This report is reproduced in two parts:
Part I goes to page 150
Part II starts at page 151

TABLE OF CONTENTS

LIST OF TABLES

page

Table

LIST OF ILLUSTRATIONS

LIST OF APPENDICES

page

CHAPTER V

THE LEARNING ALGORITHM

The algorithm for modifying strength discussed in the previous chapter is an important mechanism enabling the organism to achieve "... adaptation to the environment under sensory guidance" [Hebb,1972, p. 78]. Classifiers which prove valuable to the system for recognizing stimulus patterns, finding needed resources, or avoiding noxious objects are rewarded by an increase in strength. This increase in strength gives a classifier a higher probability of becoming active and controlling the behavior of the system. In this way, those classifiers most effective at generating functionally significant behaviors are used more and more frequently - resulting in behavior that is better adapted to the given environment. This is, however, adaptation only in the limited sense of "fine tuning" existing capabilities. If the repertoire of classifier taxa and tags in the system is poorly suited to the environment, adjusting strengths will lead to only marginal improvements in behavior. This points out the need for a learning algorithm capable of modifying classifier taxa and tags in appropriate ways.

Criteria for a Mechanism

Any consideration of learning in terms of adaptation to the environment requires, not surprisingly, a clear understanding of the organism's relationship with its environment. Before we can answer the question "How should the system learn?", it is important to have a well specified task description of what is to be learned.

> ... the question must be broken down into two questions.... The first of these takes the form: "What are the behavioral problems that this animal must solve in adapting to its environment?" The second question takes a more familiar form: "How [can] the animal learn to solve those problems?" Not only are questions of the first kind equal in importance to those of the second, they also clearly have logical priority.... Writing the task description may be a major part of the whole endeavour. [Johnston,1981, p. 133]

The challenges posed by the simulated environment for our hypothetical organism are straightforward: The organism must learn to recognize the structural regularities or patterns in the available stimulation that are functionally significant; then, it must learn which of these patterns indicate the presence of a resource and which indicate a noxious object. Moreover, the organism must learn which response - approach or avoid - is appropriate for each kind of object. Stated in terms of classifiers, the system must generate input taxa that are correlated with - and in that sense "represent" - the important patterns underlying the messages appearing on the message list. The system must learn which tags are significant and which combinations of input taxon and tag yield consistent and useful

classifications. Additionally, the system must discover which message taxa are most effective for a given combination of input taxon and tag. This specification of what the system has to learn points out the importance of one capability: Given some class of binary strings, the system must learn - based on repeated exposure to representatives or *exemplars* of the class - which structural dependencies among string attributes are characteristic of the class as a whole.

Having identified what the system has to learn, it is appropriate to consider how the system might learn it. Given the structure of classifier systems in general and the nature of the learning problem posed by the simulated environment, several criteria for a learning mechanism come to mind:

1) Obviously, the learning mechanism must infer useful generalizations - in the form of taxa - from the steady stream of messages received and transmitted. The mechanism must also infer the binary tag which is most characteristic of a given message configuration; and, therefore, most appropriate for a given set of classifiers.
2) The mechanism must be computationally efficient. The system has a relatively limited number of classifiers to work with and a limited amount of time - the organism must find resources or die - in which to make the required inferences. Equally important is the

fact that the behavior of the system can only be properly observed and analyzed when it is performed in real time.

3) All modifications must be made while the system continues to behave. This means, in particular, that existing classifiers known to be useful should be preserved by the learning process.

4) Pattern classes in the simulated environment are defined on the basis of combinations of attribute values. The value of a classifier depends on the combination of input taxon, tag, and message taxon. It is therefore important that the learning mechanism be capable of discovering meaningful combinations of pattern attributes and classifier components. Many of the well known approaches to learning in artificial systems (e.g. Samuel [1959] and Winston [1975]) assume that the parts of a representation make an independent or linear contribution to the whole.

5) Because a population of classifiers has a fixed size, a new classifier cannot be inserted until an existing one is deleted. The learning mechanism must decide how much of the population will be allocated to each kind of classifier so that the population as a whole accounts for all the important facets of the system's experience with the environment. This need to manage limited resources is an often overlooked constraint in information processing models (e.g. Cunningham and

Gray [1974]).

This is a rather stiff set of criteria. The challenge is to address these concerns in a way that keeps the system practical while at the same time avoids solutions that are ad hoc. It is very easy to design learning procedures which are no more than random "generate and test" searches of the set of possible structures (e.g. Fogel, Owens, and Walsh [1966]); or, which are not sufficiently powerful for the task domain under consideration (e.g. see Minsky and Papert's [1969] analysis of the perceptron model). How is one to know when a given learning heuristic is powerful and efficient enough for a given task? At this point a theoretical framework is an indispensable part of the design process.

A Theoretical Framework

Adaptive processes can be found in many subject areas. In genetics, for example, there is the evolutionary process in which the selective pressures of the physical world prescribe a set of viable genotypes. In control theory there are adaptive control policies that use successive approximation techniques to minimize the discrepancy between the actual state of a process being controlled and the desired state. In psychology there is the central nervous system, continually reorganizing and tuning itself so as to generate satisfactory behavior across a broad range of situations. All of these processes have a common

characterization: structures undergoing modifications so as to improve their performance.

Holland [1975] has reviewed adaptive processes in their many forms and provides a formal framework in which problems in adaptation can be rigorously defined and analyzed. Briefly, in this framework a problem in adaptation is well posed once the following have been specified:

E the set of possible environments the system might have to adapt to

P the set of feasible adaptive plans, or strategies for modifying the system structures, appropriate to the problem at hand

X the criterion for determining when one adaptive plan is better or worse than another

Given a well posed problem in adaptation, a particular adaptive system is specified by giving the following formal objects:

A the set of structures undergoing adaptation together with all potential alternative structures

W the set of operators for modifying structures. These are the mechanisms that transform one member of *A* into another

I the set of all possible inputs from the environment to the system This information allows the structures in *A* to be ranked and makes comparisons among structures possible.

p a particular adaptive plan belonging to *P*. Each such plan has the form

$$p: (I \times A) \rightarrow A$$

Given a structure and input value the plan determines when and how to modify that structure,

using elements of W, to produce a new one.

Consider, for example, how one might use this formalism to pose a problem in population genetics. The environment E is the set of environmental niches available to be exploited by individuals with the appropriate phenotype. The adaptive plans P are rules for reproducing individuals according to their fitness so that offspring are genotypic and phenotypic variants of their parents. A possible criterion X might be to compare plans based on the average fitness of the populations they produce. Each structure in A is a population of chromosomes. The admissible operators W are the familiar genetic operations of crossover, mutation, inversion, etc. I is the fitness function, indicating the ability of an individual to survive and reproduce. Finally, a class of adaptive plans P might specify replicating each chromosome according to its fitness, then applying the genetic operators with predetermined probabilities to generate the offspring. A particular plan p is then a specification of the operator probabilities.

Given this framework, an adaptive process can be discussed from two points of view. First, the process can be seen in terms of a trajectory thru A from an initial structure to some structure that is, presumably, ranked higher. At each point in the trajectory the adaptive plan determines which element of A will be tested next and, hence, which direction the trajectory will follow. This point of view emphasizes that an adaptive process is

basically a search process and that an adaptive plan is a search heuristic. For most problems of interest, adaptive plans that search *A* at random are not practical. Though such plans are guaranteed to eventually find the best structures in *A*, they require too much time to do so when *A* is very large. The criterion *X* therefore usually requires that an adaptive plan discover the better structures in *A* within some "reasonable" amount of time. One way for a plan to achieve such efficiency is to make hierarchical use of previous results [Minsky,1963]; that is, the ranking of structures in *A* already tested by the trajectory can be used to infer which regions in the search space should be explored next. An adaptive plan can use this global information to steer the trajectory towards the regions of *A* most likely to contain the best structures. This requires that the plan be capable of identifying structural properties of the elements of *A* that allow *A* to be usefully divided into regions; and, that the plan be capable of ranking and comparing regions of *A* just as it does with individual elements of *A*.

For most problems of interest, *A* is an extraordinarily large search space [Holland,1975]. Unless each element in *A* is tested, there is always some degree of uncertainty as to whether the best structure found at any given stage of the search is in fact the best structure in *A*. Finding the best or optimum structure is not always an important issue for an adaptive system [Simon,1969]. Still, the fact remains that,

when A is large, time constraints restrict the set of practical trajectories to those that test only a limited subset of A. Consequently, a second way of viewing an adaptive process is as a sampling procedure. Given that all an adaptive plan "knows" about A is based on a relatively small sample, and that samples are subject to error, it follows that an adaptive plan is faced with two seemingly conflicting objectives [Holland,1975]. First, it must test structures that might help reduce the sampling error in a given region. The more structures a plan tests in a region, the more likely it is that the plan's evaluation of the relative ranking of that region will be accurate. This implies that it is advantageous to sample each region many times. Moreover, when a region is discovered that is highly ranked, repeated sampling "exploits" the region in the sense that the best structure in the region is more likely to be found. On the other hand, regions of A not yet explored might contain the key to discovering even better structures. An adaptive plan must therefore continually test new regions of A and reduce the likelihood that promising regions will be overlooked. This tension between exploiting regions already discovered and exploring new regions must be carefully balanced if an adaptive plan is to reliably and effectively generate improvements in the face of uncertainty.

How does all this relate to classifier systems? We can pose the problem in adaptation faced by our hypothetical

organism as follows:

E the set of all pattern classes defined over binary strings of appropriate length. A class is defined by designating certain combinations of bit positions and values as necessary and sufficient for class membership.

P the set of algorithms for inferring a wide variety of such patterns directly from a limited set of binary string exemplars.

X the basis for evaluating classifier system behavior. The organism model must identify all objects in the simulated environment and generate an approach or avoid response as appropriate. Adaptive plans will be evaluated according to the rate at which behavioral errors are reduced.

A the set of all possible populations containing a fixed number M of classifiers

I the match score of a classifier with a given signal, together with the tag score and reinforcement when available.

What remains is to specify a set of operators *W* and an adaptive plan *p*. These must be chosen while keeping in mind the criteria set forth above.

Foundations for an Adaptive Plan

Before an adaptive plan can be devised to generate improved populations of classifiers, it is advisable to first look for plans capable of discovering potentially useful classifier components. The basic learning requirement for the classifier system has been characterized in terms of one capability: inferring structural dependencies among attributes in a class of binary strings

which are characteristic of the class as a whole, based on repeated exposure to exemplars of the class. Class membership as defined in the simulated environment is predicted by sets of coadapted signal attributes. A classifier input taxon' represents a hypothesis about the structure of a pattern class, indicating which combinations of attribute values are characteristic of the class and which values are irrelevant. Any heuristic capable of inferring environmentally relevant input taxa directly from signals in the environment therefore satisfies the fundamental requirement for the desired learning procedure. This suggests that a reduction in scope is appropriate. First, algorithms will be considered that can be used to infer structural characteristics - in the form of input taxa - from a class of binary strings. These algorithms will then be evaluated in terms of their ability to generate populations of taxa and, eventually, populations of classifiers. There are basically two kinds of algorithms available that have the desired capability. One is the interference matching algorithm described by Hayes-Roth [1973,1974] and the other is Holland's [1975] class of genetic algorithms.

Interference Matching. Interference matching is a general technique for inferring the structural characteristics some set of items have in common. The

---

'Recall that the input taxon specifies the set of binary signals capable of activating a classifier.

procedure is straightforward for binary strings. For each attribute, if all the strings have the same value at a given position, that value belongs in the structural characteristic. If the strings have different values at a given position, use a * in the characteristic. Thus the strings 101110 and 110100 yield the characteristic 1**1*0. More generally, consider some pattern class defined by an unknown set of characteristics, together with a set of known exemplars. A set of characteristics accounting for all the exemplars, called a "maximal decomposition", can be generated by the following simple procedure given by Hayes-Roth [1973]:

Let D be the list of characteristics

Let $\{E_1, \ldots, E_N\}$ be the set of N exemplars

1) Let i = 1 and initialize D to be empty

2) If D is not empty,

For j = 1, 2, ..., size of D

a) Form a characteristic d' by interference matching E_i with the j^{th} element of D.

b) Form a new characteristic d by interference matching all exemplars that match d'

c) If d is not on the list D, add it to the list. [Note that d becomes another element to be operated on by E_i, so steps 2a - 2c will be repeated]

3) Add E_i to D unaltered

4) If i = N, stop. Otherwise increment i by 1 and repeat

steps 2 and 3.

Exemplars	Maximal Decomposition
1001	1001
1110	---
0101	1***
0010	1110

	**01
	****
	*1**
	0101
	---
	*0**
	**10
	0***
	0010

Figure 5.1. An example of the Hayes-Roth [1973] algorithm for generating maximal decompositions. On the left is the set of exemplars. On the right is the list D of characteristics generated, shown in the order they are added to the list.

An example of this procedure is given in Figure 5.1. The list generated is a decomposition of the set of exemplars because each characteristic identifies a structural cluster in the set of exemplars. It is a maximal decomposition in the sense that all plausible clusters are accounted for in a non-redundant way. In Figure 5.1, for example, the characteristics ***1 and **0* are redundant for the given set of exemplars and are better summarized by the higher

order - that is, more restrictive - characteristic **01. The list D represents, in the most economical way, each set of characteristics that might plausibly define the pattern class represented by the exemplars.

The classification problems of interest here involve more than one unknown pattern class C_1, ..., C_M with exemplars E_{ij} for each class C_i. What is required is several sets of characteristics that correctly classify novel test items. To handle such situations, Hayes-Roth defines a performance value for each characteristic d_{ij} in the maximal decomposition D_i of the exemplars for class C_i. This value is a weighted difference of the expected number of correct versus incorrect classifications using only the given characteristic to classify the entire set of exemplars. In Figure 5.1, for example, the characteristic 1001 matches only one exemplar. It therefore makes one correct classification and three incorrect classifications over the given set of exemplars and will likely have a negative performance value. If additional pattern classes are defined, and their corresponding sets of exemplars don't include 1001, then the characteristic 1001 will correctly classify the new exemplars as non-instances of its class and thereby increase its performance value. The characteristic **** classifies every exemplar as a member of its class. Its performance over all exemplars for several pattern classes is likely to be near zero, the characteristic being correct as often as it is incorrect. In general, the most

highly valued characteristics are those d_{ij} that classify all exemplars of C_i as instances of C_i and all exemplars of other classes as non-instances. To classify any new test item, one need only sort each maximal decomposition D_i by performance value. The list containing the best performing characteristic d_{kj} that matches the test item indicates the best classification decision C_k.

The obvious drawback to using this procedure is that when the number of exemplars is large and each exemplar is of moderate length, the number of characteristics in each maximal decomposition is astronomical. Hayes-Roth [1973] suggests several heuristics for keeping the size of the set under control. The most useful one from the standpoint of classifier systems is to limit the number of characteristics - or, equivalently, classifier taxa - to those M with the highest performance values, either in each list D_i or overall. Such a strategy causes problems, however, if new exemplars are introduced after a set of taxa has been generated and pruned. This is a likely occurrence for our classifier system because as it behaves in the simulated environment, it will continually encounter new exemplars of resource and noxious patterns. One of the discarded low performing taxa might, in light of the new information, turn out to be an important taxon after all. This relates to the possibility of sampling error noted above. The Hayes-Roth algorithm can only correct such an error by recomputing the entire maximal decomposition. It is not clear how the

algorithm could be modified to handle a growing set of exemplars and still avoid the enormous costs associated with updating and storing maximal decompositions.

Genetic Algorithms. Holland's [1975] genetic algorithm seems much better suited to handling this problem. Like Hayes-Roth's interference matching technique, genetic algorithms use a working memory containing M structures. Rather than choosing the M structures on the basis of an exhaustive consideration of all relevant elements of *A*, however, genetic algorithms choose the structures by drawing a non-uniform random sample from *A*. The working memory is a sample of *A* in the sense that every structure provides partial information about the average performance value of those regions in *A* to which it belongs. Genetic algorithms use this information to efficiently revise the given set of structures. As new exemplars are introduced, only the performance values of the taxa in working memory need to be updated. A genetic algorithm can then be used to produce a set of taxa that is an increasingly non-uniform sample of *A*, the bias being toward those subsets of *A* with high expected performance. The elegance of genetic algorithms stems from the fact that they discover the taxa having the best expected performance over all exemplars without ever explicitly considering the entire maximal decomposition or computing expected performance values.

In order to understand how genetic algorithms work, it is helpful to consider first how a non-uniform sampling

procedure can be an effective search heuristic. In the previous discussion of adaptive plans as search procedures it was noted that, if a plan is to be efficient, the elements of A must have structural properties enabling the plan to usefully divide A into subregions. The space A being considered here is the set of all taxa, or strings of length k in the alphabet {0,1,#}. To make the explanation more precise, assume that every taxon has some well defined performance value and consider a particular set of regions in A called *schemata* [Holland,1975]. A schema is defined over taxa in a manner similar to the way pattern characteristics in the simulated environment were defined over binary strings. More specifically, values at certain positions in a taxon of length k are designated to be the "defining" properties of a schema. A schema is then the collection of all taxa having the required values at the indicated positions. For example, we might define a schema as the set of all taxa with a # in position 1 and a 0 in position 2. This schema can be compactly specified by the string #0¤¤...¤¤, where the symbol ¤ indicates positions not crucial to the definition of the schema. Similarly, the string ¤##...## specifies the three taxa 0##...##, 1##...##, and ###...##. In fact, any string of length k in the alphabet {0,1,#,¤} specifies a schema. Taxa belonging to the subset of A designated by a schema are said to be *instances* of that schema. Each taxon is an instance of exactly 2^k distinct schemata. The set of all schemata

therefore designates an overlapping and highly redundant set of regions covering all of *A*.

Because each taxon belongs to so many schemata, the performance value of a taxon provides information about the average performance in several regions of *A*. One straightforward search procedure that uses this information is as follows. Start with the simple inference heuristic that assumes taxa with high performance value will be found in regions having high average value. The average performance value of a region is a well defined quantity - it is simply the average of all taxa belonging to the region - but it is assumed that, because of the enormous number of elements in *A*, no search procedure is able to test all taxa and discover the true set of average values. How then can a search procedure be formulated that infers which regions are the ones with high performance? As the search trajectory is generated, there is ample information providing the basis for <u>estimates</u> of the regional averages. Each taxon is a sample point for each of the 2^k schemata it is an instance of. This suggests that a search procedure can implement the inference heuristic by maintaining estimates of the average value of each region. As the trajectory encounters new members of a region, the new values can be used to update the increasingly accurate regional averages. The subsequent direction of the search trajectory can then be biased toward regions with high estimated average value, making future samples increasingly non-uniform.

The search procedure just described makes explicit use of regions, or schemata, to guide the trajectory through *A*. An algorithm that implements this procedure with a working memory of M taxa is as follows:

1) Set t = 0. Choose an initial set B(0) of M taxa. Initialize every entry to some null value in the table V of estimated values for all schemata.
2) For every taxon b in B(t), use the performance of b to update the average value V(s) of every schema s that b is an instance of.
3) Generate a new set B(t+1) so that, insofar as is possible, the number of instances $n_s(t+1)$ of every schema is equal to

$$n_s(t+1) = [V(s)/V'] * n_s(t)$$

where V' is the average value over all schemata. This recursion means that schemata with above average estimates will have an increasing number of instances in working memory while schemata with below average estimates will have a decreasing number of instances.
4) Set t equal to t+1 and return to step 2.

It is obvious that this algorithm would be enormously expensive in time and space to implement. The table V must have entries for all 4^k schemata defined over *A*. Moreover, the fact that the schemata overlap so extensively makes it very difficult to generate a set B(t+1) containing the desired number of instances for each schema. Nevertheless, the algorithm does an effective job of searching *A* by

iteratively generating non-uniform samples of *A*.

> Setting aside the difficulties of implementation, we find that the algorithm ... samples with increasing intensity schemata that contain [taxa] of above-average [value]. Because the average [V'] increases with time, this sampling rule is a global "force" driving the search into subsets observed to contain useful [taxa]. Moreover, because the algorithm works from a database of M points distributed over [*A*], it is not easily caught on "false peaks" (local optima).... Overall, this algorithm is much more globally oriented than standard procedures, searching through a great many schemata for regularities and interactions that can be exploited. [Holland,1980, p. 262]

The overall intent and performance of this algorithm is easy to understand. It is unfortunate that it is computationally intractable. What is needed is a more elegant way to implement the same inference heuristic without paying the tremendous costs.

We are now in a position to describe genetic algorithms. Genetic algorithms are derived from a simple computational model of evolutionary genetics. The model is based on the following set of simplifying assumptions: chromosomes or genotypes are strings of a fixed length *k*; a population contains a fixed number M of genotypes; and, each genotype has a well defined fitness, or relative ability to survive and produce viable offspring. The population is a dynamic entity. New individuals are continually being generated to replace existing ones. In more detail, the basic genetic algorithm involves the following steps:

1) Set t = 0. Choose an initial population B(0) of M strings or genotypes.

2) Compute the fitness f(b) for every string b in B(t).
3) Assign each string b a normalized fitness f(b)/F where F is the average fitness of all strings in B(t).
4) Reproduce the strings according to their relative fitness: assign each string a probability proportional to its normalized fitness and draw a sample of size M from B(t) with replacement.
5) Modify the sample by stochastically applying the genetic operations of crossing over and mutation to the strings.
6) Use the modified sample as B(t+1).
7) Set t equal to t+1 and return to step 2.

The successive populations of strings designate a search trajectory thru the space *A* of all possible strings. The genotypes that survive will, over time, be those which have proven to be the most fit. This is because the reproduction rate for each string is proportional to its fitness. In this sense, the search trajectory is steered toward those types of individuals - or regions in *A* - with above average fitness.

The driving force behind the genetic algorithm is the reproduction of individuals in proportion to fitness together with the crossover operator. Reproduction according to fitness concentrates sample points in regions of high average fitness. Individuals with above average fitness contribute more than one offspring to B(t+1); individuals with below average fitness contribute less than

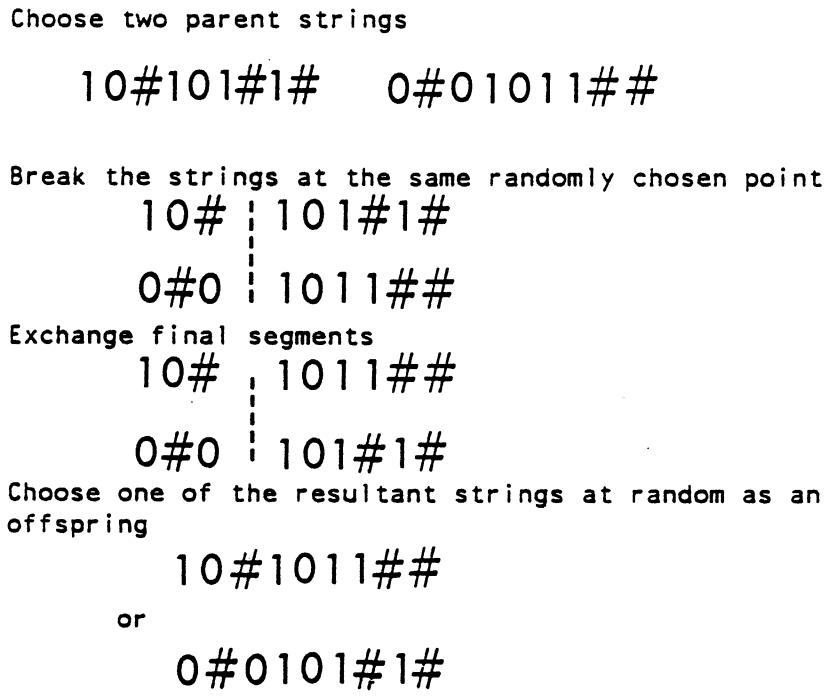

Figure 5.2. The standard implementation of the crossover operator on strings having the same length. See Smith [1980] for a discussion of what to do with varying length strings.

one offspring.[2] In the absence of other operators, reproduction will eventually generate a population of individuals identical to the best individual in the initial population. The crossover operator (see Figure 5.2) is the primary operator for generating new individuals from existing ones. Crossover generates individuals that are instances of some of their parent's schemata, as well as instances of new schemata. The operator therefore sample both familiar and unexplored regions of *A* with each

[2]The algorithm is usually implemented with a fixed population size. That means every individual gets an average of one offspring per generation.

application. In Figure 5.2, for example, the resultant string 10#1011## provides additional information about the schema ¤¤¤101¤¤¤ instanced by each parent, refining old estimates of that schema's average performance. At the same time, new schemata such as 10¤¤¤¤¤## are now being sampled as well. Each application of the crossover operator generates useful information about each of the 2^9 schemata instanced by the resultant string.

Holland [1975] notes that the interaction of crossover with reproduction generates a "pressure" toward sets of schemata that have above-average performance while at the same time maintains a robust sampling of all of *A*.

> This is closely analogous to a gas diffusing from some central location through a medium of varying porosity, where above-average porosity is the analogue of above-average performance. The gas will exhibit a quickened rate of diffusion whenever it encounters a region of higher porosity, rapidly saturating the whole region. All the while it slowly but steadily infuses enclaves of low porosity.... Thus, following the analogy, local optima in performance are thoroughly explored in an intrinsically parallel fashion. At the same time the genetic plan does not get trapped by settling on some local optimum when further improvements are possible. Instead all observed regions of high performance are exploited without significantly slowing the overall search for better optima. (p. 104)

Note that if a particular value at a particular string locus is not present in the population, the search "pressure" is effectively blocked from regions of *A* involving that value. Reproduction and crossover, in other words, can only emphasize and manipulate what is available. In Figure 5.2, for example, none of the strings has the value 1 at the

second position. This means that none of the schemata 01¤¤¤¤¤¤¤, ¤1¤¤¤#¤¤¤, etc. will be sampled. For this reason the mutation operator is required. Every new string has its values independently modified with some small probability, usually less than or equal to 0.001.[3] This is a background pressure that guarantees no value will be permanently lost from the "pool" of values available to the genetic algorithm.

A mathematical analysis of genetic algorithms has been given by Holland [1975], DeJong [1980], and Bethke [1981]. Briefly, each schema can be formally treated as a random variable whose true mean is estimated by the average fitness of the instances in the population B(t). From this point of view, three basic properties of genetic algorithms have been proven:

1) Every time the algorithm is applied to a population of size M, the estimated values of approximately $M^2*2^{k/2}$ schemata are adjusted. This information is used by the algorithm to generate a new set of M strings so that for every schema s, the number of instances $n_s(t)$ in the population is modified by an amount proportional to the average value v(s) of the instances. More precisely,

$$n_s(t+1) = [v(s)/v'] * (1-e) * n_s(t)$$

where v' is the average performance of all strings in

---

[3]This corresponds to the mutation rates commonly observed in nature.

the population and e is a very small positive number less than one. This evaluation and manipulation of a large number of schemata is performed by simple modifications of only M strings at a time, a phenomenon Holland [1975] calls *intrinsic parallelism*.

2) Iterative application of the genetic algorithm leads to a population in which the number of instances of each schema is proportional to its observed average value. In this sense, the population is a database that compactly and usefully summarizes the experience obtained from the search thru A. The genetic algorithm automatically generates and accesses this database.

3) "Because the set of schemata is such a rich cover of the space of possibilities A, almost any interactions (or correlations) between attributes ... will be 'discovered' by some schema" [Holland and Reitman,1978, p. 318]. It can be proven [Bethke,1981] that the algorithm will fail to discover highly rated schemata only when the fitness function has an unlikely set of bizarre properties.

These results indicate that the genetic algorithm stores and manipulates schemata as if it had access to the explicit table of estimated values used by the search procedure described earlier.

To better appreciate the enormous advantage of this procedure, consider again the maximal decomposition computed

by Hayes-Roth's algorithm. Assume for the moment that there is just one pattern class defined by a single unknown characteristic. Suppose the population being considered is the M highest performing taxa derived from the original set of exemplars using interference matching. Introducing a new exemplar is no problem for the genetic algorithm, since it always manipulates M taxa at a time. Only the performance value of each taxa has to be recomputed. While the population resulting from several iterations may or may not be a maximal decomposition,[4] it will contain many instances of the best schemata and is therefore a useful sample of the best taxa. Moreover, the genetic algorithm is robust enough to discover the best taxa even if the initial population it starts with is generated at random! Therefore maximal decompositions do not have to be explicitly computed in the first place.

The genetic algorithm has been empirically tested in the rigorous task domain of function optimization [DeJong,1975; Bethke,1981]. The results of these studies indicate that the genetic algorithm is superior to standard optimization techniques when the properties of the range of functions - unimodal or multimodal, convex or non-convex, continuous or discontinuous, etc. - are not known in advance. The algorithm is particularly effective when the function in question is non-linear and/or high dimensional.

---

[4]After the characteristics with low performance have been discarded, Hayes-Roth's algorithm is no longer working with a maximal decomposition either.

Moreover, the performance of the genetic algorithm as a function optimizer confirms the predictions of Holland's theoretical analysis. The genetic algorithm has also been shown to be an effective learning component for production systems in general [Smith,1980] and classifier system in particular [Holland and Reitman,1978]. It is therefore a solid choice as the basic learning component for the classifier system being considered here. The aforementioned empirical studies provide ample information about parameter settings and other details for implementing a genetic algorithm. The implementation best suited for our hypothetical organism, hereafter designated as G0, is specified in Appendix A.

Modifying the Genetic Algorithm

In its basic form, the genetic algorithm can be used to generate classifier taxa with optimum expected performance - those corresponding most closely to the defining characteristic - for a given pattern class. The search trajectory "converges" in the sense that high performance taxa proliferate and eventually "take over" the entire population. What if there is more than one pattern class to be considered, or a single class with more than one defining characteristic? The problem is no longer merely to optimize expected performance values and so discover the best individual taxon. For example, suppose there are two classes given by the characteristics 11**...** and 00**...**

respectively. Assume that exemplars from each class are equally likely to occur. The ideal population for distinguishing these classes would contain the classifier taxa 11##...## and 00##...## in equal proportions; that is, a separate subpopulation characterizing each class. The genetic algorithm as described so far will treat the two pattern classes as one class and produce a population of taxa having good performance in that larger class. The taxon ####...## will therefore make a strong bid to take over the population since it corresponds to the only single characteristic defining the entire class. The problem is obvious. Requiring each taxon to match each exemplar results in an averaging of performance that is not always desirable.

The population genetics example discussed earlier offers some useful intuitions about how to modify the genetic algorithm to handle several characteristics simultaneously with a single population. What is needed is a simple analog of the speciation and niche competition found in biological populations. The genetic algorithm should be implemented so that, for each pattern class characteristic or "niche", a "species" of taxa is generated that has high performance in that niche. Moreover, the spread of each species in the population should be limited to a proportion determined by the "carrying capacity" of its niche. The following sections describe the technical modifications to GO aimed at devising such an implementation

of the genetic algorithm.

An Appropriate Match Score. The performance value of a taxon has been described as a quantity indicating how well the taxon is matched by a given exemplar or binary string. This description must be made more explicit before any computer studies of the genetic algorithm can be started. The choice of a performance measure is clearly pivotal. Only if the taxa are usefully ranked can the genetic algorithm, or any learning heuristic, have any hope of identifying those regions of the search space likely to contain the best taxon. There are at least three criteria that seem appropriate here in considering performance measures:

1) The measure must be easy to compute.
2) It must rank taxa that match a binary string higher than those that don't match.
3) It must provide useful information about the good and bad regions of the search space.

The third criterion is perhaps the most interesting. It requires that even bad taxa provide some information about the "direction" leading to better performing regions. This makes it possible for the genetic algorithm to find its way out of and avoid low performance areas in the search space.

Holland and Reitman [1978] suggest using the following match score - hereafter called M1 - as a performance measure:

> Let NCARES be the number of specific values (0 or 1) in a taxon.

$$\text{score} = \begin{cases} \text{NCARES} & \text{if the string matches the taxon} \\ 0 & \text{otherwise} \end{cases}$$

This match score obviously satisfies the first two criteria. It does not satisfy the third, however, because all non-matching taxa get the same score of zero. Holland and Reitman avoid this deficiency in M1 by starting with an initial population of taxa containing #'s at 90% of the positions. This makes it very likely that matching taxa will be available and the genetic algorithm will have useful information to work with. Even so, the problem has not been eliminated.[5] Suppose, for example, the genetic algorithm is used to generate a population well adapted to some characteristic like 11010**. The taxa in this population will be 11010##, 1#010##, 11#10#1, 11#10#0, etc. As the search trajectory converges, the variability in the population decreases. It is therefore unlikely that the population will contain many taxa having four or more #'s. Such taxa would have a match score too low to compete over the long run and survive. Now suppose the environment changes slightly so that the characteristic is **010**; that is, the pattern class has been expanded to allow either a 0 or 1 in the first two positions. The well adapted population must not only have a taxon with four #'s in order to consistently match the exemplars of the new class, it must have those #'s in exactly the right loci. There is no

---

[5]The assumption is that the environment does not direct the algorithm to maintain a large proportion of #'s at each position.

reason to expect such good fortune since the combinations of attribute values are no longer random. The population will most likely have no taxon that matches a new exemplar like 0001011. The value of M1 for every taxon will be zero and, consequently, the genetic algorithm will have no clues about how to modify the population. It will search more or less at random, relying on the mutation operator to generate new taxa. This could all be avoided with a match score that allows the genetic algorithm to recognize #1010## and 1#010## as "near misses" and work from there rapidly toward the solution ##010##.

It might be argued that with a moderately large population of taxa and a diverse set of pattern classes in the environment, this particular dilemma is unlikely. The diversity might assure that adequate proportions of attribute values will be available at each locus. Even so, the particular combinations of attribute values are what counts in determining a match. To say that a population has adapted to some pattern class is to imply that the combinations of attribute values, or schemata, are correlated with the pattern characteristics; therefore, there is no reservoir of random combinations for matching unexpected situations.[6] This is no mere theoretical

---

[6]One solution might be to make sure there is always some given proportion of #'s at each locus, thereby assuring the population can handle novel items. It is not clear how to do this without keeping burdensome statistics, nor is it clear how such an enforced allele distribution will affect overall system performance.

inconvenience for our hypothetical organism. As the organism explores some new region of the environment, it is crucial that it make use of relevant past experience in discovering the characteristics of the new region. There might not be time to learn everything anew. The system can only make use of its relevant experience if the match score points out the schemata that are relevant.

