THE UNIVERSITY OF MICHIGAN
COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS
Department of Communication Sciences

Technical Report

THE BEHAVIOR OF ADAPTIVE SYSTEMS WHICH EMPLOY GENETIC AND CORRELATION ALGORITHMS

John D. Bagley

ORA PROJECT 01252

supported by:

DEPARTMENT OF HEALTH, EDUCATION, AND WELFARE
PUBLIC HEALTH SERVICE
NATIONAL INSTITUTES OF HEALTH
GRANT NO. GM-12236-04
BETHESDA, MARYLAND

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

December 1967

This report was also a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in The University of Michigan, 1967.

TABLE OF CONTENTS

TABLE OF CONTENTS (concluded)

LIST OF TABLES

LIST OF TABLES (concluded)

LIST OF FIGURES

ABSTRACT

A mathematician may be motivated to develop a theory purely from considerations of the theory itself, but a scientist formulates a theory with the intent that, under some natural interpretation, the theory will say something useful about some aspect of the observable world. A practitioner, on the other hand, is usually strongly motivated to solve a particular problem and less interested in the development of a theory. This thesis is intended to be a scientific study of some of those aspects of the real world which have been termed "adaptive", and attempts to develop some aspects of a theory which adequately mirrors and hopefully gives some insight into adaptive behavior.

One of the characteristics of the emerging field popularly called artificial intelligence (we prefer, and will use exclusively, the less controversial term adaptive systems) is the prevalence of the practitioners. Although this is to be expected and is indeed typical of any new area of inquiry and although much of the work being done has some value in itself, it does present difficulties both for the non-specialist who has a need to understand the significance of the ongoing efforts and for the theoretician who would like to be able to model them.

Accordingly, one of the first tasks of the scientist is that of systematization. To that end, we have developed a framework or paradigm which encompasses much of the work that has been accomplished in the field of adaptive systems. This paradigm effectively separates the learning algorithm from the heuristic aspects which lie in a meta-environment.

We then define a concept of meta-environmental depth which is intended to reflect the degree of interaction among the parameters of the meta-environment. We present an example of meta-environment which includes the game of Hexapawn and show how its depth can be made to depend upon the strategy chosen by a fixed opponent.

Next we describe a correlation adaptor which is a prototype and extension of adaptive algorithms which have appeared in the literature and develop expressions which can be used to determine the results of interaction of correlation adaptors of arbitrary level with meta-environments or arbitrary depth. These expressions are evaluated for a special case of simulated meta-environments and the mean number of trials required for their maximization under a variety of conditions is computed. These results indicate that the level of the correlation adaptor must be closely matched to the depth of the meta-environment and that it is of limited use for meta-environments of depth greater than three.

Next we formulate an alternative algorithm based upon the mechanism of natural genetic systems and demonstrate by means of simulation experiments that the genetic adaptor does well in both depth one and depth two meta-environments when compared to the correlation adaptor. The price paid for this versatility is quite small and, under certain conditions, the genetic adaptor is superior even to the matched correlation adaptor.

Finally we determine the effects of a limited range of variation of the control coefficients upon the optimizing behavior of the genetic adaptor.

In general, the results obtained in this thesis (particularly those pertaining to the genetic adaptor) have implications for two general areas — the theory and practice of problem solving and the theory of natural genetic systems.

1. INTRODUCTION

1.1 Motivation

A mathematician may be motivated to develop a theory purely from considerations of the theory itself, but a scientist formulates a theory with the intent that, under some natural interpretation, the theory will say something useful about some aspect of the observable world. A practitioner, on the other hand, is usually strongly motivated to solve a particular problem and less interested in the development of a theory. This thesis is intended to be a scientific study of some of those aspects of the real world which have been termed "adaptive," and will attempt to develop some aspects of a theory which adequately mirrors and hopefully gives some insight into adaptive behavior.

One of the characteristics of the emerging field popularly called [Fein, 1964]* artificial intelligence (we prefer, and will use exclusively, the less controversial term adaptive systems) is the prevalence of the practitioners. Although this is to be expected and is indeed typical of any new area of inquiry and although much of the work being done has some value in itself, it does present difficulties both for the non-specialist who has a need to understand the significance of the ongoing efforts and for the theoretician who would like to be able to model them.

Accordingly, one of the first tasks of the scientist is that of systematization. To that end, we have developed a framework or paradigm which encompasses much of the work that has been accomplished in the field of adaptive systems. Since this paradigm lies at the base of all of our succeeding work, we feel that it is appropriate that we begin by presenting a brief statement of our aims.

*References to the bibliography are indicated by square brackets which enclose the author's name and the year of publication.

We wish ultimately to be able to deal with "real world" problem environments which are by their very nature exceedingly difficult. In fact they are usually so difficult that any system that we propose to deal with them is almost certainly doomed to defeat unless it is able to take advantage of all of our a-priori knowledge of the environment and all of the intuition, insight, experience, heuristics, ingenuity, and downright dirty tricks that we can muster. On the other hand, it seems undesirable to have to begin afresh on each new problem with which we are faced. We would like to be able to learn something from our problem solving experience so that we might be able to formulate and prove (or at least demonstrate the feasibility of) some general statements concerning classes of adaptive systems and environments. In short, we would like to have a theory of adaptive systems in order to guide us in building systems so that our reliance on ad-hoc rules and empirical relations is minimized and so that we will have a reliable guide wherever and to whatever extent this is possible.

With these considerations in mind, we have formulated a paradigm with the hierarchical structure shown in Figure 1.1. The bottom layer, containing the box labelled "E" represents the real problem environment. The second layer which contains the boxes labelled "C", "$\overline{U}$" and "H" is intended to be the repository of all of the a priori and heuristic information that is at our disposal. The adaptor, "A", located in the third layer is the embodiment of the adaptive algorithm. It is constrained to operate in a meta-environment consisting of the real problem environment as well as those tools, contained in the second layer, which we have deemed to be the most effective vis-a-vis that environment. In the meta-environment, the adaptor's task is to maximize the estimated utility provided by the Utility Estimator, "$\overline{U}$". Since all of the specialized

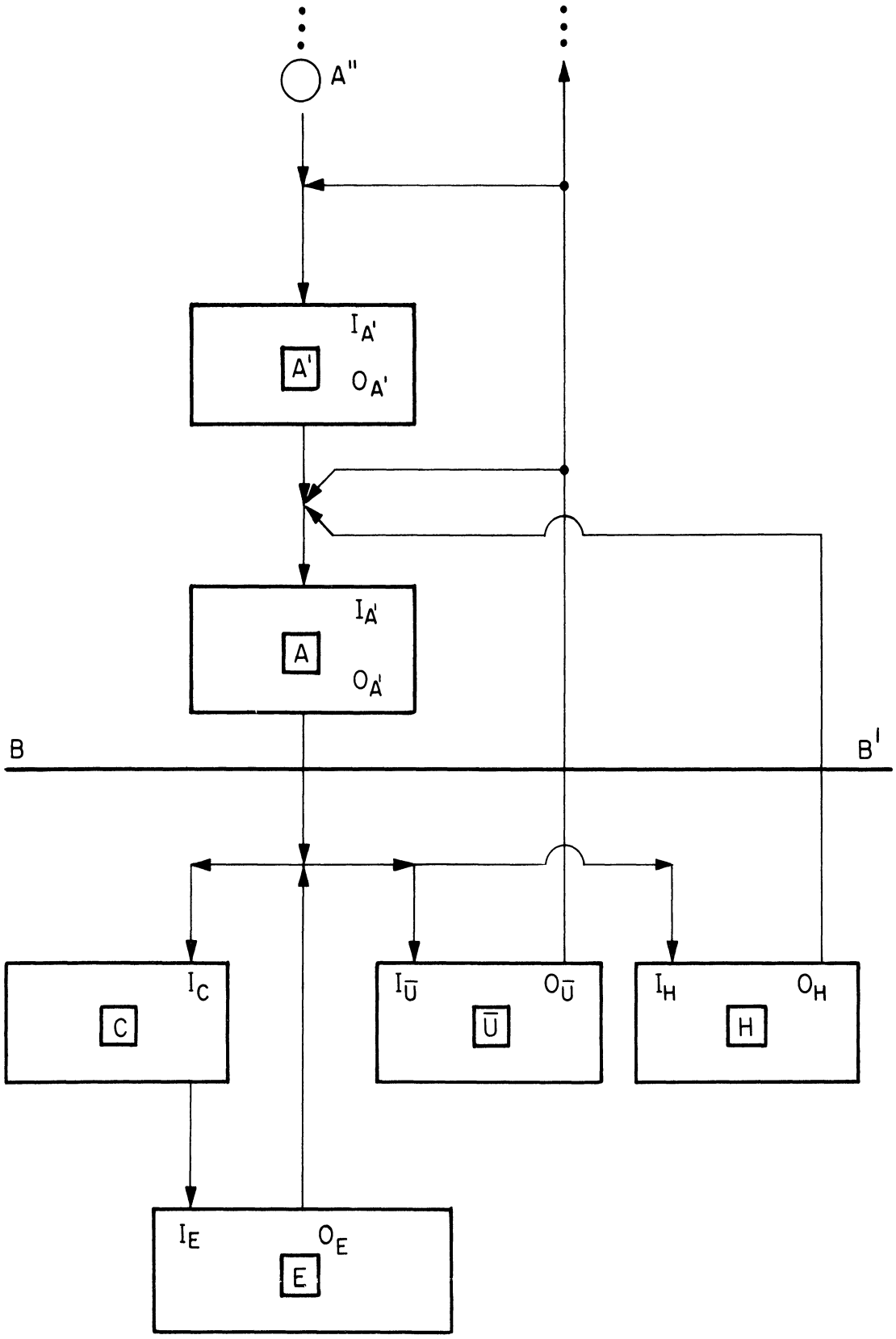

FIG. I.I STRUCTURE OF THE PARADIGM

ad hoc knowledge of the real environment is contained in the second layer, we can begin to deal with the remainder of the system, the adaptor, in more general terms and are able to consider its broader and more far-reaching aspects. Clearly, the performance of a system, any system, depends upon both the environment and the inventory of tools and methods which the system has at its disposal. Thus any attempt to compare two systems against the same real environment should strive to distill the essence, the adaptors, of the complete systems from the pre-programmed information and make the comparison on the basis of identical available tools.

The present paradigm is quite limited in terms of what we would ultimately like to be able to achieve. It is clearly possible to conceive of a much more general adaptive system — one which has the ability to, in effect, build and modify its own tools. Such a system would be much less dependent upon any information (or misinformation) that we are able to supply it initially and could be expected (if properly designed) to do well in a broad class of environments. Among other features, it would have the ability to find a best recoding of the information it receives from the environment and would be able (in our terms) to change the functioning of its own computer. We have chosen to concern ourselves with the more limited class of adaptive systems outlined here in the hope of being able to provide some insight into the design of practical system to solve real problems that arise in the near future. One way of looking at this work is to consider that we have concerned ourselves with the efficient use of heuristic methods. One might say that we are investigating what Newell and Simon [1964] refer to as simple learning techniques (although their simplicity is, for the most part, only an illusion). They look upon these techniques as the "final polishing to be applied to

any heuristic program." We agree with their general notion, but it appears to us that the selection of such techniques may be equally as important as the selection of the heuristics for any particular problem. Surely, at this stage of knowledge (or ignorance), we cannot afford to neglect either aspect.

Next, we must add a disclaimer. Although we have stated that we are ultimately concerned with the solution of real problems and we have attempted to show how the topics of this thesis are related to the solutions of such problems, the aim of this thesis is to investigate certain characteristics of adaptive algorithms. In order to achieve a measure of generality and wide range of application for the results, we will postulate both adaptors and meta-environments that will bear little detailed resemblance to the adaptors one would build or the specific meta-environments one would meet in practice. The reason for this is not hard to see. In seeking to deal with essentials, we have omitted consideration of many of the details of specific adaptors which enable them to deal with specific meta-environments. Likewise, we will postulate idealized meta-environments which mirror only certain aspects of real meta-environments. We are convinced that those aspects which we have chosen to consider here are significant to the real world so that the results we obtain can be applied with profit. We hope that we will have convinced the reader by the time we have concluded this work.

1.2 Preview

Chapter 2 contains the formal development of the aforementioned hierarchical paradigm and gives precise definitions of such concepts as adaptor, environment, meta-environment and the like. Each definition is followed by an informal description of its intended interpretation.

Chapter 3 proceeds to examine the details of three different examples of problem environments along with the adaptive systems constructed by skilled practitioners to deal with them. The Learning Machine of Friedberg [1958], the Checker Player of Samuel [1958], and various Adaptive Data Classifiers by several experimenters are fitted in detail to the paradigm and more importantly, we demonstrate how the paradigm is able to give important insight into these efforts.

Chapter 4 deals with meta-environments and with systematic methods with which to represent and classify them. The concept of meta-environmental depth is introduced, formalized, and illustrated by means of an example. A generation procedure for simulated meta-environments is given.

Chapter 5 contains a description of a particular class of adaptive algorithm which we have termed the correlation adaptor. This adaptor typifies many of the adaptive algorithms which have been propounded in the literature, but is considerably more general.

Chapter 6 examines the consequences of interaction between correlation adaptors of various levels and meta-environments of various depths. Analytical results are given for both the general case and for the specific case of the simulated meta-environments.

Chapter 7 describes a different class of adaptive algorithm — the genetic adaptor. This adaptor, as the name implies, is inspired by biological mechanisms. The emphasis is, however, on the specification of an adaptive algorithm rather than on a strict imitation of the biological situation although the natural process is modeled whenever feasible. The result should be of interest both to the practitioner and to the theoretical biologist.

Chapter 8 contains the results of simulation experiments in which the various genetic adaptors were immersed in simulated meta-environments

of depths one and two. In addition, the results of limited experimentation on the control coefficients of the adaptor are given and interpreted.

Chapter 9 concludes this thesis with a summary of its contents and a look toward the future.

2. A PARADIGM FOR ADAPTIVE SYSTEMS

2.1 Plan

Our plan is to introduce, in this chapter, the bulk of the formalism that we shall need in order to give the discussion a concrete basis and to pave the way for the main parts of this thesis.

The next section provides a brief overview of the paradigm. Following that, the formal definitions are given. Each is followed by an informal interpretation as an aid to the reader's intuition.

2.2 Orientation

Figure 1.1 illustrates the Adaptive Universe to be described in this section. We shall pass lightly over that figure proceeding from the bottom to the top.

E is intended as the environment of the adaptive system. As such it contains both the description of the problem and a payoff or utility function. It is a finite automaton of the Moore type with an associated utility function u_E.

C, the computer, is a function and is the agent which acts directly upon the environment. It receives inputs both from the environment and from the adaptor. $\overline{U}$ is the utility estimator which supplies an estimate of the action which the computer performs on the environment. It is a finite automaton of the Mealy type. H, the hint computer, is a function. Its purpose is to supplement the computer by supplying additional information to the adaptor.

A is the adaptor whose output, O_A, directs the activity of C and whose input includes the estimate of success provided by $\overline{U}$. A is a Mealy type finite automaton as are A', A'', ... etc. which perform similar functions on a higher level.

Dynamically, the interaction of the units of the universe proceeds as follows. C consults the environmental output (O_E) as well as the parameters supplied to it by A, (O_A) and on this basis generates a move (O_C). This move causes a change in E which is reflected in its output (O_E). The change is evaluated by $\overline{U}$ and the evaluation $O_{\overline{U}}$ together with the additional information generated by H, (O_H) is sent to A. A, then readjusts the parameters of C, (O_A) in such a way that, hopefully, C's future moves become better as measured by $\overline{U}$. C then generates another move and the process continues.

2.3 Definitions

Before we begin the formal definitions of the units of the adaptive universe, we will find it useful to introduce a number of general definitions and notational conventions.

