Searching Nonlinear Functions for High Values

John H. Holland

Division of Computer Science and Engineering
The University of Michigan
Ann Arbor, Michigan 48103

ABSTRACT

Many complex systems of great interest—ecologies, economies, immune systems, etc.—can be described as *adaptive nonlinear networks* (ANNs), wherein the network specifies the allowed nonlinear interactions of a large number of components. With an appropriate representation, the adaptation of an ANN can be looked upon as a search in the space $\{1,0\}^k$, using a progressively biased probability distribution, $p(t)$. Samples of this space return a value that measures the current performance of the ANN. The corresponding function $u: \{1,0\}^k \rightarrow$ Reals is usually badly nonlinear with multitudes of local optima. The possibilities for biasing $p(t)$, as information accumulates, are more readily seen if $\{1,0\}^k$ is treated as a k-dimensional space re-represented via a *hyperplane transform*. Sampling then supplies estimates of the expected value of u, under $p(t)$, over hyperplanes of various dimensions. Though it is possible in principle, it is not feasible to calculate the estimated expectations for even a small proportion of the hyperplanes for which information is available. However, it can be proved that there is a class of procedures, called *genetic algorithms*, that rapidly bias $p(t)$ to take advantage of large numbers of above-average hyperplanes. Several properties of genetic algorithms are discussed using this point of view.

INTRODUCTION

Some of the most difficult problems facing humankind involve *adaptive nonlinear networks*—the systems of evolutionary genetics, immune systems, cognitive systems, ecologies, and economies, to name a few. The behavior of such systems emerges from the aggregate influence of a multitude of parts, each of which acts locally in response to the context provided by the activity of a limited number of other parts. Because the interactions are nonlinear, the state trajectory induced by the interactions *cannot* be determined by a simple superposition of the individual acts. The state space itself is so complex that individual states never recur over feasible observation times.

The complexity is further exacerbated because the networks are adaptive—they autonomously modify the interactions of their parts, and indeed the parts themselves, to improve overall performance.

It is common to model the behavior of complex systems as an approach to some attractor, say the search for an extremum. However, as Simon [7] pointed out some years ago, it is better to think of a system such as an adaptive nonlinear network as *satisficing*. The system confronts the problems presented by its environment by "developing solutions" that are adequate, not optimal. These systems have many levels of organization, and within each level there are subsystems that act and react to exploit limited resources (roughly, energy and material). A subsystem that does a bit better in this competition will soon exert a substantial influence on the overall behavior, even though its "solution" is far from optimal.

The usual evolution of an adaptive nonlinear network (ANN hereafter) exhibits perpetual novelty. The subsystems of the ANN are continually revising their "boundaries," and the procedures within these boundaries (a process Hebb [4] typifies as fractionation and recruitment). Even when we ignore details within subsystems, the trajectory typically exhibits no repetitions (cf. the evolution of a species in biology). It is a consequence of this perpetual novelty, a consequence to be made plausible later, that these systems operate far from any global attractor. Improvements are always possible and, indeed, occur regularly.

How does one study such systems? Many of the tools of mathematics (e.g. linearity, fixed points, convergence) offer only limited help. The very size and complexity of the underlying state space is daunting. An exhaustive search of possibilities is not even conceivable; only samples are possible. There is at least a bit of comfort associated with sampling. Estimates produced by a legitimate sampling technique have a reliability that does not depend upon the size of the underlying sample space (as long as the size of the sample remains minuscule relative to the size of the space). The present paper, to exploit this advantage, views ANNs as searching for better performance by executing a progressively biased sample of the space of possibilities. We will assume that the states have some worth (*payoff* in game theory, *utility* in economics, *error* in control theory, *fitness* in genetics, and so on), and that the sampling is directed toward attaining states of high worth. In keeping with our earlier comments, emphasis will be placed upon efficiency in attaining improvements, rather than upon efficiency in approaching some optimum.

In outline, the paper proceeds as follows:

(1) The paper takes as its starting point the assumption that the ANN's component structures (rules, strategies, chromosomes, policies, or the like)

can be represented as a collection of k-bit strings, and that each of these strings can be assigned some value. There is little loss in generality in using k-bit strings—any computer model ultimately rests upon a representation in terms of binary words. However, the assumption that we can assign some value to individual components of the ANN is more problematical. The problem is one of *assigning credit*. One must assign values to individuals on the basis of their contributions to overall performance, as when one assigns *fitness* to individuals in an evolving biological population, or *worth* to individual corporations in a economy, and so on. While this is a problem of interest in its own right, we will not consider it here (the interested reader will find discussions in [6]). On the basis of this assumption we can model the ANN's search as a sampling of the space of strings $\{1,0\}^k$ using a probability distribution $p(t)$ that changes progressively as time t increases. Each $x \in \{1,0\}^k$ represents a structure to tried, and the function $u: \{1,0\}^k \rightarrow \text{Reals}$ determines the value $u(x)$ returned when x is tried. For an ANN, $u(x)$ will be a complex nonlinear function. The evaluation of a single x will be a time-consuming task as, for example, when x is a strategy for playing a game. Here we are only concerned with conditions under which the information returned—the value $u(x)$ of the structure x—will be helpful in biasing the distribution $p(t)$ that directs the search of $\{1,0\}^k$.

(2) The information accumulated by sampling u's argument space $\{1,0\}^k$ can be more transparently related to possibilities for further biasing $p(t)$ if u is re-represented using a *hyperplane transform*. The hyperplane transform uses the fact that, under the distribution $p(t)$, the function u is a random variable and subsets of the argument space $\{1,0\}^k$ are *events* having well-defined *expectations*. The hyperplane transform uses the expectations of selected sets of hyperplanes in $\{1,0\}^k$ to re-represent u. It can be shown that this transform provides a unique, invertible representation for any finite, nonlinear function u.