Consequently, it is reasonable to ask if there are better match scores than M1. To answer this question, the basic genetic algorithm G0 described in Appendix A was implemented on a PDP 11/34 computer using the C programming language. Each taxon was 16 positions long, corresponding to the word size on this particular machine.[7] In order to provide a stiff test of a match score's ability to identify relevant schemata, the initial populations were generated at random with equal proportions of 0, 1, and # at each locus. The defining pattern characteristic to be discovered was arbitrarily chosen to be 1111111111******. The criterion for evaluating the performance of the adaptive plan using M1, and all subsequent adaptive plans, is derived from DeJong's [1975] on-line performance criterion. The on-line performance of a plan, given an evaluation function f and an interval of observation T, is defined as

---

[7]This value was chosen for the sake of convenience and in no way reflects any limitations on the computational efficiency of the genetic algorithm or classifier systems. One of the function domains in Appendix A, for example, uses strings of length 240. Other applications [Smith and DeJong,1981] involve strings several times longer than that.

$$p(f,T) = (1/T) * \sum_{t=1}^{T} f(s(t))$$

where $s(t)$ is the t^{th} structure generated by the adaptive plan. This measure assumes every new structure generated has some bearing on the overall system performance. That assumption is true for systems which are adapting "on-line" while they generate behavior, a situation certainly descriptive of our hypothetical organism. DeJong suggests using an observation interval T of 6000 new structures. The plans are therefore observed over 120 *generations* or applications of the genetic algorithm.* The function f will be the taxon match score with a given exemplar. This means that on-line performance will indicate how quickly matching taxa proliferate in the population and how closely they correspond to characteristics of the pattern class.

The match score criteria mentioned earlier suggest in a straightforward way some alternatives to M1. The match score should have one range of values for non-matching taxa and a non-overlapping, higher range for matching taxa. Within a range, taxa with the greatest number of matching values should be ranked highest. This can be accomplished with the following simple point system. For non-matching taxa, award one point for each matching 0 or 1 attribute. Each # should also receive some value as a placeholder for irrelevant attributes. It is not immediately clear what an

---

*The basic plan G0 uses a population of size 50 which is completely replaced every generation. In 120 generations, therefore, 6000 individuals are generated.

appropriate value is, though it should probably be less than one to bias the adaptive plan toward finding relevant attributes when they exist. For matching taxa, the M1 score designates an appropriate number of points if it is increased by the taxon length to make sure matching taxa are ranked higher than non-matching ones. Three match scores based on these ideas are tested. Each score has the basic form

$$\text{score} = \begin{cases} \text{if the taxon matches,} \\ \quad \text{NCARES + (taxon length)} \\ \text{otherwise,} \\ \quad \text{1 for each correct attribute} \\ \quad \text{plus } e \text{ for each \#} \end{cases}$$

The three scores M2, M3, and M4 use an e of 1/4, 1/2, and 3/4 respectively.

TABLE 5.1

COMPARISON OF FOUR MATCH SCORES

Match score	Range	Optimal value	On-line value	% Progress	Sample Variance
M1	[0,16]	10	1.6438	16.4%	0.9794
M2	[0,32]	26	13.9627	53.7%	6.9063
M3	[0,32]	26	19.7919	76.1%	1.5773
M4	[0,32]	26	20.5555	79.1%	0.5685

The behavior of all four match scores is summarized in Table 5.1. The results for these and all subsequent genetic algorithm experiments represent averages over 10 computer runs. The "optimal value" entry in the table shows the score of the taxon 1111111111###### which corresponds

exactly with the pattern characteristic. The "% progress" entry shows how far the on-line performance progressed with respect to this optimal value. The dismal performance of M1 confirms the aforementioned reservations about its ability to pick out useful schemata from non-matching taxa. On two of the ten runs it failed to discover any of the correct taxa during the interval of observation; and, only twice was the final population filled with matching taxa. Surprisingly, M2 also failed to consistently produce matching taxa.[9] After examining the taxa M2 produced, however, the reason for its performance is clear. In five of the ten runs, #'s were almost entirely eliminated from the population. Recall that #'s are place holders for irrelevant attributes and irrelevant pattern attributes have their values chosen at random. During the early phases of the adaptive process the expected contribution[10] of a # to the taxon score is 1/4 using M2. For a 0 or 1 at the same locus, on the other hand, the expected contribution to the match score is 1/2. This is a clear mandate for the adaptive plan to favor either 1 or 0 for irrelevant attribute values This observation also explains why M4 is better than M3. Both lead eventually to the proliferation of matching taxa. M3, however, gives equal expected value

---

[9]Matching taxa get a minimum score of 16 which the on-line performance of M2 failed to achieve.

[10]The score of a taxon is only a sample of the random variable giving its expected score over the set of all exemplars. This makes the search for the best taxa that much more difficult.

to all alternatives at irrelevant loci. The adaptive plan is therefore prone to generate taxa having higher order than the pattern characteristic such as 1111111111####10 or 1111111111#0####. These taxa match at most half of the exemplars and therefore adversely affect the plan's performance rating. Table 5.2 shows that the differences between the match scores are considerably less significant when the initial population contains a 90% proportion of #'s at each locus. Note that M1 still does worse than the others, primarily because it does not use all the information available until every taxon in the population matches. Based on these results, M4 is the choice as the match score for our classifier system adaptive plan.

TABLE 5.2

COMPARISON OF FOUR MATCH SCORES

USING A VERY GENERAL SET OF INITIAL TAXA

Match score	Range	Optimal value	On-line value	% Progress	Sample Variance
M1	[0,16]	10	7.2472	72.5%	0.2228
M2	[0,32]	26	19.9267	76.6%	0.3461
M3	[0,32]	26	20.0535	77.1%	0.4382
M4	[0,32]	26	19.9658	76.8%	0.1581

One final observation about the taxa produced by a match score. M4 seems to successfully avoid the proliferation of taxa having higher order than the pattern characteristic. There is no guarantee, however, that the

"perfect" taxon 1111111111###### will be the most numerous one in the population. In the early stages of the adaptive process any taxon that matches has a great advantage over the rest of the population. Its schema will therefore proliferate at the expense of alternative schema. In particular, if this matching taxon has a # where a 1 is preferred and most of the 1's at that locus are on non-matching taxa, then the 1 value at that position is likely to disappear from the population. In time, of course, the mutation operator will reintroduce the 1 and the adaptive plan will find the perfect taxon. In a finite interval of observation, however, the most that can be said is that the perfect taxon is more likely to occur than any other.

Restricted Mating Strategies. If the genetic algorithm is to be used to generate a population of taxa containing many specialized subpopulations, it is no longer reasonable to produce a completely new population each generation. Only the subpopulations relevant to the current pattern class exemplar need to be modified. Given that the overall population size is fixed and the various subpopulations are not physically separated, several questions are raised: Does modifying only a fraction of the population make a difference in overall performance? How is a subpopulation identified? What is a reasonable way to allocate the limited space among the various subpopulations, especially given that the number of "niches" or pattern characteristics in the environment is not known in advance? In this

section, the first two questions are investigated and a genetic plan is developed that leads to efficient growth and development of subpopulations. The following section discusses how to allocate space among alternative subpopulations.

DeJong [1975] experimented with genetic plans in which only a fraction of the population is replaced by new individuals each generation. His results indicate that such a change has adverse effects on overall plan performance. The problem is that the plan is generating fewer samples of the search space A at a time. While the expected proportion of trials to each subset in A is unchanged, the sampling error due to finite stochastic effects becomes more severe (see Appendix A). Fewer samples per application of the algorithm means an increase in the cumulative sampling error. This, in turn, means that the search trajectory is more likely to converge to some sub-optimal region in A. The problem is illustrated in Figure 5.3. The population size of G0 is increased to 200 to make the notion of subpopulation more meaningful. The performance of two alternative plans are compared over an equal number of trials (not generations): one producing a whole new population of 200 taxa per generation; the other producing only 50 new taxa per generation with the taxa to be replaced chosen at random. Replacing taxa at random assures that the expected number of offspring for a taxon over its "lifetime" is the same for each plan. There is not only a marked

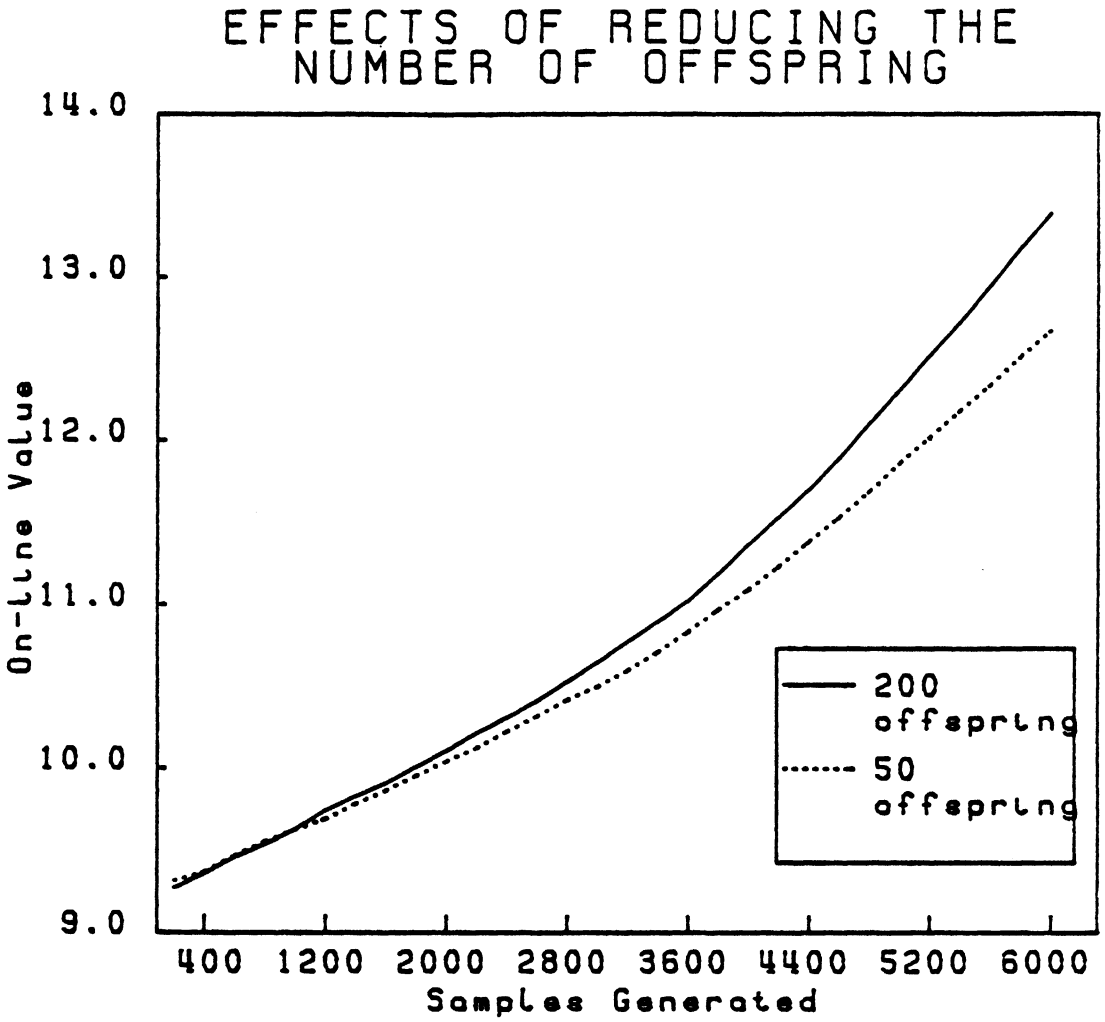

Figure 5.3. The adverse effects of producing fewer offspring per generation. Fewer samples per application of the algorithm means an increase in the cumulative sampling error.

difference in performance between the two plans as shown in Figure 5.3; there is, moreover, a substantial difference in sample variance: 0.91 when 200 taxa are generated at a time compared with 1.91 when 50 taxa are generated at a time.

The strategy adopted here to reduce the sampling error and improve performance is to make sure that the "productive" regions of the search space consistently get most of the samples. In the standard implementations of the genetic algorithm, the search trajectory is unconstrained in the sense that any two taxa have some non-zero probability of mating and generating a new offspring (sample point) via crossover. This means, in particular, that taxa representing distinct pattern classes or characteristics can be mated to produce taxa not likely to be useful for classification. As a simple example, consider the two pattern classes given by 111111**...** and 000000**...**. Combining taxa specific to each of these classes under crossover is likely to result in taxa like 111000##...## which match none of the exemplars of either class. There is no reason why such functional constraints should not be used to help improve the allocation of samples and avoid needless sampling error. It seems reasonable, therefore, to restrict the ability of functionally distinct taxa to become parents and mate with each other. This will force the genetic algorithm to progressively cluster new sample points in the more productive subsets of the search space. The clusters that emerge will be the desired specialized subpopulations.

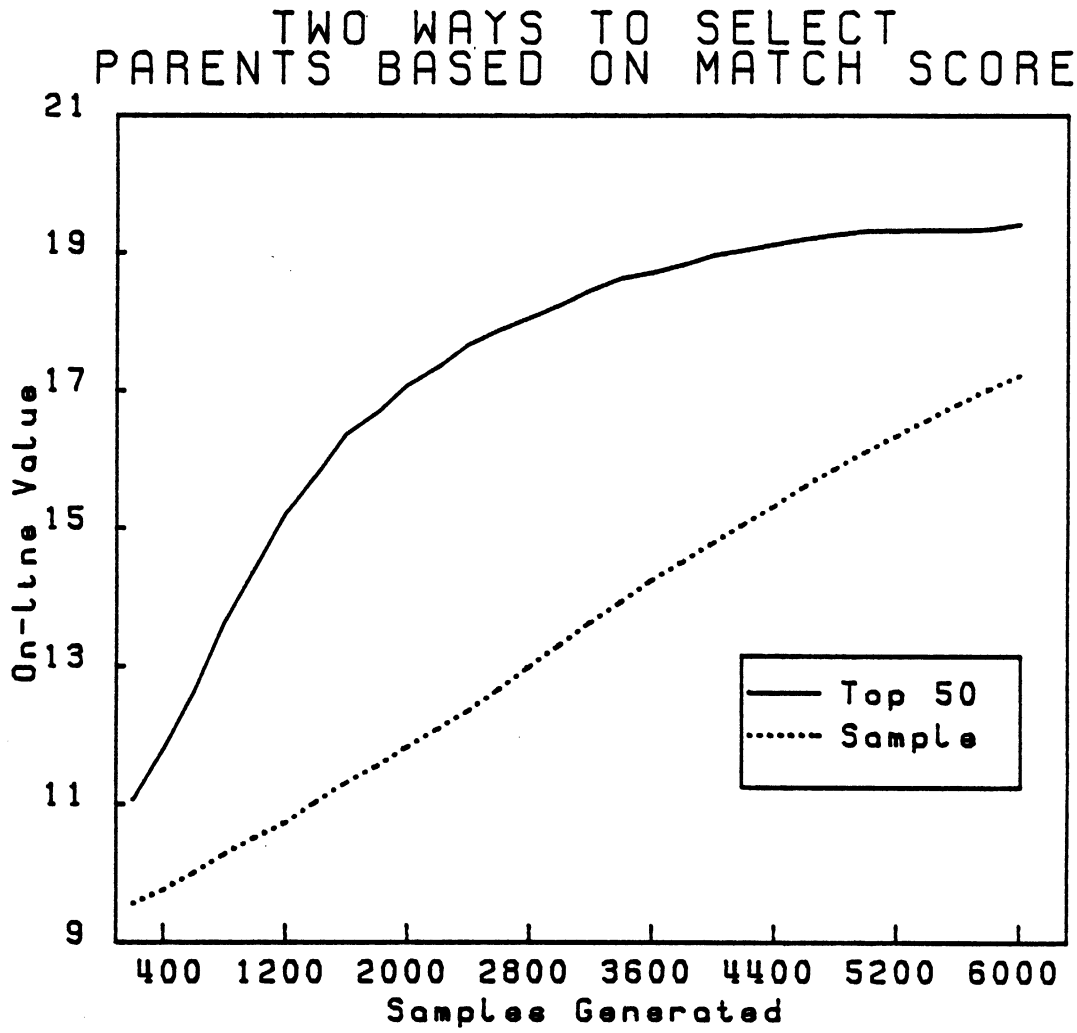

Figure 5.4. Deterministic versus stochastic designation of the relevant taxa to be used as parents for generating offspring.

How can functionally distinct taxa be identified? Any restrictive designation of parent taxa must obviously be based on match scores. Taxa relevant to the same exemplar have the same classification function. It is not obvious, however, how to best make use of that information. As a preliminary consideration of the choices available, suppose that we must choose a subpopulation of 50 parent taxa from the population of 200 and that 50 offspring will be produced. Assuming the population does not contain 50 matching taxa that could serve as parents, two ways of selecting parents come quickly to mind. Either deterministically select the 50 taxa with the highest match scores; or, more conservatively, pick a random sample of size 50 without replacement from the population using match scores to generate the probability distribution.'' Both methods were tried as modifications of G0 and their performance is compared in Figure 5.4. Deterministically choosing the best taxa generates good initial performance that quickly reaches a plateau before the best taxon is discovered. This is because there is a rapid loss of diversity, leaving the population filled with taxa having higher order than the pattern characteristic. Apparently, this plan places too much emphasis on the short term ranking of schemata. For any given irrelevant attribute, a "run" of

---

''If the total match score of all taxa involved is M, then the probability of choosing a taxon with match score m is just m/M.

several 0 or 1 values in a row is to be expected.[12] Over a short period of time, therefore, either 0 or 1 can appear to designate the best region for further sampling. It is only in the longer run that the # is proven to be the best alternative. Choosing a biased random sample from the population allows this longer term advantage to become evident. This plan makes slower but continued progress toward the best taxon in *A*.

These two alternatives are in fact at opposite ends of a continuum of strategies for choosing parents. Suppose we define a parameter SETSIZE to designate the size of the sample space used to choose the parents, relative to the number of parents, NPARENTS, to be chosen. The sample space will by convention be the SETSIZE*NPARENTS taxa with the highest match scores. Now it is clear that the deterministic method described above used the value SETSIZE = 1.0 and the stochastic method used the value SETSIZE = 4.0, where NPARENTS = 50. It is possible that some intermediate setting for this parameter will produce a plan that combines the speed of the first method with the reliability and robustness of the second. To test this hypothesis, several values of SETSIZE were used to generate alternative plans. The results in Figure 5.5 show that there is indeed a range of SETSIZE values that generate the desired behavior. Values below 2.0 have reasonable on-line

---

[12]For example, in a randomly generated sequence of Bernoulli trials a run of five 0's or 1's in a row can be expected to occur every 31 trials.

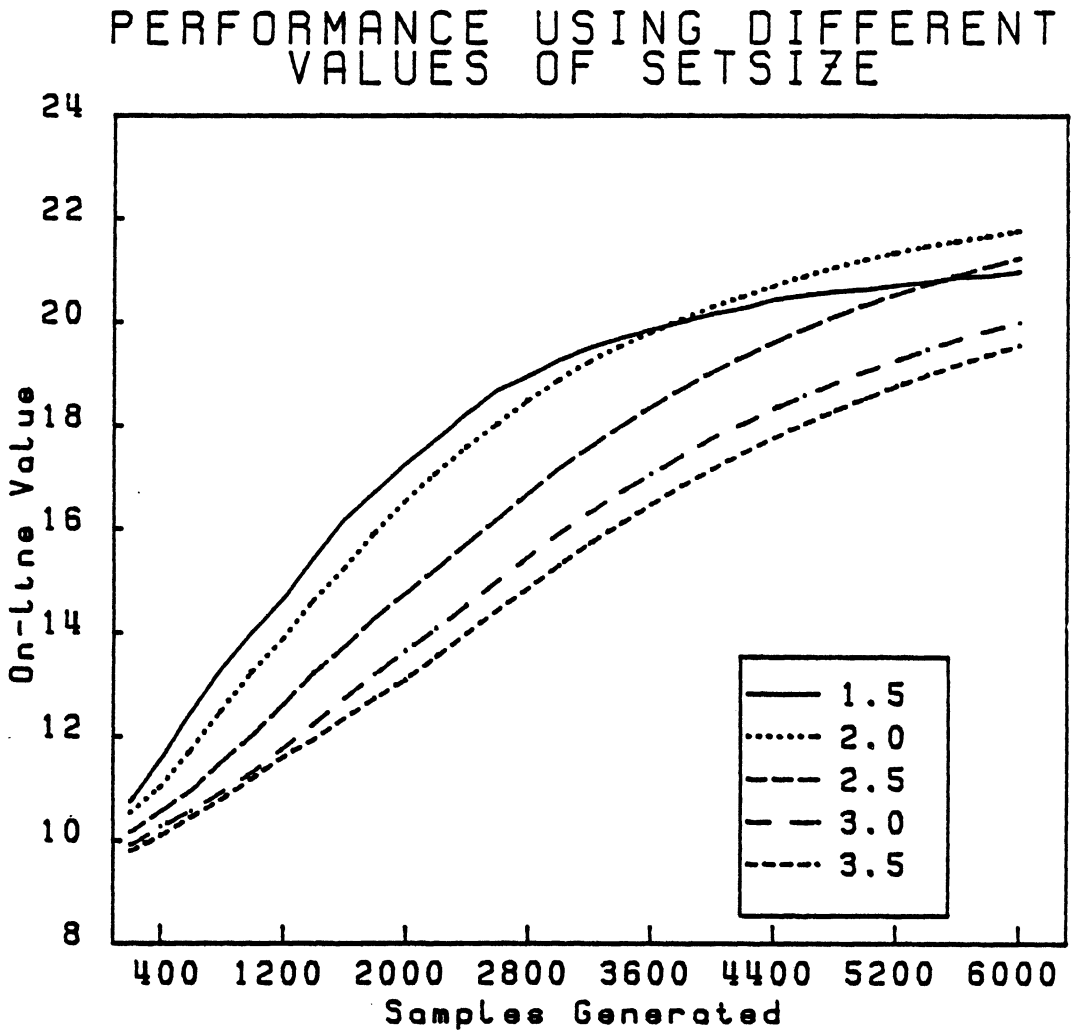

Figure 5.5. On-line performance as a function of varying SETSIZE.

performance but lack robustness in that they consistently generate taxa having higher order than the pattern characteristic and get stuck on a suboptimal performance plateau. Values above 2.5 continue to improve but the sampling strategy is too conservative to also give consistently high performance over the interval of observation.

These considerations provide the basis for a new algorithm G1 that uses a *restricted mating policy*. Only those taxa that classify the same exemplars will be allowed to mate with each other. This restriction is enforced as follows. When there are at least NPARENTS matching taxa available, then the parents are chosen at random from the set of matching taxa.[13] When there are fewer than NPARENTS matching taxa in the population, the sample space is designated as above using an appropriate value of SETSIZE. The two acceptable settings for SETSIZE determined above were tested using G1 and the results are compared in Figure 5.6. Both plans generate good performance. The value 2.0 gives the best on-line performance over the interval of observation, but the other plan is rapidly closing the gap in performance. The plan using the value 2.5 is considerably more reliable; that is, it has a sample variance of 0.6565 compared with 1.3213 when SETSIZE = 2.0.

---

[13]When there are this many matching taxa the search trajectory has very likely identified a promising region in *A*. It is therefore appropriate to consider only the matching taxa as relevant.

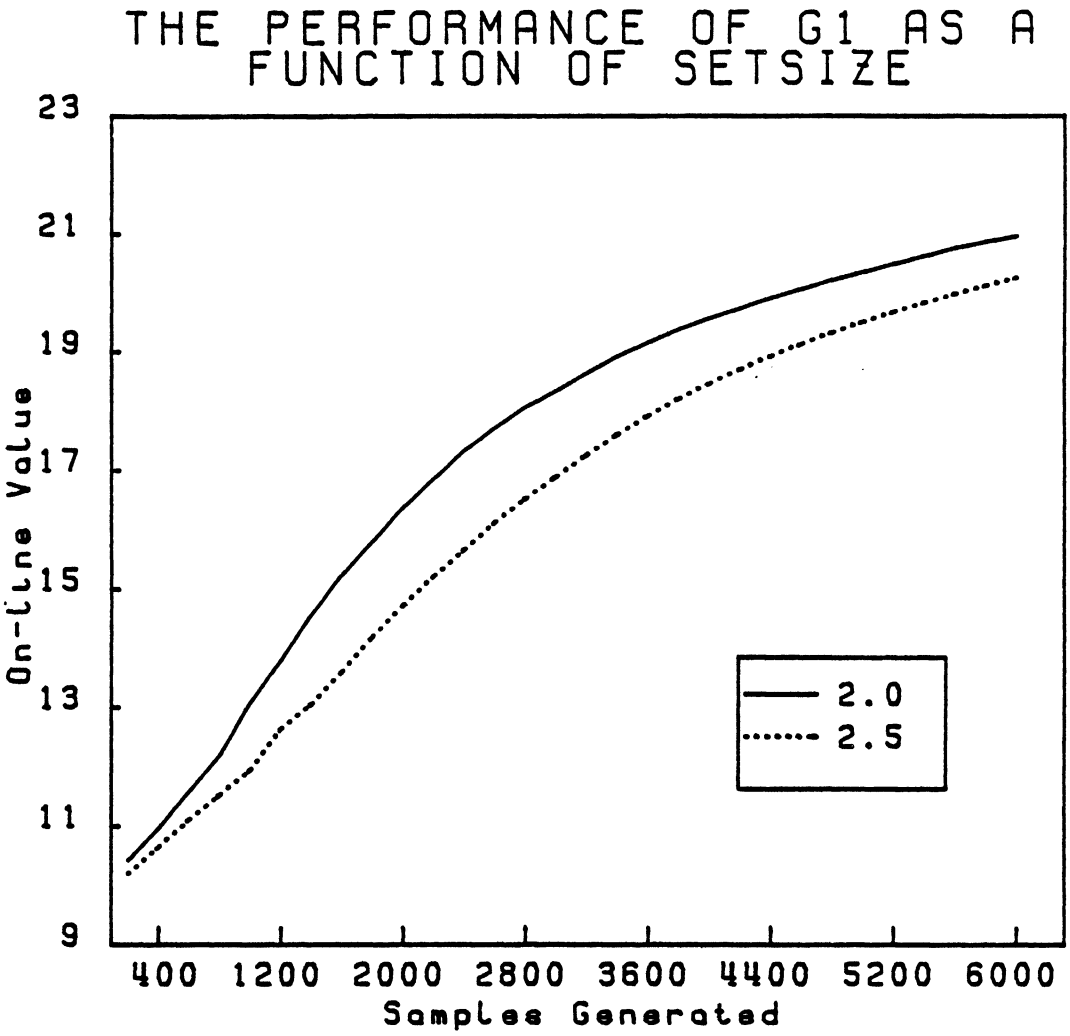

Figure 5.6. The performance of G1 using the two best values of SETSIZE.

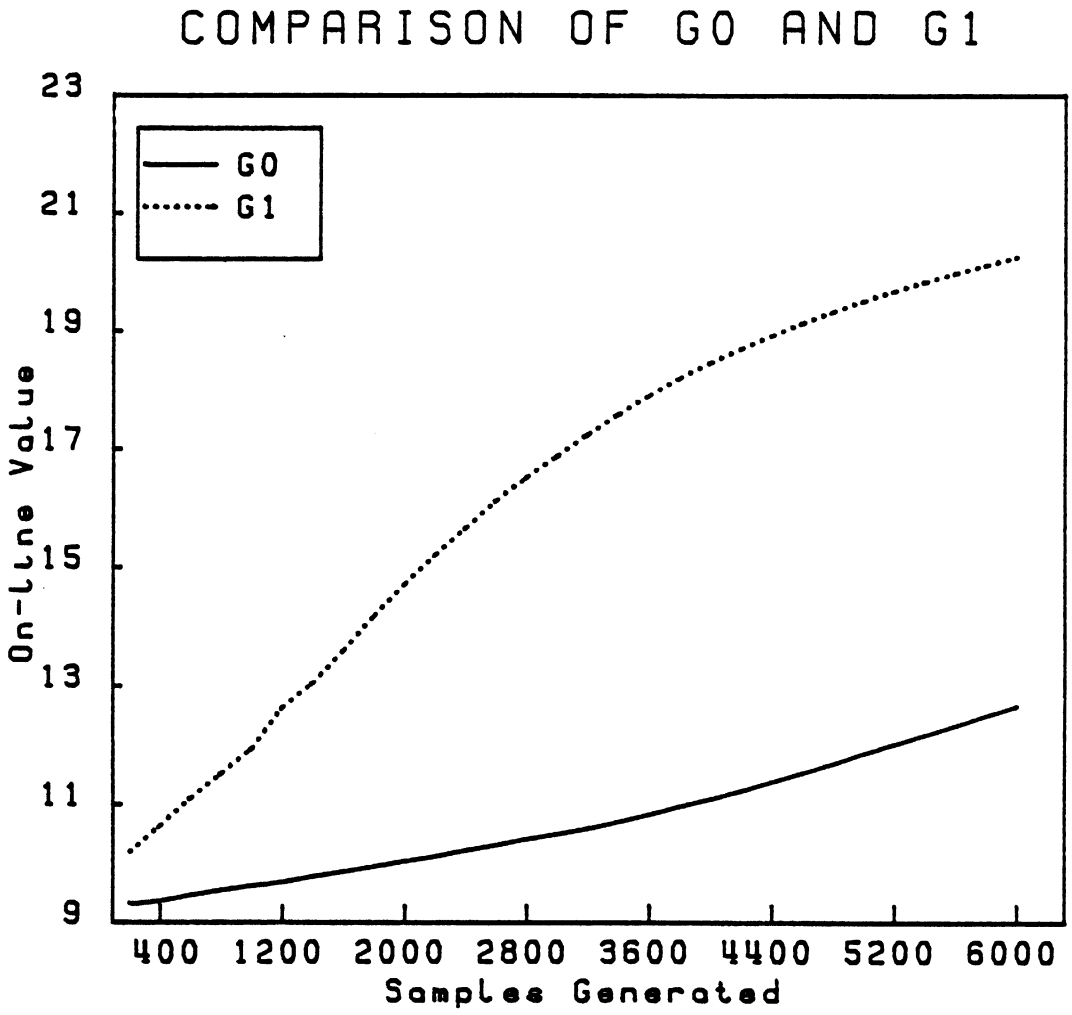

Figure 5.7. A comparison of G0 and G1.

The best overall choice is therefore SETSIZE = 2.5. To see that G1 with this value of SETSIZE is a substantial improvement over G0 without restricted mating, the performances are compared in Figure 5.7. G1 does far better producing 50 offspring than G0 and the sample variance is much lower than the 1.908 of G0. The restricted mating policy is therefore an effective strategy for usefully constraining the allocation of trials by the genetic algorithm.

Crowding. The restricted mating policy for subpopulations in G1 is analogous to the designation of species in a natural population. Under this analogy the various pattern characteristics correspond to ecological niches in that they designate local optima in the fitness function or match score. Species of taxa that have high fitness in a given niche will proliferate and fill the niche. Species that do not do well in any niche will become extinct. From this ecological perspective there is an obvious mechanism for automatically controlling the size of each subpopulation. Briefly, and very simply, any ecological niche has limited resources to support the individuals of a species. The number of individuals that can be supported in a niche is called the "carrying capacity" of the niche. If there are too many individuals there will not be enough resources to go around. The niche becomes "crowded", there is an overall decrease in fitness, and individuals die at a higher rate until the balance

between niche resources and the demands on those resources is restored. Similarly, if there are too few individuals the excess of resources results in an increase in fitness and a proliferation of individuals to fill the niche.

The idea of introducing a crowding mechanism into the genetic algorithm is not new. DeJong [1975] experimented with such a mechanism in his function optimization studies. The idea behind his crowding algorithm is to make it difficult for any one kind of individual to spread throughout the population. Instead of deleting individuals at random, a small subset of the population is randomly selected. The individual in that subset most similar to the new individual is then chosen as the one to be deleted. Clearly, the more individuals there are of a given species the more likely it is that one of them will turn up in the randomly chosen subset. After a certain point, new individuals begin to replace their own kind and the effective proliferation of a species is limited. DeJong's results show that this crowding algorithm improves the ability of the genetic algorithm to optimize multimodal functions - those designating several niches - as desired.

A similar algorithm can be implemented much more naturally here. Because an exemplar selects via match scores those taxa that are similar, there is no need to choose a random subset. Crowding pressure can be exerted directly on the set of relevant taxa. This can be done using the strength parameter associated with each classifier

and described in Chapter 4. It was shown how this parameter can be adjusted so that it estimates the expected match scores of the taxa in a given set. The strength is, in this context, a good measure of the relative fitness of a taxon in its niche. If we specify that, in the adjustment of strengths, a fixed fraction is lost by each classifier and the amount available is shared by all the classifiers in the relevant set, then in effect there is a limited amount of strength associated with each niche. When there are too many classifiers in a niche their average strength decreases because the strength adjustment causes them to lose more strength than they gain. Conversely, when there are too few classifiers with high expected match scores in a niche, the average strength of those classifiers will increase. If we further specify that taxa are deleted in inverse proportion to their strength, the changes in strength generate an effective crowding pressure. More specifically,

> Let $s(t)$ be the amount of strength in a classifier on the t^{th} time it is evoked.
> Let e be the fraction of strength lost in the process of competing to classify a message.
> Let $m(t)$ be the match score of the classifier.
> Let $M(t)$ be the total of all match scores for classifiers in the relevant set.
> Let S be the relatively constant amount of strength made available to be recovered by the relevant set as a whole.
> Then the change in strength for a classifier is given by
>
> $$s(t+1) = s(t) - e*s(t) + (m(t)/M(t))*S$$
>
> Given an appropriate choice for e, the amount of strength in a classifier will be $(m/M)*S$ where m/M is the expected relative share of strength for a classifier in a given relevant set.

This procedure generates the desired crowding effect because

the total strength in a relevant set tends to be constant. The more individuals there are in the set, the less their average strength is which means members are more likely to be deleted. Members of a relatively small relevant set will have higher average strength which means they are more likely to survive and reproduce. In this way, the total available space is automatically and dynamically managed for every relevant set or subpopulation. The number of individuals in a niche increases or decreases in relative proportion to the average strength in alternative niches.

To see how this crowding algorithm affects performance, a new plan G2 was implemented that incorporates these ideas into the G1 framework. Each taxon is given an initial strength of 320. All groups of taxa lose $e = 1/2$ of their strength on the occurrence of a pertinent exemplar and gain S = NPARENTS*320 in return. These values were chosen so that the iterative procedure converges quickly and so each taxon parameter can be stored in a machine word. As before, the strength S is allocated among group members in proportion to their match scores. One question that must be resolved is how strength is to be assigned to newly generated taxa. Even though strengths are constantly being readjusted, the initial strength has some bearing on the time of convergence to the steady state value and on the immediate potential of a taxon to survive. The initial plan strength is not appropriate for new taxa since the crowding algorithm is likely to have changed the average strength in

each group. Using the average group strength for new taxa does not seem quite right either. There are several other factors that impact strength level besides those related to crowding. The most reasonable approach seems to be to assign each taxon the average strength of its parents. This is a justifiable assumption given the basic genetic algorithm inference heuristic that high performing individuals are likely to be found in the high performing regions of *A*.

