline{f}(t)|}{m[H_j(t)]}.$$

By (11), we can conclude that $u[H_i(t)] \ge u[H_j(t)]$. Thus, by (3), it is clear that $tsr[H_i(t)] \ge tsr[H_j(t)]$.

Hence, using disruptive selection, a GA implicitly allocates more trials to schemata that have a large deviation from the mean value of a population. In the general case, we can define any kind of *nonmonotonic* fitness function $u(x) = g[f(x)]$ such that $H_i$ is more worth exploring than $H_j$ is in $P(t)$ as follows.

*Definition 9:* A hyperplane $H_i$ is *more worth exploring* than $H_j$ is in $P(t)$ ($H_i \ge_{MWE,t} H_j$) if

$$\frac{\sum_{x \in H_i(t)} g[f(x)]}{m[H_i(t)]} \ge \frac{\sum_{y \in H_j(t)} g[f(y)]}{m[H_j(t)]}. \tag{17}$$

Similarly, we can characterize the behavior of a class of genetic algorithms as follows.

*Theorem 4:* In any GA that uses a *monotonic* selection algorithm and a *nonmonotonic* fitness function, for any pair of hyperplanes $H_i$, $H_j$ in the population $P(t)$,

$$H_i \ge_{MWE,t} H_j \rightarrow tsr[H_i(t)] \ge tsr[H_j(t)]. \tag{18}$$

In fact, Theorems 3 and 4 extend the previous two theorems to a larger class of genetic algorithms. It is important to note that this extension is still consistent with Holland's Schema Theorem.

## IV. Solving a Non-Deceptive But GA-Hard Function

To verify the usefulness of using disruptive selection, we choose a class of problems that are "easy" in the sense of being nondeceptive but which are, in fact, hard for traditional GA's to optimize [18].

Let $f$ be defined as

$$f(x) = \begin{cases} 2^{L+1} & \text{if } x = 0 \\ x^2 & \text{otherwise} \end{cases} \tag{19}$$

where $x$ is an $L$-bit binary string representing the interval [0, 1]. Clearly, for any schema $H$ such that the optimum is in $H$, the average fitness of $H$ is higher than all other schemata that do not cover the optimal solution. Because they pose no deception at any order of schema partition, functions such as (19) are often called "GA-easy" [33]. However, the optimum of this function will probably never be found by a GA unless by a lucky crossover or mutation. This is because the schemata that contain the optimum have function values that vary widely, so the observed average fitnesses of the schemata do not reflect their true average fitnesses. In other words, large sampling errors are inevitable. Grefenstette [18] called this a type of "needle-in-a-haystack" function because the global optimum of the function is isolated from the relatively good areas of the search space.

In the earlier version of this paper [24], the optimal value for function (19) was found by using a steady-state approach. It is noted that the number of evaluations should be twice the results presented in [24]. This was because in [24], two newborn children were created at each generation but a factor 2 was missing when counting the number of evaluations. In this paper, we adopted a generational approach. Since the behavior of genetic algorithms is stochastic, their performance usually varies from run to run. Consequently, we replicated ten runs on this function for each combination of the following GA parameter settings: $p_c = 0.35, 0.65, 0.95$ and $p_m = 0.1$, 0.01, 0.001. Here $p_c$ and $p_m$ represent the crossover rate and mutation rate, respectively. Each search was run to 100 generations with the best five individuals recorded at each generation. The performance of a single run was taken to be the evaluation of the best individuals in the population at the end. In all cases a population size of 20 was used. The length of the binary strings was set to 10, 12, and 14 bits, respectively. Table I shows the number of successful runs out of ten runs for each combination of parameter settings. The figures in parentheses are the performance of traditional GA's. These results show that GA's using disruptive selection perform better than traditional GA's. For $p_m = 0.001$ and 0.01, the performance was not as good as that of $p_m = 0.1$. This could be because a low mutation rate prevented worse solutions from being mutated to better solutions.

TABLE I
THE NUMBER OF SUCCESSFUL RUNS OUT OF TEN RUNS

| $p_m$ | $p_c$ | L=10 | L=12 | L=14 |
|---|---|---|---|---|
| 0.1 | 0.35 | 8 (0) | 4 (1) | 1 (0) |
| | 0.65 | 5 (1) | 4 (1) | 1 (0) |
| | 0.95 | 4 (0) | 3 (0) | 1 (0) |
| 0.01 | 0.35 | 0 (0) | 1 (0) | 0 (0) |
| | 0.65 | 1 (0) | 0 (0) | 1 (0) |
| | 0.95 | 3 (0) | 0 (0) | 0 (0) |
| 0.001 | 0.35 | 0 (0) | 0 (0) | 0 (0) |
| | 0.65 | 0 (0) | 0 (0) | 0 (0) |
| | 0.95 | 1 (0) | 0 (0) | 0 (0) |

## V. SOLVING A DECEPTIVE FUNCTION

To illustrate the advantage of disruptive selection in solving a deceptive function, we shall first present a convergence analysis to estimate the number of occurrences of optima of a deceptive function in the population of a GA. Next, we shall demonstrate that disruptive selection is more reliable than directional selection in solving a deceptive function.