Definition: A Moore [1956] Model of a finite automaton is a sextuple:

$$\mathcal{A}_\alpha = \langle I_\alpha, Q_\alpha, O_\alpha, Q_{\alpha_0}, q_\alpha, o_\alpha \rangle$$

where I_α is a finite nonempty set (Input symbols)

Q_α is a finite nonempty set (Internal states)

O_α is a finite nonempty set (Output symbols)

$Q_{\alpha_0} \in Q_\alpha$ (initial state)

q_α is a function: $q_\alpha : Q_\alpha \times I_\alpha \to Q_\alpha$ (transition function)

o_α is a function: $o_\alpha : Q_\alpha \to O_\alpha$ (output function)

Definition: A Mealy [1955] Model of a finite automaton $\mathcal{A}'_\alpha$ is the same as the Moore Model except that the output function o'_α has as its domain $Q'_\alpha \times I'_\alpha$ and has as its range O'_α and, in addition, an initial output symbol $O_{\alpha_0} \in O_\alpha$ is specified.

Notational Convention: We shall define the elements of the adaptive universe in terms of the definitions given above. Subscripts will be

used to denote specific units (e.g. Q_A are the states of the Adaptor).

Definition: An Adaptive Universe Ω is an n+5 - tuple:

where $\Omega = \langle E, C, \overline{U}, H, A, A', \ldots, A^{(n)} \rangle$

Definition: E (the environment) is a triple:

$$E = \langle a_E, U_E, u_E \rangle$$

where a_E is a Moore model finite automaton and;

U_E is a finite subset of positive integers (utility values)

u_E is a function; $u_E : Q_E \times I_E \rightarrow U_E$ (Utility function)

subject to the following considerations:

We allow q_E to be a partial function because every move might not be applicable to every state. There may be constraints imposed by the "rules of the game". We will confine our attention to computers which generate only legal moves, a restriction which makes it unnecessary to include Q_E in the domain of q_E. It will become evident when the equations which govern the interconnections are written that I_E is itself a function of Q_E. Under interpretation, this means that the moves or transformations supplied by the computer have been computed on the basis of, and are applicable to, the present environmental state so that the transition function need only consider changes. The dependence on Q_E is included explicitly here only in the interest of uniformity of exposition.

O_E has the same cardinality as S_E and is restricted to be 1:1 onto in order that each state be given an unique description. In all of the finite environments which we consider u_E is, in principle, computable although there may be practical difficulties. See, for example, the comments on the Checker Player in Section 3.2.

E is the environment — the real problem that we wish to solve. The input to E, I_E, is a move or transformation. E's outputs are of two types:

(1) a description of the internal state of E, (O_E) and (2) an indication of the utility of the move with respect to the current state, (U_E). There are three points to be noted about this utility: (1) It is a true utility and gives an absolutely correct rating of move-state pairs; (2) Because of this, it may be difficult or inconvenient to compute so that it is not in general available to the rest of the formal system although it may be available to the theorist or experimenter who is supervising the situation; (3) It is a local utility in that it evaluates single move-state pairs as opposed to a global utility which ranks complete strategies or sequences of moves.

It is often the case that the local utility is used to compute a global utility which is in turn used to judge the performance of the system. Thus, for example, one might wish to discover how well the system does in generating sequences of moves and in particular those sequences which enable the system to gain the maximum amount of utility in the least number of moves. It is even possible for one to go a step higher and attempt to determine how well the system does vis-a-vis entire sets or sequences of environmental problems (see for example Friedberg's [1959] problem number 14). We shall concentrate our efforts on evaluation of systems with respect to the number of trials which they require to converge to a point of maximum utility.

Definition: C (the computer) is a triple; $C = \langle I_C, O_C, o_C \rangle$

where o_C is a function; $o_C : I_C \rightarrow O_C$.

C generates moves, (O_C), on the basis of the information it receives from the environment and the adaptor. The discussion following the definition of a strategy in the following section should serve to further clarify the nature of the Computer.

Definition: $\overline{U}$ (the utility estimator) is a Mealy type automaton,

$$\overline{U} = \mathcal{A}'_{\overline{U}}$$

and

$$O_{\overline{U}} \subseteq U_E$$

$\overline{U}$ is an estimator of the utility of the action which the computer performs on the environment. This estimate ($O_{\overline{U}}$) is fed back to the adaptor, A, and is the only indication of the worth of its generated parameter values that A receives. The inputs to $\overline{U}$, ($I_{\overline{U}}$) are derived from the outputs of A and the state description of E and thus, $\overline{U}$ might conceivably be the same as U. Since it has the same inputs as C, U could imitate the computation of C, determine the move, and use this information together with the state description to compute the true utility.

The task of the Adaptor is to maximize the value of the estimated utility (which may or may not be a good estimate of the true utility). Note that the restriction to adaptors which maximize $\overline{U}$ really involves no loss of generality since if, for example, the environment requires the minimization of some quantity, appropriate adjustments may always be made in the definition of $\overline{U}$. Thus the utility estimator provides a quantity for the adaptor to maximize even in those cases where the true utility may be impractical to compute. Ultimately, however, the performance of the system will be judged by some deus ex machina on the basis of the true utility so that no small part of the difficulty in setting a real problem into this context involves the judicious choice of the utility estimator.

Definition: A (the adaptor) is a Mealy type automaton:

$$A = \mathcal{A}'_A$$

and

$$O_A = V^P = \{O_{A_1}, O_{A_2}, \ldots, O_{A_{VP}}\}$$ (parameter value

vectors often abbreviated PVV)

where:

P is a finite nonempty set; $P = \{P_1, P_2, \ldots, P_p\}$ (parameters) and associated with each $P_i \in P$, there is a finite nonempty set; $V^i = \{V^i_1, V^i_2, \ldots, V^i_v\}$ (parameter values)*

V is a finite set; $V = \bigcup_{i=1}^{p} V^i$ and each $O_{A_j} \in O_A$ is a function; $O_{A_j} : P \to V$ such that

$$O_{A_j}(P_i) \in V^i \text{ for all } i \quad **$$

A is the adaptor whose output (O_A) is a parameter value vector which directs the activity of C and whose input includes the indication of success provided by the utility estimator $\overline{U}$. The goal of A is very elementary — it must maximize the value of feedback given to it by $\overline{U}$ by adjusting the parameter values. A, in turn, may depend for its operation upon the settings of various parameters which are controlled by higher parameter value vector generator A' as well as by hints given by the hint generator, H.

Definition: A' (the secondary adaptor) is a Mealy type automaton,

---

* We have made the assumption that each parameter may assume the same number, v, of values. This assumption holds in each of the examples given in the following chapter and really involves no loss of generality since duplicate or dummy values may be added to value sets.

** Keeping this restriction in mind enables us to simplify the notation considerably by eliminating redundant superscripts. We shall normally do this, writing for example;

$O_{A_j}(P_i) = V_K$ where the obvious meaning is;

$O_{A_j}(P_i) = V^i_k$.

It will often be the case that parameters not only have the same number of values but they also have the same values. That is:

$V = V^i$ for any i $(1 \leq i \leq p)$.

In these cases, the simplified notation is especially appropriate.

$$A' = \alpha_{A'}$$

The definitions and interpretations are analogous to those of A above.

A' adjusts the parameters of A in an attempt to optimize $\overline{U}$, and in fact there may be a complete hierarchy of similar devices A", A"', etc. One would expect that the parameter adjusters which occupy higher positions in the hierarchy would be more deliberate in their action. That is, the need for higher level adjustments becomes apparent only on a much longer time scale (in terms of the number of environmental interactions), so that the higher level parameters tend to change less frequently. We will usually consider systems in which the time scale is such that the parameters of A remain fixed throughout the period of interest.

A", A"', etc. may be similarly defined and interpreted.

Definition: H (the hint computer) is a triple:

$$H = \langle I_H, O_H, o_H \rangle$$

where o_H is a function; $o_H : I_H \rightarrow O_H$

The hint computer has been included to reflect a common situation in which hints, (O_H) are supplied to the adaptor by a device which has access to both the computer and the environment (here through I_H). Hints may take a great variety of forms and may indicate, for example, which of the parameters were actually used by the computer or which had values that contributed positively to the result, or they may suggest a direction or magnitude of change for certain parameter values. Because of this diversity, the role of hints will be considered primarily in a qualitative manner in the discussion of previous work in Adaptive Systems. The consideration of hints is but one of the many areas in which additional work remains to be done.

2.4 Recursive Connection Equations

The connection equations given in this section may be deduced immediately from the preceeding definitions and Figure 1.1. We have included them explicitly here primarily in the interest of completeness.

We have taken the liberty of deliberately introducing a bit of ambiguity at this point in order to simplify the notation. Let T be a set of ordering parameters (time if you will) such that $T = (0, 1, 2, \ldots, t-1, t, t+1, \ldots)$ — a countable set of non-negative integers. Then corresponding to each set Q defined in the previous section, we should define a function $\mathcal{Q}: T \to Q$. Instead, we shall use the symbol Q ambiguously to mean both the function ($\mathcal{Q}$) whose domain is T and the set (Q) which is the range of that function. Thus, for example, $Q(t+1) = q(Q(t), I(t))$ will be written, when strictly correct usage would demand $\mathcal{Q}(t+1) = q(\mathcal{Q}(t), \mathcal{I}(t))$ where $\mathcal{I}(t)$ has the obvious meaning. This convention is straightforward and fairly common in the literature so that its use here should cause no confusion.

Note here that Figure 1.1 as drawn continues indefinitely up off of the page. In order to lend concreteness to this study and to tackle a more realistic situation, we shall assume that $O_{A'}(t)$ is known for all $t \geq 0$. This has the effect of concentrating our attention to the layers of the diagram which are below A'.

Initial Conditions:

$O_{A'}(t)$ is known for all $t \geq 0$

$Q_A(0) = Q_{A_0}$

$Q_{\overline{U}}(0) = Q_{\overline{U}_0}$

$Q_E(0) = Q_{E_0}$

$O_A(0) = O_{A_0}$

$$O_{\overline{U}}(0) = O_{\overline{U}_0}$$

Wiring Equations: $t \geq 0$

$$I_C(t) = I_{\overline{U}}(t) = I_H(t) = \langle O_A(t), O_E(t) \rangle$$

$$I_E(t)^* = O_C(t)$$

$$I_A(t) = \langle O_{A'}(t), O_{\overline{U}}(t), O_H(t) \rangle$$

Static Equations: $t \geq 0$

$$O_C(t) = o_C(I_C(t))$$

$$O_H(t) = o_H(I_H(t))$$

$$O_E(t) = o_E(Q_E(t))$$

Dynamic Equations: $t \geq 0$

$$Q_A(t+1) = q_A(Q_A(t), I_A(t))$$

$$O_A(t+1) = o_A(Q_A(t), I_A(t))$$

$$Q_{\overline{U}}(t+1) = q_{\overline{U}}(Q_{\overline{U}}(t), I_{\overline{U}}(t))$$

$$O_{\overline{U}}(t+1) = o_{\overline{U}}(Q_{\overline{U}}(t), I_{\overline{U}}(t))$$

$$Q_E(t+1) = q_E(Q_E(t), I_E(t))$$

Under interpretation, we may think of C as a digital computer whose program is supplied by A, (O_A) and whose data is taken from E, (O_E). C's computation (O_C) is fed to the environment causing a change of state. This change is reflected in the environmental output (O_E) and is evaluated by the utility estimator and the hint generator in light of the current program. The evaluation ($O_{\overline{U}}$) and hint (O_H) as well as the output of

---

* As mentioned in Section 2.3, I_E is in reality a function of Q_E — a fact which can easily be made apparent by direct substitution in the connection equations. $I_E(t) = o_C(O_A(t), o_E(Q_E(t))$

the secondary adaptor ($O_{A'}$) are digested by the adaptor and used as a basis upon which to change C's behavior.

2.5 Composite Definitions

The following definitions are composed from the preceding basic definitions and have been formulated in order to enable us to refer to various concepts which have strong intuitive significance and which can be expressed as combinations of the basic units.

Definition: An admissible strategy, s, is a couple; $s = \langle C, O_{A_i} \rangle$

where: C is a computer and

$O_{A_i} \in O_A$ is a parameter value vector.

Intuitively, we demand that a fixed strategy determine a unique response (move) for each environmental state. An admissible strategy as defined above generates, for each environmental output O_{E_j}, the response $I_{E_j} = {}^{o}C(O_{A_i}, O_{E_j})$. This fixing of a strategy is accomplished by, in effect, opening the feedback loop from the Adaptor. This concept enables us to formulate an operational test to identify, in a concrete situation, the entity which is performing the function of the computer. In short, the computer is that device which, when given a fixed parameter value vector (e.g., O_{A_i}) will compute a move (e.g., I_{E_j}) for each environmental state description (e.g., O_{E_j}). In general, a computer with a fixed PVV is capable of responding to the environment — of playing the game — although the response may not be very effective (as measured by $\overline{U}$).

Definition: A complete system, $\overline{A}$ is a quadruple; $\overline{A} = \langle A, C, \overline{U}, H \rangle$

where: A is an Adaptor

C is a Computor

$\overline{U}$ is a utility estimator

and H is a hint generator

Definition: A meta-environment, ME, a quadruple; ME = $\langle E, C, \overline{U}, H \rangle$

where: E is an environment

C is a computer

$\overline{U}$ is a utility estimator

and H is a hint generator

The interpretation of these terms should be quite clear. The Complete System is just that part of the universe (exclusive of the higher order Adaptors) which acts upon the true environment. Similarly, the meta-environment is that portion of the universe which is acted upon by the Adaptor.

The next two definitions deal with matters that are a little more obtuse and so appear to be given in a slightly less precise manner.

Definition: The extent of an environment, designated e(E), is an integer and is the total number of distinct legal strategies which are applicable to the environment E. Proceeding formally we note that if $[I_E]^*$ is the number of moves and $[O_E]$ is the number of states; then the number of strategies is just:

$e(E) = [I_E][O_E]$ — the number of functions from the set of states to the set of moves. Unfortunately this analysis overlooks the fact that for most environments not every move is legal in every state. In such a case, $[I_E][O_E]$ certainly gives an upper bound for the extent, but often estimates for the average number of attainable states or the average number of legal moves from each state can be made so that a closer bound may be computed.

The next definition is similar, but deals with the meta-environment.

Definition: The extent of the meta-environment, designated by e(ME), is an integer and is the total number of apparently distinct admissible

---

*We use the notation [S] where S is a set to denote the cardinality of S.

strategies which may be employed by the computer of a meta-environment ME. Since the admissible strategies of a computer are determined by the parameter value vector, the extent of a meta-environment is just the total number of distinct parameter value vectors — i.e., $e(ME) = v^p$.

One very crude estimate of the efficacy of the Computer can be obtained by comparing the extent of the meta-environment with the extent of the environment. This estimate is crude and reflects only one aspect of the computer — namely its ability to reduce (in most cases of interest) the search by failing to distinguish among situations which are to be handled as equivalent and by eliminating portions of the environmental space.

Thus one quality we would hope to find in an efficient computer is that it greatly reduces the space to be searched so that the extent of the meta-environment is much less than the extent of the true environment. But this can be misleading because it's the way that utility is distributed over an environment as well as the size of the space that renders it hostile. Therefore, we expect that an efficient computer will restrict the search by lumping together those points which have the same or similar utility and by restricting the search to fruitful areas.

The amount of preprogrammed information built into the Computer is variable and is highly dependent upon the knowledge and ingenuity of its constructor (the experimentor). For example, if in the judgement of the constructor, there is not enough evidence upon which to formulate a scheme to lump or restrict the environment, that decision may be left to the Adaptor. However, each such decision which is "passed upward" increases the extent of the meta-environment and thus makes the task of the Adaptor more difficult.

Perhaps we should reemphasize that the reason for introducing formal definitions and connection equations at this point is to present

a general but concrete formulation of the universe we intend to investigate. In later chapters, we will use this formulation as the basis for various specializing and simplifying assumptions under which the results will be obtained.

2.6 Guidepost

In this chapter, we have developed a formal paradigm for adaptive systems. In the next chapter, we shall examine in detail three examples of adaptive systems that have appeared in the literature. We shall show how these fit into the general paradigm and how it is able to shed additional light on the reported results.

3. SOME APPLICATIONS AND IMPLICATIONS OF THE PARADIGM

3.1 The Learning Machine of Friedberg

We have been particularly influenced and intrigued by the work of Friedberg [1958, 1959], so that it is fitting that we begin this chapter with the consideration of that work. The following is not intended to be a review or complete exposition (for such, the interested reader is referred to Andrews [1962], Minsky [1961], or Solomonoff [1966]) but is simplified and abstracted with the emphasis on the points that are pertinent to the paradigm.