Figure 5.8 shows that the performance of G2 is indeed a small improvement over that of G1. This confirms our hypothesis that the controlled growth induced by crowding should enhance the algorithm's ability to search *A*. It should be noted that this improvement occurs even though the strengths have not necessarily stabilized. The genetic algorithm is able to compensate for noisy information. One would expect that using strengths close to the steady state values would make the search of *A* much easier and improve performance even more. To test this hypothesis, two new parameters are introduced. One is the LEARNRATE which designates the number of exemplars that must occur before the genetic algorithm is applied to some group. This parameter is used to ensure that a fixed number of iterative adjustments of strength will occur between each application of the genetic algorithm and therefore allow strengths to reach their steady state value. The value LEARNRATE = 10 is

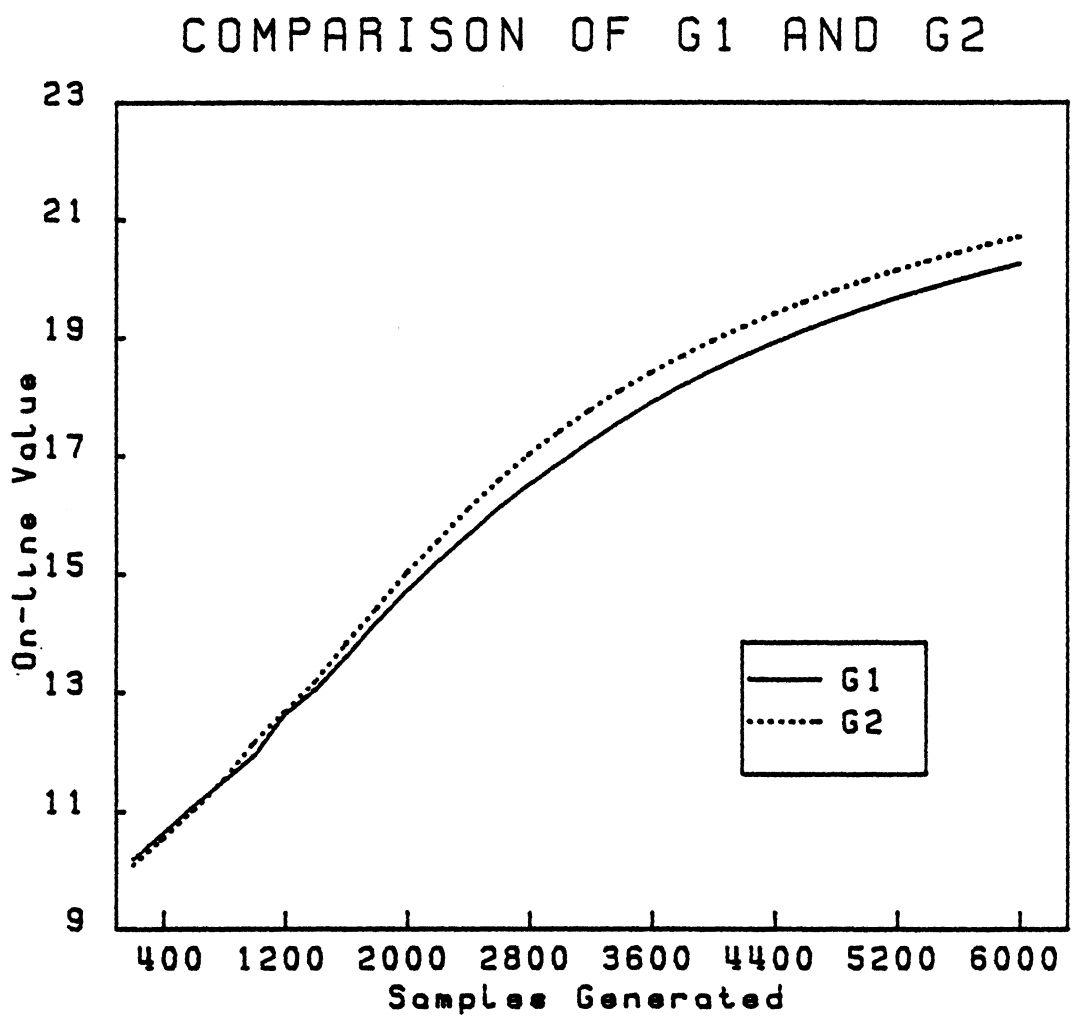

Figure 5.8. A comparison of G1 and G2.

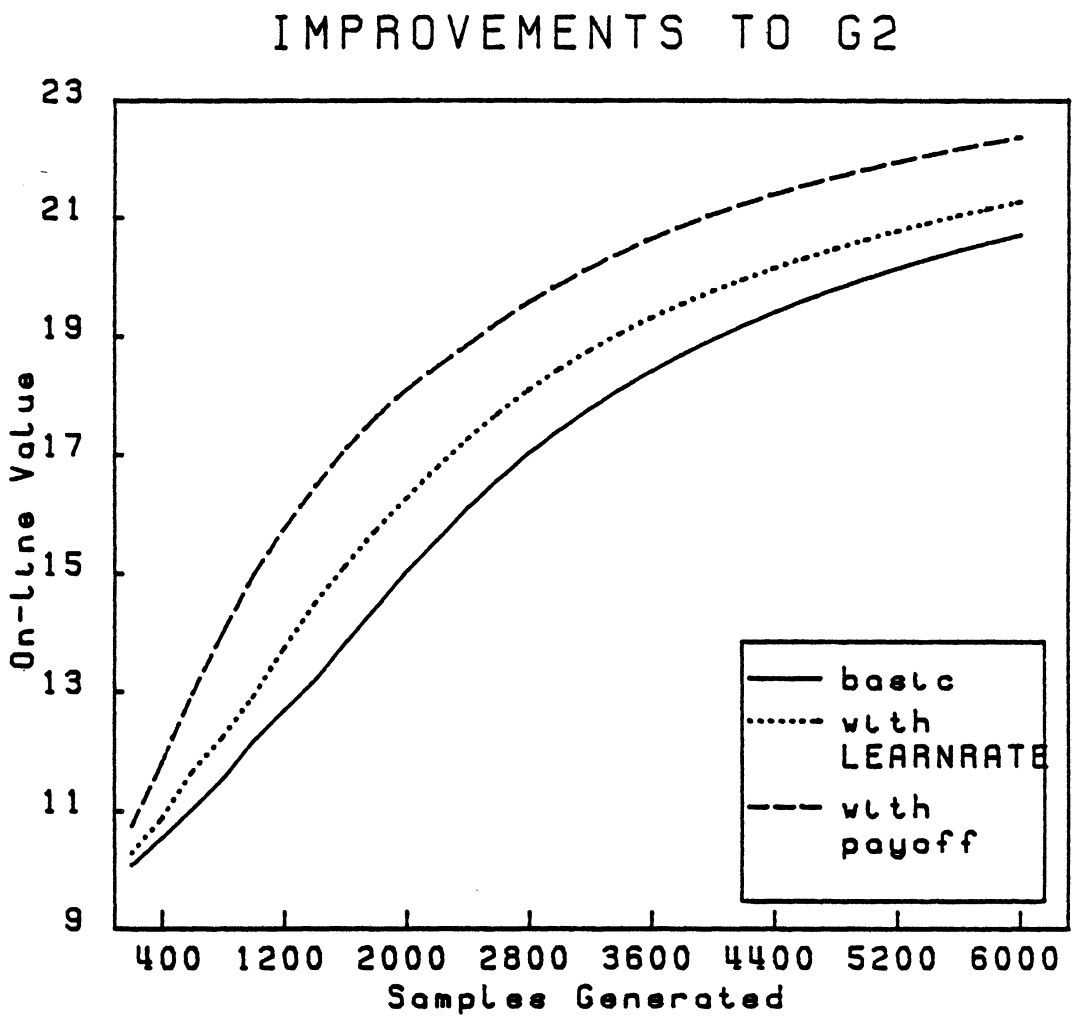

Figure 5.9. Improvements to the crowding algorithm in plan G2. First, strengths are allowed to stabilize by applying the genetic algorithm at a slower LEARNRATE. Next, payoff = strength * (match score) is used to select parents from the relevant set.

more than adequate for this purpose.[14] The other new parameter is an *eligibility* count that is associated with each taxon. This parameter is used to keep track of the number of times a subset of taxa has been relevant. Every time an exemplar is presented, the taxa in the relevant set have their eligibility - initially zero - incremented by one. When the average eligibility in a subpopulation is greater than or equal to LEARNRATE, the genetic algorithm is invoked and all eligibilities in the set are decremented by LEARNRATE. New taxa are, of course, assigned an eligibility of zero. The performance of this modified algorithm is shown in Figure 5.9. As anticipated, the new parameters resulted in an improvement in performance.

The crowding algorithm as implemented so far has improved performance, but it only has realized half of its stated purpose. Subpopulations that get overcrowded have their average strength reduced. This, in turn, reduces the expected lifetime of individuals in that group relative to the rest of the population; and, consequently, keeps the size of the group at some appropriate level. The ecological niche analogy indicates that the algorithm must do more. When a niche is under-exploited, relevant individuals should benefit from the lack of competition and proliferate to fill the niche. This does not occur with the current algorithm

---

[14]This value of LEARNRATE, together with the other plan parameters, assures us that the strength in a group is within 0.1 of its steady state value when the genetic algorithm is invoked. The value of LEARNRATE has implicitly been 1 in previous plans.

because the match score is the only criterion used to select parents and produce offspring. The match score, of course, contains no information about the relative crowding in a given niche. That information resides in the strength. It seems appropriate, therefore, to introduce another parameter - *payoff* - that combines information from both the strength and match score of a taxon. The most desirable candidate to be a parent is a taxon having a high match score, indicating it is highly relevant, and a high strength, indicating a possibly under-exploited niche. The simplest function combining strength and match score in the desired way is the product

payoff = strength * match score

The sample space for the set of parents will, as before, be generated by first determining which taxa have the highest match scores. That is the fundamental property of the restricted mating strategy G1. Now, however, the parents are chosen from that set stochastically based on payoff; and, offspring are allocated to parents according to their payoff. This enhances the fitness of individuals in an under-exploited niche by allowing them to produce more offspring. G2 was modified to include these changes and, as shown in Figure 5.9, the difference in performance is substantial. In all aspects, therefore, the crowding algorithm seems to behave as expected.

One final note about the parameters of G2. DeJong [1975] points out that once the sampling error is reduced, a

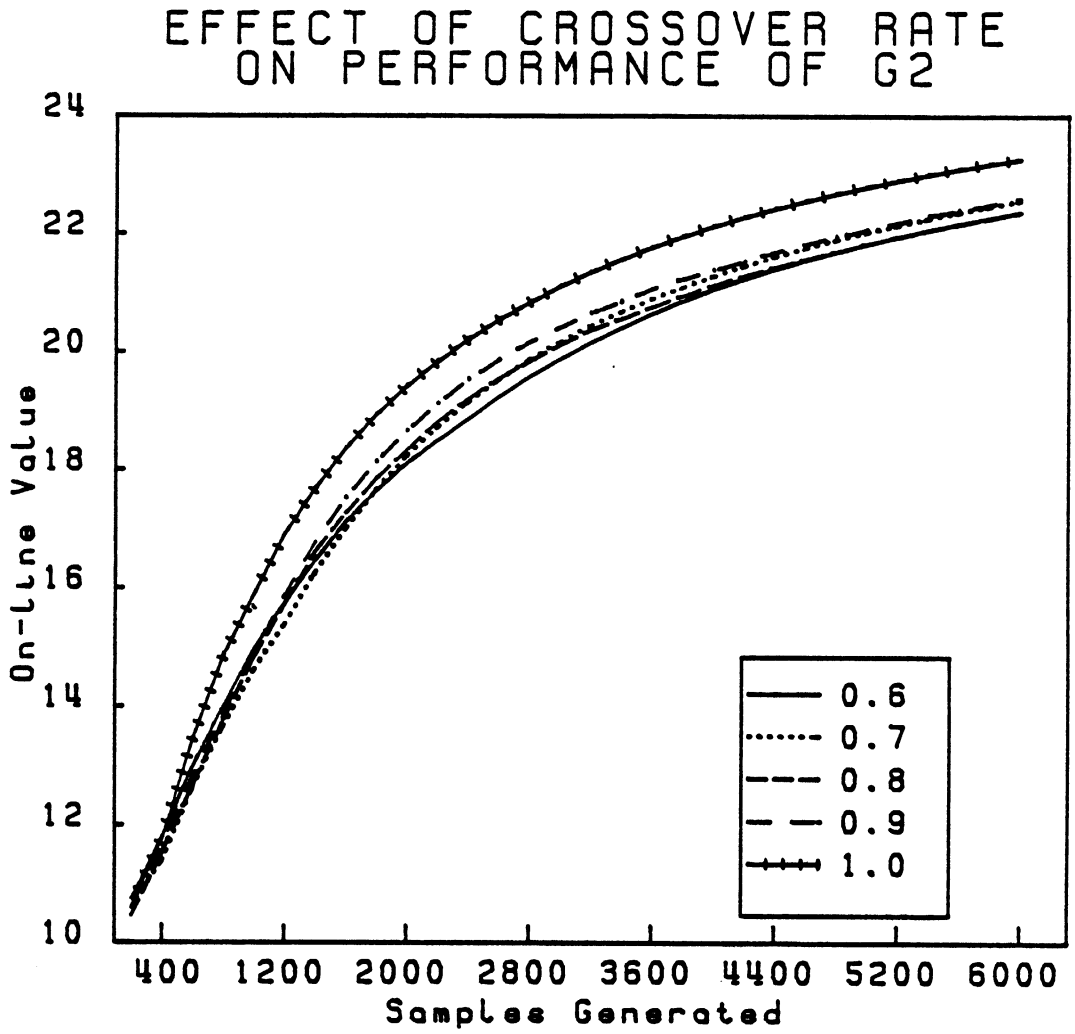

Figure 5.10. The effects of various crossover rates on the performance of G2.

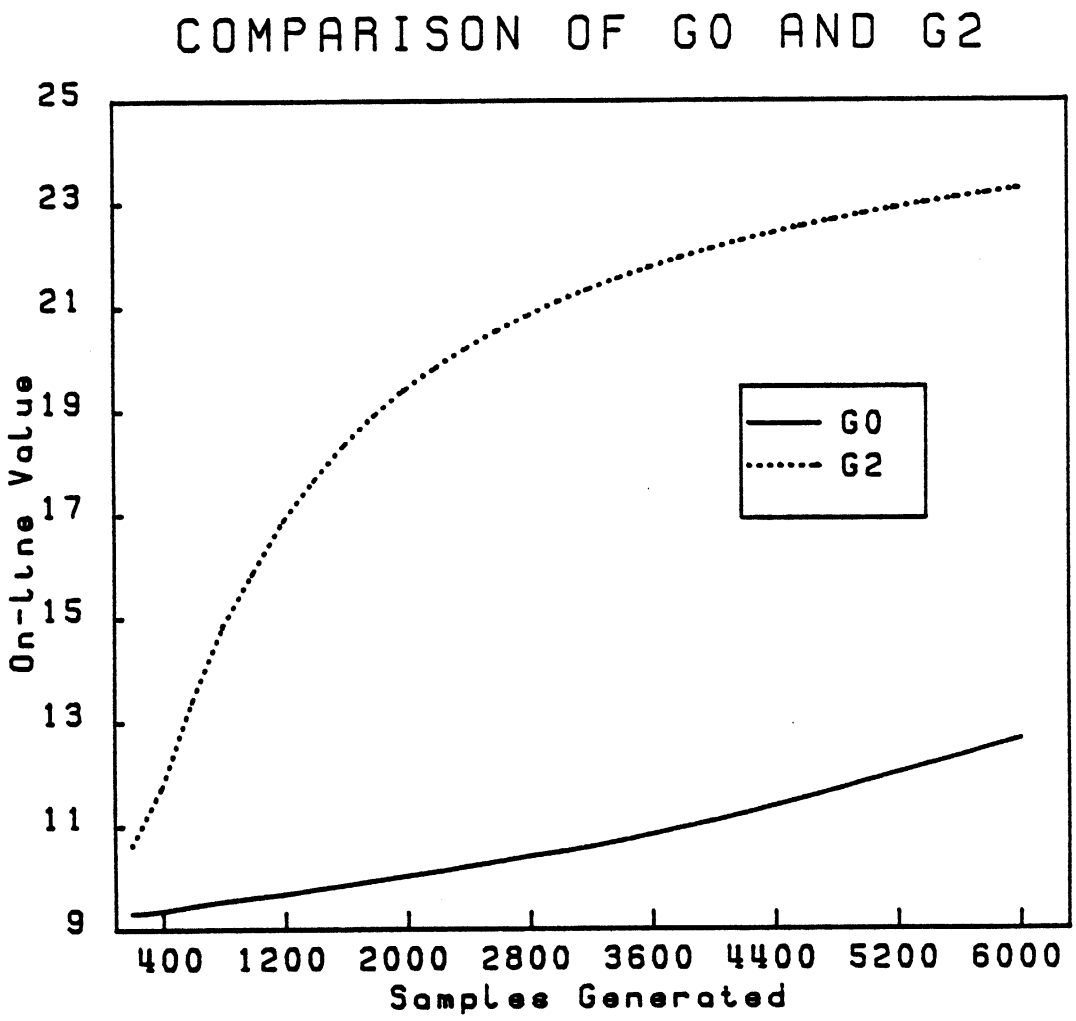

Figure 5.11. A comparison of G0 and G2.

genetic plan can benefit from increased sampling rates due to more frequent application of the crossover operator. To see whether this observation holds for G2, the crossover rate was varied to study the effects on performance. Figure 5.10 shows that, in general, increasing the crossover rate has a marginal effect on the performance of G2 with one exception: The maximum crossover rate of 1.0 produces significantly better performance than any other rate. The crowding algorithm enables G2 to search *A* thoroughly even at the highest possible sampling rate. Slower rates merely increase the likelihood of sampling error. The final modification to G2 will therefore be to use a crossover rate of 1.0. This plan is compared with G0 in Figure 5.11 to illustrate how much the adverse effects of producing fewer offspring per generation have been reduced.

Testing the Modified Algorithm

The preceding sections have discussed modifications to G0 to make it a more appropriate learning algorithm for classifier systems. In particular, the algorithm was changed to automatically generate and maintain a population containing taxa specifically tuned to each pattern class in the environment. The goal was to produce a set of subpopulations analogous to Hayes-Roth's [1973] maximal decomposition lists. It is claimed that G2 has the desired properties. In order to substantiate that claim, G2 must be tested in an environment having more than one pattern class.

Accordingly, a second pattern characteristic 0000000000****** is introduced. On each time step one of the two pattern characteristics is chosen at random and an exemplar of that characteristic is presented to the population. This particular characteristic is used because it generates none of the exemplars associated with the original characteristic 1111111111******. The two characteristics in the same environment therefore present an obvious mandate for two separate sets of taxa. Moreover, this is a simple yet important test of the robustness of the genetic algorithm. The taxon ####...####, which matches all exemplars of both classes, will be discovered quickly and make a strong bid to take over the population. It has a better match score than any non-matching taxon and, moreover, it is the only taxon having a chance to be a parent each time the genetic algorithm is invoked. The genetic algorithm must avoid getting stuck on this "local optimum", discover that there are more appropriate alternatives, and then quickly remove ####...#### from the population.

It is helpful at this point to review in more detail DeJong's [1975] analysis of the effects of crowding on the performance of genetic plans. The more offspring a plan produces per generation, the less effective crowding is in improving the plan's progress toward the optimal regions of *A*. Producing more offspring per generation means that the expected lifetime of an individual is reduced. Crowding

operates by modifying the lifetime of individuals, a strategy that becomes less effective as overall lifetimes get shorter.[15] Moreover, the larger the incremental changes in the size of various subpopulations, the less stable the relative sizes will be. If, for example, 50 offspring are produced per generation and the genetic algorithm is applied three or four times in a row to the same subpopulation, the turnover of so many individuals in a small population could have drastic effects on the size of other, supposedly uninvolved groups.[16]

TABLE 5.3

PERFORMANCE OF G2 ON TWO PATTERN CLASSES

Number of Offspring (as a fraction of number of parents)	Number of Parents		
	30	40	50
0.1	23.252	23.025	23.085
0.2	23.511	23.292	22.624
0.3	23.458	22.729	23.050
0.4	23.899	23.317	22.647

These considerations suggest that G2 should be implemented to produce very few offspring per generation.

[15] An obvious way to compensate for this effect is to increase the population size. That is an undesirable alternative here because it avoids the issue of how to manage the space available; and, because 200 taxa should be more than enough space to classify two simple patterns.

[16] This effect is not as important when only one pattern characteristic is involved because the subpopulations are correlated with schemata that overlap to such a large degree.

DeJong [1975] found that the most effective ratio of offspring to number of parents[17] is on the order of 0.2. Accordingly, G2 was tried using ratios of 0.1, 0.2, 0.3, and 0.4. In addition, the number of parents was varied to see how that might affect the development of competing subpopulations. Keeping in mind Cavicchio's [1970] results indicating that 30 is a lower bound on the number of parents that should be used, values of 30, 40, and 50 were tried. The algorithm was observed over an interval of time twice as long as previous runs. This was to make sure that each pattern class received the standard 6000 samples. Table 5.3 summarizes the results. Using 30 parents yielded the best performance for each ratio of offspring to parents. In fact, increasing the number of parents seems to have an overall detrimental effect on performance. One possible explanation for this result is as follows. When the number of matching taxa is less than the required number of parents NPARENTS, the parent taxa are chosen from the SETSIZE*NPARENTS taxa with the highest match scores. This occurs most frequently during the early stages of the adaptive process when the important first evaluations of regions in A are established. The size of this set of prospective parents is 75 when NPARENTS=30, 100 when NPARENTS=40, and 125 when NPARENTS=50. In a population of size 200, the sets associated with the two pattern characteristics are likely to overlap when NPARENTS is 40 or

---

[17]DeJong [1975] calls this ratio the "generation gap".

50. This can hinder the performance of the plan by lowering the expected rating of useful schemata. When NPARENTS=30, on the other hand, the two sets are much less likely to overlap, allowing the plan to maintain separate subpopulations from the beginning. Accordingly, the value NPARENTS=30 will be henceforth used in G2 along with the best performing ratio of 0.4 (that is, 12 offspring per generation). As a point of comparison, the performance of G2 given earlier for a single pattern characteristic, using a crossover rate of 1.0, was 23.276. The modified G2 performance of 23.899 on two pattern characteristics is an improvement in spite of the fact that the plan has twice as much work to do. Most importantly, an inspection of the populations produced by G2 in this environment shows that it does indeed generate two separate, specialized subpopulations as desired.

Summary

It was suggested near the beginning of this chapter that the genetic algorithm could be modified to be used as an effective learning component in the classifier system under study. The modifications would transform the algorithm from a straightforward function optimizer to a sophisticated heuristic, dynamically generating and maintaining a set of classifier taxa correlated with the set of defining pattern characteristics in the environment. Computer simulations indicate that such modifications do in

fact perform as expected.'[8] The first modification, a restricted mated policy, results in the isolation and development of clusters of taxa, or subpopulations, correlated with the inferred structural characteristics of the pattern environment. The second modification, a crowding algorithm, is responsible for the dynamic and automatic allocation of space in the population among the various clusters. Together, these two modifications produce a learning algorithm that satisfies all the criteria set forth at the beginning of the chapter.

It was noted that these two changes have an effect analogous to the interaction of species in a natural population. This analogy is worth pursuing. How is the dynamic balance between two clusters effected by differences in total resources available in their respective niches? What if the relative reproduction rates, as determined by the relative proportion of each kind of exemplar in the environment, differ significantly? How is the situation different when the clusters overlap to varying extents? These are all interesting questions. Investigating them using the genetic algorithm might lead to an improved version of the algorithm; and, could perhaps make some contribution to the study of population ecology. Since the

---

[8]Because the objective of this chapter was simply to develop a version of the algorithm having the desired properties - to show that it was possible and not very difficult - little time was spent carefully analyzing the effects of each parameter, the significance of differences in performance, etc. Such an analysis should be part of any future research on this algorithm.

algorithm as it stands is adequate for the purposes of this thesis, however, the questions will be left for future researchers to answer.

CHAPTER VI

IMPLEMENTATION OF THE ADAPTIVE SYSTEM

The genetic algorithm developed in the last chapter has been shown to be effective at generating populations of classifier taxa correlated with the pattern characteristics underlying a set of binary strings. It remains to be shown that the algorithm can generate populations of complete classifiers - including tags and message taxa - that are useful to the hypothetical organism as it behaves in the simulated environment. Two sets of tests will be performed to show that the algorithm has the required capabilities. First, it will be established that, given information about how to rank various tags, the algorithm can discover highly rated combinations of input taxa and tags. Some attempt will also be made to establish that the algorithm is useful for a large subset of schematic classification problems. Second, the algorithm will be installed in the hypothetical organism where message passing plays a very central role. The naive organism will then be tested in the simulated to demonstrate that the adaptive system performs as claimed.

Tags and Concept Formation

The algorithm as it has been used so far discovers those combinations of features in the stream of incoming signals that best characterize the underlying pattern structure. However, discovering the set of relevant characteristics is only half of what is required for the perceptual component in the hypothetical organism. The characteristics must be identified according to which pattern class they are characteristic of. It is only when both kinds of information are brought together - analysis into pattern characteristics, then synthesis into a representation of the pattern class - that the organism has a useful basis with which to interact with the environment. This is obviously critical when pattern classes have more than one defining characteristic. Since the pattern class definitions are unknown to the system in advance, the problem confronting the organism is accurately described as an inductive concept formation and discrimination task. This kind of learning task has long been of interest to psychologists and the artificial intelligence community.

> Inductive tasks have always been a prominent part of the artificial-intelligence landscape. The reasons for this seem twofold. For one, we have inherited a classic distinction between deduction and induction, so that the search for intelligent action should clearly look to induction. Second, American psychology has largely identified the central problem of conceptual behavior with the acquisition or formation of concepts - which in practice has turned out to mean the induction of concepts from a set of presented exemplars. [Newell,1973, p. 16-17]

A "concept" in the classifier system is a subpopulation of classifiers correlated with the characteristics of a pattern class. Each such subpopulation is identified as an entity in the system by the fact that member classifiers have similar tags. It is therefore reasonable to use a concept formation task as a vehicle for testing the ability of the genetic algorithm to discover appropriate combinations of taxa and tags.

Accordingly, each string in the population is extended to include a 16 bit binary tag. Even though the strings do not include message taxa yet, it is convenient to refer to them as classifiers. As before, pattern class exemplars are presented to the population one at a time and the input taxon match score is computed for each classifier. Recall that each exemplar is chosen at random from one of the two pattern classes given by 0000000000****** and 1111111111******. In addition, however, information about which pattern class the exemplar belongs to is made available in the form of a binary string identifier - 111...111 for the first pattern and 000...000 for the second. Each classifier gets a tag score which indicates how closely its tag matches the pattern class identifier. The tag score is computed simply by counting the number of bits the two strings have in common. The concept formation and categorization task for the adaptive system is to generate taxa that designate the structural regularities of a category and link each taxon with a tag identifying its

associated pattern class. The set of relevant classifiers is determined as usual except that, in computing payoffs and reallocating strengths, both the match score and the tag score are used. This gives the genetic algorithm information about how combinations of taxa and tags should be ranked.

It is not obvious what the best way is to combine the information from the match score and the tag score. On the one hand it seems that the two factors should have an interactive (i.e. multiplicative) relationship, placing very heavy emphasis on the ranking of the overall classifier as more than the sum of its parts. On the other hand, it is possible that a compensatory (i.e. additive) relationship might improve performance by allowing highly ranked taxa or tags to remain available as "building blocks" that can later be linked together in the proper combinations. In order to evaluate these two alternatives a new performance measure is introduced. On every cycle the genetic algorithm is applied, the set of relevant classifiers is used to compute a categorization decision for the system. Quite simply, a weighted average tag score is computed for the relevant classifiers - the weights used being the product of the classifier strength and match score. The range [0,16] of possible tag scores is divided into three regions: [16,11] for tags that are clearly correct; [0,5] for tags that are clearly incorrect; and [6,10] for tags that are too close to random to be decisively right or wrong. The system is said

to have made an "error" whenever the average tag score falls in one of the lower two regions. The performance measure is simply a count of the cumulative number of such errors made by a given adaptive plan. A plan does well under this criterion when the total cumulative error is low and eventually remains constant, indicating that the system has stopped making mistakes. Note that this is a very stiff criterion. Standard decision criteria for such a task would not count trials in which the average score was in the middle region, only record errors when the score is equal to or worse than random, etc. Under the chosen criterion, a plan does well only if it converges decisively to a correct set of classifiers.

Figures 6.1 and 6.2 compare the alternatives for combining tag and match scores in terms of on-line performance and total categorization errors. Adding the two scores together produces the best on-line performance. This is to be expected since on-line performance is computed using the match scores of input taxa, ignoring whether or not they are linked to the proper tag. A highly ranked taxon linked to the wrong tag can survive longer when the match score and tag score are compensatory, and so the plan's on-line performance should be higher. For the same reason, multiplying the two scores produces the fewest errors. Classifiers leading to incorrect decisions are eliminated more quickly. Note that both plans stop making errors at about the same time, an indication that keeping

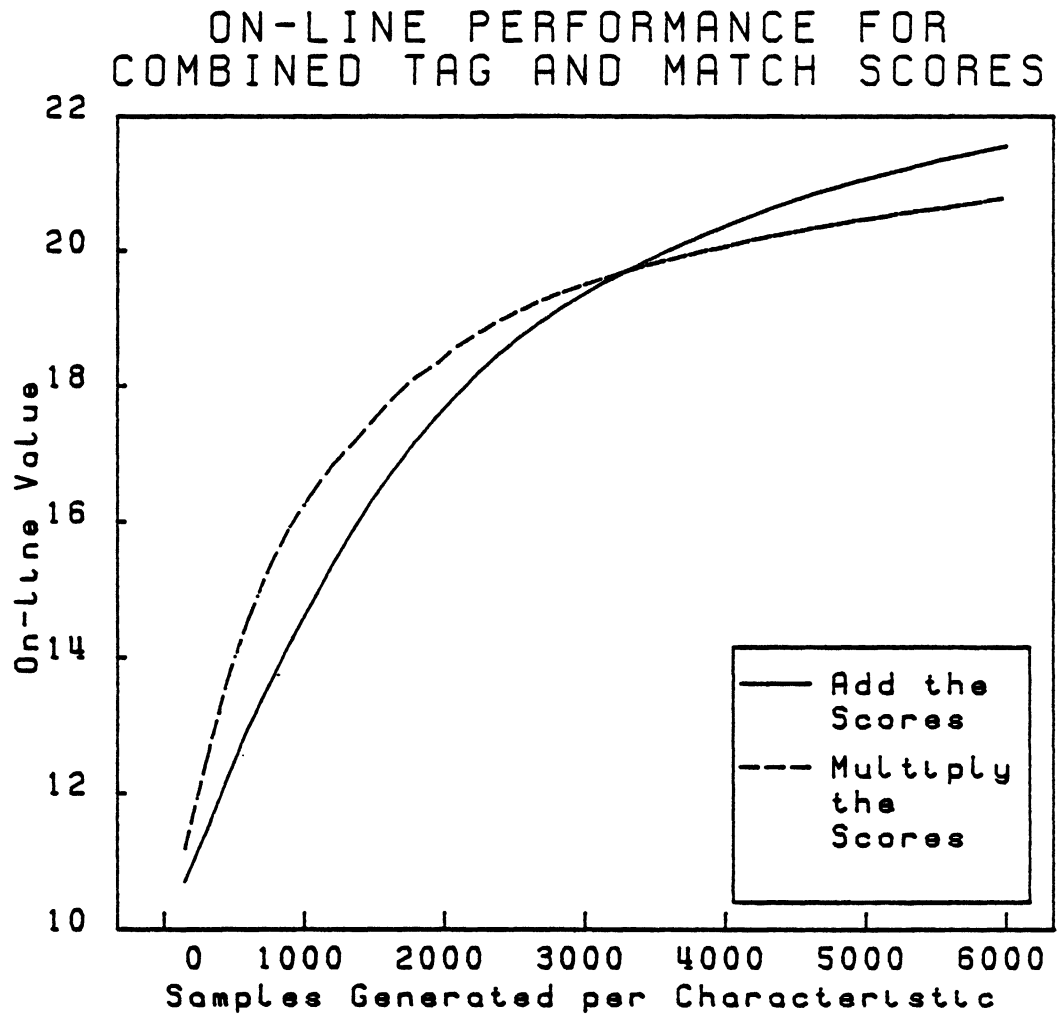

Figure 6.1. On-line performance for the additive and multiplicative combination of tag score and match score in computing payoffs and reallocating strengths.

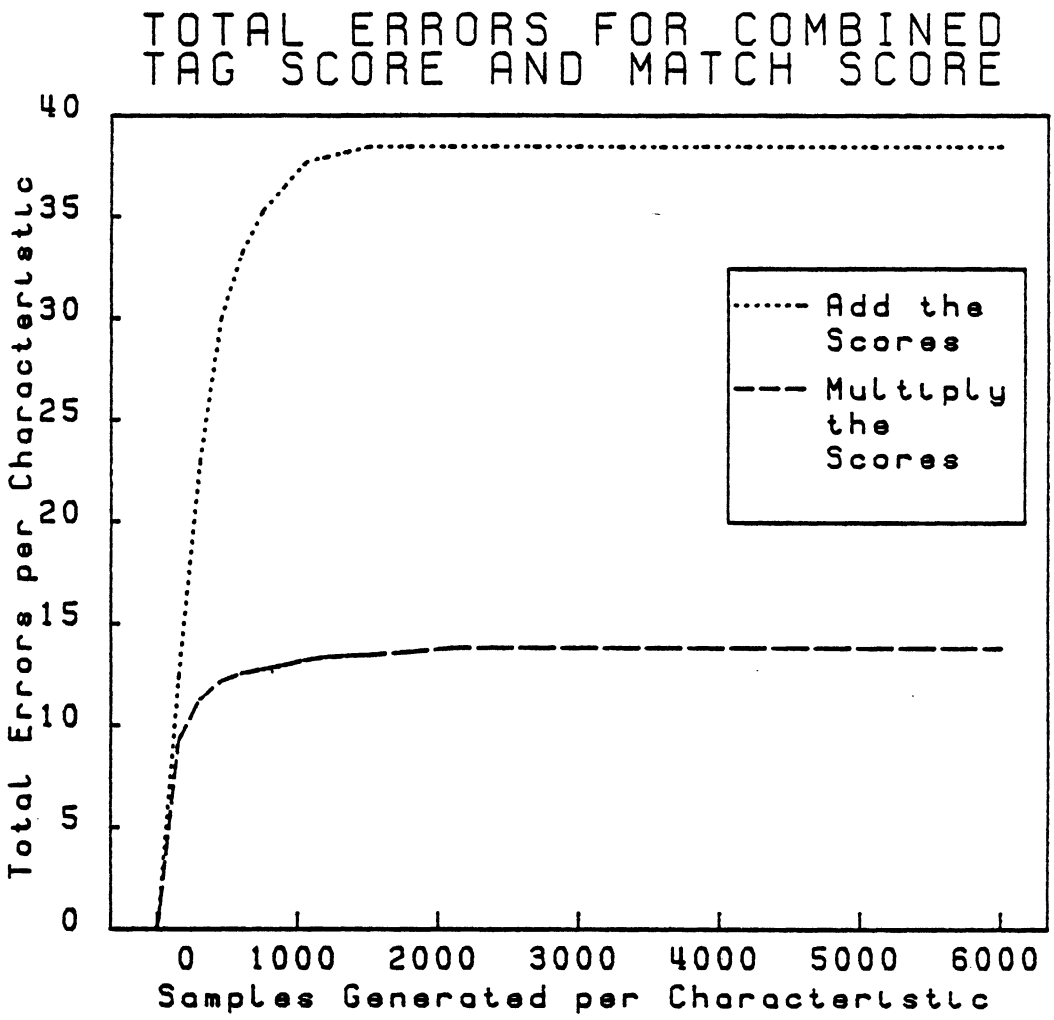

Figure 6.2. Cumulative categorization errors for the additive and multiplicative combination of tag score and match score in computing payoffs and reallocating strengths.

good taxa as "building blocks" - even though they are linked to the wrong tag - is no advantage in terms of time to convergence. This is because the genetic algorithm searches the space of possibilities thoroughly, evaluating all relevant schemata simultaneously. Moreover, while the difference in on-line performance between the two alternatives is noticeable but not large, the difference in total errors made is nearly a factor of three. The bottom line for the hypothetical organism is a capability to learn as quickly as possible with as few mistakes as possible. Failing to approach a needed resource or avoid a noxious object in the simulated environment could very well be disastrous.[1] The clear choice is therefore to multiply the tag score and match scores.