(3) Biasing a search toward (or away from) some subset of $\{1,0\}^k$, if it is to be information-based, requires an estimate that elements of that subset are, on average, better (worse) than elements elsewhere. Concentrating on hyperplanes, we note that hyperplanes of higher dimension, being larger subsets of $\{1,0\}^k$, typically receive a larger fraction of any set of samples drawn from $\{1,0\}^k$. As a consequence, estimates of the expectation $u(s)$ associated with a higher-dimensional hyperplane s will be confirmed faster than similar estimates for lower-dimensional refinements of s. Accordingly, as the number of samples increases, biases should proceed from biases based on estimates for high-dimensional hyperplanes to biases involving lower-dimensional refinements of those hyperplanes.

(4) The problem, then, is to design a feasible algorithm that, as information accumulates, provides the biases suggested by the *hyperplane transform*.

It is easy to see that, for large k, it is *not* feasible to carry out an explicit calculation of the hyperplane transform each time P_t is changed. However, it can be proved that *genetic algorithms* rapidly provide the biasing implied by the hyperplane transform without explicitly carrying out the calculations involved.

[Throughout the remainder of this paper, when the term *random* is used without further qualification, as in "*random*ly generated," it implies a sample space with a *uniform* distribution.]

THE HYPERPLANE TRANSFORMATION

The hyperplane transformation represents u in terms of its averages over certain easily defined hyperplanes in the space $\{1,0\}^k$. These hyperplanes, called *schemas* in [5], are specified by strings from the set $\{1,0,*\}^k$. In $s \in \{1,0,*\}^k$ the "*" is interpreted as a "wildcard" or "don't care" symbol. The positions occupied by 1 or 0 (i.e., those positions *not* occupied by *'s) are called the *defining loci of* s. s specifies a subset (hyperplane) of $\{1,0\}^k$ under the rule that $x \in s$ if and only if x matches s at every defining locus of s. Thus, $1**\ldots*$ designates the subset of all strings that start with a 1, and $11\ldots1*$ designates the two-element subset $\{11\ldots11, 11\ldots10\}$.

DEFINITION. For convenience, a hyperplane defined with c defining loci will be called a hyperplane of *level* c (it is a hyperplane of *dimension* $k-c$).

Each choice of a set of defining loci partitions the space $\{1,0\}^k$ into disjoint subsets containing equal numbers of elements. For example, schemas having positions 1 and 2 from the right as defining loci partition $\{1,0\}^k$ into the disjoint subsets $\{**\ldots*11, **\ldots*10, **\cdots*01, **\ldots*00\}$. We will use strings from $\{d,*)^k$ to designate the partitions specified by defining loci.

DEFINITION. Each partition $\pi \in \{d,*\}^k$ can be assigned a unique index $j(\pi)$ by the simple expedient of substituting 1's for d's and 0's for *'s throughout the string designating the partition, treating the result as a binary integer.

DEFINITION. For an arbitrary schema s, define $j(s) = j(\pi)$, where $\pi \in \{d,*\}^k$ is the (unique) partition containing the schema s.

Thus, the partition {**...*11,**...*10,**...*01,**...*00} is named by **...**dd*, and it has the index 00...011 = 3_{10}. It follows that j(**...*11) = j(**...*10) = j(**...*01) = j(**...*00) = 3_{10}.

Definition. It will be convenient to speak of the set of partitions specified by c defining bits as being the partitions at *level* c.

To define averages over the hyperplanes, we must convert $\{1,0\}^k$ to a *sample space* by imposing a probability distribution, $p:\{1,0\}^k \to [0,1]$. Then u becomes a *random variable*, and each *event* (subset) $X \subset \{1,0\}^k$ has a well-defined *marginal expectation*,

$$u(X)=\frac{\Sigma_{x\in X}\,p(x)u(x)}{\Sigma_{x\in X}\,p(x)}.$$

The average of a set of samples drawn from X constitutes an estimate of $u(X)$. In particular, each schema s can be assigned the expectation $u(s)$.

Under an algorithm that biases p as information accumulates, p becomes a function of time, $p(t)$; $u(s,t)$ then designates the expected value of u on s under the current probability distribution $p(t)$. In the development that follows, $p(s)=\Sigma_{x\in s}p(x)$ will designate the probability of the event s under p.

Because **...** designates the whole space, u(**...**) is just the expected value of u under p. Consider now a partition specified by a single defining locus, say the partition **...**d*. The two schemas that are the elements of this partition, **...*1 and **...*0, have well-defined marginal expectations under p, u(**...*1) and u(**...*0) respectively. Using the index 1 associated with the partition **...**d*, define

$$\begin{aligned}\delta_1&=\delta(**\ldots *d)\\&=p(**\ldots *1)\left[u(**\ldots *1)-u(**\ldots **)\right].\end{aligned}$$

Roughly, δ_1 measures the departure of the marginal expectation of the elements of the partition from the overall average u(**...**). It follows at once from the definition that

$$u(**\ldots *1)=u(**\ldots **)+\frac{\delta_1}{p(**\ldots *1)}.$$

It also follows that

$$u(**\ldots*0) = u(**\ldots**) - \frac{\delta_1}{p(**\ldots*0)},$$

because $u(**\ldots**) = p(**\ldots*1)u(**\ldots*1) + p(**\ldots*0)u(**\ldots*0)$. Clearly a δ can be assigned in the same way to each 2-element partition specified by a single defining locus, yielding a set $\{\delta_1, \delta_2, \delta_4, \ldots, \delta_{2^{k-1}}\}$.