In the jargon of genetic algorithms, a function is called GA-easy or GA-hard depending on whether or not genetic algorithms can find the optimum (optima) of the function. There are several approaches to studying whether a function is GA-easy or GA-hard. The most widely known approach is to study the deceptiveness of the function [32].

A deceptive function is a function for which GA's are prone to be trapped at a deceptive local optimum. To date, the study of deception in GA's has primarily focused on three different topics [6]: designing deceptive functions; understanding the effects of deception in GA solutions; and modifying GA's to solve deceptive functions. We will concentrate on the last of these and focus on functions of unitation [1]. Functions of unitation are functions for which the function value of a string depends only on the total number of 1's in that string and not on the positions of those 1's. We will modify a standard GA with disruptive selection and compare the performance of disruptive selection and directional selection by means of convergence analysis.

### A. Convergence Analysis

Before stating the convergence analysis, we first introduce several definitions. All indices of the following matrices and vectors start from zero. Capital letters denote matrices or vectors, whereas the corresponding lowercase letters signify individual elements of the matrices. Note that the double index of an element of a vector is due to the choice of matrix notation for vectors.

*Definition 10:* The occurrence vector, denoted by $\mathbf{O^g}$, is a $1\times(l+1)$ row vector in which each element represents the probability that each kind of string occurs in generation $g$. For example, the element $o^0_{1,3}$ refers to the probability of occurrence for a string of unitation $ut = 3$ in the initial population. Here, $ut = 3$ means there are three 1's in that string.

*Definition 11:* The selection vector, denoted by $\mathbf{S^g}$, is an $(l+1)\times 1$ column vector in which each element represents the probability that each kind of string is selected in generation $g$. For example, the element $s^5_{2,1}$ stands for the probability that a string with unitation $ut = 2$ is selected in generation 5.

*Definition 12:* The transition matrix, denoted by $\mathbf{T^{p_m}}$, is an $(l+1)\times(l+1)$ matrix in which each element represents the probability a string has to be mutated into another string under the mutation rate $p_m$. For example, the element $t^{0.01}_{3,4}$ denotes the probability a string of unitation $ut = 3$ has to be mutated into a string of unitation $ut = 4$ under the mutation rate $p_m = 0.01$. Each element $t^{p_m}_{i,j}$ of $\mathbf{T^{p_m}}$ is computed by

$$\sum_{k=0}^{i}\binom{i}{k}\times\binom{l-i}{j-k}\times(1-p_m)^{l-(i+j-2k)}\times(p_m)^{i+j-2k}. \tag{20}$$

Here $k$ denotes the number of 1's that are unchanged under the mutation rate $p_m$, and $l$ stands for the length of the bit string. The equation is explained as follows. Assume that there are $k$ 1's in the string that are unchanged under mutation. The number of ways to choose $k$ 1's out of $i$ 1's is the first item of the above equation. In order to have $j$ 1's, we can set only $j-k$ bits out of $l-i$ bits to be 1. The number of ways to do this is the second item of the above equation. Since there are $i-k$ bits that should be mutated from 1 to 0 and $j-k$ bits that should be mutated from 0 to 1, we need to mutate a total of $i+j-2k$ bits and keep $l-(i+j-2k)$ bits from being mutated. Finally, since the number of 1's that are unchanged under mutation can range from 0 to $i$, we take the summation over $k$ from 0 to $i$.

Note that partially to simplify the convergence analysis and partially to illustrate the effects of disruption selection, we do not consider the crossover operator in the transition matrix. In spite of this simplification, we also conducted several experiments using the crossover operator to support our argument.

*Definition 13:* Let $\mathbf{diag}(\mathbf{V})$ be the diagonal matrix of a vector $\mathbf{V}$. That is, the diagonal elements of $\mathbf{diag}(\mathbf{V})$ correspond to the elements of $\mathbf{V}$ and all other nondiagonal elements of $\mathbf{diag}(\mathbf{V})$ are zero.

Clearly, we can express the relation between the occurrence matrices of two consecutive generations as follows:

$$\mathbf{O^{g+1}} = \mathbf{O^g}\times\mathbf{diag}(\mathbf{S^g})\times\mathbf{T^{p_m}}. \tag{21}$$

*Definition 14:* The function vector, denoted by $\mathbf{F}$, is an $(l+1)\times 1$ column vector in which each element represents the function value for each kind of string. For example, $f_{3,1}$ is the function value of the strings of unitation $ut = 3$.

*Definition 15:* Let $\mathbf{abs(M)}$ be the absolute matrix of a matrix $\mathbf{M}$ such that an element of $\mathbf{abs(M)}$ equals either $m_{i,j}$, if $m_{i,j}$ is a positive number, or $-m_{i,j}$, otherwise. Here $m_{i,j}$ is an element of the matrix $\mathbf{M}$.