One property of the simulated environment has not yet been taken into account. The adaptive system has to discover the functional stimulus categories in spite of the fact that reinforcement is available irregularly and infrequently. In terms of the concept formation task being considered here, the implication is that the system must be able to learn the correct categorizations even when the binary string identifiers are available only intermittently. In order to determine if the genetic algorithm can make the desired inferences under such circumstances, two modifications are made to the system. First, each

---

[1]One can imagine other environments in which this difference in performance might be worth the increase in total errors.

classifier is given a new parameter called *tagscore*. This parameter estimates the tag score to be expected whenever the classifier in question is relevant to an exemplar. The tagscore parameter is updated for relevant classifiers every time a pattern class identifier is presented to the system. The adjustments to tagscore are made using an algorithm similar to the one used to modify strengths.[2] Tagscore is used in the revised system instead of the true tag score whenever the true score is not available.[3] The only exception to this is when payoff is computed. Since repeated adjustments make strength an estimator of the expected tag score, including tagscore in the payoff calculation is not necessary.

The second change to the system is to correlate the application of the genetic algorithm with the availability of a pattern class identifier. It is clearly preferable to have accurate tag scores when computing payoffs to be used for generating new classifiers. At the same time, the results from the last chapter indicate that it advantageous to apply the algorithm at the frequency LEARNRATE - allowing strengths to reach their steady state value. One way to satisfy both constraints is to allow pattern identifiers to

---

[2]As with strength, offspring are initialized with the average of their parent's tagscores.

[3]Information about tags must be used every time strengths are adjusted. Otherwise, the occasional inclusion of a tag score becomes mere noise in the steady stream of match score information.

induce a temporary boost[4] in eligibility for relevant classifiers. The size of the boost should be large enough to make sure that the genetic algorithm gets applied most often when pattern class identifiers are available; yet, small enough to prevent the learning frequency from being driven much beyond LEARNRATE. The value chosen is

[2*LEARNRATE*(tag score)] / (maximum possible tag score)

This value gives a full boost of 2*LEARNRATE to classifiers with a perfect tag. Since most classifiers are unlikely to have perfect tags in the early and intermediate stages of the adaptive process, the boost is large enough so that a relevant set with zero average eligibility and random tag scores can participate in the genetic algorithm. The boost is proportional to tag score so that classifiers with the most appropriate tags are more likely to contribute to the generation of new classifiers. This relationship between learning signals and what gets learned is meant to parallel the effect reinforcement has been postulated to have on the consolidation of structural changes in the central nervous system [Hebb,1972]. The pattern class identifier is a reinforcement signal that generates a non-specific change in eligibility for all classifiers. The classifiers most sensitive to this change - those having the highest tag score - are most likely to be involved in any subsequent

---

[4]The boost lasts at most for the duration of the current execution cycle and becomes zero as soon as a classifier participates in the genetic algorithm.

learning.[5]

To determine how the adaptive system behaves given these changes, the concept formation task was modified so that pattern class identifiers are made available at random, on the average of once every 19 exemplars.[6] As before, the system's categorization decision is recorded on every cycle the genetic algorithm is applied. Figures 6.3 and 6.4 compare the behavior of the modified system with the performance of the previous version. There is a large increase in the total number of errors, though this is to be expected given that the adaptive plan is now working with estimates instead of actual tag scores. The total error is still less than the level generated by adding tag scores and match scores. Moreover, the on-line performance is the best seen so far on the concept formation task. Using the estimated tag score apparently allows good taxa linked with bad tags to survive a little longer, at least until the estimate comes to reflect how bad the tag really is.

Finally, it is important to investigate which kinds of pattern classes the system is capable of learning and discriminating. The two patterns used so far are easily discriminated because they have a different value along each relevant dimension. The discussion of natural categories in

---

[5]The idea of a fixed LEARNRATE needs to be re-examined. A more event-driven criterion will be necessary in a more sophisticated environment.

[6]We shall see later how this approximates the number of signals an organism is likely to encounter before contact with an object.

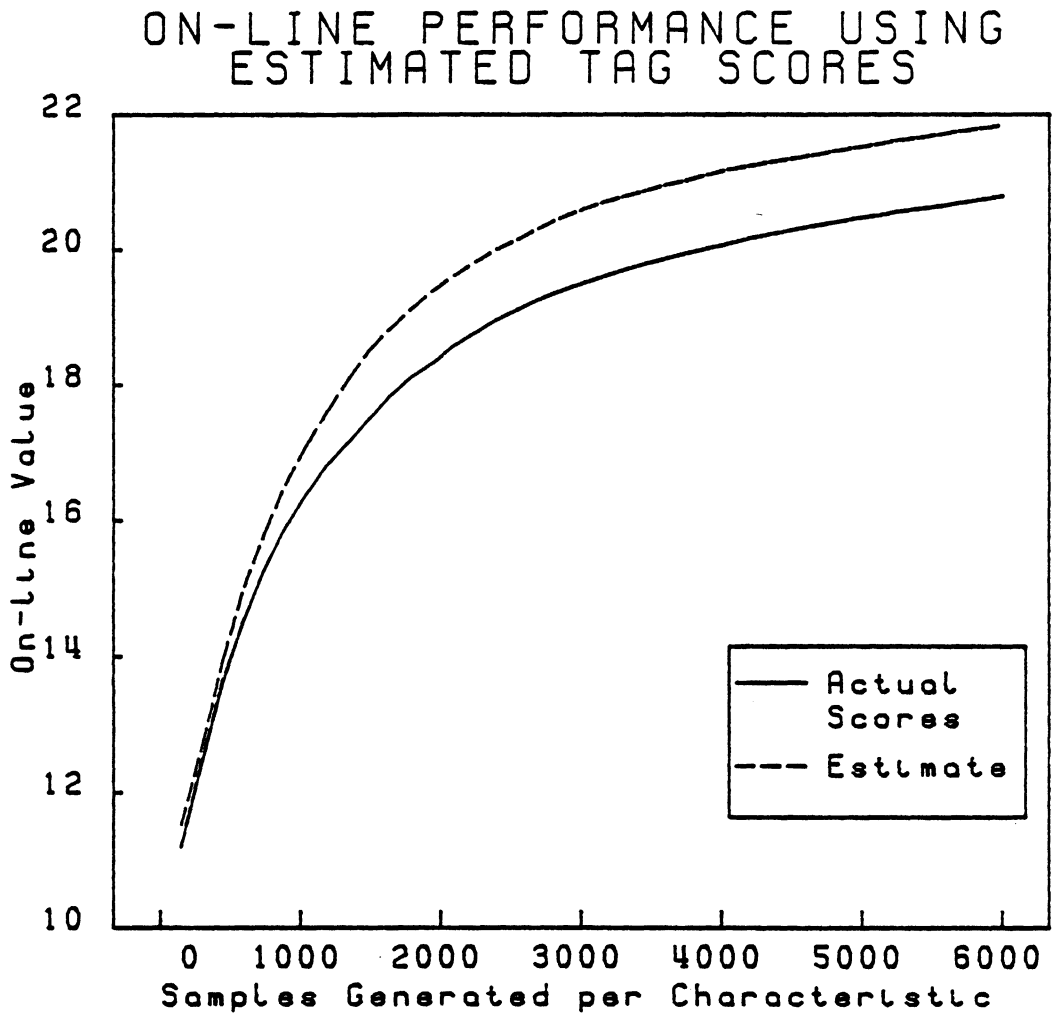

Figure 6.3. On-line performance using actual and estimated tag scores.

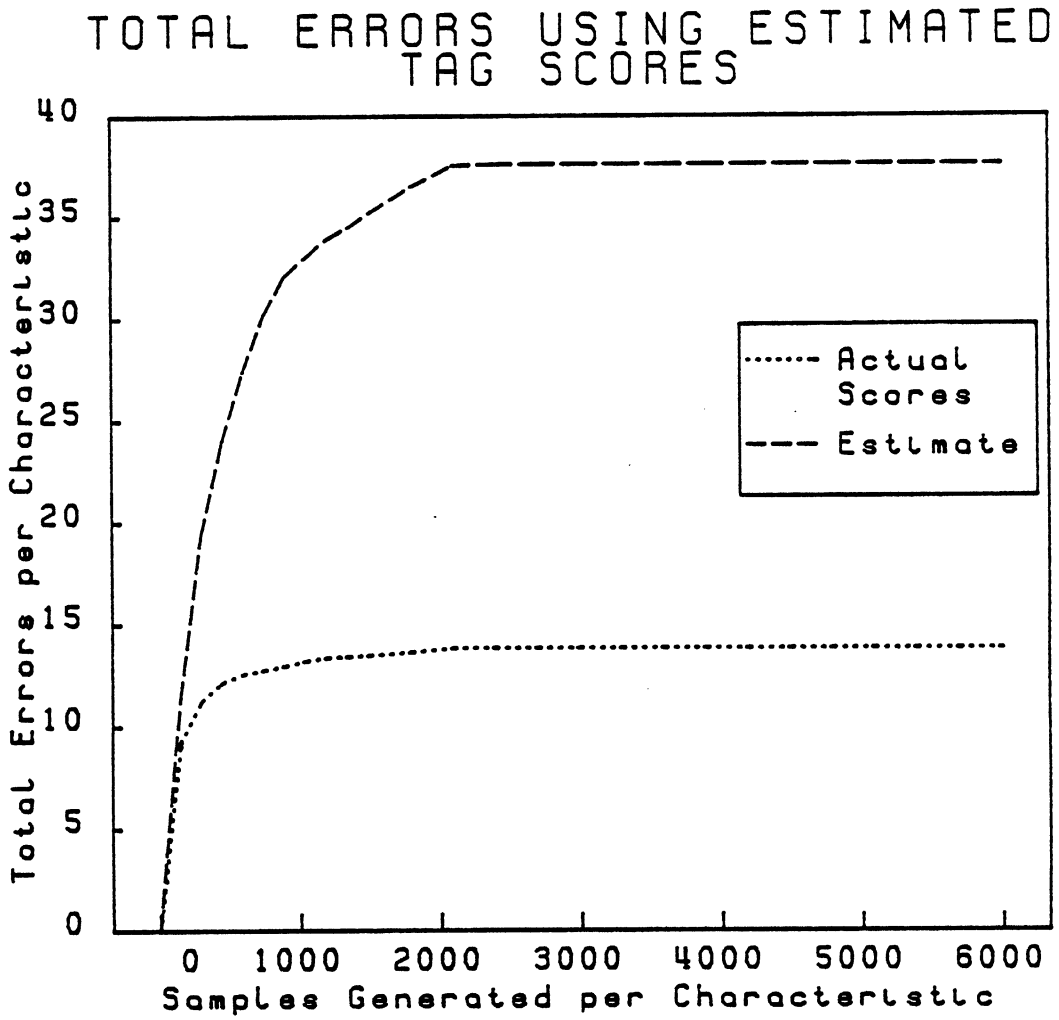

Figure 6.4. Cumulative categorization error using actual and estimated tag scores.

Chapter 4 emphasized that real-world equivalence classes are not likely to have such rigid, necessary and sufficient membership criteria. Categories may be defined with several sets of relevant dimensions - perhaps overlapping, perhaps not; categories may be defined in which the members share no features in common; and so on. The schematic approach to generating patterns was chosen for the simulated environment precisely because it allows categories having many such properties to be easily defined.

TABLE 6.1

FIVE CONCEPT LEARNING TASKS

Task Number	Pattern A	Pattern B
1	1111111111******	0000000000******
2	11111111********	********11111111
3	*****1111111111	1111111111*****
4	11111111******** ********11111111	00000000******** ********00000000
5	1111111111****** 0000000000******	******1111111111 ******0000000000

In order to get some feel for the range of schematic pattern classes within the system's competence, a set of five concept learning tasks[7] is proposed (see Table 6.1).

[7]The symmetry of the patterns used here makes it easier to quickly look at and evaluate each population at the end of an experiment. This has no bearing on the kinds of patterns the system is capable of discriminating.

The first task is the easy discrimination of the two patterns used so far. The second is somewhat more difficult because the patterns are defined on different sets of relevant dimensions. This means that there is no one dimension the system can use to reliably discriminate between the two patterns in all cases. The learning algorithm must therefore be able to infer the relevant coadapted sets of feature values.[4] This situation is complicated further in the third task because the patterns share the same values along each relevant dimension they have in common. Not only must the learning algorithm infer coadapted sets of features; it must also avoid allocating a permanent niche in the population to the common pattern *****111111*****. In the fourth task each pattern is disjunctively specified with two pattern characteristics. This is difficult because there are no common features shared by all members of a class - they only have a common "family resemblance". This task is also difficult because the system has to learn four pattern characteristics instead of two. In order to avoid having the limited space in the population slow down the performance on this task and the next one, the population size was increased from 200 to 300. The fifth task involves more family resemblance categories and is even more difficult since members in one category share several features in common with members of the other

---

[4]To avoid ambiguous classifications, the strings 111...111 and 000...000 are never presented as exemplars for any of these tasks.

category.

The performance of the system on each of these tasks is shown in Figure 6.5. The total number of errors increases with the difficulty of the task as expected. In each case the learning curve reaches a plateau, indicating that the system discovered the defining pattern characteristics. The system makes the most errors when the two categories share several features in common. This is because classifiers tuned only to those common features are discovered quickly by the genetic algorithm. These classifiers are always relevant and have a consistently high match score. The only factor that eventually works against such classifiers is a mediocre estimated tag score.' The ease with which the system copes with these various tasks - together with DeJong's [1975] success in optimizing functions that are highly non-linear, multimodal, etc. - emphasize the flexibility and power of genetic algorithms as general purpose inference heuristics.

The Hypothetical Organism as an Adaptive System

Having established that the genetic algorithm can discover useful combinations of input taxa and tags, we are now ready to demonstrate that it is capable of generating complete classifiers. This will require that the execution cycle of the classifier system be modified to incorporate

---

'Always being relevant means that the tag is never always right or always wrong. The overall estimated tag score is therefore in the middle range.

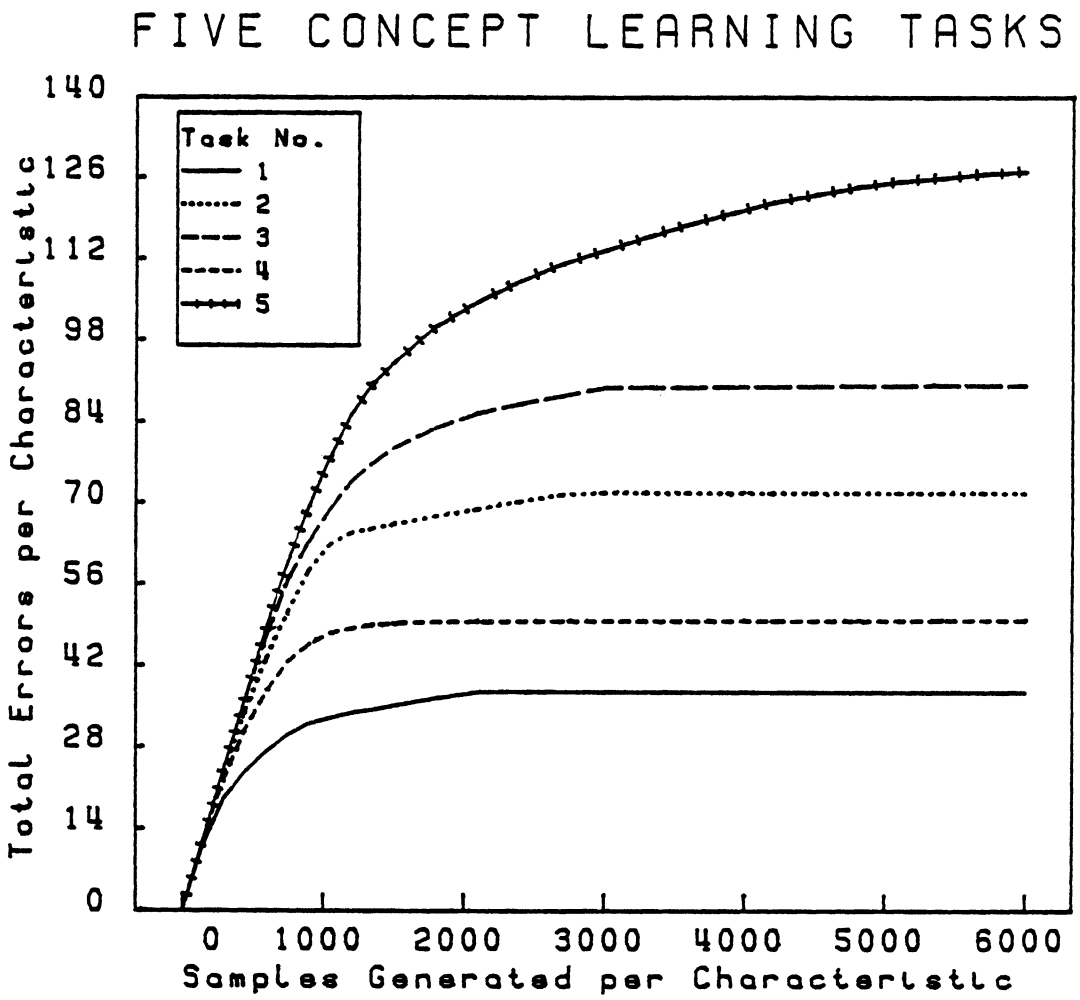

Figure 6.5. Learning curves for several categorization tasks.

the genetic algorithm - making sure the "selective pressures" generated by the system architecture direct the algorithm's search in appropriate ways. Overt behavior is the most important factor in evaluating how effective a set of classifiers is for the hypothetical organism. Accordingly, the organism will be judged as an adaptive system in terms of how well its learned behavior compares with the "perfect" behavior of the instinctive model developed in Chapter 3.

In more detail, the overall execution cycle for the classifier system now consists of the following steps:

1) Both message lists are emptied and all active classifiers are deactivated.
2) New classifiers generated during the preceding cycle are inserted into the population. Existing classifiers are selected to be replaced in inverse proportion to their strength.
3) The excitation level of every classifier is reset to zero.
4) Signals from the environment are placed on the first message list.
5) NACTIVE perceptual classifiers are chosen to become active (see below).
6) Each active classifier places exactly one message on the second message list. The message intensities are given by M_MAX * (e_i/E) where e_i is the excitation level of the i^{th} active classifier, E is the highest

excitation level in the active set, and M_MAX is the highest intensity allowed in the system for internally generated messages. The value M_MAX = 5 was arbitrarily chosen for this system.

7) NACTIVE affect classifiers are chosen to become active (see below).

8) If reinforcement - that is, contact with an aversive or appetitive object - is signalled, all active perceptual classifiers'° have their feedback estimates revised as explained in Chapter 4.

9) The affective code corresponding to the activity in the affect population is computed as follows. The tags 000...000 and 111...111 were arbitrarily chosen as the system's innate designations of "+" and "-" codes respectively. The code associated with a classifier is determined by counting the number of 1's in its tag. If there are more 0's than 1's the code is "+"; otherwise, the code is "-". Within the set of active classifiers, a weighted sum of the excitation levels associated with each kind of code is computed. The weights used are the number of 1's in the tag of "-" classifiers and the number of 0's in the tag of "+" classifiers. The code with the highest weighted sum is the affective code for the given set of active classifiers.

'°Affect classifier messages have no function in the system as described so their feedback estimates remain unchanged.

10) This code is used by the hierarchical control mechanism in its determination of an overt response. This top level cycle does not directly involve any of the system's learning mechanisms. However, since only active classifiers can generate messages and receive feedback, the parameter NACTIVE can have an effect on the system's ability to learn which messages are most effective. The more diverse the set of perception messages, the more difficult it is for the affect population to discern and become attuned to some consistent pattern characterizing those messages. This is particularly a problem in the early stages of the adaptive process, when perception messages are essentially random. A value of NACTIVE that is too high makes it difficult for the system to "focus" on and strengthen any particular message pattern. A value of NACTIVE that is too low might not always provide an adequate sample of the way the message patterns are developing. The value NACTIVE = 5 was chosen for this system as a guess based on the parameter settings used by Holland and Reitman [1978].'' It is interesting to note that this kind of limitation on the useful number of active classifiers is analogous to the well documented channel capacity limitations evident in human information processing [Miller,1956;Mandler,1967].

It should be pointed out that it is very easy to

''The limited memory available on the computer used to simulate the system made efforts to try several higher, perhaps more optimal, values of NACTIVE impossible.

implement much more powerful controls on the channel capacity of this system. Suppose, for example, that we redefine the parameter M_MAX to designate the total intensity of the messages generated by a set of active classifiers. Suppose further that we introduce two new parameters, MIN_FRACTION and MAX_FRACTION, which give bounds on the total excitation permitted in the active set as a fraction of the total excitation in the population. This makes the number of active classifiers NACTIVE a time varying quantity, increasing or decreasing as required to keep the total excitation in the active set within the given limits. These two changes give the system an "attention span" in the sense of Cunningham [1972]. When the match scores with a given message configuration are low, the number of active classifiers - or attention span - increases above the "normal" level, allowing more alternative messages to be processed. Thus, when the system encounters unfamiliar stimuli, its sensitivity is increased in the sense that there is more input-related processing activity than usual. This is a kind of primitive search operation allowing "low probability" structures to be tried in situations where no solid representation is available. Note that since M_MAX is fixed, some of the messages generated by this activity will have lower than normal intensity. In this way the system is capable of processing under the influence of "fringe" elements, a factor mentioned in Chapter 1 as being very important in human information

processing. Similarly, fewer classifiers will become active when the match scores with a given message configuration are high. In familiar situations, therefore, processing is handled compactly by a few classifiers, each of which transmits a high intensity message. One can imagine variations on this theme that might be even more powerful. The change in NACTIVE for one population might be under the control of classifiers in some other part of the system. In this way, the generation of alternatives - or searches - can be initiated and controlled by internal states of the system. Kaufmann [1979] suggests that this is one of the basic generic *transformations* of activity required for problem solving in a parallel information processing system. However, the current simulated environment does not require such sophisticated processing on the part of the organism. It is sufficient therefore for NACTIVE to remain as a constant.

The inner loop of the execution cycle, deciding which classifiers in a population are activated, involves the following steps:

1) For each message on the list,
    a) Determine the set of relevant classifiers: Matching classifiers are always relevant. If there are at least NPARENTS matching classifiers, use the set of matching classifiers as the relevant set; otherwise, select additional classifiers (to get a total of

NPARENTS relevant classifiers) by taking a non-uniform random sample based on payoff from the SETSIZE*NPARENTS classifiers with the highest match scores.

b) Determine the amount of facilitation impinging on each relevant classifier.

   i) In the affect population, facilitation is only available when the system is reinforced. Classifiers with the appropriate affect code are facilitated in proportion to their tag score. In this way, the system is provided with information about the correct tag for a given situation.

   ii) Facilitation is always available in the perception population. Tags here designate the appropriateness of a classifier for generating behavior relevant to one of the system's needs: obtaining resources or avoiding noxious signals. As explained in Chapter 3, obtaining resources is the most pressing need when the deprivation level exceeds a THRESHOLD; otherwise, "pain" aversion is the primary need. Classifiers with tags designating the required function are facilitated in proportion to their tag

score and the deprivation level (or THRESHOLD). Reinforcement leads to facilitation of perception classifiers in a manner analogous to the way affect classifiers are facilitated.

c) Increment the excitation level of each relevant classifier by an amount equal to the product of three (or four) factors: strength, match score, message intensity, and facilitation (if available).

d) Update the strength of each relevant classifier as discussed previously. If reinforcement is available, update the tag score estimate as well.

e) Compute the average eligibility in the relevant set. Reinforcement leads to a temporary boost in eligibility as noted previously.

f) If the average eligibility is greater than LEARNRATE,

   i) Apply the genetic algorithm to generate NOFFSPRING = 0.4*NPARENTS new classifiers. Parents are chosen by taking a non-uniform random sample based on payoff from the set of matching classifiers. If there are fewer than NPARENTS matching classifiers, the parents are selected from the SETSIZE*NPARENTS classifiers with the

highest match scores.

ii) Decrement the eligibility of all parents by LEARNRATE.

2) Increment the eligibility of each excited classifier by 1.

3) Assign each of the NPARENTS most excited classifiers[12] a probability of becoming active given by e_i/E where e_i is the excitation of classifier i and E is the total excitation in this *competition set* of NPARENTS classifiers.

4) Select NACTIVE classifiers from the competition set without replacement using this probability distribution.

Several points need to be emphasized here. Matching classifiers are always relevant because they are clearly pertinent to a given message. When there are not enough matching classifiers available[13], the remaining classifiers are selected as the system's "best guess" about the situation based on match scores. This selection is done stochastically, taking into account the strength of each alternative, because the eventual relevance of the classifiers chosen is uncertain. Note that matching classifiers do not automatically become parents in the

---

[12]There are certain to be at least NPARENTS excited classifiers.

[13]Each message evokes at least NPARENTS relevant classifiers so that the system always has an adequate sample of potential classifications to work with.

genetic algorithm. Parents are always chosen stochastically, preventing the system from investing too heavily in short term or random "peaks" in the "fitness" of a classifier. Only those classifiers that are consistently relevant will get a foothold in the population and proliferate. This is a pressure towards generality (more #'s) in classifier taxa, counteracting the pressure for specificity generated by the modification of strengths based on match score. The result is that the system automatically generates and maintains classifiers having the most appropriate levels of generality. There are very few examples of artificial learning systems that have mechanisms for both generalization and discrimination (e.g. Winston [1975] and Anderson and Kline [1979]). The classifier system described here discovers the proper level of generality by simultaneously testing specific and general alternatives - allowing the "selective pressures" of experience to prescribe the appropriate blend of specialists and generalists.

Since the match score is the primary factor in determining which classifiers are relevant, this system is a very simple model of sensory driven processing.

> ... when sensory information enters through the sensory system, the processes operating on it do so automatically, up through the extraction of features. Then as a result of these processes. the sensory memory is [excited], with different regions representing the different feature sets. Imagine, if you will, a memory space with regions of [excitation] flourishing here, fading away there. Each new sensory input starts up new [excitation], and the system must

> attempt to organize the structures that have been [excited].... [Norman,1976, p. 73-74]

Though the processing is sensory driven, it is not a simple "filter model" in which the information flow is unidirectional. Sensory input nominates those structures that are the best candidates for classifying the current situation. The central motive state uses facilitation to nominate those structures most pertinent to the current goal of the system. These two influences are merged in the competition process to activate a representation that is appropriate for the organism's current activities.

Classifiers compete probabilistically for activation based on their total excitation. Even though excitation is computed as the sum of various factors, the fact that only the NPARENTS most excited classifiers compete for activity makes the transition from excitation to activity a variable threshold function. This approximates a more detailed model in which highly excited elements inhibit those with low excitation. The variability of the threshold - what counts is which classifiers are most excited, not how excited they are - allows the system to respond to varying levels of signal intensity. Facilitation biases the outcome of the competition by increasing the excitation in certain regions. This makes the processing in the system context dependent in a limited though important sense. The basic machinery is in place to implement biases on activity that are generated by representations in other parts of the system.

One final point about excitation. The reader may have

noted that, since strength is used both as a measure of the "fitness" of a classifier and as part of the crowding criterion, it makes no sense to compare individual strength levels across various subpopulations. A group of classifiers having high fitness will generate many offspring, which means that the size of the group will increase and absolute strengths will go down. To compensate for this, the average strength in each relevant set is normalized to the same value before strength is used to compute excitation. The contribution of strength to the excitation of a classifier therefore depends not on the absolute amount, but on the amount of strength relative to the average strength in the group.'[4]

Simulations of the Adaptive System

We are now ready to determine if the revised version of the hypothetical organism is capable of learning how to behave in the simulated environment. Recall that the hard-wired version of the organism was designed with innate releasing mechanisms enabling the environment to select - subject to certain motivational controls - the appropriate APPROACH or AVOID motor response. In the adaptive organism these innate releasing mechanisms have been replaced with a classifier system. The signals generated by objects in the

---

[4]This change was made to avoid having "unworthy" classifiers involved in the competition for activity. The likelihood of such an event should be investigated more thoroughly. Perhaps using normalized strengths really is not necessary after all.

environment have no *a priori* significance to the organism. Contact with resource or noxious objects is reinforcing in that it triggers learning mechanisms that preserve information about important stimuli and their associated affective codes. The affective codes, in turn, have a releasing effect on the APPROACH and AVOID routines.

More specifically, the organism is designed to learn using the overall organizing construct of "respondent conditioning" [Keller,1954]. Contact with a resource or noxious object, as a primary reinforcer, automatically elicits an internal response - activation of an affect code - in the organism. Under the principle of respondent conditioning, pairing a neutral stimulus - in this case the signals generated by the object - with the eliciting stimulus will eventually cause the previously neutral stimulus to elicit the same response. The signals detected when in contact with an object therefore come to elicit the appropriate affect code even when there is no contact. Under the generalization and discrimination mechanisms of the learning heuristics, the same capability is extended to all signals generated by an object. Since APPROACH and AVOID are activated according to which affect code is designated, the organism eventually learns to make the correct response when an object signal is detected.

This scenario will take place using the classifiers, genetic algorithm, and modifications of strength described so far. Repeated exposure to signals from the environment

will lead to a population of perception classifiers with two specialized subpopulations, one for each kind of object. This occurs automatically as the genetic algorithm is invoked periodically. Reinforcement is not required. At the same time, the affect population is being modified to include classifiers that match the messages generated by the perceptual classifiers. This is a more difficult process since the messages are initially random and get sorted out very slowly into some consistent pattern.'[5] Reinforcement provides information about the appropriate tags for excited classifiers in each population and triggers a feedback mechanism that makes the current message configuration - or "connection" between the two populations - more likely to recur.

In order to demonstrate that the system performs as expected, the organism was placed in the two channel environment shown in Figure 6.6. This environment is rich in stimulation, containing twelve uniformly distributed objects, six of each type. In order to prevent the organism from dying as it acquires experience, it is assumed that the organism has its need for resources taken care of during a "training" interval of fixed length. During this time the only responses the organism can make are CONSUME, ESCAPE, and EXPLORE. After the training interval, the APPROACH and AVOID behaviors have "matured" to the point where they can

---

[5]Since the affect input taxa are also initially random, perception messages get no useful feedback.

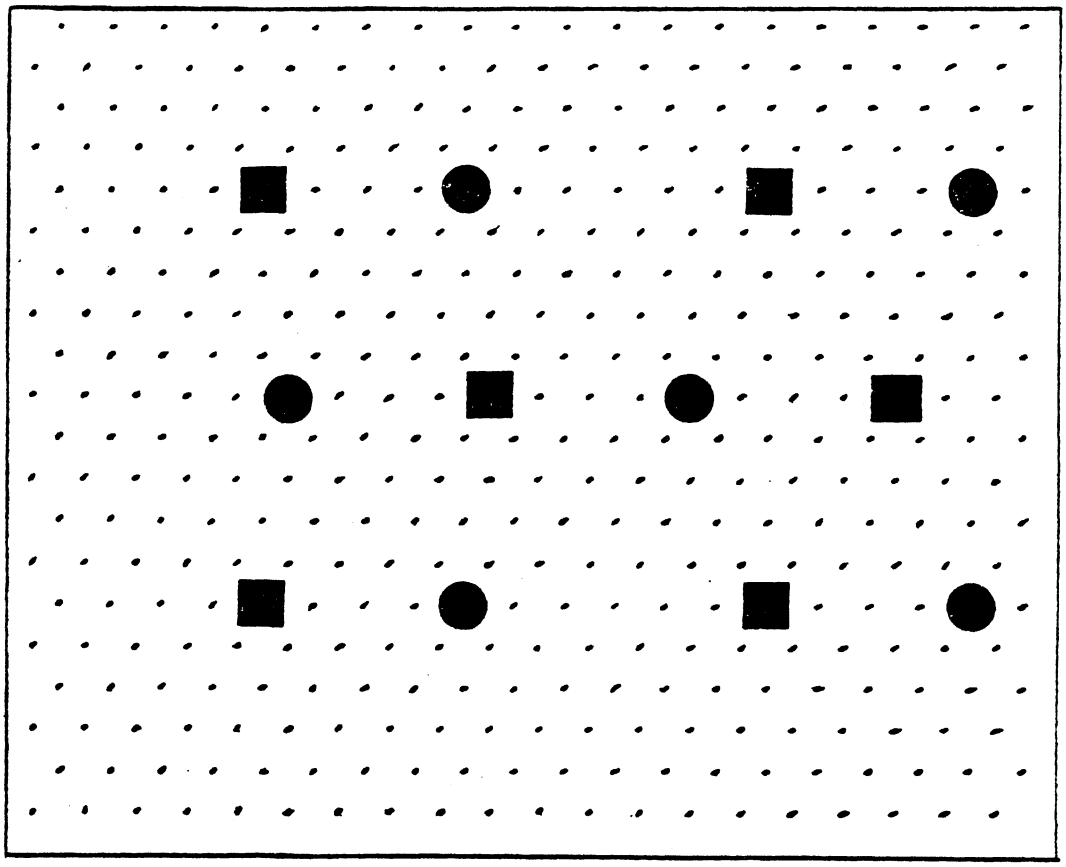

Figure 6.6 The two channel environment used for the "training" simulations. The resource objects [squares] and noxious objects [circles] each has an intensity aura like those shown in Figure 3.5.

be controlled by the activity in the classifier system. The organism must then rely on its own goal-directed processing to obtain needed resources. The computer simulation of this training interval is summarized below:

ORGANISM SIMULATION 5

Environment

The two channel environment depicted in Figure 6.6. Resource objects emit signals with the characteristic 1111111111******. Noxious objects emit signals with the characteristic 0000000000******.