Given any schema s defined on m loci, one can partition s into two subsets by selecting one *additional* defining locus. Two schemas partitioning s, call them s_1 and s_0, result; they are defined on the selected $m+1$ loci. s_1 and s_0 play much the same role with respect to s that $**\ldots*1$ and $**\ldots*0$ played with respect to $**\ldots**$. With a little care, we can set up an induction based on this analogy. It assigns a unique δ to every indexed partition, such that for an arbitrary schema s,

$$u(s) = \delta_0 + \frac{1}{p(s)} \sum_{\substack{s' \supset s \\ s' \neq **\ldots**}} 2^{-[m(s) - m(s')]} \sigma(s') \delta_{j(s')},$$

where

$\delta_0 = u(**\ldots**)$, the expectation of u under p,
$m(s')$ = the number of defining bits in s', and

$$\sigma(s') = \begin{cases} +1 \text{ if } s' \text{ has no 0's or an even number of 0's} \\ \quad \text{in its defining bits,} \\ -1 \text{ otherwise.} \end{cases}$$

Because $\{1,0\}^k$ contains 2^k points and because there are 2^k selected partitions with associated δ_j's, it is easy to show that the hyperplane representation is unique for each distinct function u and distribution p. The inverse transform is given by

$$\delta_{j(s)} = \sum_{\substack{s' \supset s \\ s' \neq **\ldots**}} (-2)^{-[m(s) - m(s')]} p(s') [u(s') - \delta_0],$$

using a schema s for which the defining bits are all 1's.

The following two lemmas are quickly established by inspection of the transform.

LEMMA. *$\delta_{j(s)}$ contributes a positive (negative) increment to $u(x)$ for exactly half of the points $x \in \{1,0\}^k$.*

DEFINITION. A partition $\pi' \in \{d,*\}^k$ will be called a *superpartition* of $\pi \in \{d,*\}^k$ when the defining bits of π' constitute a subset of the defining bits of π. (That is, each element of the superpartition π' is the union of a distinct set of elements of π,)

LEMMA. *If s' is an element of any superpartition of $j(s)$, then $\delta_{j(s)}$ makes no net contribution to $u(s')$, the expected value of u over s'.*

Because the $u(s)$ are marginal expectations, the average of any sample drawn from s under the distribution $p(t)$ constitutes an *estimate* of $u(s,t)$. The $\delta(s,t)$, as functions of the $u(s,t)$ and the biasing probabilities $p(s,t)$, can also be estimated.

USING ESTIMATES OF $u(s)$ TO BIAS THE SEARCH OF $\{1,0\}^k$

The hyperplane averages take a particularly simple form when the function u is linear over individual loci and the probabilities assigned to individual alleles are independent of one another. We will consider this case first, and then proceed to the general case. In this discussion two useful pieces of terminology from genetics will be adopted: The positions along a string will be called *loci*, and the values that can be inserted at any position will be called *alleles*.

u is a linear function of values assigned to individual loci when

$$u(x) = \sum_h u_h(x_h),$$

where $x = x_1 x_2 \ldots x_k \in \{0,1\}^k$, and $u_h: \{0,1\} \to \text{Reals}^+$ assigns values to the two alleles at locus h. The probabilities of the individual alleles are independent of each other when the probability of an arbitrary string x is given by $p(x) = \Pi_h p_h(x_h)$, where $p_h(x_h)$ is the probability that allele x_h occurs at locus h.

Consider now the δ's at level 1, that is, the $\delta_{j(s)}$ for which $j(s) = 2^h$. Each hyperplane at level 1 is specified by the single allele at its defining locus. For example, the partition of index 2 has as elements the two hyperplanes $**\ldots*1*$ and $**\ldots*0*$. When u is linear and the probabilities of the alleles in a string are independently assigned, it is a simple exercise to show that

$$\delta_{2^h} = p_h(1)p_h(0)\left[u_h(1) - u_h(0)\right].$$

Note that the formula for δ_{2^h} involves only values and probabilities assigned to the alleles, 1 and 0, at locus h. It is also easy to show that the δ's for partitions of level higher than 1 are all zero. To determine the global optimum of u, we select the allele at each locus that corresponds to the above-average hyperplane at that locus. The string made up of the alleles so selected optimizes u. Stated another way, the optimizing string lies in the intersection of all the above-average level-1 hyperplanes. Thus, if the estimates of $u(s)$ for these level-1 hyperplanes reflect the true values, an optimal string is easily determined. It follows that linear functions can be solved via searches involving only the level-1 hyperplanes, as intuition might suggest.

In the general case, a hyperplane s supplies nontrivial information when the $\delta_{j(s)}$ associated with partition $j(s)$ is nonzero. Drawing on the terminology of mathematical genetics, the $\delta_{j(s)}$ for which $j(s)$ is *not* a power of 2 amount to *epistatic* effects—departures from the expectations that would hold if u were linear. If the epistatic effects are such that best hyperplanes are different from those determined by the linear estimate based on the level-1 hyperplanes, then, typically, the search for better values of the evaluation function will be hampered until the corresponding $u(s)$ can be estimated.

To arrive at some idea of the difficulty posed by epistatic effects, note that a hyperplane at level c can be expected to receive a fraction 2^{-c} of a uniformly distributed set of samples. Thus, about half of all samples can be used to estimate $u(s)$ for any level-1 hyperplane, while the proportion drops by half each time the level is increased by 1. Since the reliability of an estimate of $u(s)$ depends upon the number of samples s receives, the rate at which relevant information accumulates drops by half each time the level is increased by 1.