To perform our convergence analysis, we first investigate the relationship between the occurrence vector, the selection vector, and the function vector. The selection vector $\mathbf{S^g}$ is a function of vectors $\mathbf{O^g}$ and $\mathbf{F}$. As stated earlier [see (1)], under the proportional strategy, the expected number of offspring reproduced by an individual is proportional to its fitness. That is, the probability of being selected for a string $x$ with unitation $ut = k$ can be expressed as

$$\begin{aligned} s^g_{k,1} &= tsr(x, g)\\ &= \frac{u(x)}{\overline{u}(g)}\\ &= \frac{f_{k,1}}{\sum_{j=0}^{l} f_{j,1}\times o^g_{1,j}} \end{aligned} \tag{22}$$

and

$$\begin{aligned} s^g_{k,1} &= tsr(x, g)\\ &= \frac{u(x)}{\overline{u}(g)}\\ &= \frac{|f_{k,1}-\overline{f}(g)|}{\sum_{j=0}^{l}|f_{j,1}-\overline{f}(g)|\times o^g_{1,j}} \end{aligned} \tag{23}$$

for directional selection and disruptive selection, respectively. Here, $\bar{f}(g)$ is the population's average performance at generation $g$.

Thus, in matrix form, we can express the selection matrix of GA's using directional selection and the selection matrix of GA's using disruptive selection as

$$\mathbf{S^g} = \frac{\mathbf{F}}{\mathbf{O^g} \times \mathbf{F}} \tag{24}$$

and

$$\mathbf{S^g} = \frac{\mathbf{abs(F - O^g \times F \times I)}}{\mathbf{O^g} \times \mathbf{abs(F - O^g \times F \times I)}}, \tag{25}$$

respectively. Here $\mathbf{I}$ is an identity column vector with dimension $(l+1) \times 1$.

Next, we derive the relationship between the occurrence matrices of two consecutive generations. The percentage of a string in a population, after the selection phase, depends on its current percentage and its productivity (i.e., target sampling rate). In addition, theoretically, a string can be produced by any other string through mutation. By observing these two facts, we can derive the relationship between the occurrence matrices of two consecutive generations. In the first generation, the percentage of occurrence for a string of unitation $ut = i$ can be expressed as

$$\begin{aligned} o^1_{1,i} &= \sum_{k=0}^{l} o^0_{1,k} \times s^0_{k,1} \times t^{p_m}_{k,i} \\ &= \sum_{k=0}^{l} o^0_{1,k} \times \frac{f_{k,1}}{\sum_{j=0}^{l} f_{j,1} \times o^0_{1,j}} \times t^{p_m}_{k,i} \\ &= \frac{\sum_{k=0}^{l} o^0_{1,k} \times f_{k,1} \times t^{p_m}_{k,i}}{\sum_{j=0}^{l} f_{j,1} \times o^0_{1,j}}. \end{aligned} \tag{26}$$

Similarly, for the second generation, we can write

$$\begin{aligned} o^2_{1,i} &= \sum_{k=0}^{l} o^1_{1,k} \times s^1_{k,1} \times t^{p_m}_{k,i} \\ &= \sum_{k=0}^{l} o^1_{1,k} \times \frac{f_{k,1}}{\sum_{j=0}^{l} f_{j,1} \times o^1_{1,j}} \times t^{p_m}_{k,i} \\ &= \frac{\sum_{k=0}^{l} o^1_{1,k} \times f_{k,1} \times t^{p_m}_{k,i}}{\sum_{j=0}^{l} f_{j,1} \times o^1_{1,j}}. \end{aligned} \tag{27}$$

In general,

$$o^{g+1}_{1,i} = \frac{\sum_{k=0}^{l} o^g_{1,k} \times f_{k,1} \times t^{p_m}_{k,i}}{\sum_{j=0}^{l} f_{j,1} \times o^g_{1,j}}. \tag{28}$$

Thus, in matrix form, we can express the relationship between the occurrence matrices of two consecutive generations as

$$\mathbf{O^{g+1}} = \frac{\mathbf{O^g} \times \mathbf{diag}\,(\mathbf{F}) \times \mathbf{T^{p_m}}}{\mathbf{O^g} \times \mathbf{F}} \tag{29}$$

and

$$\mathbf{O^{g+1}} = \frac{\mathbf{O^g} \times \mathbf{diag}\,[\mathbf{abs(F - O^g \times F \times I)}] \times \mathbf{T^{p_m}}}{\mathbf{O^g} \times \mathbf{abs(F - O^g \times F \times I)}} \tag{30}$$

for directional selection and disruptive selection, respectively.

### B. Reliability of Convergence

Note that the above equations supply only a *deterministic model* of the genetic algorithms under the assumption that the expectations are actually achieved in each generation. In fact, the behavior of a genetic algorithm is stochastic. Hence, it is worth investigating the behavior of GA's with directional selection and disruptive selection. In this section we demonstrate that GA's using disruptive selection find the optima of a deceptive function more quickly and reliably than GA's using directional selection.