Required Behavior

On contact with a needed resource, CONSUME it. On contact with an aversive object, ESCAPE. Otherwise, EXPLORE.

Mechanism

The organism is in a "protected" state: The deprivation level is reset to zero whenever it reaches 25. The classifier system has 200 perception classifiers and 200 affect classifiers generated initially at random. Each classifier has an initial strength of 320, the value used in the earlier genetic algorithm studies. The initial tagscore value is 8 and the feedback estimate is 72, both corresponding to the expected values given the random initial state. The system innately interprets the tag 000...000 as the "+" code and 111...111 as the "-" code in the affect population. In the perception population, these tags designate classifiers related to food seeking and pain aversion respectively. Other system parameter values: NACTIVE=5, M_MAX=5, NPARENTS=30, NOFFSPRING=12, LEARNRATE=10, and STRENGTH=NPARENTS*320 which is the limited "resource" available in a niche for the purposes of the crowding algorithm.

Results

Five organisms were each simulated for 10,000 execution cycles or trials. The organisms were evaluated according to how often they generated mandates for behavior that, if allowed to become overt, would be inappropriate given the system's motivational state and the stimulus signals available. The average results shown in Figure 6.7 are considerably better than random, but not as good as expected.

The extent to which an organism has adapted to the environment is measured by keeping track of the number of "errors" it makes. An error is recorded whenever the system does not behave like the hard-wired organism would behave under the same circumstances. There are two sources of error: the affective code generated might be incorrect for a given situation; or, faced with simultaneous stimulation from both kinds of objects, the competition in the

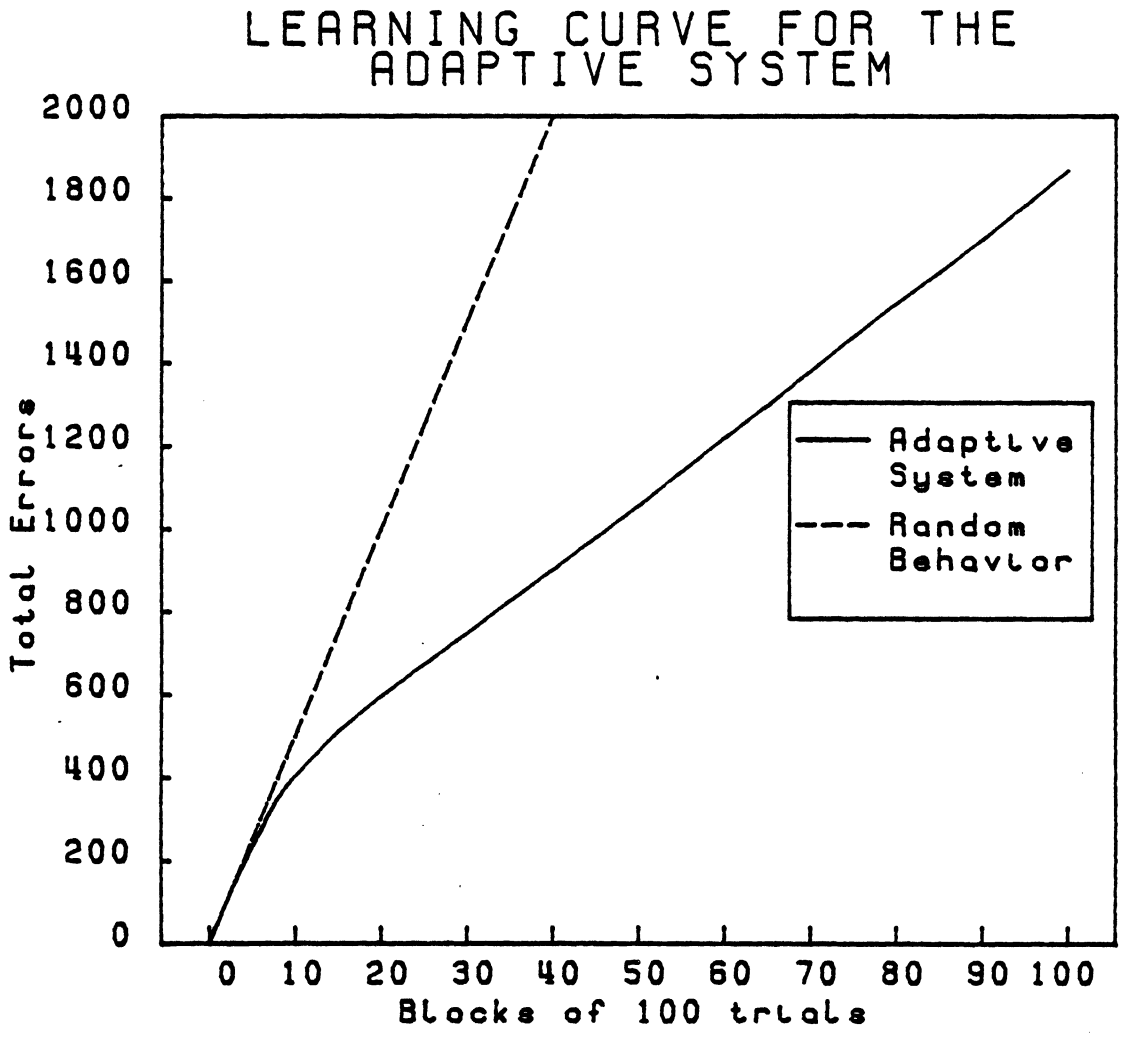

Figure 6.7. The behavior of the adaptive system during the training interval. The learning curve shows the average number of total errors made by the five organisms compared with random behavior.

perception population might be won by classifiers matching the object least relevant to the current system needs - a "failure of attention". Figure 6.7 shows the average cumulative error for five organisms simulated over 10,000 execution cycles or encounters with external stimuli.[16] Though behavior quickly does better than random, the organisms continue to make errors at a relatively steady rate over the interval of observation. This was an unexpected result, especially given the care with which the learning algorithm had been analyzed and tested.

An examination of the results shows that only two of the five organisms were responsible for the non-decreasing error rate (see Figure 6.8). The populations generated in each case exhibit the same symptoms: the affect classifiers all had similar tags and their input taxa matched the messages transmitted by <u>both</u> kinds of perception classifiers. This explains why the error rate failed to go down. After a certain point, all perceptual representations were linked to the same affective code. Interestingly, they were linked to the "-" code which - given that resources are regularly provided for - is most often the correct response. Since all representations for the "+" code were eliminated, the feedback mechanisms were of course rendered useless.

---

[16]Assuming that each type of object gets roughly 5000 of the trials, a LEARNRATE of 10 means the genetic algorithm is applied 500 times to each subpopulation. With 12 offspring generated at a time, this works out to 6000 total samples per pattern class, the interval of observation used in previous experiments.

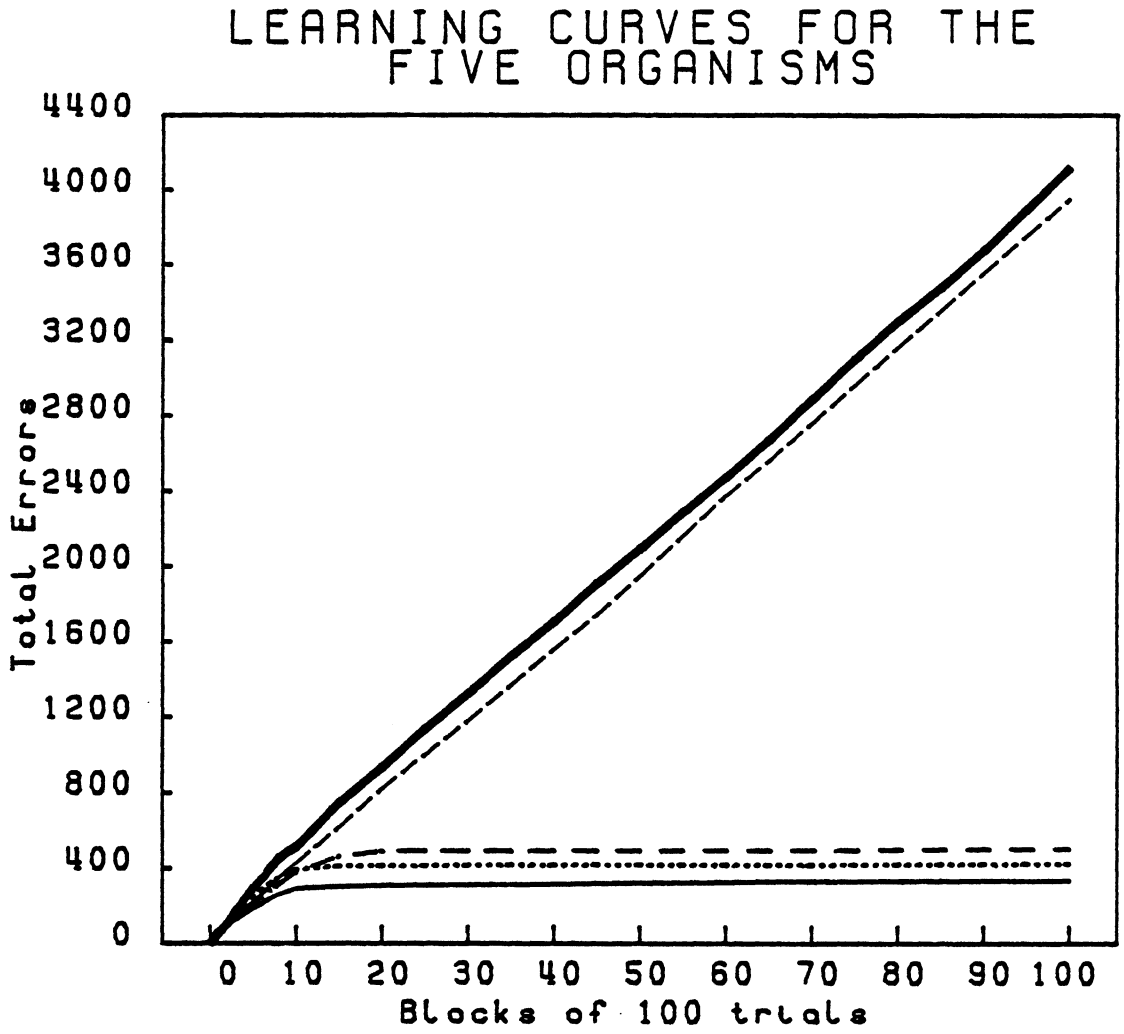

Figure 6.8. Learning curves for each of the five organisms simulated during the training interval. Two of the organisms failed to stop making errors.

Only in the presence of alternatives is the reduced strength due to repeated errors an effective penalty.

Why did the genetic algorithm converge to this particular configuration in these two organisms? Experience with genetic algorithms leads to the suspicion that somehow the selective pressures have been incorrectly specified. In a system with so many parameters and mechanisms interacting to determine the "fitness" of a classifier, it is often difficult to pinpoint the source of such an error. Given the fact that the learning algorithm was independently tested and verified, however, the problem must lie in the way the algorithm is used. Weighing these considerations[17] led eventually to the following analysis. Signals from each object in the simulated environment are available at the object locus and at 18 symmetrically distributed locations around the object (see Figure 3.5). Since the organism is reinforced only while in contact with an object, the average rate for external reinforcement is approximately once every 19 signals.[18] With the LEARNRATE set at 10, the genetic algorithm is very often applied in non-reinforcing situations. What is being learned? In the given environment the organism is most likely to be at a locus

---

[17]The fact that a set of classifiers constitutes a production system program makes analyzing the reasons for successes or failures much easier than is the case with most artificial self-organizing systems.

[18]The actual rate differs because contact with a resource is only reinforcing when the deprivation level exceeds a threshold.

where both kinds of signals are available simultaneously. The activity in the perception population is therefore likely to include both kinds of classifiers. Applying the genetic algorithm to the affect population under such circumstances - especially when this occurs more frequently than the unambiguous reinforcement situations - can generate a strong pressure for taxa matching both kinds of messages! Once the affect population is thus turned into one big non-discriminating relevant set, the crowding mechanism has no way to maintain two separate kinds of tags. The two alternatives are in direct competition and the most frequently occurring "-" tag eventually wins.

If this analysis is correct, increasing the value of LEARNRATE should solve the problem. That would make external reinforcement the primary criterion for what should be learned as was originally intended. The parameter should not be increased too much, however, lest the organism be incapable of taking advantage of the valuable information about patterns available in the stimulus aura surrounding an object. Two values of LEARNRATE are considered: LEARNRATE=20 which is roughly equal to the external reinforcement rate; and LEARNRATE=30, for which the external criterion is clearly the dominant factor. Computer simulations using these values of LEARNRATE are summarized below:

ORGANISM SIMULATION 6

Environment

The two channel environment depicted in

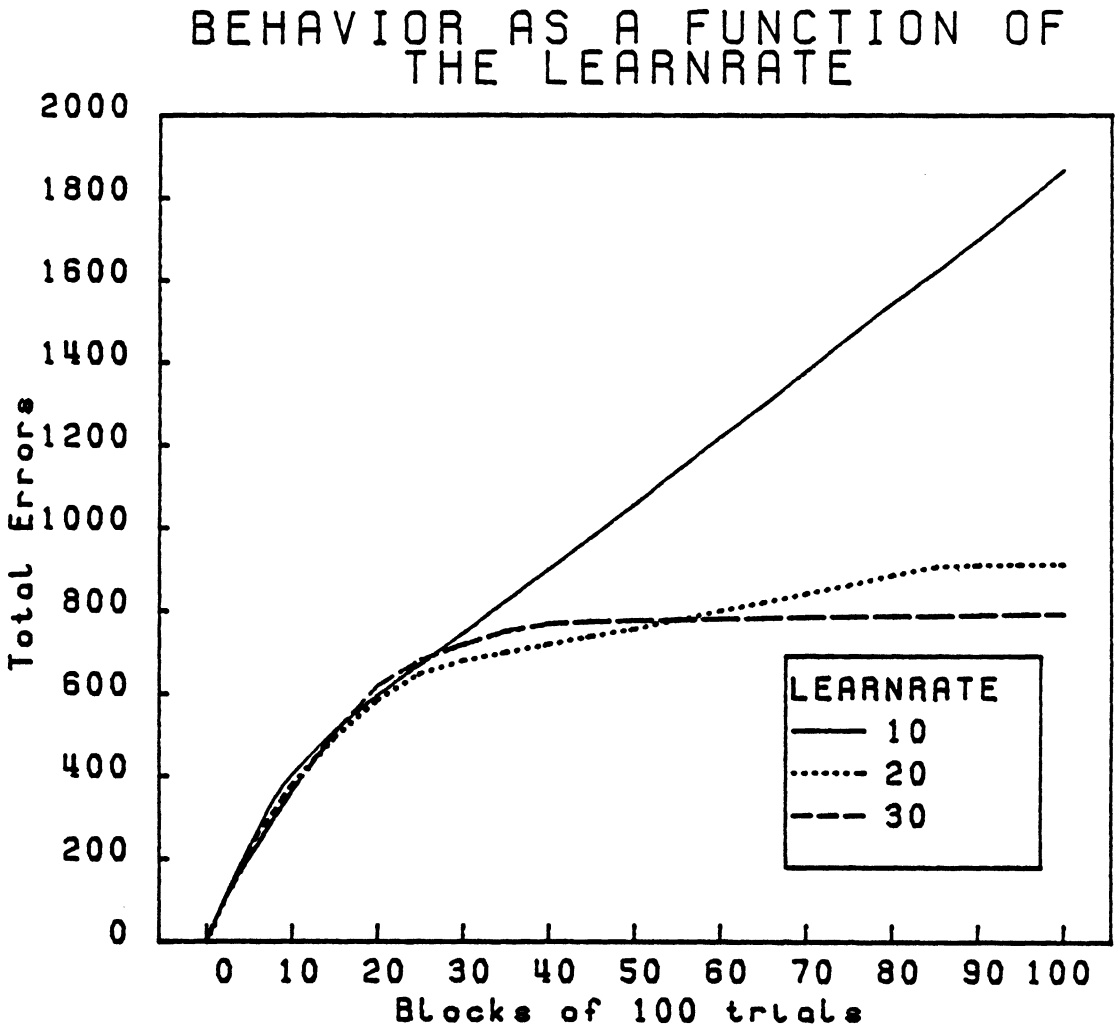

Figure 6.9. The effects of LEARNRATE on the performance of the adaptive system.

Figure 6.6. Resource objects emit signals with the characteristic 1111111111****. Noxious objects emit signals with the characteristic 0000000000******.**

Required Behavior

On contact with a needed resource, CONSUME it. On contact with an aversive object, ESCAPE. Otherwise, EXPLORE.

Mechanism

Same as in simulation 5, except that the two values LEARNRATE=20 and LEARNRATE=30 are used in two separate experiments.

Results

For each value of LEARNRATE, five organisms were simulated for 10,000 trials each. The average results in Figure 6.9 show the anticipated improvement in performance.

As hypothesized, increasing the LEARNRATE resulted in a marked improvement in performance (see Figure 6.9). The value LEARNRATE=30 is clearly superior, giving a quick and steady reduction in the error rate. Over the last 1000 trials of the interval of observation, the organisms had an average error rate of only 0.24%. Two of the organisms made no errors at all during this period. The few errors being made at this point were due to the following two factors: First, one of the organisms represented resource objects at the "wrong" level of generality, using the two classifiers 1#11##111#####1# and 1#11##111#####0# instead of 1111111111######. This is a perfectly reasonable solution given that the crowding mechanism can keep both taxa available. Under the given behavioral criteria, however, this is an unfortunate choice of representation. When two or more signals are available from an object, the hard-wired organism uses the sum of the intensities to determine the

mandate for activating the associated sensory/motor pathway. An organism using two kinds of taxa to represent that object will spread the effects of the signals over two sets of relevant classifiers, rather than accumulate them in one set. The outcome of the competition might therefore not correspond to the hard-wired result. The second more frequent source of error is that the classifier system uses a statistical computation to determine which sensory/motor pathway controls behavior. The closer the total excitation is in the two alternatives, the less predictable the outcome of the competition will be.'' The hard-wired system was of course able to detect very small differences in the various alternatives. Neither of these two sources of "error" seem significant from the standpoint of the overall functioning of the system.

The most meaningful test of how adequately an organism has learned the simulated environment is whether or not it is capable of fending for itself. Accordingly, the adaptive organism that made the most errors in the training simulation was placed in the same environment that the hard-wired organism survived in. The behavior of this organism will be a "worst case" estimate of how well a trained organism can be expected to perform in the given environment. Unless the organism makes the correct decisions in this more challenging environment and keeps the

---

''It is possible to discriminate very small differences in relative excitation using a statistical computation. See Milner's [1958] explanation of acuity in the visual system.

deprivation level within tolerable limits, it will die. The computer simulation is described below:

ORGANISM SIMULATION 7

Environment

The two channel environment depicted in Figure 6.6. Resource objects emit signals with the characteristic 1111111111****. Noxious objects emit signals with the characteristic 0000000000******.**

Required Behavior

On contact with a needed resource, CONSUME it. On contact with an aversive object, ESCAPE. For other stimulus configurations, APPROACH or AVOID as indicated by the affective code derived from the active affect classifiers. When no signals are detected, EXPLORE.

Mechanism

Same as in simulation 6 with LEARNRATE=30.

Results

The organism demonstrated its ability to survive in this environment as the deprivation level was kept below 93 during a 709,214 time step interval of observation. The average deprivation level was 10.87.

The organism was simulated for 709,214 time steps to be sure it had ample opportunity to make a tragic set of errors. The error rate over this interval was only 0.12%. This resulted in an average deprivation level of 10.87, only slightly higher than the 10.36 level of the hard-wired organism. The errors occurred in clusters separated by long periods during which no errors were made. Observations of the organism's behavior showed that, when errors occurred, the classifiers related to pain aversion were beginning to lose their solid foothold in the population. Quite simply, as noxious signals are less frequently encountered, the drop

in reproduction rate gives a decided advantage to classifiers related to food seeking.[20] However, a brief period of several "erroneous" encounters with noxious stimuli promptly restores the balance. A sample of the "program" the organism used to function in the environment is given in Table 6.2. Note that the affect input taxa have very few #'s. This is because only one kind of message is necessary to achieve the desired "connection".

TABLE 6.2

SAMPLE CLASSIFIERS FROM A TRAINED ORGANISM

Function	Classifier		
	Input Taxon	Message Taxon	Tag
"Food"	1111111111######	0110###001001000	0000000000000000
"Pain"	0000000000######	#10#00#00#000#0#	1011111110111011
"+" code	0110111001001000	001##101#1000#01	0000000000000000
"-" code	0100000000000###	0010#101#1000#00	1101101011100101

These results show that the adaptive system acquired useful structures based on experience that enabled it to match the performance of the hard-wired organism. The somewhat arduous design effort has therefore been successful. A key reason for the adaptive behavior of the organism is the non-trivial modifications of structure made

[20]Recall that since new classifiers are continually being generated, every classifier has a finite lifetime.

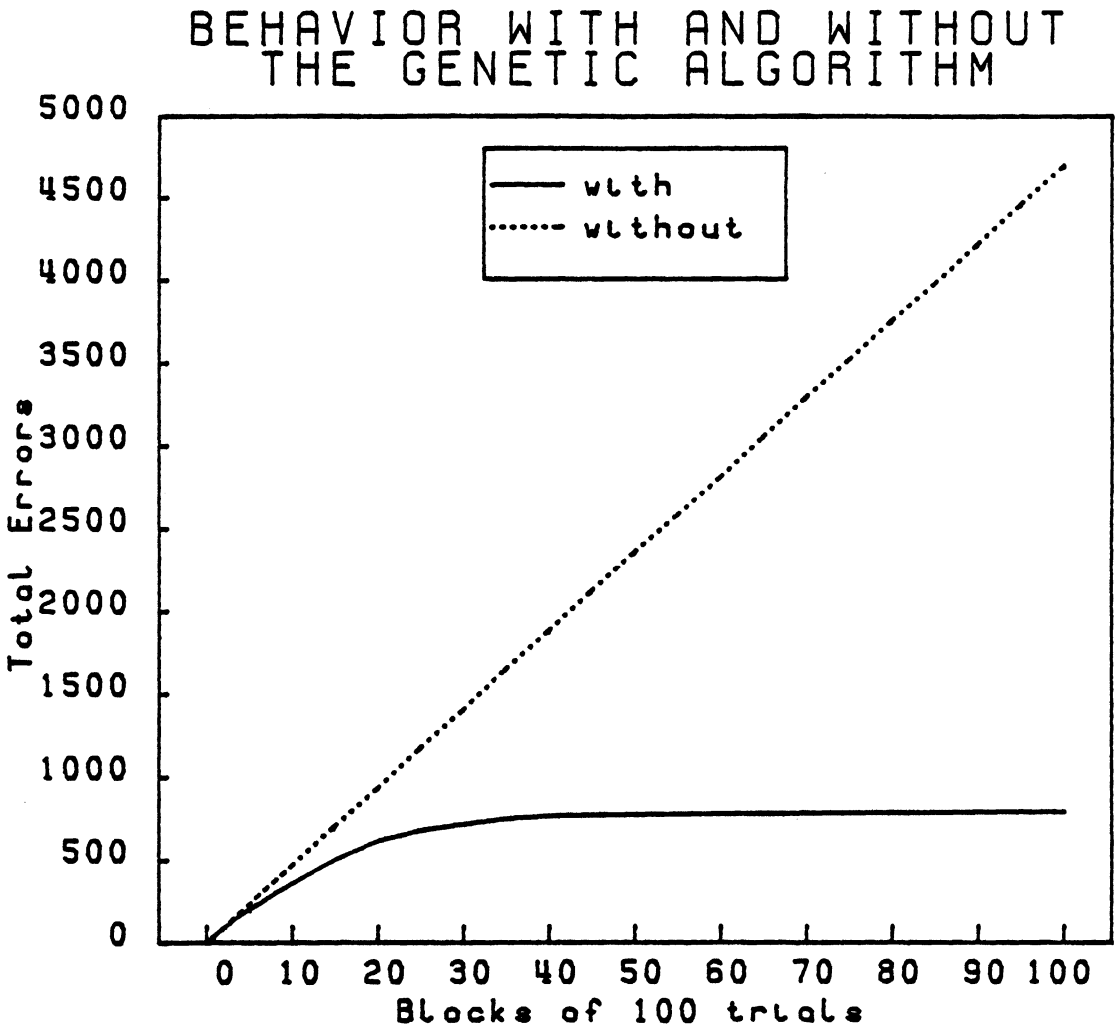

Figure 6.10. Performance of the adaptive system with and without the genetic algorithm. The learning curves represent the average behavior of five organisms.

possible by the genetic algorithm. Figure 6.10 shows the extent to which the genetic algorithm accounts for the learning capabilities of the organism. Adjusting strengths without modifying classifier taxa or tags leads to performance levels only slightly better than random behavior.

The tests of the organism done so far indicate that it can learn to generate behavior that was instinctive for the hard-wired organism. The adaptive system is of course capable of much more than this. If the environment changes in any meaningful way, the hard-wired organism has no means of modifying its behavior to remain viable.[21] The adaptive system generates a useful and flexible internal structure as a result of its experience with the environment. This internal structure, and the inherent learning capabilities of the system, give the organism a meaningful advantage during times of change. One way to demonstrate this advantage is by the transfer of learning from one situation to another.

To demonstrate the effects of prior structure on learning in new situations, two environments were devised. Each environment had appetitive and aversive objects distributed as shown in Figure 6.6. In the first environment, resource objects generate signals with the characteristic *****11111****** and noxious objects generate

---

[21]Adaptive changes can, of course, occur on an evolutionary scale. The concern here is only with changes possible during the lifetime of an individual organism.

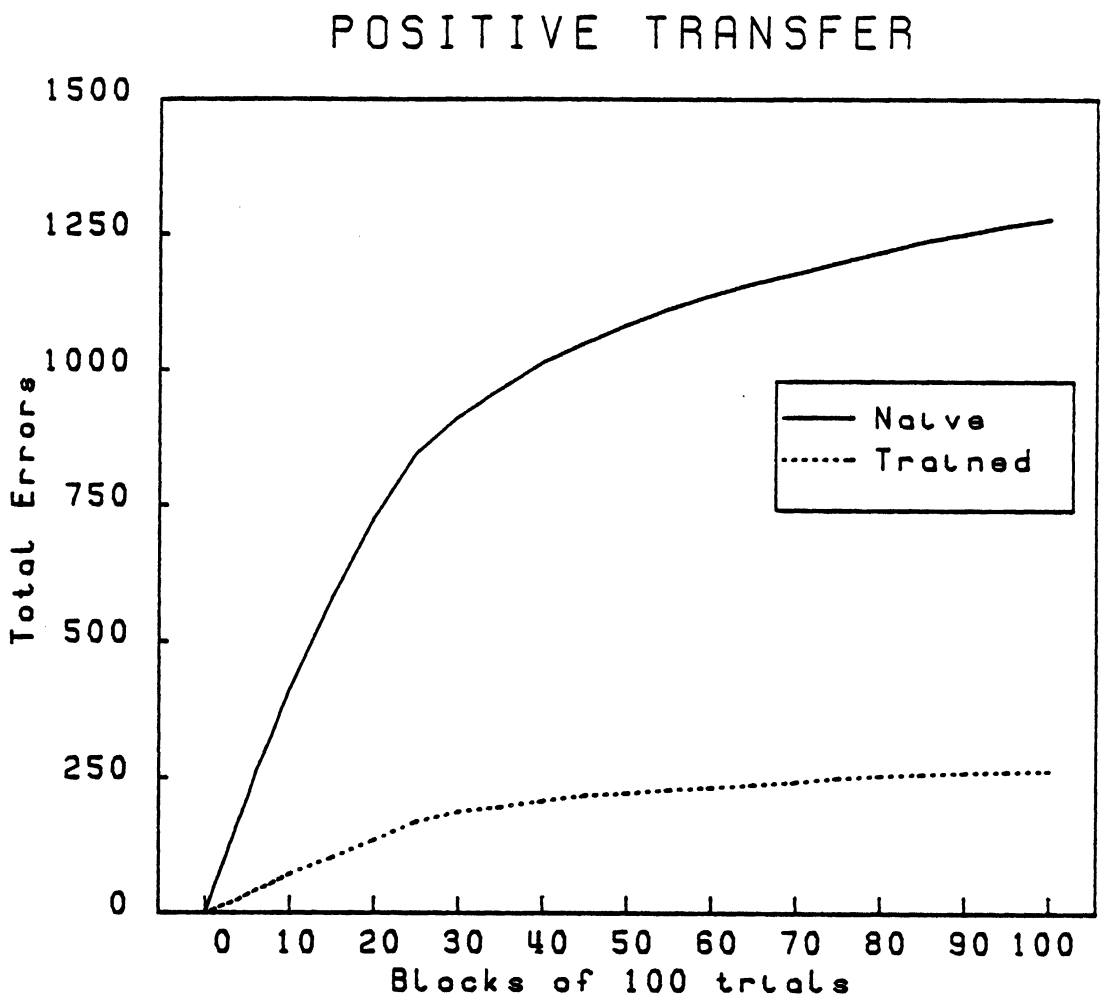

Figure 6.11. Positive effects of previous experience in learning a new environment. Note that the increased size of the pattern class slows down learning in the naive organism as compared with Figure 6.9.

signals with the characteristic *****00000******. This environment is a generalization of the one used in the training simulations in the sense that the signals generated by each object include the original set of signals - those given by 1111111111****** and 0000000000****** - as a proper subset. An organism trained in the original environment has many classifiers that are relevant in the new environment. Just as important, the classifier populations in such an organism contain numerous schemata well suited for constructing new classifiers capable of dealing with the additional signals not previously encountered. It is hypothesized, therefore, that organisms trained in the original environment should exhibit "positive" transfer of their previous experience and learn the new environment more easily than a naive organism with random classifiers. Figure 6.11 shows that this is indeed the case. Organisms trained in the original environment make fewer total errors and learn faster than their naive counterparts.

Prior structure can have detrimental effects on learning as well. The second environment was constructed so that signals have the opposite meaning from the way they were used in the training environment; that is, resource signals are defined by 0000000000****** and noxious signals are defined by 1111111111******. This environment makes the tags and message taxa in a trained organism absolutely wrong. Trained organisms are therefore in the worst possible position to learn the new environment. Figure 6.12

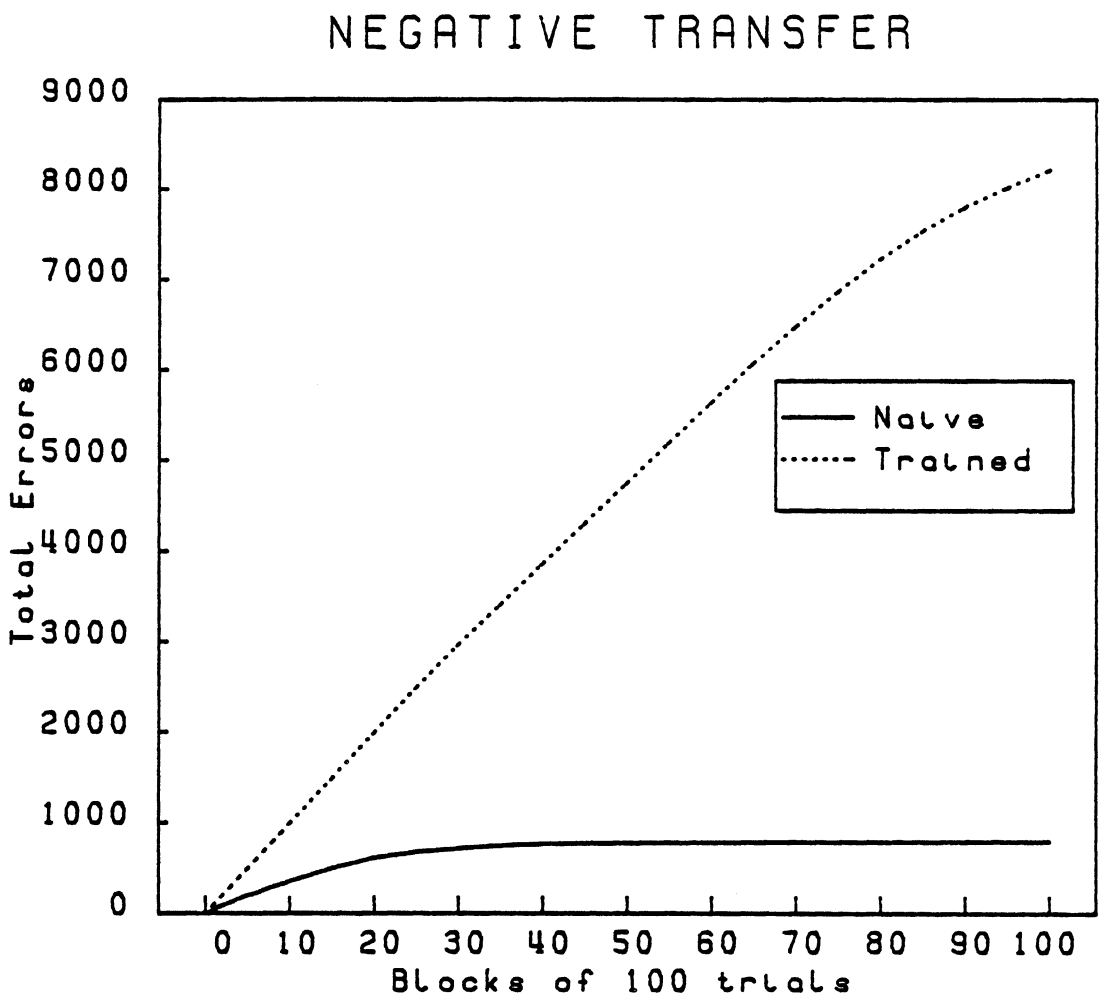

Figure 6.12. Negative effects of previous experience in learning a new environment.

illustrates this marked "negative" transfer effect. The naive organisms do substantially better than those trained in the original environment. Indeed, the latter do not begin to show improvement until the end of the interval of observation.

One final experiment was run to illustrate the way in which the system acquires a structural representation of its environment. Reversal training, in which the reinforcement contingencies of some learning situation are reversed, is a very well studied phenomenon in animal discrimination learning [Mackintosh,1974]. Initially, reversing the reinforcement contingencies has a negative effect on learning as was the case above with the hypothetical organism. If the contingencies are then reversed again, however, and the animal is subsequently trained on a repeated series of reversals, performance can improve "... and each new reversal may be learned with fewer errors than were required to learn the original discrimination. Eventually, indeed, each new reversal may be learned with only a single error" [Mackintosh,1974, p.608]. This phenomenon has been demonstrated in the learning of rats and a few other mammals. It has also been shown that goldfish and mouthbreeders are not capable of reducing errors over a series of reversals. There is still considerable debate about exactly what mechanisms are responsible for serial reversal learning.