Friedberg's universe consists of three interconnected units ("black boxes" if you like), the first of which is a hypothetical stored program digital computer known as the "Slave." Programs for the Slave are written by the second unit, the "Learner," and the two are tied together by the third unit, the "Teacher." The Teacher poses a problem to the Learner which responds by writing a program for the Slave which is then executed. The Teacher examines the results and reports to the Learner whether or not the Slave has computed the correct solution to the problem. Typically, the problem involves the computation of a simple Boolean function of a few binary variables. The variables as well as the value computed are stored in a memory bank to which both the Teacher and Slave have access — the Teacher to set initial values of the variables and to check the result, the Slave to make the transformations. The Teacher generates values for the binary variables either at random or in sequence and reports a "success" to the Learner whenever the Slave computers the correct value of the function for the given arguments. The Learner is designed so that it makes a change in the Slave's program only when a failure has been reported. Its ultimate goal is to generate a program which will enable the Slave to

compute the correct value for all combinations of argument values (i.e., to compute the desired function).

Friedberg's environment is the underlying bit-manipulation problem contained in the Teacher. Environmental States are the states of the memory bank and a move is the transformation to the memory which is made by an entire execution of the Slave's program. The true local utility is awarded on the basis of a success or failure of the transformation for a single set of arguments. Actually, Friedberg [1958] himself computed a global utility when he determined that the program given in Chart 1 of that paper would enable the Slave to perform the correct computation for the subset of environmental states attainable from the given state. The computer is the Slave and its inputs are the program and the description of its memory bank. The utility estimator, contained in the Teacher, gives a true local utility and in this instance, $Q_{\overline{U}}$ has only one element. A good case can be made for the notion that Friedberg actually had a global utility in mind since he states that he is concerned with the fraction of trials (i.e., environmental states) for which the program-Slave combination produces the correct transformation in terms of the local utility estimator. This fraction is a global utility. The fact that the Learner took no action until a trial was judged to be a failure can be interpreted to mean that its action is based upon an estimate of global utility. This estimate is $n/n+1 = \overline{U}_g$ where n is the number of successful trials before a failure. If g is the fraction of trials for which a given program will compute the correct answer, and if $\langle \overline{U}_g \rangle$ is the expected value of the estimated global utility, then it can be shown that $\langle \overline{U}_g \rangle < g$. That is, $\overline{U}_g$ is a pessimistic estimator of the global utility g.

The "Teddy" system of the follow-up paper [Friedberg et al., 1959]

employs a somewhat different arrangement in that the Learner is informed of participating instructions (those instructions actually executed or referred to). This is one form of hint.

The Adaptor is contained in the "success number" mechanism located in the Learner. The programs are the parameter value vectors (PVV's), where the instruction set corresponds to the parameter values and the locations to the parameters (i.e., $v = 2^{14}$ and $p = 64$). The adjustments which could have been made by an A' unit were evidently set initially by the experimenter and left unchanged throughout the course of the experiment. Examples of these are the scaling parameters S_m, S_i, and r, and the rate of introduction of new instructions.

The complete system consists of the Learner, the Slave and that part of the Teacher that signals success or failure while the meta-environment consists of the Slave and all of the Teacher except that part which is concerned with executive functions.

The highlights of this identification of the elements of the paradigm with the elements of the Learning Machine along with similar identifications for the other papers considered may be found in Table 3.1.1.

Because of the way that a move is defined, the computation of the extent of the environment is especially simple. There are no restrictions so that a single move may take E from any state to any other state. Thus;

$$e(E) = [O_E]^{[O_E]} = (2^{64})^{2^{64}} = 2^{2^{70}} > 10^{10^{20}}$$

The computation of the extent of the meta-environment is equally straightforward:

$$e(ME) = v^p = (2^{14})^{64} < 2^{2^{10}} < 10^{270}$$

The preceding computations as well as equivalent computations from the next two sections are summarized in Table 3.1.2.

OPUS (Author)	ENVIRONMENT	COMPUTER	ADAPTOR	$\overline{U}$	PVV
The Learning Machine (Friedberg et al)	Bit Manipulation Problems	Slave	Learner	Success or Failure	Program
The Checker Player (Samuel)	The Game of Checkers	Look-Ahead Computer	Coefficient Value Generator	Delta	Set of Polynomial Term Coefficients
Various Adaptor Data Classifiers (Widrow et al)	Pattern Classification Problems	Threshold Logic Network	Weight Changing Algorithm	Correct or Incorrect Sometimes Degree of Error	Set of Network Weights and Thresholds

Table 3.1.1 Summary of Paradigm Applications

OPUS	LOWER BOUND OF e(E)	UPPER BOUND OF e(ME)
Learning Machine	$2^{2^{70}}$ or $10^{10^{20}}$	10^{270}
Checker Player	$2^{2^{55}}$ or $10^{10^{16}}$	10^{36}
Adaptive Data Classifier	$2^{2^{25}}$ or $10^{10^{10}}$	10^{111}

Table 3.1.2 Extent Reduction of Various Computers

It should be clear at this point that the Slave does indeed reduce the search space drastically. But if we look carefully at the nature of experiment no. 1, we find that only a single initial bit and a single final bit of the memory were considered by the Teacher in assigning utility. This has the effect of partitioning the whole environment space (of functions) into four equal subsets such that within each subset, each function is equivalent with respect to the two bits considered and so is assigned the same utility. Therefore, 25% of the environment points have a maximum utility value. An equivalent measure on the meta-environment is the utility density or in this case, the fraction of programs that were perfect (stopped within the time limit and computed the correct function). This fraction is almost impossible to estimate analytically although lower bounds are easy to find (since in a few minutes, one can easily write

schemata for literally billions of perfect programs). It is not difficult to obtain a statistical estimate of the utility density by simulation methods — i.e., by taking a uniform random sample using a Homer type of Learner. Although Friedberg did not record data which would enable this estimate to be made it is evident that the density of perfect programs is less than 1/4 by a factor of at least a thousand. Thus, if Friedberg had really been interested in a solution to the bit-manipulation problem (which he wasn't), he made a very unfortunate choice of computer. It reduced the extent of the environment, but it reduced the utility density by a considerable amount as well. This is an example of a poor choice of a computer which immensely complicates the solution of a basically simple problem. The net result was to generate an extremely difficult problem (in the meta-environment) for the Learner.

It should be noted that the changes introduced in the sequel paper [Friedberg et al, 1959] had the effect of both reducing the extent of the meta-environment and, more importantly, increasing its utility density. In particular, "priming" reduced the extent of the meta-environment only slightly (from 2^{896} points to 2^{893} points) but in doing so it increased the utility density by eliminating mostly points of low utility. The problem of hostile environments or meta-environments exists almost independently of the problem of environments of large extent (the problem of immensity, as Andrews [1962] terms it) for even though an environment may be immense, it may not necessarily be hostile if a system can easily obtain sufficient information from it. Thus, for example, having read Friedberg's paper, one can easily write a perfect program for the Slave simply because he has received sufficient information to do so. Even an environment with a very low utility density may not be hostile if it is easy to avoid

the bad points and find the good ones, i.e., if the regularities of the environment become apparent. We shall discuss environmental hostility further in Chapter 4.

Having obtained some insight into the hostility of the meta-environment* we would now like to examine the effectiveness of Friedberg's Learner in coping with it.

The Learner stores both the current program and an alternate program (i.e., 2 instructions out of a possible 2^{14} for each location) as well as their associated success numbers. Even if we make the optimistic and obviously false assumption that it is possible to store in these 128 success numbers all that can be known (with respect to utility) about all of the programs that can be formed by making all possible interchanges of instructions between the current program and its alternate, this would mean that we would have knowledge of $2^{64} < 10^{20}$ programs out of the total of $(2^{14})^{64} > 10^{265}$ possible programs. If we adopt the Learner's rate of introduction of new instructions (one after each 64 failures) and if we assume that all trials are failures and that no duplicate instructions are ever introduced and that we can obtain the same type of perfect-knowledge of all the programs which can be formed by interchanges, then we still will have "explored" fewer than 10^{103} points of the meta-environment after a run of 150,000 trials. (It took 150,000 trials for the Learner to find its first perfect program.) Thus the point should be clear that even if we make the most optimistic and unrealistic assumptions (which are really assumptions about the astuteness of the Learner and the regularities of the meta-environment), it is possible for the Learner to

*Minsky's [1961] apparent failure to appreciate the hostility of the meta-environment has led him to be, perhaps, too harsh in his judgement of Friedberg's Learner.

investigate only an insignificant fraction of the space even in 150,000 trials. Therefore, we should not be surprised at the apparent lack of success of the Learner, for it is faced with a truly immense meta-environment which it explores very cautiously. These considerations of the area which the Learner can know are apt to be misleading unless we bear in mind that the Learner's objective is to find a maximum in the meta-environment and not to identify it or to explore it thoroughly.

One of the obvious faults of the success number mechanism, and a fault of which Friedberg was aware, is the fact that no provision is made to keep track of the correlations associated with strings of instructions (subroutines, in effect). He tried to avoid the necessity for this by his selection of an instruction set, but a close study of any of the published programs will show that he was not outstandingly successful. The possibility of assigning success numbers to pairs of instructions was mentioned but was not implemented because of the practical difficulties involved in his immense meta-environment.

Apparently, from the structure of the Learner which made use of active and inactive program strings, there was some attempt to mimic the recessive-dominance structure of a genetic system. However, no provision was made to imitate the recombination aspects of such systems for anything except single instructions. One of the objects of this thesis is to examine in detail two alternative algorithms which are designed to be effective in meta-environments where utility may be profitably associated with strings or sets of parameters rather than with single parameters.

3.2 The Checker Player of Samuel

We shall assume that the reader has a basic familiarity with the original paper of Samuel [1959] and shall begin by associating the elements

of Samuel's universe with the elements of our paradigm.

Loosely speaking, Samuel's environment is the game of checkers played against a well defined opponent. Board positions correspond to environmental states and moves are the environmental inputs. The state transition function plays the role of a fixed well defined opponent. The output function presents a description of the board position (state) to the computer.

The game of checkers may be represented by a finite* directed tree whose nodes correspond to board positions and whose edges correspond to moves so that the edges radiating from a node represent the legal moves from the board position associated with that node. Utility is assigned to terminal board positions on the basis of a win, a loss, or a draw of the game and starting from those terminal assignments, assignments may be made to each of the intermediate nodes on a minimax basis. (We may even distinguish between states which "guarantee" the same result by assigning a higher utility to those states which assume a quicker win or a slower loss). Thus, in principle a utility may be assigned to each state-move pair so that a true local utility is computable.

Here, as with the Learning Machines, the experimenter usually judges the system by computing a global utility — the success of an adaptor over a succession of moves or a complete game. We can even conceive of a more general global utility which measures a system's success versus some set or sequence of environmental opponents. Actually the ultimate goal would be the development of a strategy — a mapping from board positions (environmental states) to moves that guarantee a win (or a draw when a win is not

*As Samuel did, we shall restrict our attention to games which are restricted, by fiat, to a finite number of moves. Thus, a draw will be one of the possible outcomes.

possible) against every opponent. Since Checkers, as we have defined it, is a two person, finite game with complete information, we know that such an optimum, minimax, strategy exists. A less ambitious but certainly acceptable goal would be the development of a strategy that does well against good human players but perhaps poorly against highly unorthodox opponents.

Needless to say, such a computation is impractical since, as Samuel [1959] noted if moves were explored at a rate of three per nanosecond, it would take 10^{21} centuries to explore the complete game tree.

The computer employes a look-ahead procedure and a set of subroutines which assign values to board positions. These subroutines are chosen to rate board positions with respect to such factors as center control, advancement, mobility and the like. There is a weight associated with each of the subroutine terms and the sum of the weighted terms (the score of the evaluation polynomial) is used to select moves by exploring the move tree a few moves ahead, computing polynomial scores and minimaxing back to the present position. In short, given a board position and a set of weights, the computer selects a move on the basis of the highest minimax score. It is evident that with a fixed set of weights, the computer will play a complete game*: The weight vector is our parameter value vector and it is the space of weight vectors which forms the meta-environment.

The measure of performance (the utility estimate) is given by a factor which Samuel calls "delta." Delta, in fact, measures something apparently quite different from the utility of move-state pairs — it measures the stability of the polynomial score as the game progresses. The justification for the use of delta as an estimate of utility stems

---

*This is an application of the operational test for the computer which we gave in the previous chapter.

from the fact that the further the game is played, the more obvious the outcome becomes, so that the polynomial score can be considered to be a more accurate reflection of the true utility as the game progresses. The score of a perfect polynomial (one which assigned values to move-state pairs which agreed with the true utility function so that it is a minimax predictor) would remain constant throughout the entire game* (i.e., it would give the same score at each node in the path it selected through the game tree so that a polynomial score at any point gives the effect of a look ahead to the end of the game). Thus a large value of delta is undesirable and a small value desirable so that the goal of the adaptor is to minimize the value of delta. The forced inclusion of a piece advantage term with a fixed positive weight helps to assure that the polynomial does measure factors which are relevant to the true utility. We have made $\overline{U}$ a finite automaton rather than a function because of the necessity for it to "remember" the previous polynomial score. We should like to emphasize that the value of delta is the only indication of utility given to the adaptor. There is no special reward given even when the computer has won a game.

Hints indicate to the adaptor which terms of the scoring polynomial (parameters) were actually used (i.e., pertain to the present environmental state) and whether their weights (the parameter values) should have been increased or decreased in order to reduce the value of delta.

The Checker Player's Adaptor is the embodiment of the polynomial modification algorithm. To quote Samuel [1959, p. 219] "A record is kept of the correlation existing between the signs of the individual term contributions in the initial scoring polynomial and the sign of delta. After

---

*This assumes an opponent who always makes the best possible move.

each play, an adjustment is made in the values of the correlation coefficients due account being taken of the number of times that each particular term has been used and has had a non-zero value." The correlations are then used to select weights (parameter values) for the terms of the scoring polynomial. The correlation record is preserved in the states of the adaptor and, to recapitulate, changes in state are dictated by the value of delta (estimated utility) and an indication of the relevant terms and their contributions to the polynomial (hints).

As was the case with the Learning Machine, the functions of A' were assumed by the experimenter when he adjusted the settings of various parameters. Samuel discusses a number of the changes he was forced to make after an initial series of tests in order to eliminate several sources of erratic behavior. Typical parameters so adjusted were the span of remembered moves over which delta was computed and the rate of introducing fresh terms.

As in our analysis of the Learning Machine, we will find it profitable to contrast the extent of the environment of checkers with the extent of the meta-environment presented to Samuel's adaptor. Unfortunately, the task of estimating the extent of the environment (i.e., the number of strategies for the game of checkers) is not simple, however we shall make two rather crude estimates.

Recall that there are 32 squares of the board used in the game of checkers and that a square may be occupied by either a white piece, a white king, a black piece, a black king, or be empty. Thus there are 5^{32} conceivable board positions many of which could not occur as a result of a legal play of the game of checkers. A strategy is a mapping from board positions to moves, but it is not realistic to consider all legal board positions to be the domain of the mapping, since, for example,

the sequence of initial moves generated by a specific strategy may insure that a large number of legal board positions will never occur in a game played by that strategy. Thus $5^{32} > 10^{22}$ is much too large — let us assume that it is a million times too large so that the average domain of a strategy has cardinality $> 10^{16}$. Newell and Simon [1964] have estimated that there are, on the average, 10 legal moves from each board position (i.e., the average range of a strategy has cardinality = 10. Combining these figures, we obtain an estimate for the extent of the environment: $e(E) > 10^{10^{16}}$.

On the other hand, following Newell and Simon [1964], we can attempt to determine the number of strategies by a uniform tree which is an approximate of the checkers game tree. The disadvantage of such an approximation is that while the uniform tree becomes broader as the depth increases, the game tree begins to narrow after a certain depth because of merging which occurs when different sequences of moves lead to the same state. Newell has estimated that the equivalent tree for checkers has an average branching B = 10 and an average depth D = 70. We use the following reasoning to determine the number of strategies for both the first and second player of a uniform B, D tree game.