A bipolar function is defined as a function that has two global optima that are maximally far apart from each other and a number of deceptive attractors that are maximally far apart from the global optima. Here the distance is measured in Hamming space. A symmetric bipolar function of unitation is a function that has two global optima of unitation $ut = 0$ and $ut = l$ ($l$, an even integer number, is the length of the bit string), respectively, a number of deceptive attractors of unitation $ut = l/2$, and function values that are symmetrical with respect to unitation $ut = l/2$. In our study, the test bed was a six-bit symmetric bipolar deceptive function of unitation for which

$$\mathbf{O^0} = (\tfrac{1}{64}\ \tfrac{6}{64}\ \tfrac{15}{64}\ \tfrac{20}{64}\ \tfrac{15}{64}\ \tfrac{6}{64}\ \tfrac{1}{64})$$

and

$$\mathbf{F} = \begin{pmatrix} \mathbf{1} \\ \mathbf{0.2} \\ \mathbf{0.6} \\ \mathbf{0.9} \\ \mathbf{0.6} \\ \mathbf{0.2} \\ \mathbf{1} \end{pmatrix}.$$

This function was constructed by satisfying the sufficient conditions for a bipolar deceptive function [6]. Using (20), (29), (30), $\mathbf{O^0}$, and $\mathbf{F}$, we can compute the distribution of the population for GA's after any generation. Although the distribution of the entire population can be determined, we are interested only in the occurrence ratio of optima (strings 000 000 and 111 111). Here (and hereafter) the occurrence ratio of optima refers to the percentage of occurrence of the optima in a population. When the occurrence ratio of optima equals zero, the GA has failed to discover the optima. In contrast, when the occurrence ratio of optima equals one, we say that the population has completely converged to the optima.

Using the mutation rates 0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99, and 0.999, we computed the transition matrix $\mathbf{T^{p_m}}$ in (29) and (30). Fig. 1 shows the computation results of the deterministic model stated in Section V-A.

The horizontal axis identifies the mutation rate and the vertical axis indicates the occurrence ratio of optima in the final population (after 100 generations). Clearly, the curves are symmetrical with respect to $p_m = 0.5$. This is because the transition matrices are symmetrical with respect to $p_m = 0.5$ and both the occurrence vector $\mathbf{O^0}$ and the function vector $\mathbf{F}$ are symmetrical. Note that the upper bound of the occurrence ratio of optima is 50% when using disruptive selection. This upper bound is reasonable, since disruptive selection favors extreme (both superior and inferior) individuals.

It can be seen that in the range of 0.05–0.95 the proposed method has a larger occurrence ratio of optima than directional selection has. In contrast, in the ranges of [0.001 0.05] and [0.95 0.999], the proposed method has a smaller occurrence ratio of optima. However,

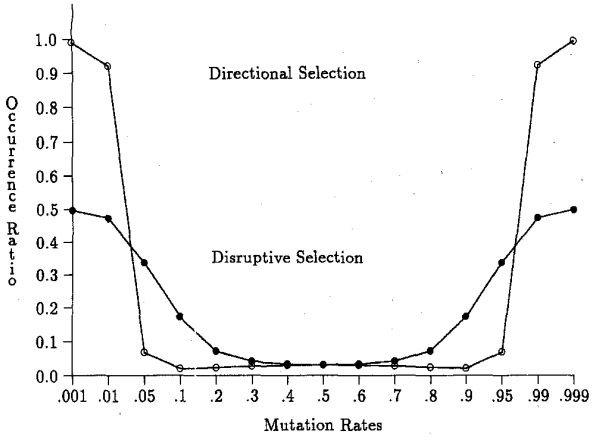


Fig. 1. Convergence of solving A symmetric bipolar deceptive function.

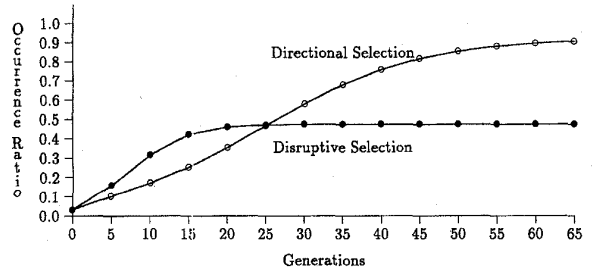


Fig. 2. Comparison of rapidity on solving A symmetric bipolar deceptive function.

computing the occurrence ratio of optima through the deterministic model, we observe that GA's using the proposed method find the optima more quickly than GA's using the conventional method. This observation is depicted (for $p_m = 0.01$) in Fig. 2.

Here, the horizontal axis indicates the number of generations. Clearly, in the early generations, GA's using disruptive selection have a higher occurrence ratio of optima than GA's using directional selection.

Since (29) and (30) involve probability, there is an intrinsic difference between experimental results obtained by actually applying GA's and computation results obtained via these equations. It is reasonable to believe that a higher occurrence ratio of optima in the early stages implies greater reliability of the ratio in the final result. To verify this conjecture, we conducted several experiments using the following parameters:

- Population size: 64
- Initial population: randomized
- Generations: 100
- Mutation rates: 0.001, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, and 0.5.