Consider, now, a function u with nonzero δ's at several different levels. By the earlier lemma, a deeper-level, nonzero δ_j has no effect on the averages $u(s')$ of elements s' of the shallower superpartitions of j. On the other hand, the effects of δ_j on $u(s)$, for any schema s in partition j can be hidden in the contributions to $u(s)$ made by nonzero δ's at shallower levels. To exploit a deeper δ_j one must find a schema s that (1) belongs to the partition j, (2) is above average under the distribution p, and (3) receives a positive contribution from δ_j. The effect of $\delta_{j(s)}$ on $u(s)$, when confounded by the contributions of shallower δ's, can only be determined when s has received enough samples to permit a reasonable estimate of $u(s)$. The deeper the level of δ_j, the longer it will take typically to accumulate information about schemas belonging to partition j. Accordingly, nonzero δ's at deeper levels typically increase the difficulty of a search.

Matters also become more complicated when the probabilities assigned to loci are no longer independent of one another. Then even linear functions may yield nonzero δ's at deep levels. Intuitively, the information supplied by a sampling procedure depends upon the underlying distribution. A biased distribution emphasizes certain regions, and the information supplied by the δ_j's is modified accordingly.

The rate at which information accumulates about the $u(s)$ dictates the order in which the search of the argument space $\{1,0\}^k$ should be biased. Under uniform random sampling, estimates of a given level of reliability accumulate first for level-1 hyperplanes, then, at half that rate, for level-2 hyperplanes, and so on. Thus we should expect a realistic search to produce biases that, at first, are largely dependent upon estimates for $u(s)$ associated with low-level hyperplanes. Biases based on higher-level hyperplanes enter as the number of samples increases. A search so directed has the concomitant advantage that a simpler problem is solved more quickly.

THE GENETIC ALGORITHM AS A HYPERPLANE-DIRECTED SEARCH PROCEDURE

(1) Description of the Genetic Algorithm

[The algorithm acts on a set $B(t)$ of M strings $\{x_1, x_2, \ldots, x_M\}$ over the alphabet $\{1,0\}^k$ with observed values $u(x_j)$. For convenience M will be taken to be an even number.]

Briefly, a genetic algorithm has the following form [6] for more details, and [3] for a wide range of variants and applications):

(1) Compute the average strength $u^{\wedge}(t)$ of the strings in $B(t)$, and assign the normalized value $u(x_j)/u^{\wedge}(t)$ to each string $x_j \in B(t)$.

(2) Assign each $x_j \in B(t)$ a probability $p(x_j, t)$ proportional to its normalized value. Using this probability distribution, select M strings from $B(t)$, forming a new population B'.

(3) Pair all of the strings in B' at random, forming $M/2$ pairs. Apply *crossover* with probability P_{cross} to each pair (and, possibly, apply other *genetic operators* such as *mutation* with probabilities P_{mut}, etc.), forming a new population B'' of M strings. *Crossover* is applied to a pair of strings as follows: Select at random a position i, $1 \leqslant i \leqslant k$, and then exchange the segments to the left of position i in the two strings (see Figure 1).

(4) Increase t by 1, set $B(t) = B''$, and return to step (1).

[Step (4) in the algorithm may be modified to prevent one kind of string from "overcrowding" $B(t)$ (see [1] and [2] for details). Contiguity of constituents, and the building blocks constructed from them, is significant under the

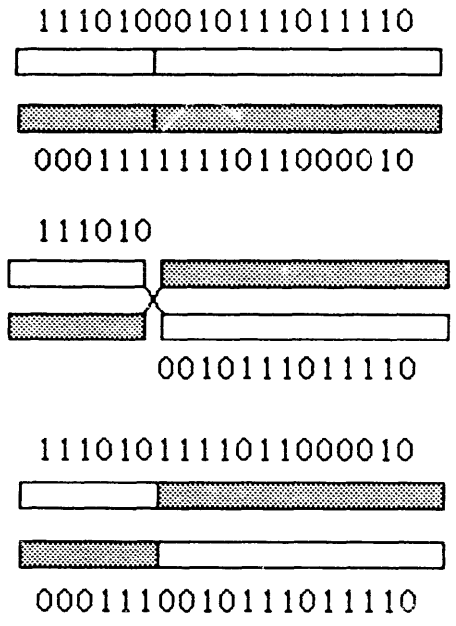


FIG. 1. An example of the crossover operator.

crossover operator. Close constituents tend to be exchanged together. Operators for rearranging the loci, such as the genetic operator *inversion*, can bias the rule-generation process so that loci that interact usefully tend to be contiguous. Other genetic operators, such as *mutation*, have lesser roles in this use of the algorithm, mainly providing "insurance" (see [5, Chapter 6, Sections 2, 3, 4] for details).]