At the k^{th} step for the first player (i.e., the $(2k-1)^{th}$ move of the game), his opponent has already made k-1 moves and consequently may be at any one of B^{k-1} nodes. Thus at his k^{th} step, the first player must choose one of the B alternatives open to him for each of the B^{k-1} nodes to which his opponent may have directed the play. That is, he must choose one out of a total of $B^{B^{k-1}}$ functions. In all, the total number of strategies for the first player for a complete game of D moves (assume D even) is $\prod_{k=0}^{\frac{D}{2}-1} B^{B^k}$. Similarly for the second player, there are a total

of $\prod_{k=1}^{\frac{D}{2}} B^{B^K}$ strategies. This line of reasoning, therefore, gives an estimate of

$$\prod_{k=1}^{34} 10^{10^k} > 10^{10^{34}}$$

strategies for the first player which is considerably larger than our previous estimate for e(E). We shall use the least upper bound.

As is usual, an estimate of the extent of the meta-environment (the apparent number of strategies available to the adaptor constrained to use Samuel's look-ahead computer) is considerably easier to compute. Each polynomial (PVV) specifies a single strategy, but there is no reason to believe that the same strategy will not be specified by more than one polynomial, so that the number of possible polynomials gives an upper bound on the number of strategies which the complete system is capable of playing. There are at most 38 terms, 22 of which are constrained to have a weight of zero and the remaining 16 of which taken on any of a total of 36 possible weights. Thus

$$e(\overline{E}) = \binom{38}{22} \cdot 36^{16} < 36^7 \cdot 36^{16} = 36^{23} < 10^{36}$$

So again as with the Learning Machine, e(ME) is much smaller than e(E) so that the computer has drastically reduced the size of the space which must be considered. It is interesting to note as is clearly shown in Table 3.1.2 that the extent of the environment and especially the extent of the meta-environment of the Checker Player are much less than those of the Learning Machine.

Because of the difficulty of estimating the true utility, we are at a loss to give any estimation of the effect of the reduction of the search space on the utility density. However, judging from the result, if the utility density were reduced (and this seems extremely unlikely), other

effects more than compensated for the loss. As mentioned above, a third effect of the computer layer is to make a transformation of the representation of the strategies so that regularities (with respect to utility) in the strategy space become more apparent. The checker player illustrates this effect admirably since meta-environmental strategies are expressed in a language which has a useful type of continuity. This continuity assures that "small changes yield small effects", i.e., polynomials which differ slightly in a single term can be expected generate strategies which play very similar games against most opponents and thus have a similar utility. This fact enabled Samuel to design a learner which was able to take advantage of the ordering of the parameter values to restrict the number of permissible next values and make use of a hint generator which suggests a direction of change.

The peculiarities of Samuel's adaptor, especially its use of hints make it impractical to make a detailed investigation of the portion of its meta-environment it was able to explore. However, it suffices to say that Samuel's adaptor was capable of changing many parameter values after each trial as contrasted to Friedberg's Learner which changed only one per trial. Clearly, Samuel's adaptor was required to search a much smaller meta-environment (10^{29} points versus 10^{270} points) which was much more regular and it conducted the search much less cautiously. The fact that the Checker Player was much more successful than the Learning Machine should not be surprising. The Checker Player shares with the Learning Machine an inability to associate utility with pairs, triples, etc. of parameters except in the few cases where parameters are formed using binary combinations of more primitive terms. The success of the Checker Player provides evidence that Samuel was reasonably successful in choosing a set of orthogonal parameters, yet it seems natural to

speculate as to the relative merits of an adaptor which is able to associate utility with combinations of parameters in a more systematic way.

3.3 Adaptive Data Classifiers

Finally, we shall apply the paradigm to systems which use networks of threshold elements to classify data. Such systems have been proposed and constructed for a variety of purposes: character recognition, weather forecasting, speech recognition, electrocardiogram analysis, etc. The work of Widrow [1962] and the group at Stanford Electronics Laboratories on the Adaline and that of Rosenblatt [1962] and the group at Cornell Aeronautical Laboratories on the Perceptron are examples of the effort being applied to systems of this type. Related schemes have been proposed by Uhr & Vosseler [1961] and Bledsoe & Browning [1959] among others.

Basically, the problem is to partition an input set (whose elements may be thought of as patterns) into a number of subsets (e.g., characters) by means of a network composed of threshold logic devices. More concretely, inputs to the network (elements of the set to be partitioned) may be thought of as being projected on a retina of photocells each of which is connected to an input (or inputs) of the network. The signal appearing at the output of the network is interpreted as a label of the class to which the input pattern belongs. The amplification factors and threshold levels of the threshold logic elements are adjusted during a training period which consists of a sequence of presentations of patterns from the input set. The network is said to converge if the desired response is obtained to each pattern presentation of interest following the training period. The principal goal is to characterize those network organizations, training rules and input spaces for which

convergence exists and for which the rate of convergence is, in some sense, optimum.* A further goal is to develop systems which generalize — i.e., correctly classify patterns on which they have not been trained. There are few concrete results relating to the attainment of this goal.

Next we shall identify the elements of the universe of the Adaptive Data Classifiers with the elements of our paradigm. The environment consists of the set of patterns which is to be partitioned so that the environmental output (state description) is a representation (usually a binary string or array) of the pattern under consideration. The environmental input (move) is the computer's "guess" of the correct class to which the pattern belongs and the true utility indicates whether or not this classification is correct. The environmental transition function controls the sequence in which patterns are presented. The computer is the network of threshold logic devices some or all of which may have variable weights associated with their input wires as well as variable threshold values. These weights and threshold are the parameters and the settings which they may assume are the parameter values. This identification of the network with the paradigm's computer is in full accord with the operational test given in the previous chapter. For, when given a set of weights and threshold levels, the network will indeed compute a move for (i.e., make an identification of) each environmental output. The utility estimator, besides indicating whether or not a correct identification has been made, often provides to the computer the differences between the weighted sums of the inputs and the threshold

---

*Mayes [1963] gives a complete analysis and establishes bounds on the number of adaptations required for convergence by various adaptive algorithms.

values of individual threshold devices (i.e., an estimate of how right or how wrong the identification is). In addition, hints are usually supplied which identify the relevant parameters (since this is just an identification of the active threshold devices, it is closely related to the environmental state description). The Adaptor embodies the training algorithm or reinforcement control system which aims, in essence, to partition the input or measurement space by means of hyperplanes. The adaptors treated in the literature differ in the precise way that corrections are applied to weights and thresholds during the training period. For example, the amount of change may be fixed or it may depend upon the error indication given by the utility estimator, or corrections may be made as a result of each trial or only after unsuccessful trials.

As in the previous sections, we shall find it profitable to compare the extent of the environment with the extent of the meta-environment. Thus far in this section, we have dealt in general terms with what is in reality a class of systems (the Perceptron — Adaline class if you will). However in order to make the following discussion more concrete, we will concentrate our attention on the simplest member of that class — the single threshold device with K binary inputs and one binary output.* As we have noted in previous sections, the selection of a specific computer has the effect of restricting the performance of the system. In the present case, the selection of a threshold logic device network limits the system in that it can only partition the input space with hyperplanes. The extent of the meta-environment is just the number of strategies or functions from an input space of cardinality 2^K to a move space of cardinality 2. Clearly $e(E) = 2^{2^K}$. On the other hand

*This tactic is regularly employed by the proponents of such systems.

the development of a similar formula for threshold logic devices is still an open question, however, 2^{K^2} has been given [Fein, 1964] as a crude upper bound on the number of such functions. That is $e(ME) < 2^{K^2}$. For $K = 35$ (a reasonable sized retina for character recognition), $2^{K^2} = 2^{35^2} < 10^{111}$ while $2^{2^K} = 2^{2^{35}} < 10^{10^{10}}$ so that once again, the choice of computer has limited the strategies to such an extent that the system can be effective on an extremely small proportion of the potential environments. Whether this selection of a computer has been appropriate for the class of environments to which the Perceptrons and Adalines have been applied (i.e., whether the use of these devices has increased or decreased the utility density of such environments) is difficult to determine because of the lack of both a good theory and quantitative experimental results. It is true, however that the use of such computers has produced meta-environments which have the same continuity property encountered in the Checker Player meta-environment. This has certainly been of some advantage. As shown in Table 3.1.2, for this example, although the extent of the environment is the smallest of the three cases studied, the extent of the meta-environment for the Adaptive Data Classifier is larger than that of the Checker Player. We find that the adaptor is capable of changing many parameter values after each trial and therefore explores its meta-environment quite freely. Similarly we note that although Uhr and Vossler [1961] make use of restricted binary combinatorial parameters, no attempt has been made to develop adaptors with a general capability of associating utility with combinations of parameters.

3.4 Guidepost

Having considered a paradigm encompassing adaptive systems and

their environments and having shown how the research of a number of other workers in this area fits into the framework we have developed, we wish next to consider in detail the meta-environments in which they may be immersed.

4. ON META-ENVIRONMENTS

4.1 Idealization of the Meta-Environment

Chapter 2 contains a precise and formal definition of a meta-environment. The definition was made complete and very general in order to make evident its connections to the real world. At this point, we shall make several simplifications of that model in order to reduce its complexity to a level which is suitable to the goals and scope of this investigation.

Recall that the meta-environment as defined was intended to include everything beneath the line B-B' in Figure 1.1. Our first simplification will be to ignore the hint generator. This has been done not because hints are not an underexplored and potentially fruitful area for investigation but because we have chosen to concentrate our efforts on other aspects of the problem.

In formulating the paradigm in Chapter 2, our intent was that the adaptor A would introduce a strategy (PVV) to the computer C. The computer would then employ this strategy in generating tactics with which to confront the environment E. Although we made note of the fact that a global utility might be computed, we specified the estimated utility $O_{\overline{U}}$ was in fact an estimate of the local utility and therefore a measure of the "goodness" of the tactical moves of the computer. At this point, it should be apparent that our main interest lies in the interaction between the adaptor and the meta-environment. This being the case, we will assume that the estimated utility $O_{\overline{U}}$ available to the adaptor is indeed a global utility. It indicates how well a program computes the desired function rather than how well a single instruction operates. It indicates whether the complete game has been won or lost rather than

how good a single move was. It indicates how well the pattern recognizer evaluates a complete alphabet rather than a single character. Further we shall assume that this evaluation takes place over a sufficiently large number of observations that the effects of the states of the computer and environment can be effectively ignored. This means that $O_{\overline{U}}$ is an estimate of the utility of the strategy as it is applied to all conceivable states to which it is applicable. Note that the strategy may itself restrict the range of states which both the computer and the environment may attain. All of this has the effect of forcing the meta-environment to be a function which evaluates the strategies of the adaptor. Thus the task of the adaptor is to generate those PVVs which maximize the meta-environmental function. Note that both the domain (parameter value vectors) and the range (estimated utility values) of the meta-environmental function are finite so that there is no question of the existence of an adaptor which will maximize the meta-environment — an enumerative adaptor will surely do so. Adaptors will be judged on the expected or average number of trial PVVs they must generate before they generate one which maximizes the meta-environment. A "good" adaptor is one which is able to maximize a meta-environment after having generated few PVVs.

Now, function maximizing is an old art and the essence of the art lies in choosing constraints which partition the class of functions which are to be maximized. Conventionally these constraints concern such things as the modality of the function, the existence of derivitives, continuity, convexity and the like. In this thesis, we have chosen to concentrate our attention on one particular aspect of the meta-environment and this aspect has dictated the choice of constraints.

The aspect of meta-environments that we shall be concerned with deals with interaction between parameters. Historically this has been recognized

as a particularly knotty problem and has been attacked by Samuel [1959], Uhr and Vossler [1961] and others. Anyone who has had the experience trying to obtain an optimum picture on a television receiver by adjusting size, linearity, contrast, and sync controls which interact with one another should have an appreciation for the problems involved.

4.2 A Partition of Meta-Environments — The TEA

Let us suppose that we are to maximize a ME function each of whose parameters are orthogonal or independent. It is clear that the maximization may be accomplished by the following relatively straightforward procedure: 1) Choose a parameter and select arbitrary values for all the other parameters. 2) Vary the values of the chosen parameter to obtain a relative maximum utility value. 3) Hold the chosen parameter at a value which relatively maximizes the utility and choose another parameter which has not been previously chosen. 4) Go to step 2 and repeat the process until all parameters have been assigned values. The parameter value vector thus chosen will be one which maximizes the meta-environment.

The algorithm just described is indeed enumerative; but it does not enumerate all possible parameter value vectors. The fact that it works implies that the computer and its parameters were matched to the environment in a very special way. Consequently the maximum of the resultant ME could be found by enumerating parameters individually. This is certainly a very special case but one whose existence has been assumed by various workers in the field of adaptive systems. Friedberg [1959] made this assumption when he had his learner associate "success numbers" with individual instructions rather than with sequences of instructions. Similarly in the Checker Player and in various Adaptive Data Classifiers,

we find that the main emphasis is on associating "goodness" with individual parameters rather than with pairs or triples, etc. of parameters.

In an analogous manner, we can define a sequence of algorithms which seek the maximum of a function by enumerating pairs, triples, etc. of parameters. Let us call these "Tuple Enumerating Algorithms" (TEAs). The algorithm which enumerates individual parameters will be denoted TEA(1); that which enumerates parameter pairs will be denoted TEA(2) and so on. The purely enumerative algorithm which enumerates all possible PVVs is just TEA(p). We may then use the TEAs as a device to define precisely what we mean by parameter interaction. A ME will be said to be of depth d when it can be maximized by a TEA(d) but not by any TEA(d') where $d' < d$. It should be apparent that this definition of depth does partition the set of all MEs (which have the same domain and range) and that this depth partition is meaningful to the problem of parameter interaction. It is also effective since it has been stated constructively. However as a practical matter, it is too unwieldy to be used in the analysis or synthesis of MEs for either simulation or theoretical work. For that reason, we have been rather informal in the definition of the TEA and we shall define another related partition on the class of ME functions.

4.3 An Alternative Partition — The SCF

Of all of the ME functions of p arguments, there are some for which the following equation holds: $f(p_1,p_2,\ldots,p_p) = f_1(p_1) + f_2(p_2) + \ldots + f_p(p_p)$. We shall call these ME functions Sum Canonical Functions of type one denoted SCF(1). For other ME functions, the following is true: $f(p_1,p_2,\ldots,p_n) = f_{12}(p_1p_2) + f_{13}(p_1p_3) + \ldots + f_{p-1\,p}(p_{p-1},p_p)$ where not all of the subfunctions may be further broken down into SCF(1) subfunctions.

Those functions will be said to be of SCF(2) type. It is clear that this notion may be readily extended to define MEs of type SCF(3), SCF(4), etc. and that the resulting scheme does partition the class of all ME functions. This SCF partition does turn out to be much more desirable since it may be used in practice for both the synthesis and the analysis of MEs. The question remains as to the relationship between the TEA partition which reflects that quality of the ME which we wish to study and the SCF partition which we have the tools to deal with. That question will be taken up in the following section.

4.4. Relationships Between TEA and SCF Partitions of MEs

If a ME function is SCF(1), then it must be TEA(1). This is easily seen since the tuple enumerating algorithm need only maximize each of the subfunctions in turn and their sum will surely be the maximum of the ME function. The generalization of this relationship which states that if a function is SCF(α), then it must be TEA(β) where $\beta \leq \alpha$ does not hold in general. However, it is true for the subset of SCF functions called simulated MEs described in section 4.6.

We would hope that the converse of this relationship would hold, but unfortunately this is not true and that can best be shown by a simple counter-example. Let f be a ME function of three binary valued parameters with the following SCF(2) form: $f(P_1,P_2,P_3) = f_{12}(p_1,P_2) + f_{23}(P_2,P_3)$ where f_{12} and f_{23} are defined by Table 4.4.1.

Value Pair	Subfunction Value f_{12}	f_{23}
00	5	0
01	7	2
10	0	3
11	2	4

Table 4.4.1. Definition of the SCF(2) ME Subfunctions

P_1	P_2	P_3	f
0	0	0	5
0	0	1	7
0	1	0	10
0	1	1	11
1	0	0	0
1	0	1	2
1	1	0	5
1	1	1	6

Table 4.4.2 Complete Definition of the ME Function

The reader may verify that the complete function as defined by Table 4.4.2 will be maximized by the TEA(1) algorithm.