It is hypothesized that our adaptive organism should

demonstrate some improvement in performance over a series of reversals. Very briefly, the initial errors should be high due to the negative transfer effects noted above. As the system experiences more reversals, however, it should eventually develop two sets of representations - one for each reinforcement contingency. At this point merely readjusting strength levels is all that is necessary to assure correct behavior for a given contingency; and, this can be done quickly after only a few trials. If this analysis is correct, it might shed some light on the mechanisms at work in serial learning behavior in animals. To test this hypothesis, five naive organisms were subjected to a series of eight reversals. On each reversal an organism was trained until it achieved an arbitrarily chosen criterion of 25 correct responses in a row. The resulting performance, shown in Figure 6.13, was inconclusive. One of the organisms (see Figure 6.14) definitely exhibited the anticipated improvement over the series of reversals. For the most part, however, improvement was either inconsistent or non-existent.

An analysis of the populations generated by each organism revealed the reason for these results and points out the major weakness of the current model. Every "species" of classifier must reproduce in order to hold on to its space in the population. By the time the organism had reached criterion on a given reversal, the classifiers learned during the previous reversal were likely to have

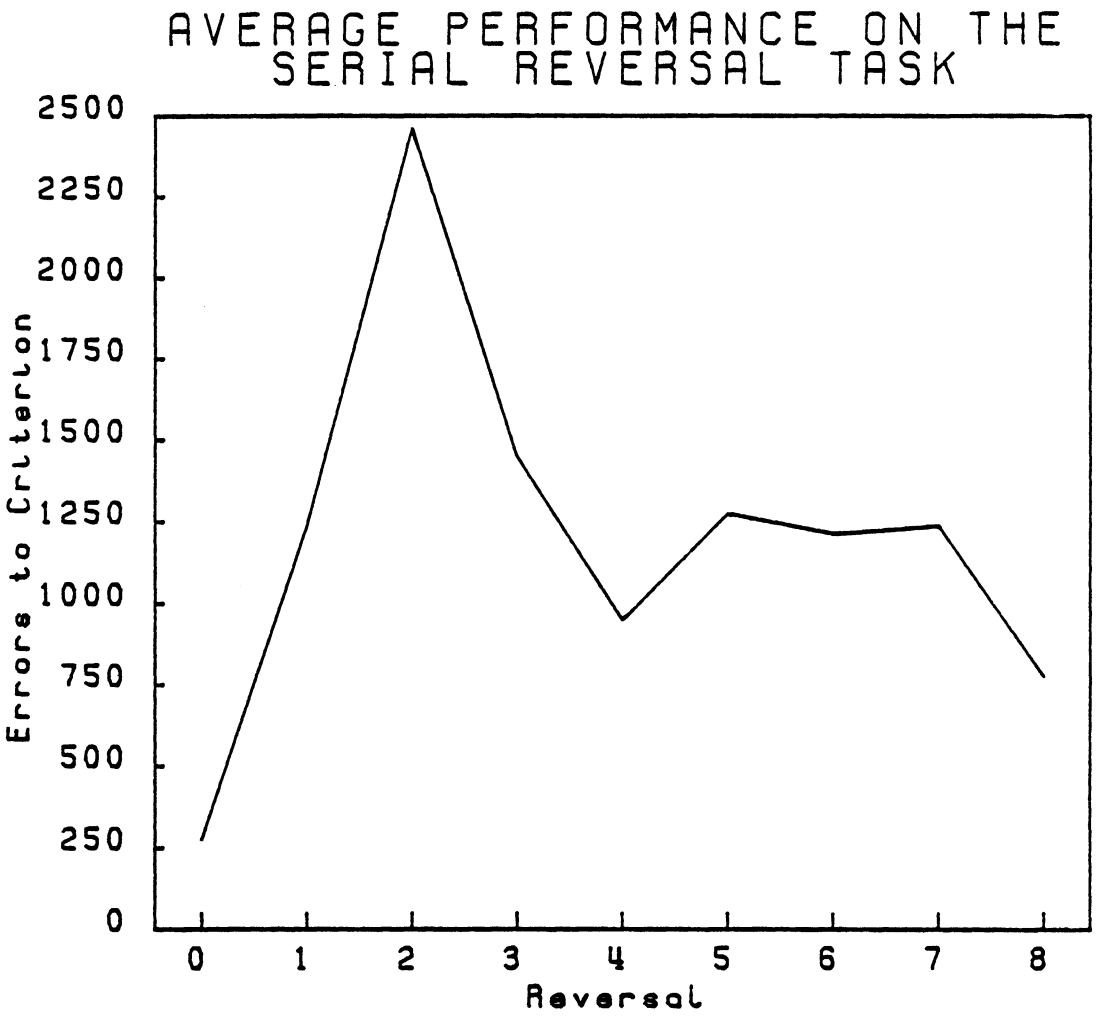

Figure 6.13. Average results of five organisms in the serial reversal experiment.

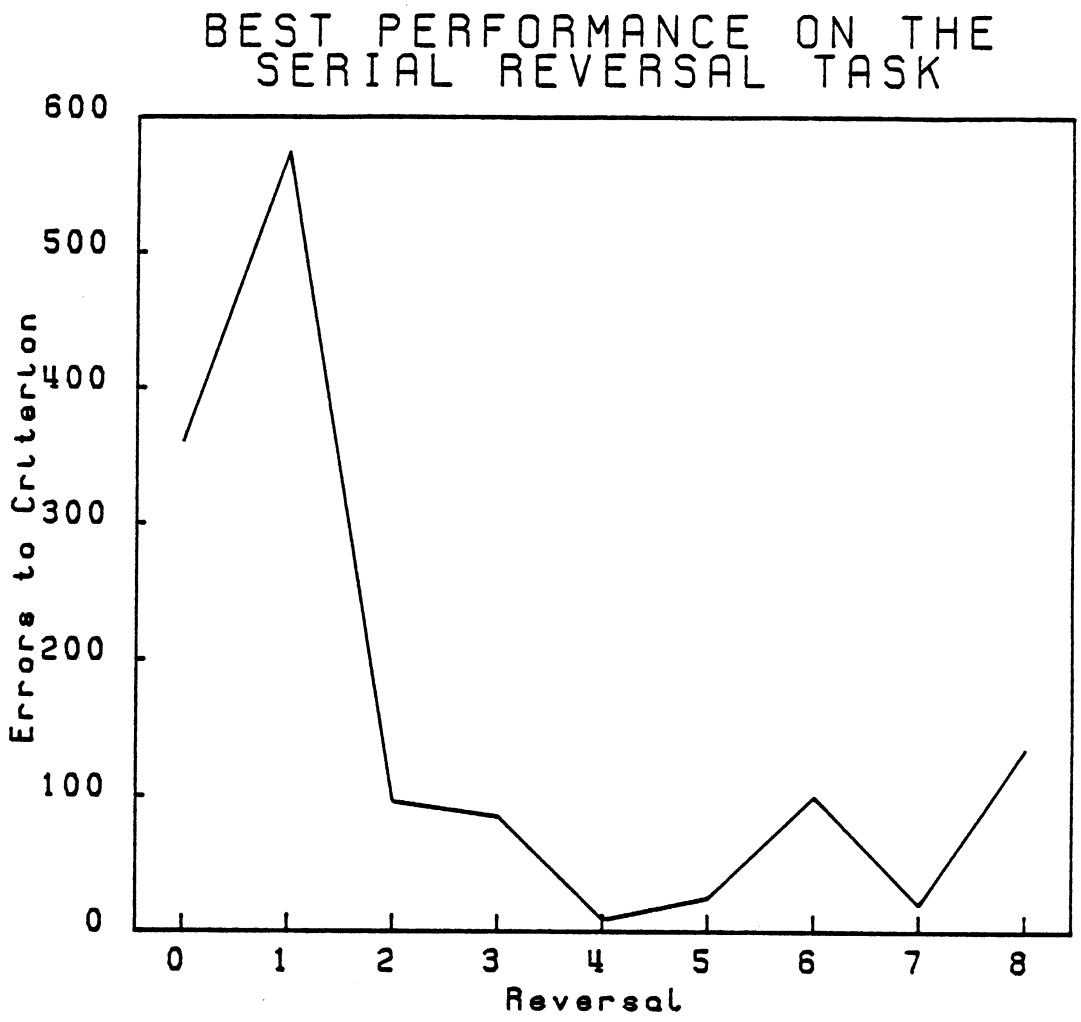

Figure 6.14. The best performance observed in the serial reversal learning experiment.

been deleted - that is, become "extinct" due to the drastic change in the environment. This made it difficult, if not impossible, for the system to improve over the series of reversals. There was no guarantee that information could be preserved from one reversal to the next. While the extinction of species is a reasonable thing to happen under sudden environmental change in a biological population, such extinction in the population of classifiers is, more often than not, undesirable. The emphasis on recency and short term memory in the system is too great. The system must be capable of storing and retaining information even if it is not likely to be accessed again for a while.[22]

Clearly, introducing a longer term memory into the system requires that some classifiers be given a special status[23] in which they are not deleted and reproduced often or even at all. This must be done with care, since the criteria used for transferring a classifier from short to long term memory will have a decisive impact on the system's ability to effectively manage its limited storage space. A useful analogy for how to do this can be found in Hebb's

---

[22]The problem can be made considerable less severe, though not solved, by increasing the population size and thereby lengthening the expected lifetime of each classifier. The 28K word space limitation of the computer on which these simulations were run, however, made increasing the number of classifiers impossible.

[23]It might be necessary to have a graded notion of memory in which classifiers can become increasingly "long term" over time. A classifier might therefore be in any one of several possible states determining its longevity in the population.

[1949] discussion of the growth of cell assemblies. In its early stages an assembly is large, diffuse, and weakly interconnected. Repeated learning leads eventually to a compact, strongly connected set of elements. Some elements that were part of the original assembly are "fractionated" during the learning process and become available to be used in other assemblies. In an analogous way, a set of relevant classifiers that is well learned - as evidenced by the size of the set, the average match score, the amount of diversity, etc. - can be compactly summarized with just a few classifiers. A process that makes certain classifiers in such a set progressively more resistant to change and the rest of the set more vulnerable will lead to the desired economy. It should be emphasized that any such process must be carefully thought out and parsimoniously integrated with the other mechanisms in the system. The matter is sufficiently complex to warrant an extensive research effort and will not be considered further here.

CHAPTER VII

SUMMARY AND CONCLUSIONS

> I am coming more and more to the conviction that the rudiments of every human behavioral mechanism will be found far down in the evolutionary scale and also represented even in primitive activities of the nervous system. [Lashley,1951, p. 134]

This thesis began with the premise that intelligent behavior can be profitably viewed as an adaptation to a task environment. From this standpoint, understanding intelligent behavior requires an understanding of the ways an environment can pose problems in information processing for some potential information processing system; and, an understanding of how a system can be organized to solve those problems and adapt to its environment in non-trivial ways. The issues raised here are dealt with routinely and successfully by animate systems in their ordinary commerce with the real world. Accordingly, it was decided that studying the behavior of a simple hypothetical organism - one that must find resources and avoid noxious stimuli - in a carefully chosen simulated environment might illuminate some of the basic informational and structural prerequisites for intelligent behavior.

The research proceeded in three stages. First, the information available in natural environments was analyzed in terms of its implications for the functioning of potential organisms. A class of simulated environments was then designed that embodies many of the important functional properties characteristic of natural environments. Second, an organism relying on instinctive mechanisms was constructed that was capable of surviving in several of these environments. This instinctive organism was developed incrementally, with each information processing mechanism added only as required to meet the demands of a series of increasingly sophisticated environments. The organism model that emerged from this process proved capable of locating resources and avoiding noxious stimuli in these environments by generating temporal sequences of actions. Third, the task for the organism was made more difficult with the introduction of a large measure of uncertainty into the information available in the environment. In order to meet this final challenge, the instinctive organism was redesigned to be an adaptive system capable of discovering the significance of various stimuli based on experience. This required, among other things, a powerful learning heuristic to continuously modify the components of the system so that useful structures are maintained while potentially better ones are generated and evaluated. Simulation results show that, from an undifferentiated initial state, the system organizes itself based on

experience until the appropriate behaviors are discovered.

Statistical Information Processing

The information processing challenges confronting the adaptive organism are summarized by the following characteristics of the organism/environment interface:

1) The organism is permitted access to only a limited sample of all the information potentially available at any given moment.
2) The organism has limited resources for processing information. These resources must be managed so as to make best use of the organism's information processing capabilities.
3) There is a meaningful and functional distinction between proximal cues and their distal source.
4) The cues available from any given object are highly diverse and variable.
5) There are structural regularities or *patterns* underlying object/cue relationships that have potential significance to the functioning of the organism.

These characteristics define an environment for the adaptive system that is complex and uncertain.

In order to function under these conditions, the system was endowed with a "perceptual" component enabling it to sift through the variants in the proximal cues and discern the distal source of stimulation. The perceptually driven

processing used here is in sharp contrast to the symbol manipulation strategies common to most artificial intelligence systems. Objects in an uncertain environment are not simply "there" as clean and definite clusters of information ready to be assigned symbolic representations and processed abstractly. Psychologists have long recognized that perception is an achievement. A system designed to handle this somewhat messy aspect of reality is likely to be more concerned with basic processes that enable it to function at some acceptable level than with general purpose strategies for manipulating symbols.

The organism model was implemented as a *classifier system* [Holland,1976]. Classifier systems are basically rule-based systems in which several rules or *classifiers* can be active at once, there are no fixed priorities determining the order in which rules can be activated, and the syntax of each rule is simple enough to make powerful learning heuristics applicable. A classifier is a pattern sensitive element that receives, processes, and transmits simple messages. Associated with each classifier is a *strength* indicating its effectiveness in processing messages and a *tag* identifying its function. As a rule-based system, the system presented here is notable in that it is open to substantial modification in ways other than the usual isolated changes or addition of new rules. The system's collection of classifiers is continually evaluated and revised using a carefully modified version of the *genetic*

algorithm [Holland,1975]. This is a powerful learning heuristic that is provably more efficient than the simple insertions and modifications characterizing most other learning strategies. The system is also differs from other rule-based systems in that it processes information in a statistical manner. The classifiers that become active and control the behavior of the system are determined by a probabilistic computation involving facilitation, competition, proximity, and strength. Informational significance is attributed to the orderly behavior of groups of classifiers, rather than to the potentially unreliable activity of any single classifier.

The adaptive system was tested in the simulated environment and demonstrated that it could quickly acquire the knowledge necessary to function effectively. Further experiments showed that the system is capable of discriminating a large class of schematic patterns; and, that prior learning experiences transfer to novel situations. The system also proved extremely efficient to run. A typical experiment lasting 10,000 time steps required only about three hours of computer time on a PDP 11/34. This allowed the behavior of the system to be observed in "real time" on a graphics screen depicting the organism and its environment.

There has been a tradition of resistance in artificial intelligence to statistical computations and representations, especially those related to learning.

Minsky [1963], for example, expresses doubts about whether

> ... such "incremental" or "statistical" learning schemes should play a central role in [artificial intelligence] models.... The more intelligent one is, the more often he should be able learn from an experience something rather definite; e.g., to reject or accept a hypothesis, or to change a goal.... The heart of problem-solving is always, we think, the combinatorial part that gives rise to searches, and we should usually be able to regard the complexities caused by "noise" as mere annoyances, however irritating they may be. (p. 428)

This thesis argues that complex and uncertain information has to be dealt with directly and cannot always simply be filtered out or ignored as insignificant "noise". Brunswik [1956] has noted that a system can function effectively in an uncertain environment by making use of a diverse set of generic representations. The classifier system developed in this thesis is a simple scheme for implementing such a strategy. Entities in the environment are represented by groups of classifiers. The statistical structure of this representation is embedded in the distribution of strength and the variability among classifiers in a group. Because the members of a group compete to become active, a representation is processed only in terms of those aspects most likely to be relevant in a given context. The performance of the system on non-trivial tasks demonstrates that this statistical approach can be an effective way to function in a complex and uncertain task domain.

More Sophisticated Systems

Given the success of the adaptive system in the

simulated environment, it is reasonable to ask if this overall framework lends itself to studying more sophisticated modes of intelligent behavior. In considering how the system might be extended to deal with more complex tasks, we must begin by understanding what additional challenges an environment might present to the system. The properties and mechanisms enabling the system to meet those challenges can then be characterized.

In the simulated environment every object has some innate motivational significance to the adaptive system. This eases the processing burden on the organism because the relevance of every stimulus is easily derived. Moreover, objects are distributed in the environment in a basically uniform fashion. This makes random exploration an adequate strategy for searching the terrain. How is the situation changed if these two aspects of the simulated environment are complicated by the introduction of neutral objects and a non-uniform distribution of resources and noxious objects?

One can imagine circumstances in which a non-uniform distribution of resources would be irrelevant to the behavior of an organism. For example, the clumps of grass in a field might have a non-uniform spatial distribution, but that would make little difference to a moderately sized grazing animal that can get to any clump easily. In the absence of such special circumstances, however, an uneven distribution of resources - especially given large distances to be covered - requires that the organism take a more

spatially oriented approach to the environment. In other words, in order for the organism to find resources in a reasonable amount of time, or at all, the organism must be able to determine if it is headed in the "right" direction. Similarly, if some directions lead to danger and/or noxious objects, it is important to know when a direction is "wrong" as well.

Neutral stimuli can simplify the navigation task for an organism by serving as landmarks to guide the organism from one place to the next; or, by providing directional cues enabling the organism to maintain some general heading. However, if the environment is one in which getting lost is to be avoided - because of predators, a need to return to some home base, etc. - these straightforward aids to navigation might be insufficient. For example, it is easy for an organism blindly steered by landmarks to get lost if a landmark is missed because of some change in the landmark's appearance, an unanticipated detour, or a momentary lapse in attention. Similarly, maintaining a general heading is not always an adequate strategy for locating objects or places. Consider an environment which has obstacles that must be circumnavigated, resources that are far away, or locations that are accessible only from certain directions. After the system has found its way around some obstacle, the orientation to its destination is likely to have changed. When the distance from start to goal is large, a small error in the original heading can

lead to locations far removed from the desired one. In an environment where getting in the general area of a target location is not good enough, an adaptive system must be able to predict and correct for the outcome of its movements. It must be able to determine where it is as well as where it is going.

An organism possessing the detailed knowledge required for this kind of spatial behavior - knowledge about objects, places, the relations between them, and the likely outcome of potential actions - has the potential to use this knowledge in other, non-spatial information processing activities.

> If the organism carries a 'small scale model' of external reality and of its own actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilize the knowledge of past events in dealing with the present and future, and in every way to react in a much fuller, safer, and more competent manner to the emergencies which face it. [Craik,1943, p. 61]

The kind of "model" being described here is usually referred to as a *cognitive map* [Tolman,1949].

This brief discussion makes it clear that, by manipulating just a few parameters, environments can be designed within the current simulation framework that demand sophisticated "cognitive" processing on the part of the organism. What kind of changes must be made to the organism so that it can meet the challenges of such an environment? Kaplan [1973] has identified three critical properties a cognitive map must have: generic representations,

motivational coding, and network structure. The hypothetical organism already makes use of generic representations and motivational codes. The advantages of these aspects of structure have been discussed at length. Network structure refers to the fact that representations or map "nodes" are not related to each other in a simple linear or even hierarchical fashion. The associative relations of any given node indicate the situations that precede it, follow it, are implied by it, etc. Because representations are generic, the set of all possible associations is rich and complex. The assumption is that networks are the most economical way to store this huge amount of information. Though sequences are characteristic of experience (one event follows another), tree-like structures are characteristic of how memory is used to trace the possible associates from any given point. Since any representation is a potential starting point for generating such a tree, the most reasonable storage strategy is to embed all the potential sequences and trees in a network.[1]

The potential for a rich associative structure exists in a classifier system but the hypothetical organism is designed so that only one associative connection is needed for each representation. This is clearly the aspect of the model that must be elaborated before the organism can build a more sophisticated model of its environment. The

---

[1]See Kaplan and Kaplan [1982] for a comprehensive treatment of the issues related to cognitive maps.

machinery is already in place. Every classifier generates messages that are "processed" versions of incoming signals. Each classifiers is therefore capable of sending several different messages and so be part of many associative sequences.

One obvious first approximation to a model having extensive associations is a system that is capable of only a simple one-step prediction or "lookahead". It is instructive to briefly consider how the current hypothetical organism might be modified to achieve such a level of functioning. First, the system would have to have the ability to activate representations of objects "off-line" in the absence of sensory input and without necessarily generating overt behavior. This allows the organism to activate each of the possible "next" representations, evaluate them, and decide on the most appropriate course of action. It would be dangerous if this off-line processing seriously impaired the system's ability to respond to the current stimulus configuration. In particular, if the activity in the perceptual components is tied up evaluating possible future states, the organism might fail to attend to some important input signal. This suggests that the off-line processing should take place somewhere further removed from the sensory interface.[2] One way to achieve this is to provide the organism with a set of "associative" classifiers

---

[2]It is also helpful if the internal time step for the system is faster than the rate of change for significant events in the environment.

to serve as the model-building components of the system. Representations using associative classifiers must be developed for the objects in the environment, their affective codes (if any), and their sequential relationships based on the organism's experience. This can be accomplished using the existing pattern recognition machinery if the "input" signals for the associative classifiers are the tags from active classifiers in other parts of the system. If we specify further that associative classifiers can send messages to each other, the machinery for sequential predictions is in place. Note that it is also possible for the tags of active associative classifiers to be used as input for other associative classifiers. This is the beginning of a very sophisticated procedure, allowing the system to recognize its own internal states and build hierarchical models even further removed from the original sensory stimulation. Such a model could be the basis for "top-down" processing and control in the system.

This brief discussion leaves many questions about specific implementation details unanswered. It does serve, nevertheless, to point out that the framework developed here is broad enough to investigate many aspects of "cognitive" functioning. The same general principles apply if the "environment" for the adaptive system is a complex data base that cannot be searched exhaustively or the state space for some difficult problem solving task. In each case, searching the domain is much easier with a "map" that points

the way to promising regions.

Structuring a Task Environment

Very often in the design of an artificial system, little time is spent deciding how to present information from the environment to the system. It is usually assumed that the major design effort should be directed towards internal representations and processes; and, that external information can be easily converted into a useable form. One of the points emphasized in this thesis is that the informational aspects of the environment have a direct bearing on what kinds of information processing mechanisms can be effective, particularly with regard to learning.

Analyzing the way animate systems interact with natural environments leads to some very basic, though often ignored, observations. The stimulus cues available at the system/environment interface represent all the information an adaptive system can directly extract from the environment. Natural environments are complex in the sense that the amount and variety of cues and their interrelationships is overwhelmingly large. Any viable system must therefore be discriminating and selective about what information it extracts; indeed, information reduction must be a fundamental and pervasive aspect of the system's processing strategies. Natural environments are uncertain in the sense that proximal cues are not always perfectly correlated with their distal source. Given the complexity of natural

environments, it is rarely cost-effective for an organism to strive for unerring accuracy in discerning a distal object. Faced with the dilemma of needing to know a great deal yet having to act quickly, natural systems place an emphasis on speed. Organisms compensate for "sub-optimal" accuracy with a capability for making "good guesses" about a distal stimulus based on previous experience and by avoiding drastic errors; maintaining, in other words, a reasonable correspondence with the environment. This strategy is possible because the proximal cues in a natural environment have structure that is indicative of their distal source.

Natural object categories - though they have fuzzy or ill-defined boundaries - seem to have a basic level of organization at which prototypical structural patterns can be discerned. These patterns are an economical and reasonably accurate summary of the information required for classification of and interaction with members of the category. By being sensitive to patterns, an organism can generalize over the variants in proximal stimulation to "focus" on and respond to the distal source. This helps reduce the complexity and uncertainty of the environment to more manageable proportions.

What do these insights imply about how information should be structured in more artificial domains? Object information in most artificial task domains is provided in the form of abstract symbols and symbolic expressions. Symbols are usually defined in a rather general way as

> ... physical patterns that can occur as components of another type of entity called an expression (or symbol structure). Thus a symbol structure is composed of a number of instances (or tokens) of symbols related in some physical way (such as one token being next to another). [Newell and Simon,1976, p. 116]

This definition clearly acknowledges the centrality of the notion of pattern with respect to symbols. In practice, however, the pattern structure of symbols and expressions is most often not correlated with the structure of the entities being designated.

Choosing an arbitrary set of symbols to designate the entities in an environment can be a mistake. The structure of the information in a task domain is a crucial factor affecting the complexity of any system having to interact with it. If the symbolic information available does not embody the functionally meaningful structures in the environment, a system will require sophisticated interpretive processes to figure out which symbols are relevant, how they are related to each other, etc. The enormous computational burden of such processing in a complex environment seems to be at the root of many doubts about the ability of artificial systems to function in realistic task domains [Dreyfus,1972]. What is being suggested here is that the complexity of the mechanisms that interpret environmental information can be substantially reduced by paying more attention to the representation of the environment. When "... the likeness between sign and significate is rich and systematic, ... it can be exploited

in using the symbolic representation as an aid to intelligent thinking about the thing symbolized" [Boden,1977, p. 15]. A systematic relationship between the structure of symbolic expressions and the structure of the objects they designate can give an information processing system a crucial advantage in coping with a complex and uncertain task domain. In particular, if the patterns in the environment give a useful summary of the important functional categories, then pattern-directed processing is an effective way to generate appropriate behavior.

Implications for Artificial Intelligence

Learning systems have been a research topic in artificial intelligence from the beginning. Early work placed considerable emphasis on self-organizing systems like nerve nets and perceptrons. It was hoped that by providing these systems with the proper training experiences, they might discover and utilize the knowledge required for some designated behavior. It proved to be very difficult, however, to construct a system that was capable of learning to perform any interesting task. These efforts were therefore largely abandoned in favor of systems endowed with huge amounts of domain-specific knowledge and sophisticated heuristics for manipulating symbolic knowledge representations.

There has recently been a renewed interest in learning systems as an important aspect of artificial intelligence

research.

> Over the past five years there have been many signs of a revival in the learning enterprise, but with a viewpoint quite different from the original nerve net or simple reinforcement paradigms.... What is characteristic of the new generation of learning programs is that most of them are basically problem-solving programs, capable of undertaking heuristic search with a reasonable armory of methods like generate-and-test and means-ends analysis. They do not start out in anything like the barebones fashion of the earlier generation of self-organizing systems. [Simon,1981, p. 20]

These recent learning programs are plagued by a different kind of problem. Given that they rely on having access to large amounts of detailed knowledge and several potentially useful "knowledge sources", there is a need for some way to efficiently acquire this knowledge and make building these systems more practical [Hayes-Roth,1976]. Indeed, building and maintaining each system is such a costly and specialized endeavor that comparative studies of alternative techniques, control structures, etc. are extremely difficult.

The adaptive system presented in this thesis is best described as a hybrid. Because it is a rule-based system it can be extensively pre-programmed with whatever knowledge is deemed appropriate in a given task domain. Because the system can acquire knowledge in non-trivial ways based on its experience, all of the information does not have to be provided in advance. The initial "knowledge base" can serve as a stable framework around which the system can expand and reorganize its knowledge of the environment. This, of course, substantially reduces the costs associated with

knowledge acquisition. Consequently, many versions of such a system can be tested and compared in the same task domain to test the effectiveness of alternative mechanisms, heuristics, organizations, etc. Similar comparisons become possible across task domains. In this way, a deeper and more systematic understanding of information processing strategies can be achieved.

Moreover, the system presented here is a model of how activity in a collection of simple computational elements - operating in parallel, activated stochastically, interacting cooperatively and/or competitively - can be orchestrated to produce reliable behavior in a challenging environment. From this perspective, the model touches on several issues related to information processing in cognitive systems: the generic representation of objects, motivational control of behavior, primitive modes of attention, statistical computations, management of limited processing resources, retention and use of affect codes, and non-trivial mechanisms for learning. These issues have been addressed in a way that is computationally feasible and that allows for rigorous testing.

Finally, it should be emphasized that the system and principles developed in this thesis are only a first step in coming to understand information processing from a "functionalist" point of view.

> One characteristic of these results is that they ... lie at a relatively low level in the overall canvas of intellectual functions, a level often dismissed with contempt by those who purport to

> study "higher, more central" problems of intelligence. Our reply to such criticism is that low-level problems probably do represent the easier kind, but that is precisely the reason for studying them first. When we have solved a few more, the questions that arise in studying the deeper ones will be clearer to us. [Marr,1977, p. 40]

The successful behavior of the model, together with the breadth of the material covered in the design process, leads one to conclude that the methodology and general principles presented here point to a very promising avenue of research. By focusing on a succession of environmental challenges, each requiring some increment in information processing sophistication, the design of complex information processing systems becomes more manageable; and, at the same time, more informative about fundamental issues.

APPENDICES

APPENDIX A

The Reproductive Plan G0

Genetic algorithms are basically adaptive search strategies. The successive populations of individual strings constitute a search trajectory thru the space *A* of all possible strings. The direction of this trajectory is biased toward those regions in *A* containing individuals of high average fitness. The trajectory, in other words, is generated using the simple inference heuristic that the best individuals will be located in the highest performing regions. Using the robust definition of regions - or, *schemata* - given by Holland [1975], it can be shown that this heuristic fails only for very bizarre kinds of fitness functions [Bethke,1981].

DeJong's [1975] work is, to date, the most extensive study of the performance of genetic algorithms. Using the well-defined problem domain of function optimization, DeJong compared the performance of various implementations of the algorithm with each other and with standard optimization techniques. In order to evaluate the efficiency and robustness of each alternative, a set of five test functions, F1 - F5, was used as a representative task

environment. These functions posed various challenges to the candidate function optimizers: F2 was non-convex, F3 was discontinuous, F4 was high-dimensional and noisy, and F5 was multimodal. F1 was an "easy" function having none of these properties. Two performance measures were used to evaluate the optimizers as adaptive plans: on-line performance and off-line performance. On-line performance of an optimizer on a function f in the task environment E is defined as

$$x(f,t) = (1/T) * \sum_{t=1}^{T} f(s(t))$$

where $s(t)$ is the t^{th} point evaluated by the optimizer[1] and T is the interval of observation. This measure emphasizes the initial and interim performance of an optimizer as it progresses toward the optimum. Such performance is important in situations where the points generated by an adaptive plan are used "on-line" in some real time system. Off-line performance is defined as

$$x'(f,t) = (1/T) * \sum_{t=1}^{T} f'(s(t))$$

$$\text{where } f'(s,t) = \min_{1 \le y \le t} f(s(y))$$

This is a more traditional criterion for an optimizer, measuring only the rate of progress toward the optimum. The corresponding measures for the whole environment E are simply the average of the measure on each function.

Given this framework, DeJong shows that genetic

---

[1]Each string $s(t)$ is a simple binary encoding of some point in the function domain.

```
GENETIC-ALGORITHM:

     Set t=0
     Generate an initial population B(0) of M strings at
     random
     Set N=M

→    Compute f(s) for each string s in B(t)

     Save the best string s'

     Compute the selection probabilities
        p(s_i) = f(s_i) / [ Σ_{j=1}^{N} f(s_j) ]

     Compute the expected number of offspring
         c(s_i) = p(s_i) * N

     Generate M new strings for B(t+1):
     FOR  j = 1 to N

          Choose a uniform random number r in [0,1]

          IF  r ≤ the probability of crossover P_c
              THEN

                   Call SELECT twice to get 2 parents s_1
                   and s_2

                   Decrement c(s_1) and c(s_2) by 1/2

                   Apply crossover to s_1 and s_2
                   Randomly select one of the resultant
                   strings to use as offspring j

              ELSE

                   Call SELECT to get a parent s

                   Decrement c(s) by 1

                   Use s as offspring j

          Apply mutation to each gene position of offspring
          j with probability P_m

     IF  the best string s' is not in B(t+1)
          THEN

               Set N=M+1

               Append s' to B(t+1)

          ELSE
               Set N=M

     Set t=t+1

SELECT:

     Set x=0

     WHILE  x≤0

          Choose a string s at random from B(t) using the
          selection probabilities

          Set x = c(s)

     Return s
```

Figure A.1. The genetic algorithm that was DeJong's [1975] best function optimizer where M=50, P_C=0.6, and P_m=0.001.

algorithms do better than standard optimization techniques on the functions in E. In particular, the version of the genetic algorithm shown in Figure A.1 - hereafter referred to as R0 - had the best overall performance in an interval of observation $T = 6000$. Despite this success, however, the performance of the genetic algorithm falls short of the theoretical expectations. The problem is that, while the theory was developed using limit theorems and the law of large numbers, any implementation of the algorithm uses only a very small population or set of sample points. Consequently, the cumulative sampling error from several iterations of the algorithm can lead to trajectories much different from those theoretically predicted. The success of R0 shows that DeJong made considerable progress in eliminating some of these finite stochastic effects. There remains, however, substantial room for improvement. The following sections describe changes designed to improve the performance of R0 on E. In accordance with Bethke's [1981] suggestion, all performance results are normalized to the same range to help guarantee consistent comparisons.

Scaling

The computation of selection probabilities in R0 tacitly assumes that the function values are all non-negative. In order to handle negative values, all function values $f(s)$ are rescaled to the value

$$g(s) = \text{FMAX} - f(s) + 1$$

where FMAX is the largest function value observed so far. Each of the functions in E is to be minimized. This scaling assures that all values will be positive and that the smallest values have the highest selection probabilities. Unfortunately, this scale also introduces an undesirable side effect. As the function value of the average individual in the population improves via the genetic algorithm, the difference FMAX-$f(s)$ gets increasingly large. Eventually this difference can get so large that the scaled function values no longer reflect the true range of values in the population. For example, the maximum value for F1 is 78.6 and the minimum is zero. When the best string in the population has value 5 and the worst string has value 10, the best string is relatively speaking twice as valuable as the worst. Under scaling, however, the values are altered to 69.6 and 78.6 respectively assuming FMAX=78.6. This reduces the relative advantage of the best string from 2.0 to 1.13. Similarly, adding 1 to the scaled value can have undesired consequences. If the range of function values in the population lies within the unit interval, for example, then the relative value of two strings is greatly distorted by the scale.

Finding an acceptable scale is a very difficult problem. No attempt will be made here to solve the problem since negative function values do not occur in the classifier system under consideration. However, it is hypothesized that any improvement in the current scale will

improve significantly the performance of R0. To test this hypothesis, a new plan R1 is tested using the scale

$$g(s) = \text{CURR\_MAX} - f(s) + 1$$

where CURR_MAX is the largest function value in the current population. This scale clearly has some of the problems of the original scale. It is an improvement in that it is flexible with respect to the changing range of values in the population. Plan R1 was tested on all five functions. The results for this and all subsequent tests were averaged over ten runs. The performance of R1 is compared in Table A.1 with that given by Bethke [1981] for R0.