Each application of a GA consisted of 50 reinitialized runs. After replicating 50 runs, we computed the mean number of instances of the optima in the final population and the variation of the observed values. Table II presents the experimental results; here

- $E$ is the expected number of instances of the optima
- $M$ is the mean number of instances of the optima
- $SD$ is the standard deviation of the 50 numbers of instances of the optima
- $CV$ is the coefficient of variation, defined as $SD/M$.

A subscript 1 signifies directional selection and 2 disruptive selection.

TABLE II
RELIABILITY OF SOLVING A SYMMETRIC BIPOLAR DECEPTIVE FUNCTION

| $p_m$ | $E_1$ | $M_1$ | $SD_1$ | $CV_1$ | $E_2$ | $M_2$ | $SD_2$ | $CV_2$ |
|---|---|---|---|---|---|---|---|---|
| 0.001 | 63.50 | 33.10 | 32.13 | 0.97 | 31.83 | 31.18 | 3.25 | 0.10 |
| 0.01 | 59.00 | 30.60 | 29.60 | 0.97 | 30.29 | 29.98 | 3.67 | 0.12 |
| 0.05 | 4.35 | 4.70 | 7.03 | 1.50 | 21.55 | 21.06 | 4.20 | 0.20 |
| 0.1 | 1.37 | 1.42 | 1.97 | 1.39 | 11.1 | 10.02 | 3.56 | 0.36 |
| 0.2 | 1.56 | 1.78 | 1.54 | 0.87 | 4.58 | 4.10 | 1.85 | 0.45 |
| 0.3 | 1.78 | 2.02 | 1.52 | 0.75 | 2.70 | 2.68 | 1.46 | 0.54 |
| 0.4 | 1.94 | 2.28 | 1.60 | 0.70 | 2.14 | 2.24 | 1.46 | 0.65 |
| 0.5 | 2.00 | 2.20 | 1.37 | 0.62 | 2.00 | 1.90 | 1.41 | 0.74 |

TABLE III
RELIABILITY OF SOLVING A SYMMETRIC BIPOLAR DECEPTIVE FUNCTION (WITH CROSSOVER)

| $p_m$ | $p_c$ | $M_1$ | $SD_1$ | $CV_1$ | $M_2$ | $SD_2$ | $CV_2$ |
|---|---|---|---|---|---|---|---|
| 0.001 | 0.35 | 6.4 | 20.24 | 3.16 | 31.6 | 3.06 | 0.1 |
| | 0.65 | 0.0 | 0.0 | - | 31.8 | 3.12 | 0.1 |
| | 0.95 | 0.0 | 0.0 | - | 33.1 | 3.54 | 0.11 |
| 0.01 | 0.35 | 5.8 | 18.34 | 3.16 | 32.1 | 4.84 | 0.15 |
| | 0.65 | 0.0 | 0.0 | - | 30.9 | 3.93 | 0.13 |
| | 0.95 | 0.0 | 0.0 | - | 31.0 | 3.89 | 0.13 |
| 0.05 | 0.35 | 1.6 | 2.50 | 1.56 | 21.0 | 4.69 | 0.22 |
| | 0.65 | 0.6 | 0.52 | 0.86 | 23.1 | 3.35 | 0.14 |
| | 0.95 | 1.1 | 1.45 | 1.32 | 22.6 | 6.06 | 0.27 |
| 0.1 | 0.35 | 1.7 | 2.21 | 1.3 | 10.1 | 4.93 | 0.49 |
| | 0.65 | 1.4 | 1.17 | 0.84 | 12.1 | 2.96 | 0.24 |
| | 0.95 | 1.0 | 1.49 | 1.49 | 10.3 | 5.31 | 0.52 |
| 0.2 | 0.35 | 1.4 | 1.17 | 0.84 | 3.0 | 1.76 | 0.59 |
| | 0.65 | 1.7 | 1.57 | 0.92 | 4.1 | 1.44 | 0.35 |
| | 0.95 | 2.2 | 1.62 | 0.74 | 2.7 | 1.89 | 0.7 |
| 0.3 | 0.35 | 2.3 | 1.34 | 0.58 | 2.9 | 1.37 | 0.47 |
| | 0.65 | 1.2 | 0.63 | 0.53 | 2.2 | 1.69 | 0.77 |
| | 0.95 | 2.3 | 1.34 | 0.58 | 2.6 | 1.58 | 0.61 |
| 0.4 | 0.35 | 2.3 | 1.25 | 0.54 | 1.7 | 1.34 | 0.79 |
| | 0.65 | 2.5 | 1.43 | 0.57 | 1.5 | 1.5 | 1.0 |
| | 0.95 | 2.3 | 1.34 | 0.58 | 1.9 | 1.52 | 0.8 |

Obviously, directional selection usually resulted in a higher variation than disruptive selection did. This result verifies our conjecture; an early, slight deviation from the computation value eventually leads to a great divergence from the expected result. This effect was clear in solving such a deceptive function, since the deceptive attractors are the second-best solution and are in the majority. Note that, for $p_m = 0.5$, the behavior of a GA is just like a random search. Thus, a contrary result is not surprising.