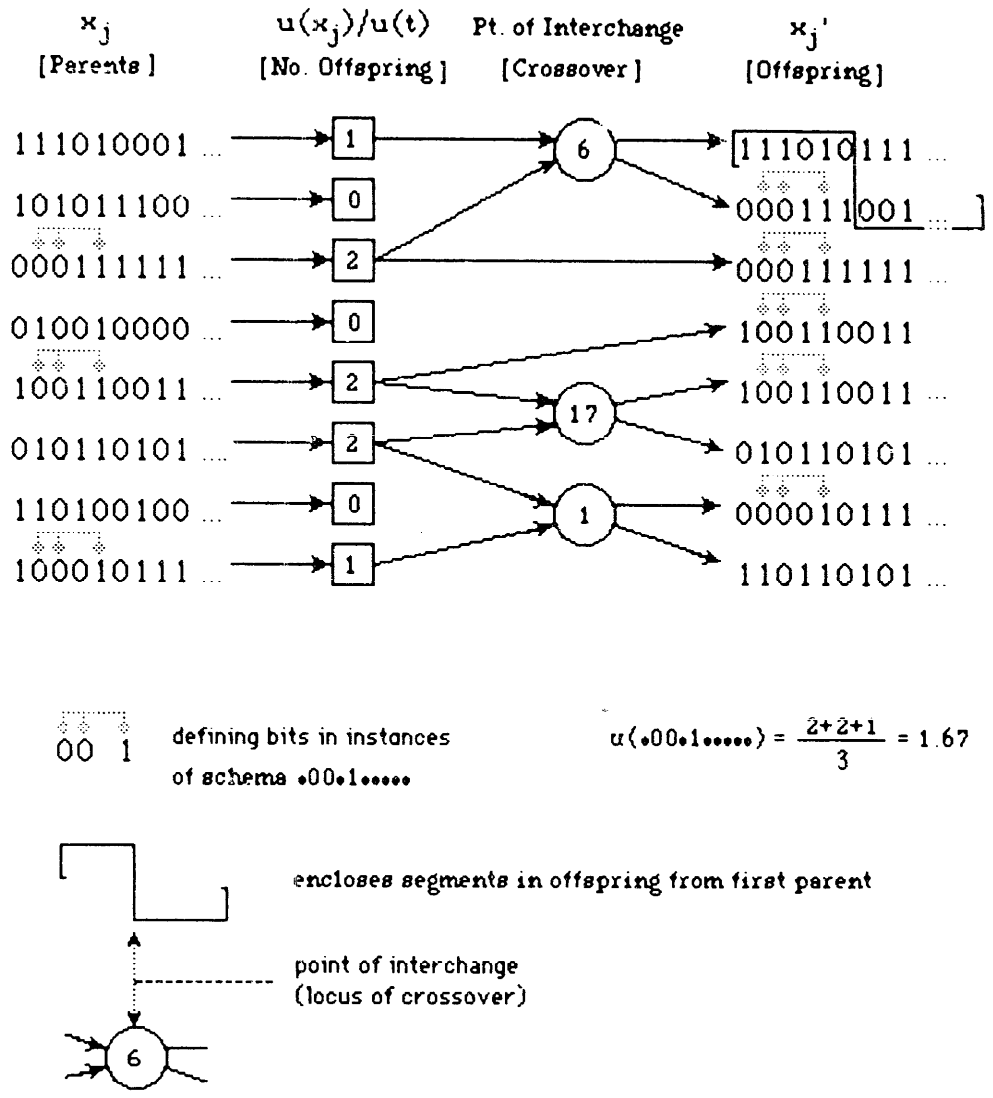
An example of the action of the algorithm is shown in Figure 2.

The fundamental theorem for genetic algorithms (see [5]) can be rewritten as a theorem about progressively biasing a probability distribution over the space $\{1,0\}^k$:

THEOREM. $p(s,t+1) \geqslant [1-\lambda(s,t)][1-P_{\text{mut}}]^{d(s)}[u(s,t)/u(t)]p(s,t)$, *where* $p(s,t+1)$ *is the expected fraction of the population that will be occupied by the instances of* s *at time* $t+1$ *under the genetic algorithm, given that* $p(s,t)$ *is the fraction occupied by* s *at* t.

The factors on the right-hand side are:

(1) $[u(s,t)/u(t)]$, *the ratio of the observed average value of the schema* s *to the overall population average.* This term determines the rate of change of $p(s,t)$, subject to the "error" terms $[1-\lambda(s,t)][1-P_{\text{mut}}]^{d(s)}$. If $u(s,t)$ is



The population consists of strings ("chromosomes") of length k from the set $\{1, 0\}^k$ based on the 2-letter alphabet {1, 0} (2 possible "alleles" at each locus).

FIG. 2. An example of the genetic algorithm acting on schemas.

above-average the proportion of schema s increases (if the error terms are small), and vice versa.

(2) *$\lambda(s,t)$ and P_{mut}, the "error" terms resulting from the breakup of instances of s because of crossover and mutation, respectively.* Specifically, $\lambda(s,t)p(s,t) = P_{cross}[l(s)/(k-1)]p(s,t)$ is an upper bound on the *crossover loss*, the loss of instances of s resulting from crosses that fall within the interval of length $l(s)$ determined by the outermost defining loci of the schema. $[1-P_{mut}]^{d(s)}$ gives the proportion of instances of s that escape a mutation at one of the $d(s)$ defining loci of s.

(The underlying algorithm is stochastic, so this equation only provides a bound on expectations at each time step. Using the terminology of mathematical genetics, this equation supplies a *deterministic model* of the algorithm under the assumption that the expectations are actually achieved on each time step.)

Proof outline (see [5] for details).

(1) Consider a schema with $M(s,t)$ instances in the population $B(t)$. The average value of these instances is given by $u(s,t) = \Sigma_{x \in s} u(x)/M(s,t)$. If each of these instances is copied with probability $u(x)/u^{\wedge}(t)$, there will be $\Sigma_{x \in s} u(x)/u^{\wedge}(t) = u(s,t)M(s,t)/u^{\wedge}(t)$ instances of s expected in B' after the copying. (The actual number of course will be subject to sampling error).

(2) When the point of crossover falls within the outer limits of the defining positions for a schema, the defining bits of the schema will be separated in the offspring (otherwise they are passed on intact). Under such circumstances, it is possible (but not necessary) that neither offspring is an instance of the schema, so that there is a "loss" of one instance in the process. Because the point of crossover is chosen at random in each case, the probability that the cross falls within the outer defining positions is given by $l(s)/(k-1)$, where $l(s)$ is the number of crossover points between the outer defining positions of s. Thus $P_{cross}[l(s)/(k-1)]$ gives an upper bound on the probability that a given instance of s will be lost because of crossover during the formation of B." A similar calculation provides the loss rate because of mutation.