The reason for the relationship between the SCF and the TEA partitions (as well as the secret of constructing counterexamples like the one given) can be seen most easily by examining the nature of the partition. The TEA is a very specialized criterion which is relevant to a single aspect of the ME function — that of the relationship of the parameters to the maximum points. On the other hand, the SCF criterion applies to the description of the whole function. In the example given, those values which render the ME intractable to description as a SCF(1) do not affect its maximum value. In other words, every subfunction may be broken down or if it cannot, the only terms which prevent the breakdown are those which do not effect the maximum. Viewed in this manner it can be expected that although counterexamples are easy to construct, most "natural" MEs which are SCF(d) will also be TEA(d). In the example given in the next section, the three MEs which are exhibited will turn out to have identical SCF and TEA depths. In succeeding sections, unless otherwise stated, the term depth will be used to refer to the partition induced by the SCF and a ME which is of type SCF(d) will be referred

to simply as a ME(d).

4.5 The Game of Hexapawn — An Example

Thus far we have given a definition of meta-environmental depth. As motivation, we have mainly appealed to the reader's intuition. In this section we will give an example in order to illustrate that our notion of depth does have some relation to the real world. The example will consist of a computer to play the game of Hexapawn (the environment) against a fixed opponent. We will then exhibit three opponents such that the meta-environment is of depth 1, 2, or 3 depending upon which opponent is being challenged.

Hexapawn is a game played on a 3 x 3 portion of a chess board by two players each of whom has three pieces which may be moved like the pawns in chess. The object of the game is to have one of your pieces reach the opponent's home row or to render the opponent incapable of moving. Each game must terminate with a win or a loss; there are no ties. Details of the game (as well as of a correlation type adaptor to play it) may be found in Gardner [1962]. It suffices to state here that it is a simple game (about as difficult as tic-tac-toe) but one which may easily be extended to any desired level of complexity (Octapawn, Decapawn, etc. and even to modified games of checkers).

The players of the game will be designated as the "first mover" and the "second mover" for obvious reasons. The first mover will be the computer whose strategy may be adjusted by the adaptor. The second mover, the computer's opponent, will be represented by a fixed strategy and different opponents by different fixed strategies. The states of the game are related to board position and move number and moves cause a transition from one state to another. There are two classes of states;

non-terminal states from which another move is possible and terminal states from which no other move can be made and which end the play. The game states will be given labels in the sections below. However, in order to avoid superfluous terminology, moves will be denoted by the states to which they lead rather than by some additional designation.

Terminal states represent plays that have been either won or lost. Throughout this section, the orientation will be that of the first mover so that a win is a win for the first mover and a loss is a loss for the first mover. In order to simplify the game tree, there will be some consolidation of terminal board positions so that end positions which represent the same result as far as winning or losing the play are concerned will be assigned the same terminal state. The utility value which is fed back to the adaptor is derived from the terminal state attained. In order to enrich the range of utility values (a strictly binary won or lost indication is rather sterile) we shall make a distinction in terminal states based upon the move in which the play was terminated. This is meant to reflect the notion that an early win is better than a later win and that a deferred loss is better than a precipitous loss. As a further state consolidation measure, if a non-terminal board position can lead only to a single terminal state regardless of the choices on subsequent moves of either player, that board position will be assigned to the terminal state to which it must lead. The terminal states are denoted by two characters. The first is a letter — either a W or an L which indicate a win or a loss. The second is a number which indicates the move on which the termination occurred. There are five terminal states and these will be assigned utility values as shown in Table 4.5.1.

State	Utility Value
W3	5
W5	4
W7	3
L4	2
L6	1

Table 4.5.1 Utility Value Assignment

As a result of the consolidation outlined above, each non-terminal state must have at least two branches extending from it. In the complete game tree, there are 46 non-terminal states. 23 of these (labeled A - W) represent board positions which may be encountered by the first mover. The remaining 23 (labeled 1 - 23) represent board positions which may be encountered by the second mover. Table 4.5.2 gives the pictures of the board positions represented by each of the non-terminal states. The first mover's pieces are represented by the character "O". This table may be used to construct the complete consolidated game tree. A strategy for either mover consists of a choice (that is a one of the successor states) for each state which it can encounter.

We will next describe a parameterized computer so constructed that each parameter value vector will specify to the computer a strategy for the first mover. This strategy may then be tested against the fixed opponent by playing the game. The utility value obtained from the terminal state will be the value of the ME function for the given PVV. By enumerating all possible PVVs and computing the values of the ME function by playing the game, we can obtain the entire ME function in explicit form. By examining this function, we can readily determine the depth of the ME. In our example, we shall describe a fixed computer and demonstrate that the associated ME may be of depth one, two or three depending

LABEL	PREDECESSOR	BOARD	SUCCESSORS
A		XXX OOO	01 02 03
B	01	X X X OO	04 L4 L4
C	01	X X OX OO	05 W3
D	01	XX O X OO	W3 06 07
E	02	XX XO O O	W3 08
F	02	XX X O O	09 10 11 12
G	02	XX X O O	13 14 15 16
H	02	XX OX O O	06 W3
I	03	XX X O OO	08 17 W3
J	03	X X XO OO	W3 18
K	03	X X X OO	L4 19 L4
L	04 09	X XX O	L6 20

TABLE 4.5.2 HEXAPAWN BOARD CODING (PART 1)

LABEL	PREDECESSOR	BOARD	SUCCESSORS
M	05	X OOX O	W5 W5 W7
N	06 09	X OXX O	W5 21 W5
O	07	X OXO O	W7 W5
P	16 08	X XXO O	22 W5 W5
Q	10 14	X X O	W5 L6
R	11 15	X X O	L6 W5
S	11	X OX O	21
T	14	X XO O	22
U	16 19	X XX O	23 L6
V	17	X OXO O	W5 W7
W	18	X XOO O	W7 W5 W5

TABLE 4.5.2 HEXAPAWN BOARD CODING (PART 2)

LABEL	PREDECESSOR	BOARD	SUCCESSORS
01	A	XXX O OO	B C D
02	A	XXX O O O	E F G H
03	A	XXX O OO	I J K
04	B	X X XO O	W5 L4 L L6
05	C	X X OO O	L4 M W5
06	D H	XX OOX O	W5 N
07	D	XX O O O	W5 O L4
08	E I	XX XOO O	W5 P
09	F	XX OX O	L N L4 L4
10	F	X O O	L4 Q
11	F	XX O O	S R
12	F	XX XO O	L4 L4 L6

TABLE 4.5.2 HEXAPAWN BOARD CODING (PART 3)

LABEL	PREDECESSOR	BOARD	SUCCESSORS
13	G	XX OX O	L6 L4 L4
14	G	XX O O	Q T
15	G	XX O O	L4 R
16	G	XX XO O	L4 L4 P U
17	I	XX O O O	W5 L4 V
18	J	X X OO O	W5 W L4
19	K	X X OX O	L6 U L4 W5
20	L	X XO	L6 W7 L6
21	N S	X OOX	L6 W7
22	T P	X XOO	W7 L6
23	U	X OX	L6 W7 L6

TABLE 4.5.2 HEXAPAWN BOARD CODING (PART 4)

only upon the fixed strategy of the opponent second mover.

Our object in formulating the computer for playing the game of Hexapawn has been to give an example of MEs of various depths and not to demonstrate an efficient way to generate a computer to play board games. As a result, we have taken a very direct approach to the computer. The computer is controlled by three parameters each of which can assume one of four values. Thus there are $4^3 = 64$ possible parameter value vectors. Each of the states which the first mover can encounter has one of the parameters associated with it and the value which that parameter assumes determines the move that will be made when the computer encounters that state. The move table for our specific first mover computer is shown in the upper left corner of Table 4.5.3. As can be seen, state A is controlled by parameter 1 and will cause a move to states 01, 02, 03 or 01 as Parameter 1 takes on values V1, V2, V3 or V4 respectively. State B is controlled by parameter 3 and so on through the complete table. This computer is universal in the sense that it does not exclude any move which is possible for the first mover. Table 4.5.4 shows the computer table states sorted by the parameters which control them. Returning to Table 4.5.3, we note that the upper right hand corner contains a fixed strategy for the second mover. A small digital computer was programmed to enumerate the parameter value vectors and execute the resulting 64 plays of Hexapawn. The results are shown in the bottom of the table. The explicit form of the ME function may be expressed as $f(P_1, P_2, P_3) = f_1(P_1) + f_2(P_2) + f_3(P_3)$ where the definitions of f_1, f_2 and f_3 are given in Table 4.5.5. This is clearly a depth one meta-environment.

Table 4.5.6 includes the same first mover computer, but the second

THE PLAYERS

FIRST MOVER						SECOND MOVER	
STATE	PAR	V1	V2	V3	V4	STATE	MOVE
A	1	01	02	03	01	01	B
B	3	L4	L4	04	L4	02	G
C	2	05	W3	05	W3	03	J
D	1	07	06	W3	07	04	L
E	2	W3	08	W3	08	05	M
F	2	09	11	12	10	06	N
G	3	16	15	14	13	07	O
H	1	W3	06	W3	06	08	P
I	2	W3	08	W3	17	09	L4
J	3	18	18	W3	18	10	L4
K	1	19	L4	19	L4	11	S
L	2	L6	20	20	20	12	L6
M	1	W5	W7	W5	W7	13	L6
N	2	21	W5	21	W5	14	T
O	1	W5	W7	W5	W7	15	R
P	2	22	W5	W5	W5	16	U
Q	1	W5	L6	W5	L6	17	V
R	3	L6	L6	L6	W5	18	W
S	3	21	21	21	21	19	U
T	3	22	22	22	22	20	L6
U	2	L6	23	23	23	21	W7
V	1	W7	W5	W7	W5	22	W7
W	3	W5	W5	W7	W5	23	L6

RESULTS OF A COMPLETE SET OF PLAYS

P2	P3	PLAY	P1=1	PLAY	P1=2	PLAY	P1=3	PLAY	P1=4
1	1	1	1.0	17	2.0	33	4.0	49	1.0
1	2	2	1.0	18	2.0	34	4.0	50	1.0
1	3	3	2.0	19	3.0	35	5.0	51	2.0
1	4	4	1.0	20	2.0	36	4.0	52	1.0
2	1	5	1.0	21	2.0	37	4.0	53	1.0
2	2	6	1.0	22	2.0	38	4.0	54	1.0
2	3	7	2.0	23	3.0	39	5.0	55	2.0
2	4	8	1.0	24	2.0	40	4.0	56	1.0
3	1	9	1.0	25	2.0	41	4.0	57	1.0
3	2	10	1.0	26	2.0	42	4.0	58	1.0
3	3	11	2.0	27	3.0	43	5.0	59	2.0
3	4	12	1.0	28	2.0	44	4.0	60	1.0
4	1	13	1.0	29	2.0	45	4.0	61	1.0
4	2	14	1.0	30	2.0	46	4.0	62	1.0
4	3	15	2.0	31	3.0	47	5.0	63	2.0
4	4	16	1.0	32	2.0	48	4.0	64	1.0

TABLE 4.5.3 COMPUTER VS OPPONENT NO. 1

THE FOLLOWING STATES ARE CONTROLLED BY P1

STATE	PAR	V1	V2	V3	V4
A	1	01	02	03	01
D	1	07	06	W3	07
H	1	W3	06	W3	06
K	1	19	L4	19	L4
M	1	W5	W7	W5	W7
O	1	W5	W7	W5	W7
Q	1	W5	L6	W5	L6
V	1	W7	W5	W7	W5

THE FOLLOWING STATES ARE CONTROLLED BY P2

STATE	PAR	V1	V2	V3	V4
C	2	05	W3	05	W3
E	2	W3	08	W3	08
F	2	09	11	12	10
I	2	W3	08	W3	17
L	2	L6	20	20	20
N	2	21	W5	21	W5
P	2	22	W5	W5	W5
U	2	L6	23	23	23

THE FOLLOWING STATES ARE CONTROLLED BY P3

STATE	PAR	V1	V2	V3	V4
B	3	L4	L4	04	L4
G	3	16	15	14	13
J	3	18	18	W3	18
R	3	L6	L6	L6	W5
S	3	21	21	21	21
T	3	22	22	22	22
W	3	W5	W5	W7	W5

TABLE 4.5.4 COMPUTER STATE TABLE SORTED BY CONTROLLING PARAMETER

p	$f_1(P_1)$	$f_2(P_2)$	$f_3(P_3)$
1	0	1	0
2	1	1	0
3	3	1	1
4	0	1	0

Table 4.5.5 Definition of Depth 1 ME Subfunctions

THE PLAYERS

FIRST MOVER						SECOND MOVER	
STATE	PAR	V1	V2	V3	V4	STATE	MOVE
A	1	01	02	03	01	01	B
B	3	L4	L4	04	L4	02	F
C	2	05	W3	05	W3	03	J
D	1	07	06	W3	07	04	L
E	2	W3	08	W3	08	05	M
F	2	09	11	12	10	06	N
G	3	16	15	14	13	07	O
H	1	W3	06	W3	06	08	P
I	2	W3	08	W3	17	09	L4
J	3	18	18	W3	18	10	L4
K	1	19	L4	19	L4	11	S
L	2	L6	20	20	20	12	L6
M	1	W5	W7	W5	W7	13	L6
N	2	21	W5	21	W5	14	T
O	1	W5	W7	W5	W7	15	R
P	2	22	W5	W5	W5	16	U
Q	1	W5	L6	W5	L6	17	V
R	3	L6	L6	L6	W5	18	W
S	3	21	21	21	21	19	U
T	3	22	22	22	22	20	L6
U	2	L6	23	23	23	21	W7
V	1	W7	W5	W7	W5	22	W7
W	3	W5	W5	W7	W5	23	L6

RESULTS OF A COMPLETE SET OF PLAYS

P2	P3	PLAY	P1=1	PLAY	P1=2	PLAY	P1=3	PLAY	P1=4
1	1	1	1.0	17	1.0	33	4.0	49	1.0
1	2	2	1.0	18	1.0	34	4.0	50	1.0
1	3	3	2.0	19	1.0	35	5.0	51	2.0
1	4	4	1.0	20	1.0	36	4.0	52	1.0
2	1	5	1.0	21	3.0	37	4.0	53	1.0
2	2	6	1.0	22	3.0	38	4.0	54	1.0
2	3	7	2.0	23	3.0	39	5.0	55	2.0
2	4	8	1.0	24	3.0	40	4.0	56	1.0
3	1	9	1.0	25	2.0	41	4.0	57	1.0
3	2	10	1.0	26	2.0	42	4.0	58	1.0
3	3	11	2.0	27	2.0	43	5.0	59	2.0
3	4	12	1.0	28	2.0	44	4.0	60	1.0
4	1	13	1.0	29	1.0	45	4.0	61	1.0
4	2	14	1.0	30	1.0	46	4.0	62	1.0
4	3	15	2.0	31	1.0	47	5.0	63	2.0
4	4	16	1.0	32	1.0	48	4.0	64	1.0

TABLE 4.5.6 COMPUTER VS OPPONENT NO. 2

mover's strategy has been changed by changing the move at state 02 from G to F. The change is reflected in plays 17 - 32 shown in the column headed "P1=2" and the resulting ME is of depth 2. The ME function may be expressed as: $f(P_1,P_2,P_3) = f_1(P_1,P_2) + f_2(P_2,P_3) + f_3(P_1,P_3)$ where f_1, f_2 and f_3 are defined in Table 4.5.7.