Since it is well known that the power of GA's does not come from selection and mutation only, we also conducted experiments including the crossover operator to support our argument. Here, we replicated ten runs for each combination of parameter settings. Table III presents the results. It can be seen that disruptive selection is clearly superior to directional selection in solving this type of deceptive problem. From Tables II and III we can see that the crossover operator does not provide benefits when disruptive selection is used and it brings drawbacks when directional selection is used. This could be because the test function is deceptive for which GA's are prone to be trapped at a deceptive local optimum, thus the crossover operator plays a negative role.

### C. Why Disruptive Selection Works

To explain why disruptive selection works, we characterize the deceptive function by its landscape. Fig. 3 shows the landscape of a symmetric bipolar deceptive function in unitation space.

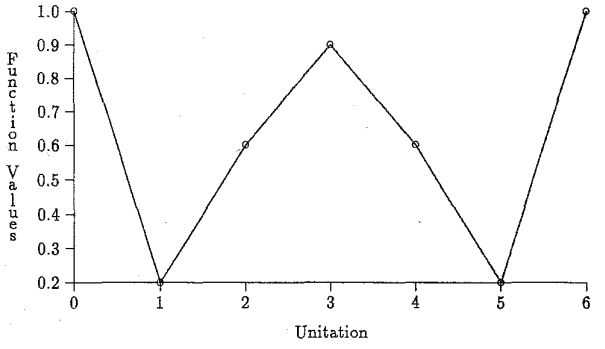


Fig. 3. Landscape of a symmetric bipolar deceptive function in unitation space.

TABLE IV
PROBABILITY OF BEING MUTATED INTO THE GLOBAL OPTIMA

| $p_m$ | $ut=0$ | $ut=1$ | $ut=2$ | $ut=3$ | $ut=4$ | $ut=5$ | $ut=6$ |
|---|---|---|---|---|---|---|---|
| 0.001 | 0.994 | 0.001 | 0.0 | 0.0 | 0.0 | 0.001 | 0.994 |
| 0.01 | 0.941 | 0.01 | 0.0 | 0.0 | 0.0 | 0.01 | 0.941 |
| 0.05 | 0.735 | 0.039 | 0.002 | 0.0 | 0.002 | 0.039 | 0.735 |
| 0.1 | 0.531 | 0.059 | 0.007 | 0.0 | 0.007 | 0.059 | 0.531 |
| 0.2 | 0.262 | 0.066 | 0.017 | 0.004 | 0.017 | 0.066 | 0.262 |
| 0.3 | 0.118 | 0.052 | 0.026 | 0.009 | 0.026 | 0.052 | 0.118 |
| 0.4 | 0.052 | 0.037 | 0.030 | 0.014 | 0.030 | 0.037 | 0.052 |
| 0.5 | 0.031 | 0.031 | 0.031 | 0.031 | 0.031 | 0.031 | 0.031 |

It is easy to see that the global optima are surrounded (in a Hamming sense) by worst solutions and that the local optima (strings with unitation $ut = 3$) are surrounded by better solutions (strings with unitation $ut = 2$ or $ut = 4$). These features imply that worst solutions, those with either unitation $ut = 1$ or $ut = 5$, have a greater chance of being mutated into optimal solutions and that better solutions are prone to be mutated into local optima. To ascertain whether this implication is true, we used (20) to compute the probability a string has to be mutated into the global optima (strings 000 000 and 111 111) and the local optima (strings with unitation $ut = 3$) under the mutation rate $p_m$. Tables IV and V show the computation results from (20). These results confirm the implication.

From Tables IV and V, we observe one common feature of these data, namely, there are two types of attractors, global optima and local optima, in the landscape. The force of attraction is dependent on the Hamming distance between one point and the attractor. The nearer a point is to an attractor, the stronger the force of attraction is. This feature explains why traditional GA's are sometimes misled to deceptive attractors and why disruptive GA's perform well.

## VI. Discussion and Conclusions

Since all traditional GA's use a monotonic fitness function and apply the "survival-of-the-fittest" principle to reproduce the new population, they can be viewed as a process of evolution that is based on directional selection. In this paper, we have proposed a type of disruptive selection that uses a nonmonotonic fitness function. The major difference between disruptive selection and directional selection is that the new method devotes more trials to both better solutions and worse solutions than it does to moderate solutions, whereas the traditional method allocates its attention according to the performance of each individual.