(3) It follows that the number $M(s,t+1)$ of instances of s to be expected after copying and crossover is bounded below by

$$[1-\lambda(s,t)][1-p_{mut}]^{d(s)}\frac{u(s,t)}{u(t)}M(s,t).$$

(4) But $p(s,t)$, the fraction of instances of s in the population $B(t)$, is by definition $M(s,t)/M$, so that

$$p(s,t+1) \geqslant [1-\lambda(s,t)][1-P_{\text{mut}}]^{d(s)}\frac{u(s,t)}{u(t)}p(s,t). \qquad \blacksquare$$

In any population that is not too small, distinct schemas will almost always have distinct subsets of instances if the number of instances is relatively small. For example, in a randomly generated population of size 2500, any schema defined on 8 loci can be expected to have about 10 instances. There are

$$\binom{2500}{10} \simeq 3\times 10^{27}$$

ways of choosing this subset, so that it is extremely unlikely that the subsets of instances for two such schema will be identical. (Looked at another way, the chance that two schemas have even *one* instance in common is less than $10\times 2^{-8} = \frac{1}{25}$ if they are defined on disjoint subsets of loci.) Because the sets of instances are overwhelmingly likely to be distinct, the observed averages $u^{\wedge}(s,t)$ will be determined mostly by independent samples. As a consequence, the rate of increase (or decrease) of a schema s under a genetic algorithm is largely uncontaminated by the rates associated with other such schemas. Loosely, the rate is uninfluenced by "crosstalk" from the other schemas.

From the point of view of sampling theory (applied to populations large enough that sampling without replacement is insignificantly different from sampling with replacement), 20 or 30 instances of a schema s constitutes a sample large enough to give some confidence to the corresponding estimate of $u(s)$. Thus, for such schemas, the biases $p(s,t)$ produced by a genetic algorithm over a succession of generations are neither much distorted by sampling error nor smothered by "crosstalk."

To gain some idea of how many schemas are so processed, consider the following:

THEOREM. *Select some bound e on the crossover error, and pick k' such that $k'/k \leqslant e/2$. (The theorem is only of interest when $ek/2 \gg 1$.) Consider a population of size $M = c_1 2^{k'}$, where c_1 is a small integer (say $c_1 < k^{1/3}$). If M is obtained as a uniform random sample from $\{1,0\}^k$, the number of schemas propagated with an error less than e greatly exceeds M^3.*

Proof outline.

(1) Consider a "window" of $2k'$ contiguous loci in a string of length k such that $2k'/k \leqslant e$. Clearly any schema having all its defining loci within this window will be subject to a crossover error less than e.

(2) There are

$$\binom{2k'}{k'} \simeq \frac{2^{2k'}}{(\pi k')^{-1/2}}$$

ways of selecting k' defining positions in the window, and there are $2^{k'}$ different schemas that can be defined using any given set of k' defining loci. Therefore, there are approximately $2^{3k'}/(\pi k')^{-1/2}$ distinct schemas with k' defining positions that can be defined in the window.

(3) A population of size $M = c_1 2^{k'}$ obtained by a uniform random sampling of $\{1,0\}^k$ can be expected to have c_1 instances of *every* schema defined on k' defining positions. Therefore, for the given window, there will be approximately $M^3/c_1^3(\pi k')^{-1/2}$ schemas having instances in the population and defined on some set of k' loci in the window.

(4) The same argument can be given for schemas of length $k' - 1$, $k' - 2, \ldots,$ and for $k' + 1, k' + 2, \ldots,$ with values of

$$\binom{2k'}{k' \pm j}$$

decreasing in accord with the binomial distribution. There are also $k - k' - 1$ distinct positionings of the window on strings of length k. It follows that many more than M^3 schemas, with instances in the population of size M, increase or decrease at a rate given by their observed marginal averages with a crossover error less than e. ∎

A genetic algorithm's ability to meaningfully bias the sampling rate of a large number of schemas while processing a relatively small set of instances is called *implicit parallelism* (né *intrinsic parallelism* [5]).

(2) Effects of the δ's on the Search Generated by a Genetic Algorithm

The fundamental theorem makes it clear that the biases $p(s, t+1)$ produced by a genetic algorithm at time $t+1$ depend directly upon the observations $u^\wedge(s,t)$ and biases $p(s,t)$ at time t. The hyperplane transform **H**, applied to the sample space defined by the new distribution $p(t+1)$, determines a new set of $\delta_i(t+1)$.

Consider a function that is complex enough that a random sample of M arguments is very unlikely to contain an argument that yields the optimum value of the function. (Typically a function with m or more nonzero δ's will be such a function if $\log_2 M \ll m$.) Let partition i be a deeper partition with a large associated $\delta_i(t+1)$. By the earlier lemmas, $\delta_i(t+1)$ will contribute positively to $u(s)$ for half the elements s of this partition, but it will not contribute to $u(s')$ for any elements s' of superpartitions of partition i. It follows that $\delta_i(t+1)$ can be exploited only if the population $B(t+1)$ contains one or more instances of a schema s in partition i for which (1) $\delta_i(t+1)$ makes a positive contribution (i.e., it has the appropriate sign), and (2) the contributions of the δ's from shallower levels are such that $u(s)$ is above average. If there is no instance of such an s in the population at $B(t)$, then s can only be formed by recombination (crossover) or mutation. That is, if there are no instances of such an s in the population, $\delta_i(t+1)$ can be exploited in the near future only if the schema s can be reached from currently exploited hyperplanes via a few recombinations and mutations.