P_i	P_j	f_1	f_2	f_3
1	1	0	1	0
1	2	0	1	0
1	3	0	1	1
1	4	0	1	0
2	1	0	1	0
2	2	2	1	0
2	3	3	1	0
2	4	0	1	0
3	1	0	1	3
3	2	0	1	3
3	3	0	1	4
3	4	0	1	3
4	1	0	1	0
4	2	0	1	0
4	3	0	1	1
4	4	0	1	0

Table 4.5.7 Definition of Depth 2 ME Subfunctions

Table 4.5.8 shows the same computer vs still another opponent. This opponent plays the perfect strategy. Note that he wins (i.e., causes the first mover to lose) all games in a minimum number of moves. It is a characteristic of the game of Hexapawn that the second mover can always force a win. It should be evident from the results of play that this combination also represents a depth two ME. We might note in passing that we have matched many computers of the form shown with many opponents and that the result most often reflected a depth two ME. If the meta-environment composed of the game of Hexapawn with utility function and computer of the form given here can be said to have a "natural"

THE PLAYERS

FIRST MOVER						SECOND MOVER	
STATE	PAR	V1	V2	V3	V4	STATE	MOVE
A	1	01	02	03	01	01	B
B	3	L4	L4	04	L4	02	F
C	2	05	W3	05	W3	03	K
D	1	07	06	W3	07	04	L4
E	2	W3	08	W3	08	05	L4
F	2	09	11	12	10	06	N
G	3	16	15	14	13	07	L4
H	1	W3	06	W3	06	08	P
I	2	W3	08	W3	17	09	L4
J	3	18	18	W3	18	10	L4
K	1	19	L4	19	L4	11	S
L	2	L6	20	20	20	12	L4
M	1	W5	W7	W5	W7	13	L4
N	2	21	W5	21	W5	14	T
O	1	W5	W7	W5	W7	15	L4
P	2	22	W5	W5	W5	16	L4
Q	1	W5	L6	W5	L6	17	L4
R	3	L6	L6	L6	W5	18	L4
S	3	21	21	21	21	19	L4
T	3	22	22	22	22	20	L6
U	2	L6	23	23	23	21	L6
V	1	W7	W5	W7	W5	22	L6
W	3	W5	W5	W7	W5	23	L6

RESULTS OF A COMPLETE SET OF PLAYS

P2	P3	PLAY	P1=1	PLAY	P1=2	PLAY	P1=3	PLAY	P1=4
1	1	1	1.0	17	1.0	33	1.0	49	1.0
1	2	2	1.0	18	1.0	34	1.0	50	1.0
1	3	3	1.0	19	1.0	35	1.0	51	1.0
1	4	4	1.0	20	1.0	36	1.0	52	1.0
2	1	5	1.0	21	2.0	37	1.0	53	1.0
2	2	6	1.0	22	2.0	38	1.0	54	1.0
2	3	7	1.0	23	2.0	39	1.0	55	1.0
2	4	8	1.0	24	2.0	40	1.0	56	1.0
3	1	9	1.0	25	1.0	41	1.0	57	1.0
3	2	10	1.0	26	1.0	42	1.0	58	1.0
3	3	11	1.0	27	1.0	43	1.0	59	1.0
3	4	12	1.0	28	1.0	44	1.0	60	1.0
4	1	13	1.0	29	1.0	45	1.0	61	1.0
4	2	14	1.0	30	1.0	46	1.0	62	1.0
4	3	15	1.0	31	1.0	47	1.0	63	1.0
4	4	16	1.0	32	1.0	48	1.0	64	1.0

TABLE 4.5.8 COMPUTER VS PERFECT OPPONENT (NO. 3)

THE PLAYERS

FIRST MOVER						SECOND MOVER	
STATE	PAR	V1	V2	V3	V4	STATE	MOVE
A	1	01	02	03	01	01	B
B	3	L4	L4	04	L4	02	F
C	2	05	W3	05	W3	03	K
D	1	07	06	W3	07	04	L4
E	2	W3	08	W3	08	05	L4
F	2	09	11	12	10	06	N
G	3	16	15	14	13	07	L4
H	1	W3	06	W3	06	08	P
I	2	W3	08	W3	17	09	L4
J	3	18	18	W3	18	10	L4
K	1	19	L4	19	L4	11	R
L	2	L6	20	20	20	12	L4
M	1	W5	W7	W5	W7	13	L4
N	2	21	W5	21	W5	14	T
O	1	W5	W7	W5	W7	15	L4
P	2	22	W5	W5	W5	16	L4
Q	1	W5	L6	W5	L6	17	L4
R	3	L6	L6	L6	W5	18	L4
S	3	21	21	21	21	19	L4
T	3	22	22	22	22	20	L6
U	2	L6	23	23	23	21	L6
V	1	W7	W5	W7	W5	22	L6
W	3	W5	W5	W7	W5	23	L6

RESULTS OF A COMPLETE SET OF PLAYS

P2	P3	PLAY	P1=1	PLAY	P1=2	PLAY	P1=3	PLAY	P1=4
1	1	1	1.0	17	1.0	33	1.0	49	1.0
1	2	2	1.0	18	1.0	34	1.0	50	1.0
1	3	3	1.0	19	1.0	35	1.0	51	1.0
1	4	4	1.0	20	1.0	36	1.0	52	1.0
2	1	5	1.0	21	2.0	37	1.0	53	1.0
2	2	6	1.0	22	2.0	38	1.0	54	1.0
2	3	7	1.0	23	2.0	39	1.0	55	1.0
2	4	8	1.0	24	4.0	40	1.0	56	1.0
3	1	9	1.0	25	1.0	41	1.0	57	1.0
3	2	10	1.0	26	1.0	42	1.0	58	1.0
3	3	11	1.0	27	1.0	43	1.0	59	1.0
3	4	12	1.0	28	1.0	44	1.0	60	1.0
4	1	13	1.0	29	1.0	45	1.0	61	1.0
4	2	14	1.0	30	1.0	46	1.0	62	1.0
4	3	15	1.0	31	1.0	47	1.0	63	1.0
4	4	16	1.0	32	1.0	48	1.0	64	1.0

TABLE 4.5.9 COMPUTER VS OPPONENT NO. 4

depth, that depth is two. If we take this perfect second mover and alter its strategy by changing the move at state 11 from S to R, we obtain the results shown in Table 4.5.9. Note that the only change from the previous set of plays occurred at play 24 which resulted in a win for the first player. This introduces one small weakness in the perfect opponent and makes for a much more interesting set of plays. The resulting ME function cannot be broken down and is clearly of depth 3.

To recapitulate: Our example contains a board game, a fixed computer, a fixed utility function and an assortment of opponents with fixed strategies. The resulting ME can be of depth one, two or three depending only upon the opponent selected.

4.6 Simulated Meta-Environments

Our purpose in the succeeding sections is to outline procedures for the generation of several classes of meta-environments. Each class is characterized by a specified depth (as defined previously by means of the SCF). These MEs which we will denote "simulated meta-environments" will be used in subsequent chapters as the basis for both analysis and simulation. They will be generated in such a way that the utility density and distribution under random sampling will be identical for all depths. They are admittedly artificial and are designed to be devoid of all structure except the depth characteristic whose effect on adaptive systems we are investigating. We begin by giving a fairly detailed description of the process for depth one MEs and then explain how it may be extended to generate MEs of depth two and greater.

As we have stated, a ME is a function (i.e., ME : $O_A \rightarrow O_{\overline{U}}$) which assigns an estimated utility to each parameter value vector (PVV). Each individual ME function is specified by an array of numbers and we shall

first describe the generation of this ME array and then describe how the array is used to assign values to PVVs. In the sections which follow, we shall assume that p = v = 8. These are the values which will be used in the simulated meta-environments.

4.7 The Generation of Depth One Simulated MEs

For depth one, the ME array has eight rows and eight columns where the rows are intended to correspond to parameter values and the columns to parameters. Thus the element in the third row of the fourth column corresponds to the third value of the fourth parameter. The elements of the array are integers drawn from the set {1, 2, 3, 4} and are selected in the following manner: 1) Two rows of the first column are selected at random and the corresponding elements are set equal to 1. 2) Two of the remaining six elements are selected at random and set equal to 2. 3) The process is repeated with 3 and 4 so that the elements of the first column are completely specified. This first column now contains two ones, two twos, two threes and two fours placed at random. 4) The process is repeated for each of the remaining seven columns.

The computation of the estimated utility assigned to a PVV by the ME represented by the array is quite straightforward. To each of the eight parameter values in a particular PVV, there corresponds a single element in the ME array. The sum of these elements or subutility values is the value of the ME for that PVV. The range of the ME function thus defined is the integers from 8 to 32 inclusive.

Next, as a control, let us investigate the performance of an adaptor which generates PVVs purely at random vis-a-vis the depth one ME just described. We may consider the ME to be a random variable which assigns a number (the estimated utility) to each of the PVVs. We can then

examine the resulting distribution of estimated utility values. Since the random variable is composed of the sum of eight independent random variables, this distribution is very close to being normal. The probability of the generation of a PVV which is assigned the maximum value (32) is just $(.25)^8$ since each of the 8 elements in the sum must be a 4 and this occurs independently with probability .25. The probability of not attaining a maximum in n trials is $(1 - (.25)^8)^n$ which may be approximated (for large n) by $e^{-n(.25)^8}$. Thus the probability of attaining at least one maximum in n trials is the complement of that value or $1 - e^{-n(.25)^8}$ The expected number of trials before a maximum is reached is just $\frac{1}{(.25)^8} = 65{,}536$. The Table 4.7.1 gives the probabilities for attaining at least one maximum value in n trials for various values of n. These

Number of Trials n	Probability of Success $1 - e^{-(.25)^8 n}$
655	.01
3,277	.05
65,536	.63
196,608	.95
301,466	.99

Table 4.7.1 Random Adaptor vs ME(1)

figures provide a yardstick against which we may judge the performance of more sophisticated adaptors. The extent of the ME is 8^8 or 16.8 million so that there are $(.25)^8 \cdot 8^8$ or 256 points of maximum utility.

4.8 Higher Depth Simulated MEs

The array for a depth two ME has $V^2 = 64$ rows and $p = 8$ columns. In an obvious extension of the depth one case, the rows correspond to

pairs of parameter values. Note that there are $\binom{8}{2} = 28$ parameter pair combinations, but only 8 columns in the ME array. Thus, unlike the depth one situation, not all of the parameter combinations can be represented. The eight parameter pair combinations which correspond to the columns of the array will be chosen at random from among the twenty-eight possible combinations in such a way that each parameter is represented in precisely two columns. In general, in the depth of ME array, each column will represent precisely d parameters and each parameter will be represented in precisely d columns. The elements of the array are then randomly assigned numbers from the set {1, 2, 3, 4} with each number present in equal proportions in the columns as in the depth one ME. The value of the ME for any PVV may be obtained by summing the subutility values in the rows corresponding to value pairs in the PVV for each of the eight parameter pairs which correspond to the columns. An additional complication arises in the generation of ME arrays for higher depth MEs. Note that for depth two a given value for a single parameter specifies subsets of each of the two columns in which it is represented. The problem arises when the interaction is such that certain combinations of subutility values may not be attained or may be attained with a different probability under random sampling than is required to maintain uniformity with the depth one case.

The following example of an ME with 4 binary parameters and ME array values drawn from the set {1, 2} illustrates the problem. The ME array is given in Table 4.8.1.

	Parameter Pair			
Value	12	23	34	14
00	1	2	1	1
01	1	1	2	2
10	2	1	1	2
11	2	2	2	1

Table 4.8.1 Example of a Defective ME(2) Array

We note that each column has 2 ones and 2 twos and we would expect that if we enumerated the ME function, the values would be distributed as would a depth one ME with the same parameter and value structure so that the distribution would be as shown in Table 4.8.2. However if we

Utility Values	4	5	6	7	8
#	1	4	6	4	1

Table 4.8.2 Desired Distribution of Utility Values

enumerate this depth two ME function, we find that the distribution is as given in Table 4.8.3 so that the anticipated maximum value of 8 cannot be attained.

Utility Values	4	5	6	7	8
#	2	2	6	6	0

Table 4.8.3 Actual Distribution of Utility Values

In order to prevent occurrences of this sort we must impose another type of uniformity in the ME array. There are many ways to do this but we shall describe only one of them which we have employed to generate depth two ME functions. First of all, we may arrange the ordering of the column labeling so that each parameter occurs first in one and only one column label. Then the values (1-4) are placed in the columns in such a way that each subset which is specified by a single value of the first parameter is uniform in the sense that it has two ones, two twos, two threes and two fours. Note that this property will not in general hold true for the subset specified by the second parameter. The effect of this is to assure that the choice of a single parameter value

restricts the possible subutility values only in a single column. It should be noted that this does not force the ME to be of depth one since that would require the subset's values rather than merely its distribution be invariant. Table 4.8.4 contains an example of a depth two ME array which does yield the desired distribution of utility values:

	Parameter Pair			
Value	12	23	34	41
00	1	2	1	1
01	2	1	2	2
10	2	1	1	2
11	1	2	2	1

Table 4.8.4. Example of a Constrained ME(2) Array

This concept may be readily extended to simulated MEs of higher depth. It means insuring that all but one of the subsets specified by the values must have the uniformity characteristic described. It is this characteristic which insures the validity of the relationship between simulated MEs and TEAs described in section 4.4. For the depth two and higher simulated meta-environments the range of estimated utility values is the integers 8 to 32 inclusive and the statistics under random sampling are identical to those of the simulated depth one ME.

We have had to pay a price in order to achieve uniformity under random sampling. Because we were forced to choose only a fraction of the parameter combinations, we have in a sense diluted the depth character of the ME. The effect of this dilution will be seen in the results given in subsequent chapters.

4.9 Guidepost

In this chapter, we reduced the complexity of our paradigm in such a way that the meta-environment became a function. We then concentrated our attention on one particular aspect of the meta-environmental function

— that of interaction between parameters and formulated a definition of meta-environmental depth that highlighted that aspect. Because the idealized definition proved to be unwieldy for our purposes, we formulated another related definition of depth. Next, in order to illustrate the validity of our concept of ME depth, we gave an example of a real environment — the game of Hexapawn played against a number of fixed opponents along with a computer and showed that the combination was a meta-environment of depth one, two or three depending upon the opponent chosen. Finally, we gave a procedure for generating simulated meta-environments with specified depth characteristics which exhibit uniform behavior under random sampling.

In the chapters that follow, we will describe two classes of adaptive algorithms and their behavior when immersed in simulated meta-environments.

5. THE CORRELATION ALGORITHM — DESCRIPTION

5.1 Introduction

In this chapter, we shall be concerned with a particular class of adaptive algorithm or adaptor which we have termed the "correlation adaptor." It is our intent that this algorithm should reflect the essential features of a number of adaptive systems that have appeared in the literature and we shall endeavor to point out the correspondences as we proceed.

Recall that in our paradigm, the function of the adaptor is to generate parameter value vectors and that our criterion for success for an adaptor is the number of trials required for it to generate a PVV which merits the maximum utility. The fewer trials required, the "better" the adaptor is.

The heart of the correlation adaptor (CA) is the experience array (EA) which contains, at any time, information which summarizes the results of the adaptor's previous encounters with the current meta-environment. The adaptor uses the experience array as the basis for the generation of subsequent parameter value vectors. It is convenient to divide the correlation adaptor into two parts according to function. The recording algorithm is that part of the CA which enters the results of a trial (of a PVV in a ME) into the experience array. The generation algorithm is that part of the CA which uses the EA to generate a new PVV.

In the next sections, we shall examine the components of the Correlation algorithm in more detail. It may be either deterministic or probabilistic and though we mention both cases, we shall concentrate upon the latter.

5.2 The Experience Array

The memory or storage states of the CA are contained entirely within the EA. Under interpretation, the values in the EA are intended to represent the correlation of PVVs with the utility they merit from the ME. The utility attained by a particular PVV is the result of the entire PVV, however, internally, the CA may choose to associate this utility with the individual parameters of the PVV, or with pairs of parameters, triples of parameters, and so on. We shall reflect this choice by defining several distinct classes of CAs each of which is distinguished by the experience array which it uses. Thus the correlation algorithm which associates utility in its EA with individual parameters will be referred to as the level 1 CA (or CA(1) for short) and the CA which associates utility with parameter pairs will be referred to as the level 2 CA (CA(2)) and so on. We may also define a zero level CA in which utility is associated with parameter values irrespective of parameter type. Both the success numbers of Friedberg [1958] and the correlations of Samuel [1959] as well as the weights associated with arrays of adaptive threshold elements correspond to level 1 experience arrays. Samuel as well as Uhr and Vossler [1961], Chow and Liu [1966], and others have considered experience arrays of level 2 and higher, but not as systematically as the present treatment. (This is understandable since they were more interested in attaining practical ends and it is apparent that complete experience arrays of levels greater than one become very unwieldly. We reiterate that our interest lies more in exploring the implications of the higher level CAs rather than using them to aid in the solution of real problems.)