TABLE A.1

THE PERFORMANCE OF R0 AND R1 ON *E*

Function	On-line		Off-line	
	R0	R1	R0	R1
F1	2.95	2.3218	0.145	0.1810
F2	0.8899	0.7175	0.0057	0.0074
F3	6.382	4.342	3.273	2.47
F4	3.263	1.8501	1.412	0.9067
F5	6.681	6.206	0.469	0.5256
E	4.033	3.0874	1.061	0.8181

The results in Table A.1 show that R1 performs considerably better - that is, is more robust - than R0 over *E* using either on-line or off-line criteria. This confirms the hypothesis that the original scale had a detrimental effect on overall plan performance. It is interesting to

note that all of the on-line results for R1 are better than R0, while all the off-line results are only better - though substantially so - on F3 and F4. This illustrates two points. First, it confirms DeJong's observation that there is usually a tradeoff involved in on-line versus off-line performance. The bold exploration of *A* conducive to good off-line performance is often at odds with the more conservative sampling strategies leading to good on-line performance. Second, the dramatic improvement in off-line performance on F3 and F4 confirms our intuitions about why the original scale was a mistake. These two functions are associated with very large domains *A* relative to the other functions. F3 has a domain containing 10^{15} alternative solutions. F4 has a domain containing 10^{72} alternatives. By contrast, the next largest domain contained only 10^{9} alternatives. It is precisely in such large search spaces, where the plan has a "long way to go", that one would expect the original scale to make continued improvement very difficult.

Improved trial allocation

In any implementation of the genetic algorithm, a new population is generated by taking a finite sample using the selection probabilities. Because the sample is finite, and usually small, it is subject to sampling error. This means there is almost always some difference between the expected number of trials, or offspring, for any given schema and the

actual number allocated by the algorithm. Once this sampling error gives one schema an unwarranted advantage over another, the variance increases and the allocation of trials moves even further away from the expected proportions. This phenomenon is called genetic drift in population genetics.

In order to combat this problem, DeJong used a simple heuristic to reduce the disparity between the expected and actual number of trials. The expected number of offspring $c(s)$ for each individual is explicitly computed before any offspring are produced. As an individual produces offspring this number is decremented: by 0.5 if crossover is involved and by 1.0 otherwise. An individual is no longer eligible to produce offspring once $c(s)$ is less than or equal to zero. This heuristic was shown to improve both the on-line and off-line performance of the genetic algorithm on E. It is reasonable to assume that more improvement can be made if the sampling error is reduced even further.

To test this hypothesis, an improved version of the heuristic is proposed. The new heuristic is based on the "remainder-stochastic" sampling method described by Brindle and Sampson [1979]. Whenever $c(s)$ is greater than or equal to the number of offspring k being selected for - k is either 0.5 or 1.0 - then proceed as before. Otherwise, select an eligible individual with probability $c(s)/k$. For example, if $c(s) = 1/4$ then individual s has a probability of 1/2 of producing an offspring involving crossover. A

random number generator is used to make this decision. If the decision is negative, then the selection probabilities are used to select another candidate. DeJong's heuristic merely dampens the accumulation of sampling error. Brindle and Sampson show that the remainder-stochastic method has a much lower variance than pure stochastic sampling.

TABLE A.2

THE PERFORMANCE OF R1 AND R2 ON *E*

Function	On-line		Off-line	
	R1	R2	R1	R2
F1	2.322	1.8166	0.181	0.1133
F2	0.7175	0.5964	0.0074	0.011
F3	4.342	4.162	2.47	2.483
F4	1.85	1.713	0.9067	0.8234
F5	6.206	5.029	0.5256	0.9119
E	3.087	2.663	0.8181	0.8685

A new plan R2 was implemented by using the new selection heuristic with plan R1. The on-line and off-line performance of R2 on *E* is compared with R1 in Table A.2. On-line performance is improved on *E* while off-line performance suffers. Note that the drop in off-line performance is due almost exclusively to poor performance on the multimodal function F5. DeJong observed the same phenomenon, to a lesser degree, when he introduced his sampling heuristic. Apparently, reduction in sampling error means more rapid search of above average schema. This is

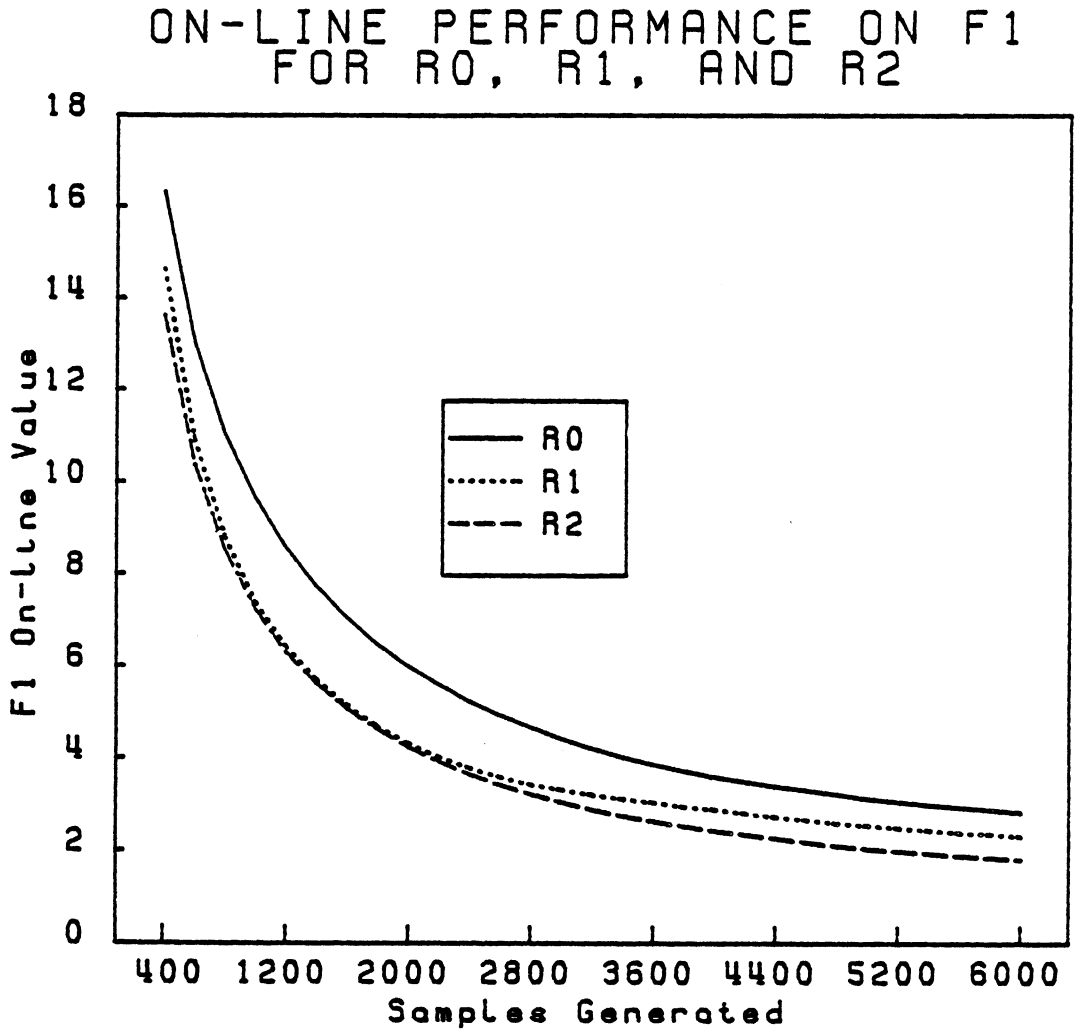

Figure A.2. On-line performance of R0, R1, and R2 on F1.

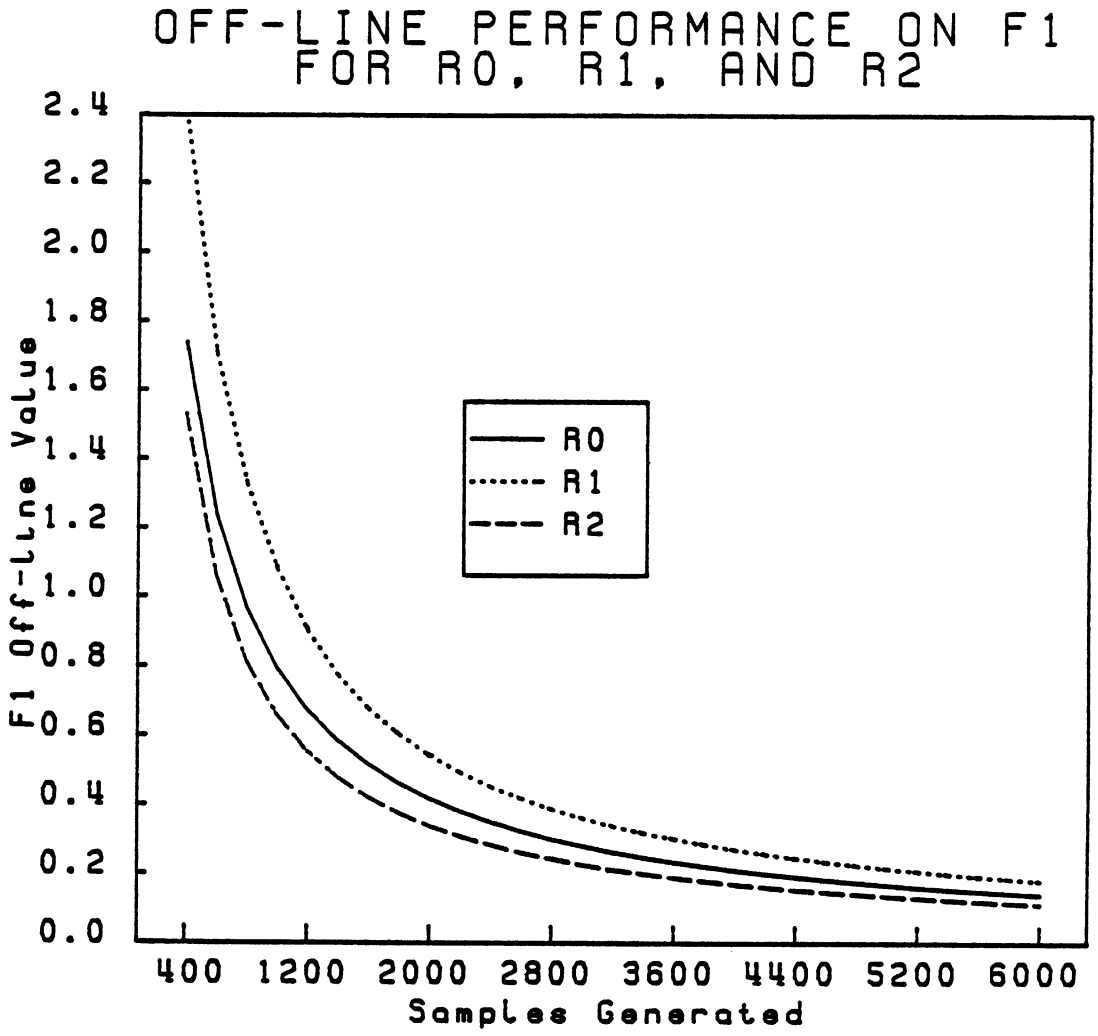

Figure A.3. Off-line performance of R0, R1, and R2 on F1.

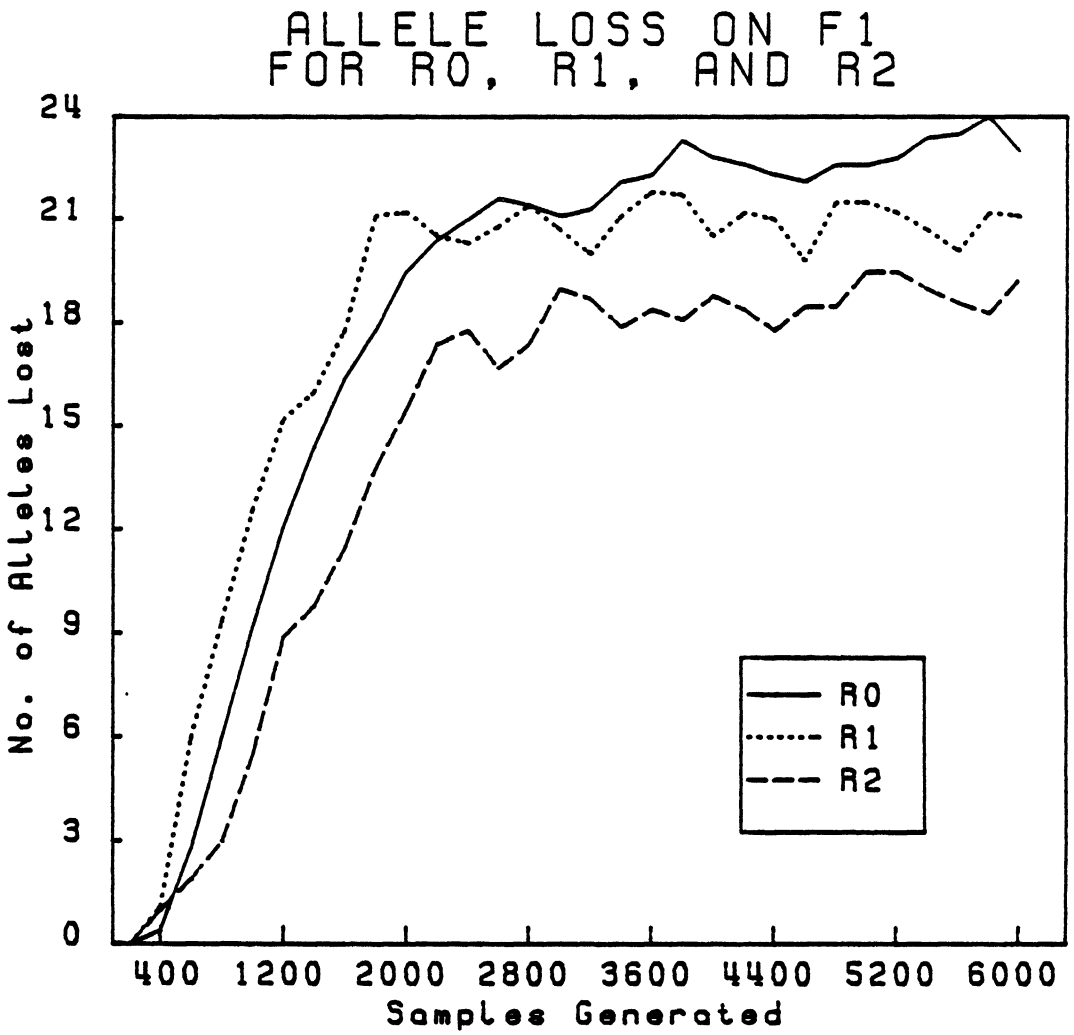

Figure A.4. Allele loss for R0, R1, and R2 on F1.

helpful on a unimodal function; but, on a difficult multimodal function like F5, the plan can rapidly converge on a good local optimum and lose alternative values - or *alleles* - at string loci crucial for finding the true optimum. Overcoming this implementation problem requires introducing a "crowding pressure" to control the growth of every schema. This is considered in detail in Chapter 5. The loss of alleles due to sampling error is particularly noticeable for F1, a symmetric function for which each allele - or, more accurately, each first order schema - has equal value and which therefore should show allele loss only when the optimum is found. The performance of R0, R1, and R2 on F1 are compared in Figures A.2 and A.3. The allele loss rates are compared in Figure A.4. Each string in the population was 30 positions long, meaning there were 30 alleles to be potentially lost. These results suggest that the selection method of R2 does indeed substantially reduce sampling error and thereby improve performance.

Saving Both Offspring

The fact that reducing allele loss is linked to improved performance suggests another implementation issue that warrants discussion. The crossover operator produces two resultant strings with each application; yet, the genetic plans R0 thru R2 only use one of these strings as an offspring and discard the other. This seems to be a mistake. Discarding a string might be an unwarranted source

of allele loss when the string is the last instance of some first order schema. It could also be a source of sampling error when the discarded string is a sample point for under-represented regions of the search space. This would lead, of course, to even more allele loss. It is therefore hypothesized that using both strings produced by crossover should improve the performance of R2 on *E*.

To test this hypothesis, R2 was modified to use both strings from a crossover. This involved, among other things, decrementing the expected number of offspring $c(s)$ by 1 for each cross instead of 0.5. The resulting plan will be called R3. Figures A.5 - A.7 compare the behavior of R3 on F1 with that of R2. The improvement in the allele loss rate is dramatic, confirming our hypothesis. There is further improvement in on-line performance and a negligible difference in off-line performance. Table A.3 compares R3 and R2 on *E*. Despite continued problems with off-line performance on F5, R3 shows overall improvement in both on-line and off-line performance over *E*.

Generalized Crossover

One final observation about the crossover operator. It has been noted that crossover is the primary search operator for genetic plans. What has not been emphasized is that the search is most effective for those schemata whose non-nil values are close together on the string. This is because the further apart those values are, the more likely they are

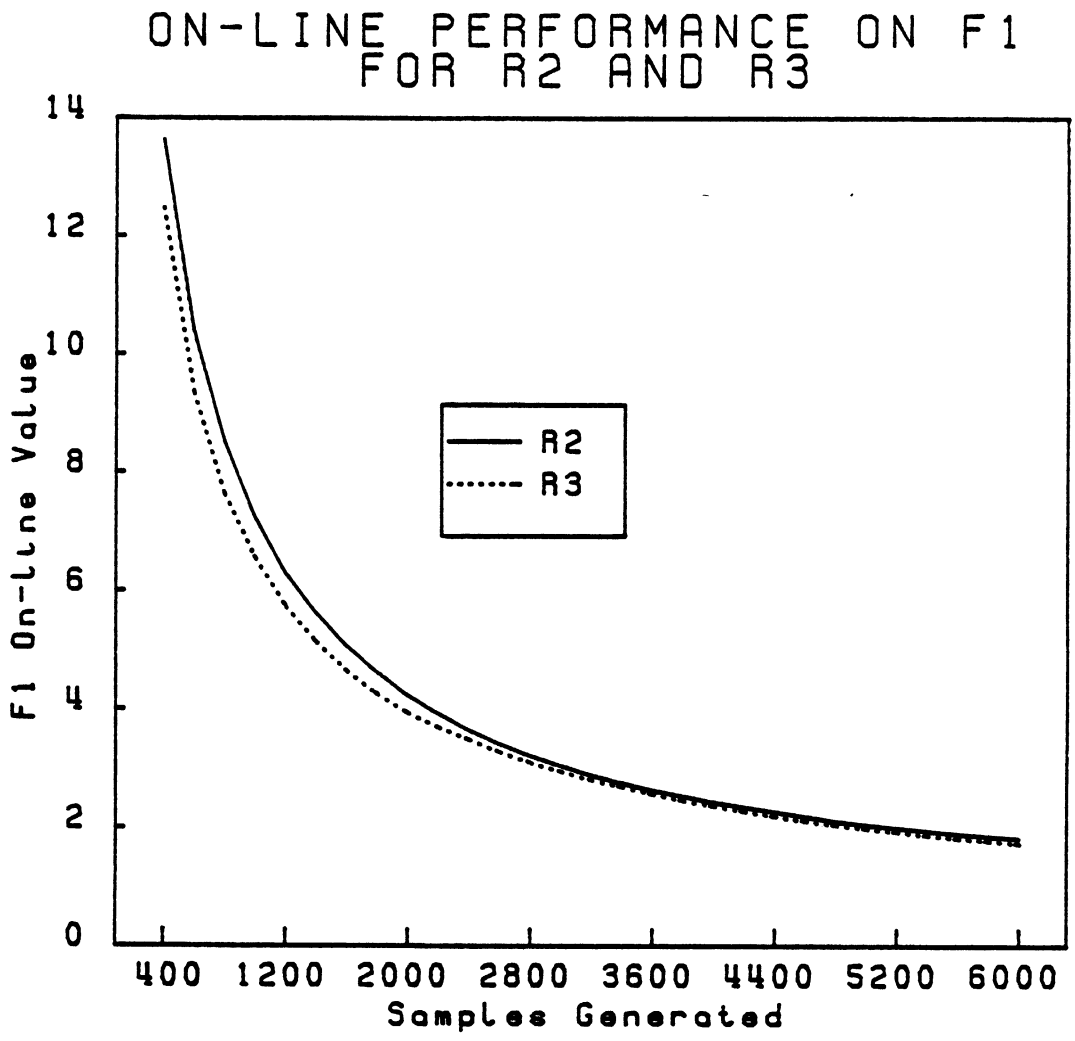

Figure A.5. On-line performance of R2 and R3 on F1.

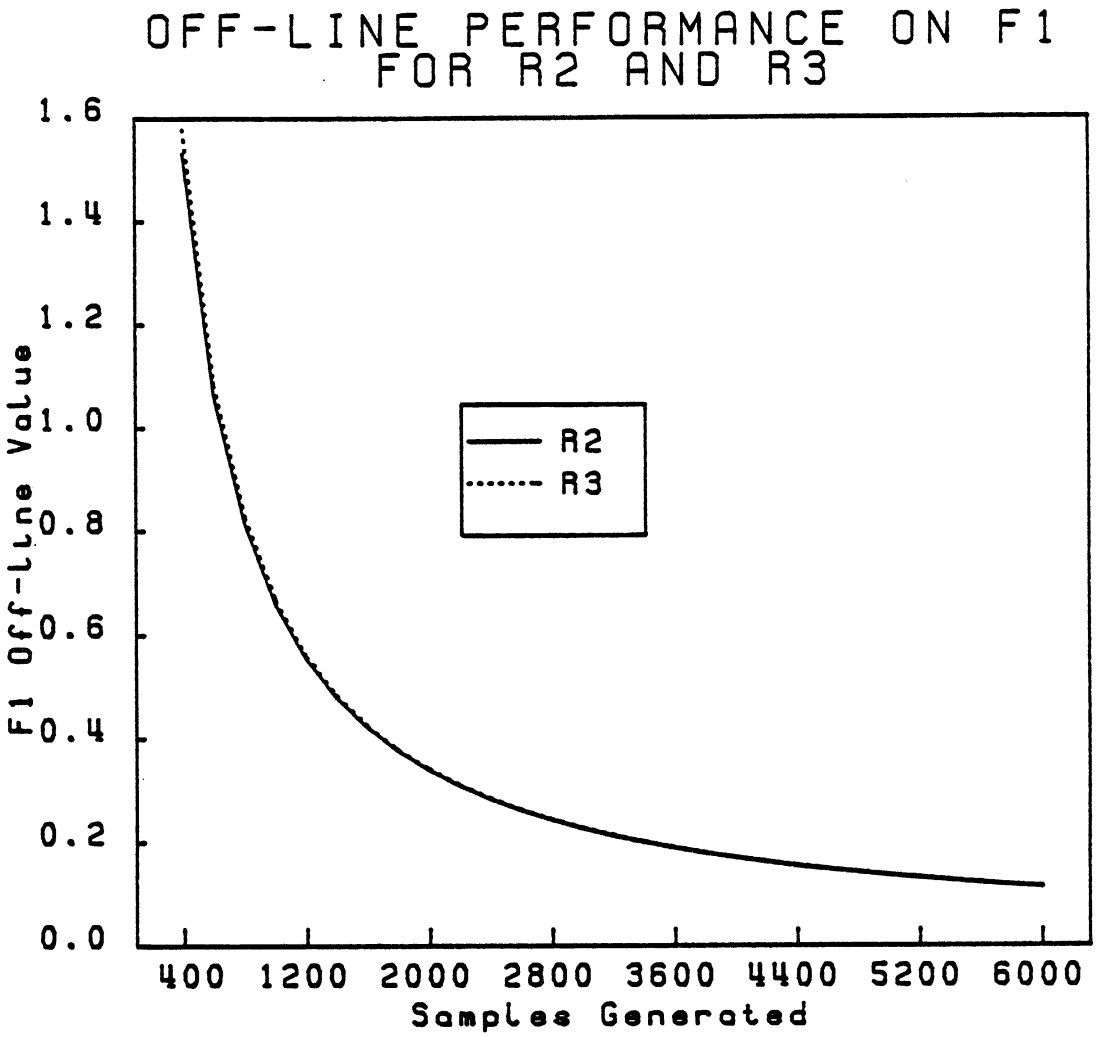

Figure A.6. Off-line performance of R2 and R3 on F1.

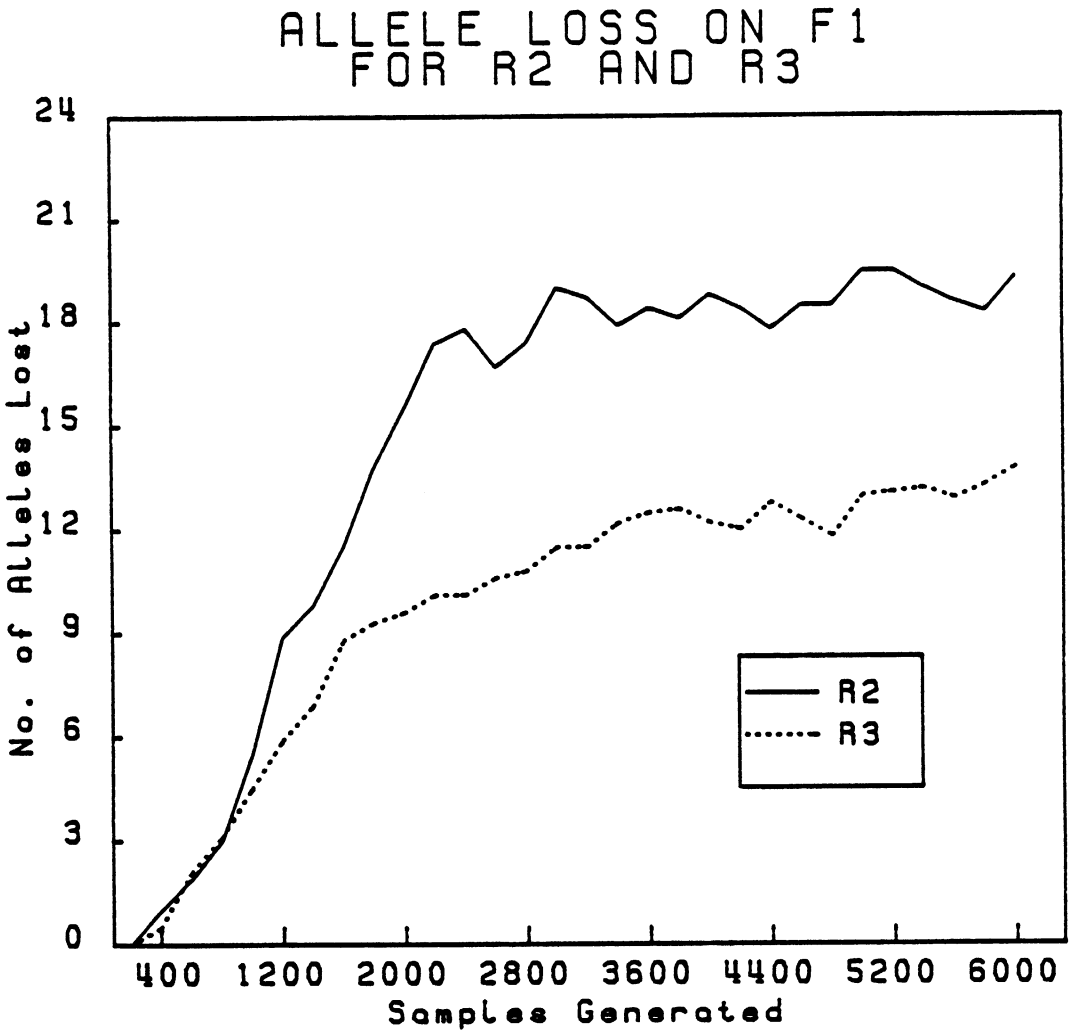

Figure A.7. Allele loss for R2 and R3 on F1.

TABLE A.3

THE PERFORMANCE OF R2 AND R3 ON *E*

Function	On-line		Off-line	
	R2	R3	R2	R3
F1	1.8166	1.732	0.1133	0.1136
F2	0.5964	0.5687	0.011	0.0069
F3	4.1617	3.9496	2.4828	2.4265
F4	1.7134	1.6689	0.8234	0.7974
F5	5.0294	4.7674	0.9119	0.9613
E	2.6635	2.5374	0.8685	0.8611

to be separated by crossover; and, consequently, the less likely one of the resultant strings will be an instance of that schema. For example, crossover will break up the schema 11####...## with probability $1/l$ where l is the length of the string. The schema 1####...###1, on the other hand, will be broken up with probability 1.0 because the non-nil values are as far apart as possible. Holland [1975] has shown that this disruptive effect of crossover does not seriously effect the algorithm's ability to robustly search the entire space. Still, any reduction in this disruption should improve the effectiveness of crossover and the performance of the genetic plan.

DeJong [1975] has pointed out that such a reduction can be achieved by a simple modification to the crossover

operator.[2]

> If we think of a chromosome as a circle with the first gene immediately following the last then it becomes immediately clear that there are in fact 2 crossover points: one fixed at position zero and the other randomly selected. An immediate generalization to the present crossover operator is to allow both crossover points to be randomly selected. (p.149)

The advantage to be gained is that schemata like 1¤¤¤...¤¤¤1 are no longer always disrupted by crossover. This should be reflected in improved performance.

DeJong [1975] implemented the generalized operator and discovered that allele loss on F1 became even more severe. Off-line performance on *E* improved as expected since more of the search space is presumably easier to access. On-line performance however decreased. The degradation of on-line performance on *E* and, especially, the increase in allele loss on F1 are both surprising results. An examination of DeJong's implementation of generalized crossover reveals a flaw that might explain the problem. Figure A.8a shows the outcome of DeJong's algorithm applied to two strings. Note that the third segment in the second string is never involved in the final result. This is an inadvertent source of allele loss that can be easily corrected as shown in Figure A.8b. Here, all the alleles from the parent strings are accounted for in the offspring.

---

[2]The disruptive effect of crossover is countered by the inversion operator in biological systems. Studies by Frantz [1972], however, suggest that for genetic plans the inversion operator is only effective over very long intervals of behavior.

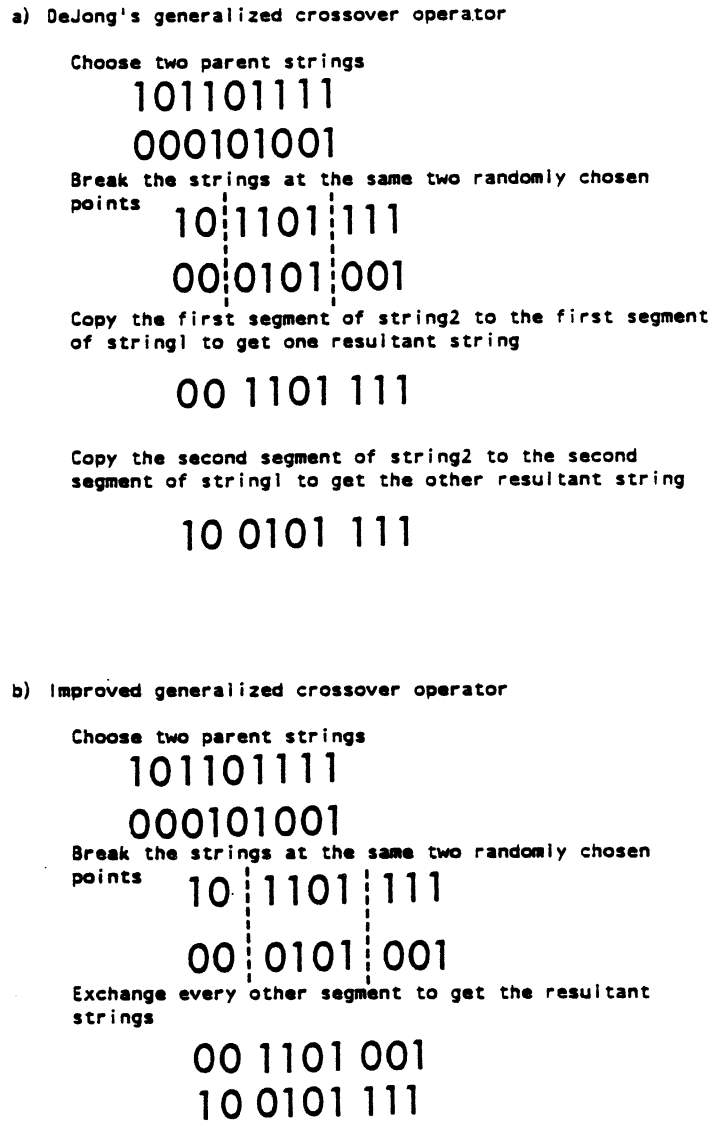

Figure A.8. Two versions of the generalized crossover operator.

The improved generalized crossover operator was inserted in R3 to implement a new plan R4. Figure A.11 shows that, as expected, generalized crossover reduces the rate of allele loss on F1. There is a degradation of on-line and off-line performance on F1, but Table A.4 shows an overall improvement in both measures for R4 over *E*. The

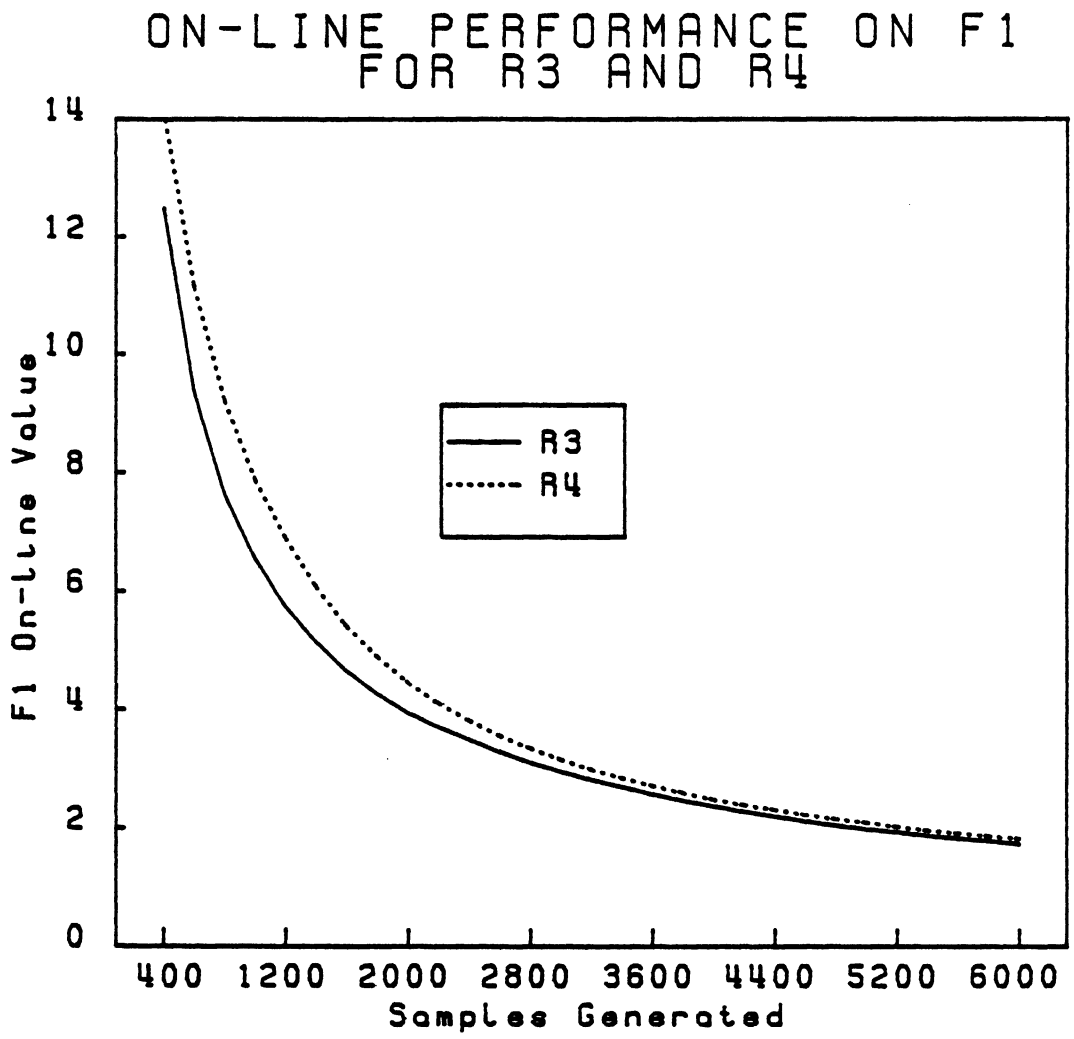

Figure A.9. On-line performance of R3 and R4 on F1.