By looking at the levels in which the nonzero δ's are distributed, one can attain a qualitative understanding of the trajectory induced by a given nonlinear function u. Consider again an initial population $B(0)$ that has been generated using a uniform random distribution over $\{1,0\}^k$. If the size M of that population is 2^m, then we can expect multiple copies of all schemas with fewer than m defining loci. If, as earlier, we set a bound e on the error produced by the genetic operators, requiring $[1-\lambda(s,t)][1-P_{\text{mut}}]^{d(s)} < e$, then in excess of M^3 of these schemas will be processed with an error less than e (for an appropriately chosen M and a sufficiently small mutation rate). Because the above-average schemas increase their instances exponentially (with exponent $[1-e][u(s,t)/u(t)]$), they soon come to occupy a substantial fraction of the population. Sampling then is concentrated on these hyperplanes and their intersections (though not exclusively).

While the above-average schemas with instances in the initial population are being exploited, new schemas exploiting deeper δ's are discovered in two ways: (1) mutation of loci that are near the loci defining one of the initially exploited schema, (2) recombination, under crossover, of fragments of the initially exploited schemas. These new schemas will join the set of schemas that occupy a substantial fraction of the population, and hence are sampled intensively, only if $[1-e][u(s,t)/u(t)] > 1$.

Each of these discovery processes is worth looking at in more detail:

Mutations of loci contiguous to the defining loci of an above-average schema s can provide instances of schemas not previously present in the population. Each such schema, s', is a refinement of s (i.e., s contains s') and hence is an element of a deeper level partition. Consider, then, a partition of s composed of 2^h hyperplanes obtained by specifying the values for some set of h loci contiguous to s. Because the mutations are allocated randomly, the number of samples $n(s')$ of s' will be approximately $2^{-h}n(s)$, where $n(s)$ is the number of samples allocated to s. The *central limit theorem* assures that the averages of different sets of samples of any s' will be distributed approximately as a Gaussian distribution, whatever the probabilities assigned to the elements x in s'. The sampling process (mutation operator) thus produces an estimate of $u(s')$ with a variance that decreases as $\sqrt{2^{-h}n(s)}$. Very roughly, then, one would expect to reliably discover s' for which $u(s') > u(s)$ at a rate on the order of $\sqrt{2^{-h}n(s)}$. In other words, mutation will discover improvements in the vicinity of s at a rate that falls off as the square root of the number of samples allocated to s. In biological terms, this would correspond to an *adaptive radiation* wherein variants of the prototype s provide incremental improvements.

This process of "exploring the neighborhood" via mutations contrasts sharply with the jumps produced by crossover. To develop the contrast, consider a randomly generated population with instances of two above-average schemas s_1 and s_2, where $d(s_1) \geqslant d(s_2)$. Let the defining bits of s_1 and s_2 be such that there are no instances of the schema s designating the intersection at s_1 and s_2, and let $u(s) > \max\{u(s_1), u(s_2)\}$.

Under these conditions, on the order of $d(s_2)/2$ mutations must accumulate in some instance of s_1 before an instance of s appears in the population. The mutation rate can of course be increased to make the accumulation more rapid, but only at the cost of making it increasingly *unlikely* that s will be "copied" into successive generations (see the example just below). Mutations tend to explore in a linear way—the depth of the exploration is a linear function of the number of generations elapsed.

On the other hand, a single crossover between parents that are instances of s_1 and s_2, respectively, can yield an instance of s. That is, an above-average schema s at depth $2b$ that falls in the intersection of established schemas s_1 and s_2 at depth b can be discovered in a single generation. This doubling of depth in successive generations comes about whenever established schemas can be combined as "building blocks" to yield improved schemas. Under these conditions, crossover explores with directed exponential increases in depth.

(3) An Example

The following example gives a quantitative measure of the difference between mutation and crossover:

Let schema s be defined on contiguous loci, i.e. $l(s) = d(s) - 1$. Divide the defining loci of s into disjoint subsets of contiguous loci, forming schemas s' and s'' that have s as their intersection. Let $e(s)$ be an upper bound on the allowable rate of transcription errors for s. [That is, if the transcription error rate exceeds $e(s)$, then even an above-average schema s with an instance in the population $B(t)$ is unlikely to have instances that persist through successive generations.]

Let s^* be any schema properly contained in s' and properly containing s. Assume that for all such $s^*, u(s') > u(s^*)$, while $u(s) > u(s')$. That is, the schema s' is "surrounded" by a "desert" of lesser-valued schemas $\{s^*\}$, at the edge of which is a "higher peak" s. For ease of calculation, assume that s' has a single persistent instance in the population.

Consider, first, the time expected to elapse, under mutation alone, before the "peak" s is attained from s'. Note that the bound on transcription error sets an upper bound on the mutation rate because $(1 - P_{\text{mut}})^{d(s)} > 1 - e(s)$, and this yields the inequality

$$P_{\text{mut}} < \frac{e(s)}{d(s)}.$$

For an instance of s' to be an instance of s, the alleles at all the defining loci $D(s'')$ of s'' must match the corresponding alleles of s. Schemas s^* that are refinements of s' (but not equal to s) cannot persist, because $u(s^*) < u(s')$. As a consequence, there is no way for instances of s' to gradually accumulate the mutations that "lie on the way" to s. To attain s all the required mutations must occur simultaneously in some single instance of s'. In a randomly generated population, we can expect that half of the alleles at the loci $D(s'')$ match the corresponding alleles of s. Thus $d(s'')/2$ mutations must occur simultaneously in some instance of s'. This will occur with probability

$$(P_{\text{mut}})^{d(s'')/2} < \left(\frac{e(s)}{d(s)}\right)^{d(s'')/2},$$

with a corresponding search time

$$T_{\text{mut}} > \left(\frac{d(s)}{e(s)}\right)^{d(s'')/2}.$$

If we look to the corresponding calculations for crossover we first find that the bound on transcription error sets a bound