The following is a more detailed interpretation of the experience array. The formal definition is given constructively below in the discussion of the recording algorithm.

A ℓ^{th} level experience array X is a two dimensional array of elements x_{ij} where each element x_{ij} is an estimate of the correlation with success (i.e., high utility) of the i^{th} parameter value ℓ-tuple in the j^{th} parameter ℓ-tuple. For example with $\ell = 1$, x_{ij} is the correlation associated with the i^{th} value of the j^{th} parameter while for $\ell = 2$, x_{ij} is the correlation associated with the i^{th} value pair of the j^{th} parameter pair and so on. For level 1, the elements of the experience array are closely related to the entity Fisher [1958] calls the average excess of utility associated with allele i at position j. We have extended this notion to apply to pairs, triples, etc. of alleles at pairs, triples, etc. of positions.

In general, for a level ℓ CA, there are V^{ℓ} value ℓ-tuples and $\binom{p}{\ell}$ parameter ℓ-tuples (symmetry requires that the parameter ℓ-tuples include no duplicates). Thus there are a total of $V^{\ell}\binom{p}{\ell}$ elements in the ℓ-level experience array. In order to simplify the exposition in the following section, we find it convenient to introduce the following notation for ℓ-tuples of parameters and values: p_i^{ℓ} is the i^{th} parameter ℓ-tuple $(1 \leq i \leq \binom{p}{\ell})$ and V_j^{ℓ} is the j^{th} value ℓ-tuple $(1 \leq j \leq V^{\ell})$ where the ordering is the natural one. Thus for example: $p_j^1 = p_j$ $(1 \leq j \leq p)$, $v_i^1 = v_i$ $(1 \leq i \leq v)$ and $v_1^2 = (v_1, v_1)$ etc. In a similar manner we may define x_{ij}^{ℓ} and X^{ℓ}. When the meaning is clear from the context, we shall often omit the superscript for the sake of simplicity. Figures 5.2.1 and 5.2.2 are diagramatic representations of the level one and level two experience arrays respectively.

5.3 The Recorder Algorithm

Formally, the experience recorder algorithm corresponds to the state transition function of the adaptor. Its purpose is to record in

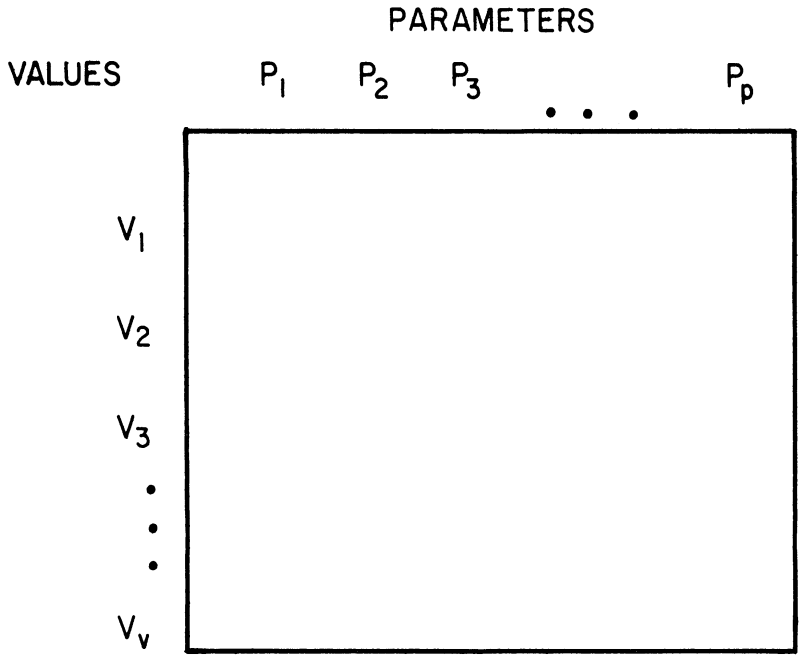

FIG. 5.2.1 LEVEL ONE EXPERIENCE ARRAY

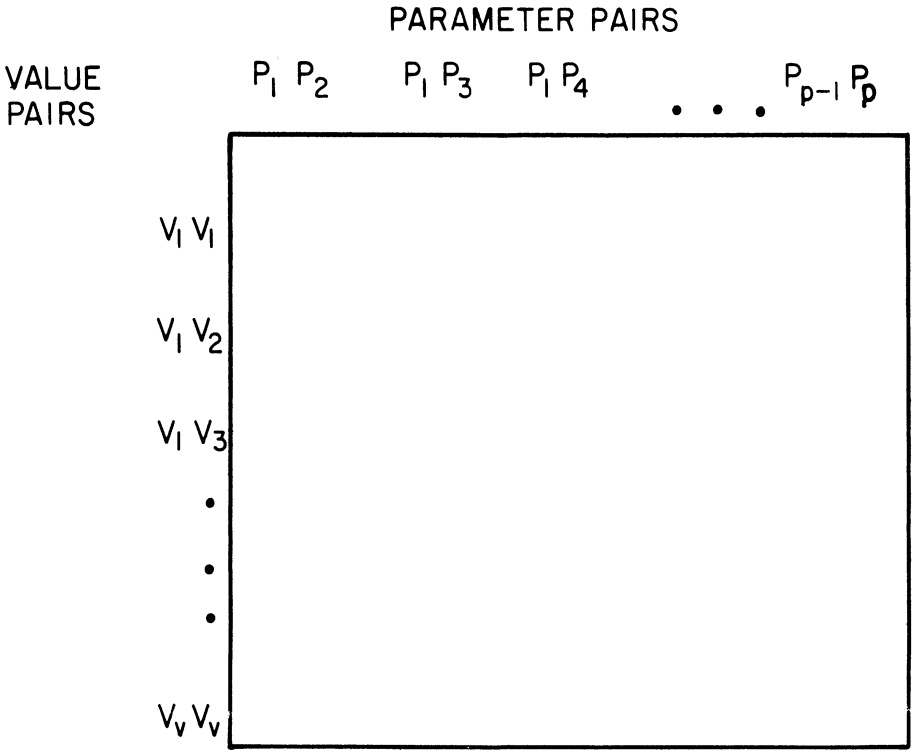

FIG. 5.2.2 LEVEL TWO EXPERIENCE ARRAY

the array X information gained from a trial (vs the ME) of a PVV($O_A(t)$). Any specific PVV has a value for each parameter, a value pair for each parameter pair and so on, so that corresponding to the X array for any level, there is one and only one element in each column which corresponds to a given PVV. It is this relevant entry which will be changed as the result of the trial the corresponding PVV.

At any time, each element (x_{ij}) of the experience array X reflects the average utility earned by the relevant PVVs sampled to date. In order to compute the average, it is necessary to record the number of times that each entry has been relevant. To accomplish this, let each element x_{ij} consist of a pair (y_{ij},z_{ij}) where y_{ij} is interpreted to be the sum of the utilities of all of the PVV's for which the (i,j) the entry has been relevant and z_{ij} is interpreted as the number of those relevant PVVs (the number of times tried). The desired average value is just y_{ij}/z_{ij} (z_{ij} will be restricted to positive integer values). It will be convenient to consider the X array as being split into two separate arrays Y and Z with the obvious interpretation.

In order to be able to state the recorder algorithm more compactly, we shall define another array — the relevance array R. The relevance array has the same dimensions as the X array and there is a relevance array associated with each PVV. The R array is composed of ones and zeros and has a single one in each column which corresponds to the relevant entry in that column.

More precisely:

$$R_i = (r_i)_{jk} \quad \text{and}$$

$$(r_i)_{jk} = 1 \text{ if } O_{A_i}(p_k) = v_j$$

$= 0$ otherwise for $1 \leq j \leq v^{\ell}$

$1 \leq k \leq \binom{p}{\ell}$

$1 \leq i \leq v^{p}$

The recorder algorithm may now be stated as follows:

$Z(t+1) = Z(t) + R(t)$ (the operation + is matrix addition)

$Y(t+1) = Y(t) + R(t) \circ O_{\overline{U}}(t)$ (where $O_{\overline{U}}(t)$ is a scalar and the operation $\circ$ is the multiplication of a matrix by a scalar)*

5.4 The Generator Algorithm

Formally, the generator algorithm corresponds to the output function of the adaptor. Its purpose is to generate (using the information contained in the X array) an optimum PVV (i.e., one which will return the maximum utility). Because the space of all possible PVVs is immense (though finite), in practice it is possible to sample only a minute portion of it. The sampling is of necessity biased since its purpose is not to completely explore the space so as to be able to predict the utility of an arbitrary PVV but to find the points of maximum utility. Consequently, the areas of high utility will tend to be more thoroughly sampled, while those of lower utility will be avoided. In essence, the adaptor conducts a search for regularities in the space of PVVs.

For level one, it is necessary to select a single value for each parameter, i.e., to choose a single row in each column of the X array.

---

*We might recall that one of the hints that we have mentioned is the one that identifies participants. The natural method of handling such hints is to restrict reward and punishment to those parameter value tuples which actually participated in the acquisition of utility. Such a scheme can be expressed quite naturally by the present notation by modifying the definition of the relevance array to read:

$$(r_i)_{jk} = 1 \text{ if } O_{A_i}(p_k) = v_j \text{ and } p_k \text{ is a participant}$$

The order in which the columns are selected is immaterial. However, for levels greater than one, this is not the case since choices within one column may limit the consistent choices available in other columns. For example, in a level two adaptor if p_1p_2 is chosen to be v_3v_4, the choice of values for p_2p_3 cannot be made freely from among all pairs v_iv_j, but must be constrained to those pairs whose first element is v_4. It may be seen that for levels greater than one the order in which columns are considered is significant so that it becomes necessary for the generator algorithm to include a column sequences selector as well as a mechanism for selecting a row within a given column. This situation comes about because of a redundancy in the P-level adaptor for ℓ greater than one since although there are only p parameters, there are $\binom{p}{\ell}$ columns in the X array so that each parameter is represented in a total of $\binom{p-1}{\ell-1}$ columns. This redundancy means that fewer than the total number of columns must be chosen in the generation process. Let C_{min} be the minimum number of columns required to be chosen and C_{max} the maximum. It is evident that:

$$C_{min} = \text{the least integer} \geq p/\ell$$

and

$$C_{max} = p - \ell + 1.$$

Thus far, we have considered only what might be termed only completely redundant P-level adaptors — those which consider all $\binom{p}{\ell}$ ℓ-tuples of parameters. There are many reasons for considering incomplete or non-redundant adaptors. For example, a priori information may be available about the existence of "neighborhoods" among the parameters (i.e., subroutines under Friedberg's interpretation) or considerations of implementation may force a reduction in the size of the X array and the complexity of the adaptor. Chow and Liu [1966] confine their study to level one adaptors and incomplete level two adaptors with "chain" and

"tree" dependencies. Note that we shall consider only those adaptors which are complete in the more restricted sense that each parameter is represented in at least one column. A minimum of C_{min} columns and a maximum of $\binom{p}{\ell}$ will be required for any adaptor.

In the following treatment, we shall assume that the elements of the X array ($x_{ij} = y_{ij}/z_{ij}$) are bounded positive rational numbers. From the previous description of the recording algorithm, there is no reason to believe that this will be the case, but a subsequent section will describe a series of transformations to the basic X array which will insure this property.

5.5 The Column Selector

The basic task of the column selector is to assign a positive real number c_j to each column of the X array and then select the sequence of columns on the basis of their associated values of c_j. This selection may be either deterministic (columns are selected in order of decreasing c_j) or probabilistic (let the j^{th} column be chosen with probability equal to $c_j/\sum_k c_k$ We list below several methods of assigning values of c_j to the columns accompanied by an intuitive rationale for each. We shall examine the implications of each method in the next chapter.

1) $c_j = \max_i x_{ij}$ (The column with the best element in terms of utility should be given preference.)

2) $c_j = \sum_i x_{ij}$ (The column with the best aggregate performance should be given preference.)

3) $$c_j = \frac{\max_i x_{ij} - \min_i x_{ij}}{\frac{1}{m^{\ell}} \sum_i x_{ij}}$$ (The column with the maximum relative deviation is the column where a choice makes the most difference so that it should be given preference.)

4) $c_j = \sum_i |x_{ij} - \bar{x}_j|$ where $\bar{x}_j = \frac{1}{m^{\ell}} \sum_i x_{ij}$

(This is very similar to 3. The column which has the greatest total deviation from its mean should be given preference.)

5) c_j = constant (This leads either to enumerative or uniformly random column selection. Its speed and simplicity recommend it.)

5.6 The Row Selector

Within a given column, selection of a particular row may again be either deterministic or probabilistic. In the former case, the value tuple v_i corresponding to the largest consistent x_{ij} in the column j is chosen. In the latter, the value tuple v_i is chosen with probability $x_{ij}/\sum_i x_{ij}$ where the j^{th} column has been selected and i ranges over the set of consistent value tuples.

Next we shall discuss two basic transformations to the X array which are required to put it into a form suitable for the generator algorithm.

The generator algorithm as described thus far assumes that every x_{ij} in the EA has a positive value, but it is clear from the recorder algorithm that the only x_{ij}'s which have been given values are those which have been relevant in some PVV. What is to be done about the "unexplored" points? If they are ignored by the generator algorithm, they will never be tried and, in fact, it is unclear how the entire adaptor will be started. The most straightforward solution is to give these unexplored points an artificial x value until they become explored. This is the path that we have chosen.

Let a_1 be a small positive rational and assign to each unexplored x (i.e., to each x_{ij} whose corresponding $z_{ij} = 0$) a quantity equal to a_1 times the average value of utility returned thus far. Thus if $a_1 = 1$,

this is equivalent to the assumption that the unexplored points have a value equal to the current sample average. Initially X is assumed to be a constant array ($x_{ij} = a_2$ for all i,j) so that the first PVV generated by probabilistic adaptor will be drawn from a uniform distribution.

In the generator algorithm as described thus far there is no control for the regulation of the adaptor's reliance on its accumulated experience. For example, if the environment were extremely hostile, it would seem to be more efficient for the adaptor to pay less attention to its accumulated experience (since it is probably irrelevent or misleading) and concentrate on random sampling. On the other hand, if the environment is very regular, then the experience gained can be trusted and the adaptor would do well to avoid excessive random exploration. In order to accomplish this adjustment we shall introduce another coefficient.

Let a_2 be a coefficient which is added to each element of the X array before the generator algorithm is executed. If a_2 is a large positive number, its effect is to raise the level of the entire array so that the individual variations between elements acquired by experience exert far less control over the generation process. On the other hand, if a_2 is negative, the opposite effect obtains and the variations become accentuated so that more weight is given to accumulated experience. If a_2 is zero, its effect is neutralized. In this section, a_2 will be treated as a constant. In a subsequent section, we shall examine the consequences of allowing it to vary.

More formally, if the unmodified X array is denoted by X' and the modified array by X, the procedure described above may be expressed as follows:

$$x_{ij}(0) = a_2$$

$$x_{ij}(t) = (y'_{ij}/z'_{ij}) + a_2 \quad \text{for } z'_{ij} \neq 0,\ t > 0$$

$$x_{ij}(t) = a_1 \sum_{k=1}^{t} \frac{\overline{U(K)}}{t} + a_2 \quad \text{for } z'_{ij} = 0,\ t > 0$$

Finally, we shall close this chapter with a brief qualitative discussion of hints and their applicability to correlation adaptors.

5.7 Hints

We have already described a mechanism for introducing a priori knowledge of parameter neighborhoods by means of appropriate selection of columns in an incompletely redundant adaptor. There may also be situations where it is profitable to consider neighborhoods among values so that value selection is constrained by the current or past values of the same parameter. Thus the weights on the Checker Player bear a relationship to one another so that there is a greater distance between some than others. The same holds true for the weights on Adaline elements. However, such a relationship is by no means apparent in the Learning Machine. It hardly seems plausible to argue that the "distance" from a STORE instruction to an ADD instruction is somehow smaller than the "distance" from a STORE instruction to a TRANSFER instruction. If a meaningful neighborhood relation does exist, the problem of optimum step size may be considered.