The experimental results reported here show that GA's using the proposed method easily find the optimum of a function that is nondeceptive but GA-hard. Since the sampling error is inevitable,

TABLE V
PROBABILITY OF BEING MUTATED INTO THE LOCAL OPTIMA

| $p_m$ | $ut=0$ | $ut=1$ | $ut=2$ | $ut=3$ | $ut=4$ | $ut=5$ | $ut=6$ |
|---|---|---|---|---|---|---|---|
| 0.001 | 0.0 | 0.0 | 0.004 | 0.994 | 0.004 | 0.0 | 0.0 |
| 0.01 | 0.0 | 0.001 | 0.038 | 0.942 | 0.038 | 0.001 | 0.0 |
| 0.05 | 0.002 | 0.02 | 0.156 | 0.753 | 0.156 | 0.02 | 0.002 |
| 0.1 | 0.015 | 0.066 | 0.245 | 0.591 | 0.245 | 0.066 | 0.015 |
| 0.2 | 0.082 | 0.174 | 0.312 | 0.419 | 0.312 | 0.174 | 0.082 |
| 0.3 | 0.185 | 0.256 | 0.320 | 0.349 | 0.320 | 0.256 | 0.185 |
| 0.4 | 0.276 | 0.300 | 0.315 | 0.320 | 0.315 | 0.300 | 0.276 |
| 0.5 | 0.313 | 0.313 | 0.313 | 0.313 | 0.313 | 0.313 | 0.313 |

traditional GA's do not perform well with functions that have large variance within schemata. However, using disruptive selection, a GA implicitly allocate more trials to schemata that have a large deviation from the mean value of a population. This statistic provides a good estimate of the real variance of the schema. Experimental results also show that GA's using disruptive selection find the optima of a deceptive function more quickly and reliably than GA's using directional selection do. This could be because the global optima of a deceptive function are surrounded by the worst solutions and the local optima are surrounded by better solutions. Since disruptive selection also favors inferior individuals, GA's using disruptive selection are immune to traps. Although we have tested this method on only two such functions, it might be applied successfully to other kinds of problems. Since disruptive selection favors both superior and inferior individuals, GA's using disruptive selection will very likely perform well on problems easily solved by traditional GA's. If GA's using disruptive selection should not work well on them, we can implement a parallel GA in which disruptive selection and directional selection are used in different nodes and migration of good solutions occurs between different nodes periodically. Thus, as a supplement to directional selection, disruptive selection promises to be helpful in solving problems that are hard to optimize using traditional GA's.

## Acknowledgment

The authors would like to thank the three anonymous reviewers and the editor for their valuable suggestions for improving this paper.

## References

[1] D. H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*. Boston, MA: Kluwer, 1987.
[2] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proc. First Int. Conf. on Genetic Algorithms and Their Applications*. Hillsdale, NJ: Lawrence Erlbaum, 1985, pp. 101–111.
[3] ——, "Reducing bias and inefficiency in the selection algorithm," in *Proc. Second Int. Conf. on Genetic Algorithms and Their Application*. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 14–21.
[4] B. Bhanu, S. Lee, and J. Ming, "Self-optimizing image segmentation system using a genetic algorithm," in *Proc. Fourth Int. Conf. on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1991, pp. 362–369.
[5] G. A. Cleveland and S. F. Smith, "Using genetic algorithms to schedule flow shop releases," in *Proc. Third Int. Conf. on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1989, pp. 160–169.
[6] K. Deb, J. Horn, and D. E. Goldberg, "Multimodal deceptive functions," Univ. of Illinois at Urbana-Champaign, IlliGAL Report no. 92003, 1992.
[7] K. A. DeJong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Mich., Ann Arbor, MI, 1975.
[8] ——, "Learning with genetic algorithms: An overview," *Machine Learning*, vol. 3, pp. 121–138, 1988.
[9] M. Dorigo and U. Schnepf, "Genetic-based machine learning and behavior-based robotics: A new synthesis," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 1, pp. 141–154, 1993.
[10] L. J. Eshelman, R. A. Caruana, and J. D. Schaffer, "Biases in the crossover landscape," in *Proc. Third Int. Conf. on Genetic Algorithms*. San Mateo, CA, Morgan Kaufmann, 1989, pp. 10–19.