$$1 - P_{\text{cross}} \frac{l(s)}{k} > 1 - e(s),$$

or

$$P_{\text{cross}} \frac{l(s)}{k} < e(s),$$

which implies that

$$k > P_{\text{cross}} \frac{l(s)}{e(s)} = P_{\text{cross}} \frac{d(s) - 1}{e(s)}$$

using the fact that $l(s) = d(s) - 1$ for schema s. Under crossover, an instance of s' will have an offspring that is an instance of s if two things happen: (1) the other parent in the cross is an instance of s'', and (2) the cross occurs exactly at the juncture of s' and s''. In a randomly generated population, a randomly selected parent will be an instance of s'' with a probability $2^{-d(s'')}$, and the crossover will occur at the required juncture with probability $1/k$. Thus the overall event will occur with probability

$$\frac{2^{-d(s'')}}{k} = \frac{2^{-d(s'')}}{P_{\text{cross}}\{[d(s) - 1]/e(s)\}}$$

with a corresponding search time

$$T_{\text{cross}} < \frac{d(s)}{e(s)} 2^{d(s'')}$$

using the fact that $P_{\text{cross}} \leqslant 1$.

Even for relatively small "deserts" these waiting times are enormously different. For example, with $d(s') = 16$, $d(s'') = 8$, and $e(s) = 0.1$,

$$T_{\text{mut}} > 3.4 \times 10^9$$

and

$$T_{\text{cross}} < 6.2 \times 10^4.$$

COMMENTS

Most searches of complex spaces conducted by genetic algorithms should exhibit the same broad characteristics:

At the outset, there should be a rapid biasing of sampling probabilities toward hyperplanes s that

(1) have instances in the initial population, and
(2) have an average value $u^\wedge(s,0)$ sufficiently above the population average $u^\wedge(0)$ to overbalance errors in copying s induced by the genetic operators.

In particular, if the mutation rate is low, this is a requirement that the "length" $l(s)$ of the schema be short enough that $l(s)/k < [u^\wedge(s,0)/u^\wedge(0)] - 1$. If the observed $u^\wedge(s,0)$ is a good estimate of the actual marginal expectation $u(s)$, then the number of instances of s will increase exponentially in subsequent generations.

As the exponential increase continues, the increasing number of instances of above-average schemas can be expected to drive the population average $u^\wedge(t)$ upward. For any given s this forces $u^\wedge(s,t)/u^\wedge(t)$ ever closer to 1. The exponential increase of s then tapers off. Eventually, unless s represents the best that can be achieved, continued increases in the population average cause a decrease in the number of instances of s, unless some refinement of s offers further improvement.

During the time that a schema has a large number of instances in the population, the genetic algorithm acts to provide many *new* instances of it. The mutation operator treats the schema as a focal point, generating a variety of new instances in its "neighborhood." Crossover, and other forms of recombination, generate new instances by treating the schema as a "building block" that can be used in combination with other building blocks.

The fate of a newly discovered instance of a deep, above-average schema s depends upon the manner of its discovery. Consider a schema s with a length $l(s)$ large enough to indicate a large crossover error. Under normal circumstances, this crossover error would be enough to quickly destroy instances of s. However, if s has been discovered by recombination of well-established building blocks, things happen differently. The well-established building blocks occupy large fractions of the population, so the parents are likely to hold several building blocks in common. As a result, crosses that normally would break up instances of s now just recreate s, because they exchange pieces of identical building blocks. Thus, s increases its representation despite the large crossover error.

Generally, crossover is the operator that can be expected to yield substantial improvements based on deeper δ's. Crossover implements the heuristic that "good" structures are constructed of "good" building blocks (cf. Simon's [7] discussion of the architecture of complexity). This amounts to a conjecture that new nonzero δ's are associated with the *intersections* of hyperplanes already known to be associated with nonzero δ's. Of course, the conjecture may prove untrue for many intersections, but it need only be true upon occasion for improvements to be made.

It should be recalled that improvement is the object of the search; the global optimum may involve δ's so deep that they will never be uncovered in feasible times. Implicit parallelism, by assuring that the genetic algorithm usefully searches large numbers of schema combinations in each successive generation, makes it likely that *some* useful intersections will be uncovered. It *is* possible to design a function u that often "guides" the genetic algorithm away from good regions, but it is hard to design a function that keeps the algorithm away from improvements over long intervals.

Overall, and in qualitative terms, the search of a complex nonlinear function with nonzero δ's at many levels typically exhibits continual small improvements, punctuated by saltations to schemas involving deeper δ's. This behavior is a direct consequence of the manner in which genetic operators exploit sparse deeper δ's. From a biological perspective, it is interesting that this succession of "punctuated equilibria" occurs without the intervention of higher-order selection principles.

REFERENCES

1 A. D. Bethke, Genetic Algorithms as Function Optimizers, Ph.D. Dissertation, Univ. of Michigan, Ann Arbor, 1980.

2 K. A. DeJong, Adaptive system design—a genetic approach, *IEEE Trans. Systems Man Cybernet.* 10:9 (1980).

3 J. J. Grefenstette, *Genetic Algorithms and Their Applications*, Erlbaum, Hillsdale, N.J., 1987.

4 D. O. Hebb, *The Organization of Behavior*, Wiley, New York, 1949.

5 J. H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, Ann Arbor, 1975.

6 J. H. Holland, K. J. Holyoak, R. F. Nisbett, and P. R. Thagard, *Induction: Processes of Inference, Learning, and Discovery*, MIT Press, Cambridge, Mass., 1986.

7 H. A. Simon, *The Sciences of the Artificial*, MIT Press, 1981.