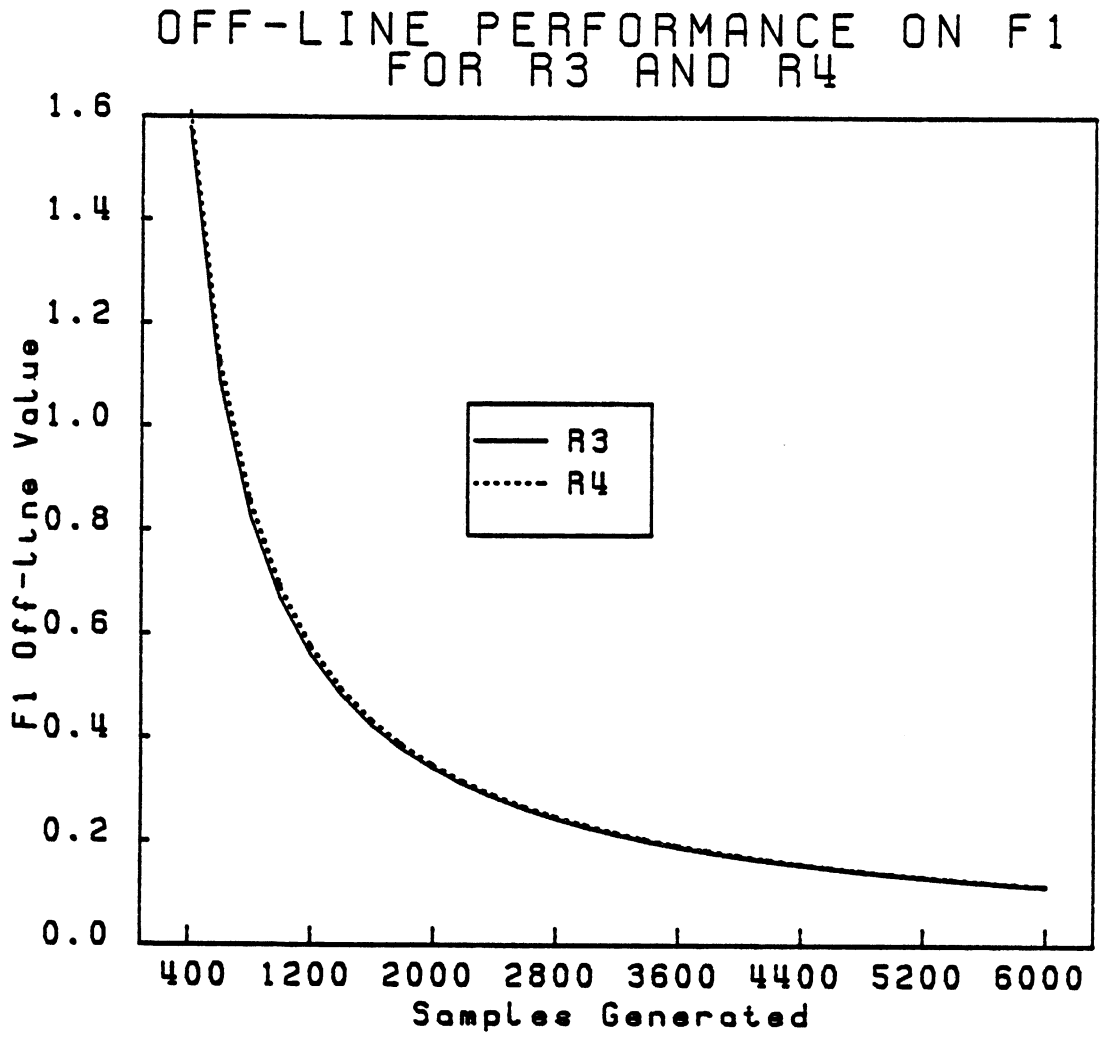

Figure A.10. Off-line performance of R3 and R4 on F1.

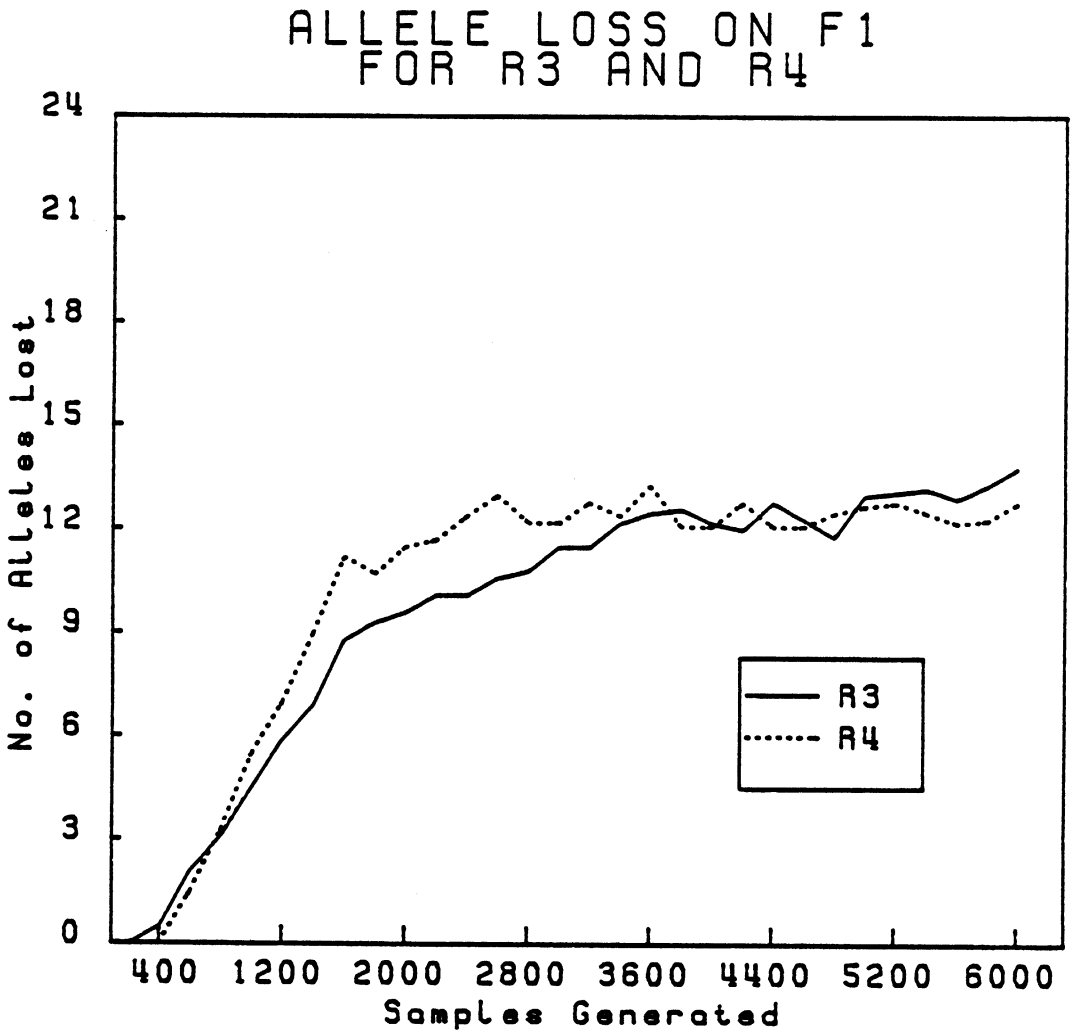

Figure A.11. Allele loss for R3 and R4 on F1.

improvement in off-line performance is particularly remarkable. These results confirm DeJong's theoretical analysis of the generalized crossover operator.

TABLE A.4

THE PERFORMANCE OF R3 AND R4 ON *E*

Function	On-line		Off-line	
	R3	R4	R3	R4
F1	1.7325	1.8165	0.1136	0.1153
F2	0.5687	0.5775	0.0069	0.0033
F3	3.9496	3.4953	2.4265	1.9712
F4	1.6689	1.6137	0.7974	0.6998
F5	4.7674	5.0751	0.9613	0.7079
E	2.5374	2.5156	0.8611	0.6995

Summary

It is worthwhile pointing out how much of an improvement R4 is over R0. Table A.5 compares the two plans and shows that, in almost every way, R4 is far superior. Figure A.12 shows the reduction in sampling error as indicated by the allele loss rate on F1. Again, the improvement is substantial. R4 will therefore serve as the foundation for the learning algorithm to be used in the classifier system.

One aspect of R4 needs to be modified, however, before it can be used in a classifier system. The function value associated with a classifier will vary over time; that is,

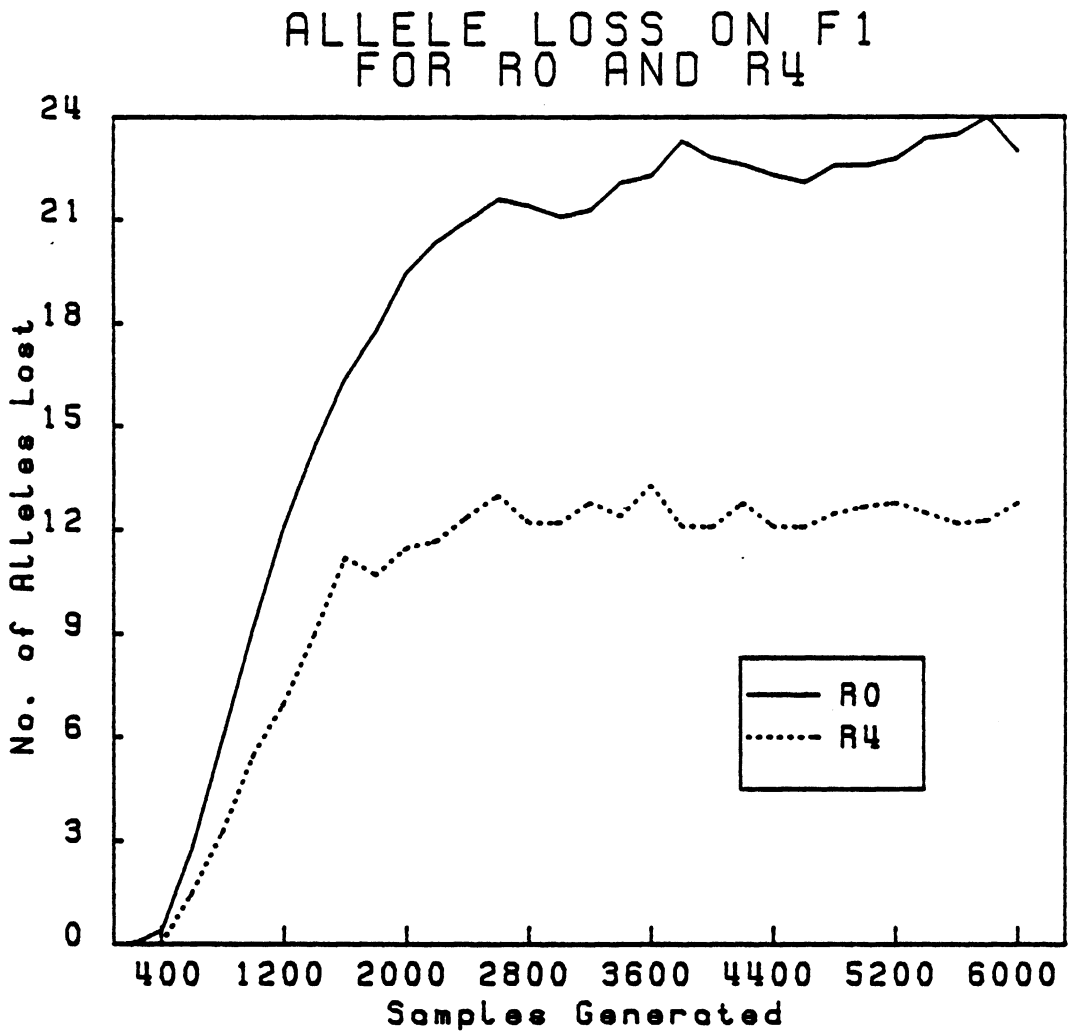

Figure A.12. Allele loss for R0 and R4 on F1.

TABLE A.5

THE PERFORMANCE OF R0 AND R4 ON *E*

Function	On-line		Off-line	
	R0	R4	R0	R4
F1	2.95	1.8165	0.145	0.1153
F2	0.8899	0.5775	0.00566	0.0033
F3	6.382	3.4953	3.273	1.9712
F4	3.263	1.6137	1.412	0.6998
F5	6.681	5.0751	0.469	0.7079
E	4.033	2.5156	1.061	0.6995

in some contexts a classifier will be highly rated, in other contexts it will have a low rating. It no longer makes sense, therefore, to save the best string from one generation to the next. The plan must search for those strings with the best expected function value. Accordingly, that extra[3] step in the algorithm will be deleted. The final version of the algorithm, called G0 in Chapter 5, is shown in Figure A.13.

[3]Saving the best string was never necessary in the first place. If the only copy of a string is deleted the algorithm will quickly generate it again if it was highly rated. What counts is the set of schemata being produced and maintained, not the instances in the population at any given time.

```
GENETIC-ALGORITHM G0:

     Set t=0
     Generate an initial population B(0) of M strings at
     random

→    Compute f(s) for each string s in B(t)

     Compute the selection probabilities
        p(s_i) = f(s_i) / [ Σ_{j=1}^{M} f(s_j) ]

     Compute the expected number of offspring
         c(s_i) = p(s_i) * M

     Compute the even number of offspring N_c to be generated
     by crossover
                  { M*P_c      if M*P_c is even
           N_c =  { M*P_c±1    otherwise (choose at random)

     Generate M new strings for B(t+1):
     Set i = 0, j = 0
     WHILE  i < M

          Choose a uniform random number r in [0,1]

          IF  (M-i)*r < (N_c-j)
               THEN

                    Call SELECT twice to get 2 parents s_1
                    and s_2

                    Decrement c(s_1) and c(s_2) by 1

                    Apply crossover to s_1 and s_2
                    to get offspring i and i+1

                    Apply mutation to each gene position of
                    the offspring with probability P_m

                    Set i = i+2, j = j+2

               ELSE

                    Call SELECT to get a parent s

                    Decrement c(s) by 1

                    Use s as offspring i

                    Apply mutation to each gene position of
                    the offspring with probability P_m

                    Set i = i+1

     Set t=t+1


SELECT:

     Set x=0

     WHILE  x<1

          Choose a string s at random from B(t) using the
          selection probabilities

          IF  c(s)≥1
               THEN

                    Set x = 1

               ELSE

                    Choose a uniform random number r in
                    [0,1]

                    IF  r < c(s)
                         THEN

                              Set x = 1

     Return s
```

Figure A.13. The genetic plan G0 where M=50, P_c=0.6, and P_m=0.001.

APPENDIX B

List of System Parameters

Parameter	Value	Comment
BADSIG	111...111	Tag associated with aversive stimulation
EPSILON	0.5	Fraction of strength lost by a classifier when its strength is modified
FEEDBACK	72	Initial feedback estimate
GOODSIG	000...000	Tag associated with appetitive stimulation
INIT	320	Strength for each classifier when the system is initialized
LEARNRATE	10	The frequency with which the genetic algorithm is used
M_MAX	5	Maximum intensity assigned to any message
NACTIVE	5	Number of classifiers activated per time step
NCLS	200	Number of classifiers in a population

NOFFSPRING	12	Number of new classifiers generated by the genetic algorithm
NPARENTS	30	Number of parents used in the genetic algorithm and the minimum number of relevant classifiers per message
SETSIZE	2.5	When there are fewer than NPARENTS matching classifiers, parents (and, additional relevant classifiers) are chosen from the SETSIZE*NPARENTS classifiers with the highest match scores
STRENGTH	NPARENTS*INIT	The strength shared by relevant classifiers when their strengths are modified. This is the "carrying capacity" of every "niche"
TAGSCORE	8	Initial estimated tag score
THRESHOLD	10	The deprivation level at which food seeking is motivated

REFERENCES

REFERENCES

Anderson, J. and Kline, P. (1979) "A Learning System and its Psychological Implications". Proceedings of the Sixth International Joint Conference on Artificial Intelligence, p. 16-21.

Arbib, M. A. (1972) The Metaphorical Brain. New York: Wiley.

Bartlett, Sir F. C. (1932) Remembering: A study in experimental and social psychology. Cambridge, England: The University Press, 1964.

Benedikt, M. (1978) "Isovists and Isovist Fields". Paper presented at the Ninth Annual Environmental Design Research Association Conference, Tucson, Arizona, 1978.

Bindra, D. (1978) "How adaptive behavior is produced: a perceptual-motivational alternative to response-reinforcement". The Behavioral and Brain Sciences 1:41-91.

Bethke, A. D. (1981) "Genetic Algorithms as Function Optimizers". Ph.D. dissertation, University of Michigan.

Boden, M. (1977) Artificial Intelligence and Natural Man. New York: Basic Books.

Bransford, J. (1979) Human Cognition: Learning, Understanding and Remembering. Belmont, California: Wadsworth.

Brindle, A. and Sampson, J. (1979) "Analysis of Frequency Error in Three Sampling Algorithms". Unpublished paper, Department of Computing Science, University of Alberta.

Bruner, J. S. (1957) "Going Beyond the Information Given". In Bruner, J. S. *et al.* Contemporary Approaches to Cognition. Boston: Harvard University Press.

Brunswik, E. (1956) Perception and the Representative Design of Psychological Experiments. Second edition, Berkeley: University of California Press.

Campbell, D. T. (1966) "Pattern Matching as an Essential in Distal Knowing". In Hammond, K. (Ed.) The Psychology of Egon Brunswik. New York: Holt, Rinehart and Winston.

Cavicchio, D. J. (1970) "Adaptive Search Using Simulated Evolution". Ph.D. dissertation, University of Michigan.

Charlesworth, W. R. (1976) "Human Intelligence as Adaptation: An Ethological Approach". In Resnick, L. B. (Ed.) The Nature of Intelligence. Hillsdale, New Jersey: Erlbaum.

Craik, K. J. W. (1943) The Nature of Explanation. New York: Cambridge University Press.

Cunningham, M. A. (1972) Intelligence: Its Organization and Development. New York: Academic Press.

Cunningham, M. A. and Gray, H. J. (1974) "Design and Test of a Cognitive Model". International Journal of Man-Machine Studies 6:49-104.

Davis, R. and King, J. (1976) "An Overview of Production Systems". In Elcock, E. W. and Michie, D. (Eds.) Machine Intelligence 8. New York: Wiley.

DeJong, K. A. (1975) "Analysis of the Behavior of a Class of Genetic Adaptive Systems". Ph.D. dissertation, University of Michigan.

DeJong, K. A. (1980) "Adaptive System Design: A Genetic Approach". IEEE Transactions on Systems, Man and Cybernetics 10:566-574.

Dennett, D. (1978) Brainstorms. Montgomery, Vermont: Bradford Books.

Doran, J. E. (1968) "Experiments with a Pleasure-seeking Automaton". In Michie, D. (Ed.) Machine Intelligence 3. New York: American Elsevier.

Dreyfus, H. L. (1972) What Computers Can't Do: A Critique of Artificial Reason. New York: Harper & Row.

Erman, L., Hayes-Roth, F., Lesser, V., and Reddy, D. (1980) "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty". Computing Surveys 12:213-253.

Feigenbaum, E. A. (1977) "The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering". Proceedings of the Fifth International Joint Conference on Artificial Intelligence, p. 1014-1029

Findler, N. and Allan, A. (1973) "Studies on the Behavior of an Organism in a Hostile Environment". Journal of the Institution of Computer Sciences 4:58-69.

Flynn, J. P. (1978) "Sensory vs. motor effects of brain stimulation". The Behavioral and Brain Sciences 1:58-59.

Fogel, L., Owens, A., and Walsh, M. (1966) Artificial Intelligence Through Simulated Evolution. New York: Wiley.

Fraenkel, G. S. and Gunn, D. L. (1961) The Orientation of Animals. New York: Dover.

Frantz, D. R. (1972) "Non-linearities in genetic adaptive search". Ph.D. dissertation, University of Michigan.

Garner, W. R. (1974) The Processing of Information and Structure. Potomac, Maryland: Erlbaum.

Garner, W. R. (1978) "Aspects of a Stimulus: Features, Dimensions, and Configurations". In Rosch, E. and Lloyd, B. (Eds.) Cognition and Categorization. Hillsdale, New Jersey: Erlbaum.

Gibson, J. J. (1966) The Senses Considered as Perceptual Systems. Boston: Houghton Mifflin..

Gibson, J. J. (1979) The Ecological Approach to Visual Perception. Boston: Houghton Mifflin.

Goldstein, I. and Papert, S. (1977) "Artificial Intelligence, Language, and the Study of Knowledge". Cognitive Science 1:84-123.

Haber, R. N. (1978) "Perception of Visual Space in Scenes and Pictures". In Harmon, L. (Ed.) The Interrelationships of the Communicative Senses. National Science Foundation Report.

Hayes-Roth, F. (1973) "A Structural Approach to Pattern Learning and the Acquisition of Classificatory Power". Proceedings of the First International Joint Conference on Pattern Recognition, p. 343-355.

Hayes-Roth, F. (1974) "Schematic Classification Problems and their Solution". Pattern Recognition 6:105-113.

Hayes-Roth, F. (1976) "Patterns of Induction and Associated Knowledge Acquisition Algorithms". In Chen, C. H. (Ed.) Pattern Recognition and Artificial Intelligence. New York: Academic Press.

Hebb, D. O. (1949) The Organization of Behavior. New York: Wiley.

Hebb, D. O. (1972) Textbook of Psychology. Third Edition. Philadelphia: W. B. Saunders Company.

Hewitt, C. (1977) "Viewing Control Structures as Patterns of Passing Messages". Artificial Intelligence 8:323-364.

Hilgard, E. R. (1978) "The Goals of Perception". In Kaplan, S. and Kaplan, R. (Eds.) Humanscape: Environments for People. North Scituate, Massachusetts: Duxbury Press.

Hilgard, E. and Bower, G. (1975) Theories of Learning. Fourth edition. Englewood Cliffs, New Jersey: Prentice-Hall.

Hochberg, J. (1974) "Higher-Order Stimuli and Inter-Response Coupling in the Perception of the Visual World". In Macleod, R. B. and Pick, H. L. (Eds.) Perception: Essays in Honor of James J. Gibson. Ithaca, New York: Cornell University Press.

Holland, J. H. (1962) "Information Processing in Adaptive Systems". Volume 3 of the Proceedings of the International Union of Physiological Sciences, p. 330-338.

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

Holland, J. H. (1976) "Adaptation". In Rosen, R. and Snell, F. (Eds.) Progress in Theoretical Biology 4. New York: Academic Press.

Holland, J. H. (1980) "Adaptive Algorithms for Discovering and Using General Patterns in Growing Knowledge-Bases". International Journal of Policy Analysis and Information Systems 4:217-240.

Holland, J. H. and Reitman, J. S. (1978) "Cognitive Systems Based on Adaptive Algorithms". In Waterman, D. A. and Hayes-Roth, F. (Eds.) Pattern-Directed Inference Systems. New York: Academic Press.

Holldobler, B. and Lumsden, C. (1980) "Territorial Strategies in Ants". Science 210:732-739.

Jacobs, W. (1972) "How a Bug's Mind Works". In Robinson, H. W. and Knight, D. E. (Eds.) Cybernetics, Artificial Intelligence, and Ecology. New York: Spartan Books.

James, W. (1892) Psychology: The Briefer Course. New York: Harper Torchbook, 1961.

John, E. R. (1967) Mechanisms of Memory. New York: Academic Press.

Johnston, T. D. (1981) "Contrasting approaches to a theory of learning". The Behavioral and Brain Sciences 4:125-173.

Kaplan, S. (1970) "The Role of Location Processing in the Perception of the Environment". In Archea, J. and Eastman, C. (Eds.) EDRA 2: Proceedings of the Second Annual Environmental Design Research Association Conference., Pittsburgh.

Kaplan, S. (1973) "Cognitive maps in perception and thought". In Downs, R. and Stea, D. (Eds.) Image and Environment. Chicago: Aldine.

Kaplan, S. (1976) "Adaptation, Structure, and Knowledge". In Moore, G. T. and Golledge, R. G. (Eds.) Environmental Knowing: Theories, Research, and Methods. Stroudsburg, PA: Dowden, Hutchinson and Ross.

Kaplan, S. (1978) "Perception of an Uncertain Environment". In Kaplan, S. and Kaplan, R. (Eds.) Humanscape: Environments for People. North Scituate, Massachusetts: Duxbury Press.

Kaplan, S. and Kaplan, R. (1982) Cognition and Environment: Functioning in an Uncertain World. Monterey, California: Brooks/Cole.

Kaufmann, G. (1979) Visual Imagery and its Relation to Problem Solving. New York: Columbia University Press.

Keller, F. S. (1954) Learning: Reinforcement Theory. New York: Random House.

Kuipers, B. (1979) "On Representing Commonsense Knowledge". In Findler, N. V. (Ed.) Associative Networks. New York: Academic Press.

Lachman, J. and Lachman, R. (1979) "Theories of Memory Organization and Human Evolution". In Puff, C. R. (Ed.) Memory Organization and Structure. New York: Academic Press.

Lashley, K. S. (1942) "The problem of cerebral organization in vision". In Cattell, J. (Ed.) Biological Symposia, Volume 7, Lancaster, PA: Cattell Press.

Lashley, K. S. (1949) "Persistent Problems in the Evolution of Mind". Quarterly Review of Biology 24:28-42.

Lashley, K. S. (1951) "The problem of serial order in behavior". In Jeffress, L. A. (Ed.) Cerebral Mechanisms in Behavior: The Hixon Symposium. New York: Wiley.

Lenat, D. (1975) "Beings: Knowledge as Interacting Experts". Proceedings of the Fourth International Joint Conference on Artificial Intelligence, p. 126-133.

Luria, A. R. (1973) The Working Brain: An Introduction to Neuropsychology. New York: Basic Books.

Lynch, K. (1960) The Image of the City. Cambridge: MIT Press.

Mackintosh, N. J. (1974) The Psychology of Animal Learning. New York: Academic Press.

MacLaren, L. (1978) "A Production System Architecture Based on Biological Examples". Ph.D. dissertation, University of Washington.

Mandler, G. (1967) "Organization and Memory". In Spence, K. W. and Spence, J. T. (Eds.) The Psychology of Learning and Motivation. Volume 1. New York: Academic Press.

Marr, D. (1977) "Artificial Intelligence - A Personal View". Artificial Intelligence 9:37-48.

Marr, D. (1979) "Representing and Computing Visual Information". In Winston, P. H. and Brown, R. H. (Eds.) Artificial Intelligence: an MIT Perspective, Volume 2, Cambridge: MIT Press.

McCorduck, P. (1979) Machines Who Think. San Francisco: W. H. Freeman and Co.

Medin, D. and Schaffer, M. (1978) "Context Theory of Classification Learning". Psychological Review 85:207-238.

Mervis, C. and Rosch, E. (1981) "Categorization of Natural Objects". Annual Review of Psychology 32:89-115.

Midgley, M. (1978) Beast and Man: The Roots of Human Nature. New York: Meridian.

Miller, G. A. (1956) "The magic number seven, plus or minus two". Psychological Review 63:81-97.

Milner, P. M. (1958) "Sensory Transmission Mechanisms". Canadian Journal of Psychology 12:149-158.

Milner, P. M. (1970) Physiological Psychology. New York: Holt, Rinehart, and Winston.

Minsky, M. (1963) "Steps Toward Artificial Intelligence". In Feigenbaum, E. and Feldman, J. (Eds.) Computers and Thought. New York: McGraw-Hill.

Minsky, M. (1975) "A Framework for Representing Knowledge". In Winston, P. H. (Ed.) The Psychology of Computer Vision. New York: McGraw-Hill.

Minsky, M. (1979) "K-Lines: A Theory of Memory". A.I. Memo No. 516, Massachusetts Institute of Technology, Cambridge.

Minsky, M. and Papert, S. (1969) Perceptrons. Cambridge: MIT Press.

Neisser, U. (1967) Cognitive Psychology. New York: Appleton-Century-Crofts.

Neisser, U. (1976) Cognition and Reality. San Francisco: W. H. Freeman and Co.

Newell, A. (1973) "Artificial Intelligence and the Concept of Mind". In Shank, R. C. and Colby, K. M. (Eds.) Computer Models of Thought and Language, San Francisco: W. H. Freeman and Co.

Newell, A. and Simon, H. A. (1976) "Computer Science as Empirical Inquiry: Symbols and Search". Communications of the ACM 19:113-126.

Norman, D. (1976) Memory and Attention. Second edition. New York: Wiley.

Norman, D. A. (1981) "Twelve Issues for Cognitive Science". In Norman, D. A. (Ed.) Perspectives on Cognitive Science. Norwood, New Jersey: Ablex.

Olds, J. (1969) "The Central Nervous System and the Reinforcement of Behavior". American Psychologist 24:114-132.

Palmer, S. (1978) "Fundamental Aspects of Cognitive Representation". In Rosch, E. and Lloyd, B. (Eds.) Cognition and Categorization. Hillsdale, New Jersey: Erlbaum.

Plum, T. (1972) "Simulations of a Cell-Assembly Model". Ph.D. dissertation, University of Michigan.

Posner, M. and Keele, S. (1970) "Retention of Abstract Ideas". Journal of Experimental Psychology 83:304-308.

Pulliam, H. R. and Dunford, C. (1980) Programmed to Learn. New York: Columbia University Press.

Reed, S. K. (1972) "Pattern Recognition and Categorization". Cognitive Psychology 3:382-407.

Rosch, E. (1978) "Principles of Categorization". In Rosch, E. and Lloyd, B. (Eds.) Cognition and Categorization. Hillsdale, New Jersey: Erlbaum.

Rosch, E., Mervis, C., Gray, W., Johnson, D., and Boyes-Braem, P. (1976) "Basic Objects in Natural Categories". Cognitive Psychology 8:382-439.

Samuel, A. L. (1959) "Some studies in machine learning using the game of checkers". IBM Journal of Research and Development 3:210-229.

Scott, P. D. (1979) "Conditional Control Transfer Mechanisms in the Neocortex: 1. The Basic Model". Journal of Theoretical Biology 81:423-452.

Shaw, R. and Bransford, J. (1977) "Introduction: Psychological Approaches to the Problem of Knowledge". In Shaw, R. and Bransford, J. (Eds.) Perceiving, Acting, and Knowing: Toward an Ecological Psychology. Hillsdale, New Jersey: Erlbaum.

Shaw, R. and McIntyre, M. (1974) "Algoristic Foundations to Cognitive Psychology". In Weimer, W. B. and Palermo, D. S. (Eds.) Cognition and the Symbolic Processes. Hillsdale, New Jersey: Erlbaum.

Simon, H. A. (1969) The Sciences of the Artificial. Cambridge: MIT Press.

Simon, H. A. (1981) "Cognitive Science: The Newest Science of the Artificial". In Norman, D. A. (Ed.) Perspectives on Cognitive Science. Norwood, New Jersey: Ablex.

Smith, J. Maynard (1978) "Optimization Theory in Evolution". Annual Review of Ecology and Systematics 9:31-56.

Smith, S. F. (1980) "A Learning System Based on Genetic Adaptive Algorithms". Ph.D. dissertation, University of Pittsburgh.

Smith, T. R. and DeJong, K. A. (1981) "Genetic Algorithms Applied to the Calibration of an Information Driven Model of U.S. Migration Patterns". Proceedings of the Twelfth Annual Simulation and Modeling Conference, Pittsburgh.

Thomas, J. C. (1977) "Cognitive psychology from the perspective of wilderness survival". IBM Research Report No. RC6647, Yorktown Heights, New York.

Tinbergen, T. (1951) The Study of Instinct. New York: Oxford University Press.

Tolman, E. C. (1948) "Cognitive Maps in Rats and Men". Psychological Review 55:189-208.

Turvey, M. (1977) "Contrasting Orientations to the Theory of Visual Information Processing". Psychological Review 84:67-88.

von Neumann, J. (1958) The Computer and the Brain. New Haven: Yale University Press.

Walter, W. G. (1953) The Living Brain. London: Duckworth.

Welker, W. (1976) "Brain Evolution in Mammals: A Review of Concepts, Problems, and Methods". In Masterton, R. B. *et al.* (Eds.) Evolution of Brain and Behavior in Vertebrates. Hillsdale, New Jersey: Erlbaum.

Whitehead, B. A. (1977) "A Neural Network Model of Human Pattern Recognition". Ph.D. dissertation, University of Michigan.

Wickelgren, W. A. (1969) "Learned specification of concept neurons". Bulletin of Mathematical Biophysics 31:123-142.

Winograd, T. (1980) "Extended Inference Modes in Reasoning by Computer Systems". Artificial Intelligence 13:5-26.

Winograd, T. (1981) "What Does It Mean to Understand Language?" In Norman, D. A. (Ed.) Perspectives on Cognitive Science. Norwood, New Jersey: Ablex.

Winston, P. H. (1975) "Learning Structural Descriptions From Examples". In Winston, P. H. (Ed.) The Psychology of Computer Vision. New York: McGraw-Hill.

Woodworth, R. S. (1947) "Reinforcement of perception". American Journal of Psychology 60:119-124.

Zajonc, R. B. (1980) "Feeling and Thinking: Preferences need no inferences". American Psychologist 35:151-175.

Zeigler, B. P. (1976) Theory of Modeling and Simulation. New York: Wiley.