[11] D. E. Goldberg and R. Lingle, Jr., "Alleles, loci, and traveling salesman problem," in *Proc. First Int. Conf. on Genetic Algorithms and Their Applications*. Hillsdale, NJ: Lawrence Erlbaum, 1985, pp. 154–159.
[12] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[13] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 69–93.
[14] J. J. Grefenstette, R. Gopal, B. Rosmaita, and D. V. Gucht, "Genetic algorithms for the traveling salesman problem," in *Proc. First Int. Conf. on Genetic Algorithms and Their Applications*. Hillsdale, NJ: Lawrence Erlbaum, 1985, pp. 160–168.
[15] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-16, no. 1, pp. 122–128, 1986.
[16] ——, "Credit assignment in rule discovery systems based on genetic algorithms," *Machine Learning*, vol. 3, pp. 225–245, 1988.
[17] J. J. Grefenstette and J. E. Baker, "How genetic algorithms work: A critical look at implicit parallelism," in *Proc. Third Int. Conf. on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1989, pp. 20–27.
[18] J. J. Grefenstette, "Deception considered harmful," in *Foundations of Genetic Algorithms 2*, D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1992, pp. 75–91.
[19] S. A. Harp, T. Samad, and A. Guha, "Towards the genetic synthesis of neural networks," in *Proc. Third Int. Conf. on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1989, pp. 360–369.
[20] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Mich. Press, 1975.
[21] ——, "Searching nonlinear functions for high values," *Appl. Math. Comp.*, vol. 32, pp. 255–274, 1989.
[22] C. L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," in *Proc. Fourth Int. Conf. on Genetic Algorithms*.. San Mateo, CA: Morgan Kaufmann, 1991, pp. 450–457.
[23] K. Kristinsson and G. A. Dumont, "System identification and control using genetic algorithms," *IEEE Trans. Syst., Man, and Cybern.*, vol. 22, no. 5, pp. 1033–1046, 1992.
[24] T. Kuo and S. Y. Hwang, "A genetic algorithm with disruptive selection," in *Proc. Fifth Int. Conf. on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1993, pp. 65–69.
[25] G. E. Liepins and M. D. Vose, "Deceptiveness and genetic algorithm dynamics," in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 36–50.
[26] B. F. J. Manly, *The Statistics of Natural Selection on Animal Populations*. London, UK: Chapman and Hall, 1984.
[27] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," in *Proc. Third Int. Conf. on Genetic Algorithms*. San Mateo, CA, Morgan Kaufmann, 1989, pp. 379–384.
[28] J. D. Schaffer and A. Morishima, "An adaptive crossover distribution mechanism for genetic algorithms," in *Proc. Second Int. Conf. on Genetic Algorithms and Their Applications*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1987, pp. 36–40.
[29] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. Third Int. Conf. on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1989, pp. 2–9.
[30] G. Syswerda and J. Palmucci, "The application of genetic algorithms to resource scheduling," in *Proc. Fourth Int. Conf. on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1991, pp. 502–508.
[31] D. Whitley and J. Kauth, "GENITOR: a different genetic algorithm," in *Proc. Rocky Mountain Conf. on Artificial Intelligence*, 1988, pp. 118–130.
[32] D. Whitley, "Fundamental principles of deception in genetic search," in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 221–241.
[33] S. W. Wilson, "GA-easy does not imply steepest-ascent optimizable," in *Proc. Fourth Int. Conf. on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1991, pp. 85–89.

# A Constructive Approach for Nonlinear System Identification Using Multilayer Perceptrons

Ju-Yeop Choi, Hugh F. VanLandingham, and Stanoje Bingulac

***Abstract*—This paper combines a conventional method of multivariable system identification with a dynamic multi-layer perceptron (MLP) to achieve a constructive method of nonlinear system identification. The class of nonlinear systems is assumed to operate nominally around an equilibrium point in the neighborhood of which a linearized model exists to represent the system, although normal operation is not limited to the linear region. The result is an accurate discrete-time nonlinear model, extended from a MIMO linear model, which captures the nonlinear behavior of the system.**

## I. Introduction

Since real-world systems resist being modeled in precise mathematical terms due to unknown dynamics and, typically, a noisy environment, it is very difficult to determine an exact model for a complex nonlinear system. Consequently, there is a need for a nonclassical technique which has the ability to accurately model these physical processes. It has been shown that multi-layer perceptrons (MLP's), one of the many forms of artificial neural networks (ANN's) is a universal function approximator, i.e., with sufficient training on appropriate input/output data, MLP's can represent arbitrarily closely any smooth vector map [1], [2]. Although the theory of linear system identification may now be considered to be a mature discipline, new techniques, particularly for *nonlinear* system identification, continue to be of interest. In this paper such a method is addressed in the context of using neural networks [3], [4]. Neural networks of various types and structures (paradigms) have been found to be efficient tools for identifying nonlinear systems, e.g., through Volterra series models, GMDH models, SONN models and radial basis functions [5]–[8]. Among the researchers of the control community using ANN's over the past two decades, Narendra [9]–[11] has used dynamic ANN's as components in dynamical systems, concentrating on system identification and control of nonlinear plants. Pao introduced the functional-link net which constructs a nonlinear mapping at the input layer to reduce the complexity of ANN's [12]. Although there are many techniques available for the corresponding *linear* identification problem, MLP's may be regarded as a nonclassical technique which can accomplish similar results using only input/output data, i.e., without prior model information. Most importantly, MLP's do not require the usual assumption of linearity. Thus, although it is true that neural networks can offer little, if any, improvement over existing methods of identification of linear systems, they do present a potential for capturing the complex nonlinearities of a wide class of industrial processes in a universal manner never before imagined [13]. However, there are many difficult problems to overcome, such as when the nonlinear system is found to be both complex and unstable. This latter condition complicates the "training" of the MLP [14]. One approach is to stabilize the system locally. Such stabilization of a nonlinear dynamic system can be done for systems which are

Manuscript received August 27, 1993; revised May 18, 1994, and December 28, 1995.

J.-Y. Choi and H. F. VanLandingham are with the Bradley Department of Electrical Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0111 USA (e-mail: hughv@vt.edu).

S. Bingulac is with the Department of Electrical and Computer Engineering, Kuwait University, 13060 Safat, Kuwait.

Publisher Item Identifier S 1083-4419(96)02308-4.

1083–4419/96$05.00 © 1996 IEEE