In many meta-environments, transitions from one value to another may be restricted by hints or may depend upon some history of values. These restrictions and dependencies may affect both the direction of the step and its magnitude and should be taken into account in the design of the adaptor for specific meta-environments. Beyond this brief mention, we shall not consider such factors in this thesis.

5.8 Guidepost

Having described the operation of the correlation algorithm, we shall next be concerned with evaluating it. In particular we are concerned with the performance of CAs of various levels immersed in meta-environments of various depths. A reasonable hypothesis is that the optimum results will be obtained in the matched case where the level of the adaptor is the same as the depth of the meta-environment. Degradation should be expected when the level is less than the depth because the adaptor does not collect enough information to be able to accurately model the ME. Less degradation should be expected when the level is greater than the depth because the adaptor is processing information which is redundant with respect to the ME. In the chapter that follows, we shall examine the behavior of the correlation adaptor analytically in order to be able to test this hypothesis.

6 THE CORRELATION ALGORITHM — QUANTITATIVE CONSIDERATIONS

6.1 General

In this chapter, we shall be concerned with the performance of correlation adaptors of various levels immersed in meta-environments of various depths. Recall that our criteria for "goodness" of an adaptor is simply the expected number of trials required to generate a PVV which elicits the maximum utility from the ME. In the sections that follow we will report on our attempts to assess the class of correlation adaptors by both analytical and experimental means.

We may identify three phases of interaction of a correlation adaptor with a meta-environment as follows:

Phase I is an initial exploration phase. It is characterized by a high proportion of untried points in the experience array and consequently most of the parameter value tuples are selected at random. Phase I ends when the experience array contains a small proportion of untried points.

Phase II is an information gathering phase. Most of the values contained in the X array are the result of limited encounters with the ME and they may be unreliable. During phase II, the sampling continues and ultimately the estimated utilities which the EA contains become more accurate reflections of the ME. Phase II ends when further sampling of the ME can produce no significant changes in the EA.

Phase III is an execution phase. Most of the information in the EA is as accurate as it is going to get under the circumstances and the adaptor must now continue to generate PVV's as based upon the acquired information. Phase III ends when a maximum has been found.

It should be clear that the divisions between the phases as

described above are not clear cut and as a matter of fact at a given point an adaptor may exhibit characteristics of two or even three phases. These descriptions could be made much more precise so that a quantitative determination of an adaptor's phase could be made. However, it seems that such definitions would be of limited utility and would actually tend to confuse the very real notion of adaptor phase. The duration of Phase I is determined by the value of a_1. If a_1 is small, very little exploration is done and the adaptor continues to gather information about the points it has already explored. If a_1 is large, initially most of the adaptor's resources are devoted to exploration and this continues until there are no remaining unexplored points. In a like manner, the value of a_2 affects the transition between Phase II and Phase III. If a_2 is positive, the adaptor tends to ignore accumulated experience and attempts to continue to extract more information from the ME. If a_2 is negative, the adaptor tends to rely heavily upon accumulated experience even though it may still be unreliable.

Having made these general remarks, let us now consider some specific cases.

6.2 Phase I Considerations

Recall that during Phase I, there remain one or more unexplored points in the X array — (i.e., value tuples v_i for parameter-tuples p_j such that v_i hasn't yet been tried for p_j or equivalently $z_{ij} = 0$). The rate at which these unexplored points are being sampled is dependent upon the coefficient a_1 in the following manner. Consider a specific column which contains a total of $m = v^{\ell}$ rows of which r are unexplored. If we assume that the average utility attained by the m - r explored points is u, then future trials are allotted to exploration and production

in the following manner. Each of the unexplored points is assigned an x value equal to $a_1\bar{U}$ while each of the explored points has an x value equal to $\bar{u}$ on the average. Thus in any column, the sum of the x values of the unexplored points is $ra_1\bar{U}$ while the sum of the x values of the explored points is $(m-r)\,\bar{u}$. Direct application of the row selector portion of the correlation algorithm yields the following expression for the probability (P_{ur}) that one of the r unexplored points will be selected for the column in question.

$$P_{ur} = \frac{ra_1\bar{u}}{ra_1\bar{u} + (m-r)\bar{u}} = \frac{a_1 r}{(a_1-1)r + m}$$

If we define P_{rn} to be the probability that r points remain unexplored (in a single column) after n trials, we may write:

$$P_{r,0} = 0 \qquad r \neq m$$

$$P_{m,0} = 1$$

(because all points are initially unexplored.) And:

$$P_{r,1} = 0 \qquad r \neq m-1$$

$$P_{m-1,1} = 1$$

(because an unexplored point is certain to be chosen on the initial trial) and in general for $n > 0$:

$$P_{r,n} = P_{r,n-1}\,\frac{(m-r)}{m + (a_1-1)r} + P_{r+1,n-1}\,\frac{a_1(r+1)}{m + (a_1-1)(r+1)}$$

In our investigation of Phase I, we are concerned with the number of trials required to explore the ME sufficiently to fill the experience array. The value $P_{0,n}$ is the probability that there are no unexplored points in a single column after n trials. But there are many columns in the experience array ($\binom{p}{\ell}$ of them to be precise). If we make the assumption that exploration within the columns takes place independently and in parallel (note that this is not the same as the assumption that row

selection takes place independently of column selection), then the probability that the entire experience array is completely explored after n trials is just $(P_{0,n})^s$ where $s = \binom{p}{\ell}$ and $m = v^{\ell}$. The expected number of trials required for complete exploration is just:

$$\sum_{n=1}^{\infty} n(P_{0,n})^s.$$

Table 6.2.1 summarizes the results of computations made on a digital computer using the above expression. We have used p = v = 8 as in the simulated meta-environments.

It should be noted that it is possible to develop a closed form expression for $P_{r,n}$ using the difference equation method. We judge such a result to be incommensurate with the effort required to obtain it.

EXPECTED NUMBER OF TRIALS REQUIRED TO EXPLORE THE X ARRAY

ADAPTOR LEVEL = 1

A1	EXPECTED TRIALS
0.5000	61.
0.7500	44.
1.0000	35.
1.2500	29.
1.5000	26.
1.7500	23.
2.0000	21.

ADAPTOR LEVEL = 2

A1	EXPECTED TRIALS
0.5000	950.
0.7500	656.
1.0000	509.
1.2500	420.
1.5000	361.
1.7500	319.
2.0000	288.

TABLE 6.2.1 PHASE I DIRECT COMPUTATION

A second approach to the problem of Phase I X array exploration can be adapted from the Coupon Collectors Problem [Feller, 1957, p. 211].

Let us first consider the exploration of a single column. Let X_k be the number of trials following the selection of the k^{th} distinct parameter value up to and including the selection of the next new value. X_0 is clearly equal to 1 since the initial trial must select a value which has not previously been selected. Let S_k be the number of trials required for the selection of k distinct values. Then:

$$S_k = X_0 + X_1 + \ldots + X_{k-1} .$$

The probability of selecting a new value assuming that k have already been selected is just:

$$p = \frac{a_1(m-k)}{a_1(m-k) + k}$$

and

$$q = 1 - p = \frac{k}{a_1(m-k) + k} .$$

The value of X_k is one plus the number of failures preceding the first success in Bernoulli trials with p as given. The expectation and variance of X_k are given as:

$$E(X_k) = 1 + q/p = \frac{1}{p} = \frac{a_1(m-k) + k}{a_1(m-k)}$$

$$Var(X_k) = q/p^2 = \frac{E(X_k)q}{p} = \frac{E(X_k)k}{a_1(m-k)}$$

S_m is the number of trials required for complete column exploration and since the X_k are independent, its expected value and variance may be simply written:

$$E(S_m) = \sum_{k=0}^{m-1} E(X_k)$$

$$Var(S_m) = \sum_{k=0}^{m-1} Var(X_k)$$

In this case, the conditions for the central limit theorem are fulfilled and we may assume that S_m is normally distributed (at least for large values of m). We now wish to determine the expected number of trials required for all n columns of the X array to be explored. This requires finding the expected maximum of n observations of S_m. The density of the random variable v which is the area under $f(S_m)$ between the largest and the smallest observations in a sample of size n is the beta density (Mood and Greybill [1963], p. 405). The expected value of v is simply:

$$\int_0^1 vp(v)dv = \frac{n-1}{n+1}$$

Since $f(S_m)$ is symmetrical, the area under left hand tail is $\frac{n}{n+1}$. This fact coupled with the assumption of normality gives the expected maximum of n observation of S_m which is the expected number of trials required for exploration of the complete X array. Table 6.2.2 illustrates the results of computations using the normal approximation method.

Comparison of these results with those previously given reveals agreement to within 13% even for small values of n. In the direct computation, we expect the results to be slightly high because very small values of probability are lost on account of truncation and exponent underflow. This has the effect of prolonging the computation and therefore increasing the value computed for the expected number of trials. In the normal approximation computation, the assumption of normality can be expected to yield results that are slightly low because the actual distribution is truncated on the high end. In effect, these two computations have given us results which bracket the true values. There significance lies not in the precise values obtained, but in the fact that for levels greater than four, Phase I of the correlation adaptor

PHASE I NORMAL APPROXIMATION.

EXPECTED NUMBER OF TRIALS REQUIRED TO EXPLORE THE X ARRAY

ADAPTOR LEVEL = 1

A1	EXPECTED TRIALS
0.5000	55.
0.7500	40.
1.0000	32.
1.2500	27.
1.5000	24.
1.7500	22.
2.0000	20.

ADAPTOR LEVEL = 2

A1	EXPECTED TRIALS
0.5000	831.
0.7500	576.
1.0000	448.
1.2500	372.
1.5000	321.
1.7500	285.
2.0000	257.

ADAPTOR LEVEL = 3

A1	EXPECTED TRIALS
0.5000	9216.
0.7500	6316.
1.0000	4866.
1.2500	3996.
1.5000	3416.
1.7500	3002.
2.0000	2691.

TABLE 6.2.2 PHASE I NORMAL APPROXIMATION COMPUTATION
(PART 1)

PHASE I NORMAL APPROXIMATION.

EXPECTED NUMBER OF TRIALS REQUIRED TO EXPLORE THE X ARRAY

ADAPTOR LEVEL = 4

A1	EXPECTED TRIALS
0.5000	91809.
0.7500	62573.
1.0000	47955.
1.2500	39184.
1.5000	33337.
1.7500	29161.
2.0000	26028.

ADAPTOR LEVEL = 5

A1	EXPECTED TRIALS
0.5000	863143.
0.7500	586322.
1.0000	447924.
1.2500	364888.
1.5000	309532.
1.7500	269994.
2.0000	240342.

ADAPTOR LEVEL = 6

A1	EXPECTED TRIALS
1.0000	4025230.

TABLE 6.2.2 PHASE I NORMAL APPROXIMATION COMPUTATION
(PART 2)

will require a greater number of trials than the random adaptor requires to maximize the meta-environment. This indicates the first limitation on the usefulness of the correlation adaptor.

6.3 Phase II Considerations

Recall that Phase II is concerned primarily with the exploration of the meta-environment. During Phase II, the correlation adaptor is attempting to, in effect, determine the contents of the ME array which lies behind the particular class of meta-environments we have postulated.

In order to be able to obtain an estimate of the time spent in Phase II (i.e., "time" in terms of the number of samples of the ME which the CA must make), we shall make the assumption that the CA samples the ME at random. This is of course not strictly true since the adaptor is not only sampling to discover the content of the ME, but is also skewing future samples based upon past experience. However, since during Phase II, the ME has not been sampled enough to give the adaptor a good picture of its contents (so that, initially, the skewed samples are as likely to be in error as not), the assumption of random sampling does not seem unreasonable. At any rate, the results derived on the basis of this assumption should give a good picture of the relative influence of Phase II on meta-environments of different depths.

The ME array for the class of simulated meta-environments with which we have been concerned has exactly $p = 8$ columns and $v^d = 8^d$ rows for a depth d simulated meta-environment. In addition, it possesses a certain statistical regularity in its columns. Each column has precisely one quarter of its elements equal to one, one quarter of its elements equal to two, etc. If parameter value vectors were to be generated randomly, and an ME array of the type under consideration here used to

assign values to them, it is clear that each of the v^d elements in a column has an equal chance of being included in the sum that is formed to assign the value. Thus over a sufficiently long number of trials n, each element will be sampled on the average, n/v^d times. Each time that it is sampled, it is sampled in the company of p-1 other elements and the adaptor is informed only of the sum of the p elements. The question that we wish to answer is "How many times must an individual element be sampled before the adaptor can determine its value?" Suppose that we are concerned with discovering the value of an element of the ME array, E_{ij}. We may obtain information about the value of E_{ij} by examining the utility values of parameter value vectors having value i for parameter j. The utility values of these parameter value vectors may be considered to be composed of two parts: $E_{ij} = f_j(v_i)$ which is constant and a random variable which consists of the sum of p-1 individual random variables each which have identical means and variances. These individual random variables are produced by the random sampling on the p-1 other columns. Each has a mean equal to the ME array column mean: $\sum_{i=1}^{4} .25(i) = 2.5$ and a variance equal to the ME array column variance: $\sum_{i=1}^{4} (2.5-i)^2 = 1.25$.

If a large number of parameter value vectors is evaluated, the mean value of the resulting utility will be very close to the value E_{ij} plus p-1 times the column mean. To be more precise, the situation herein described allows application of the Central Limit Theorem so that we may state that if n is the number of trials required to obtain enough information to determine the value of E_{ij} within ε percent with a probability equal to p, then:

$$n = v^d \left(\frac{125}{\varepsilon}\right)^2 x^2$$

where x is the root of $\Phi(x) - \Phi(-x) = p$ and $\Phi(x)$ is the normal distribution function.

Table 6.3.1 gives the results of computations using the above expressions and various values of d, ϵ, and p. The numbers given here must be interpreted with caution since the object of the adaptor is to maximize the meta-environment rather than to explore it and maximization occurs on the basis of a sampling which is presumably far more efficient than random sampling. In spite of these warnings, it is evident that Phase II considerations make it very unlikely that the correlation adaptor will be useful for meta-environments of depth greater than three or four since here again, the number of trials required to complete the phase is greater than the expected number of trials required by a random adaptor to maximize the meta-environment.

6.4 Phase III Considerations — General

In Phase III, not only is the experience array filled in with values (as a result of Phase I), but those values represent the meta-environment as well as it can be represented (as a result of Phase II). Thus we shall assume that in Phase III, the elements of the experience array remain substantially unchanged as a result of the ongoing sampling. We shall proceed by determining what those values are and then using them to compute the number of trials required to attain the maximum utility value for various values of a_2.

The next sections contain both general and particular results dealing with Phase III and they are presented in an order which proceeds from the simplest cases to the more complicated. The first two sections are concerned with the level one adaptor and consider it first immersed in a depth one ME and then in MEs of higher depth. Within the sections, general results are first formulated and then applied to the particular simulated ME which we have described previously. The final section which

MEAN NUMBER OF TRIALS REQUIRED TO ESTIMATE SUB-UTILITY

DEPTH = 1

CONFIDENCE LEVEL	PERCENT ESTIMATION ERROR 50.0	25.0	10.0	5.0	1.0
0.500	22.	90.	568.	2274.	56868.
0.750	66.	264.	1653.	6615.	165398.
0.900	135.	541.	3382.	13528.	338211.
0.950	192.	768.	4801.	19207.	480199.
0.990	331.	1326.	8293.	33173.	829343.

DEPTH = 2

CONFIDENCE LEVEL	PERCENT ESTIMATION ERROR 50.0	25.0	10.0	5.0	1.0
0.500	181.	727.	4549.	18198.	454950.
0.750	529.	2117.	13231.	52927.	1323189.
0.900	1082.	4329.	27056.	108227.	2705695.
0.950	1536.	6146.	38415.	153663.	3841599.
0.990	2653.	10615.	66347.	265389.	6634745.

DEPTH = 3

CONFIDENCE LEVEL	PERCENT ESTIMATION ERROR 50.0	25.0	10.0	5.0	1.0
0.500	1455.	5823.	36396.	145584.	3639601.
0.750	4234.	16936.	105855.	423420.	10585514.
0.900	8658.	34632.	216455.	865822.	21645560.
0.950	12293.	49172.	307327.	1229311.	30732796.
0.990	21231.	84924.	530779.	2123118.	53077960.

TABLE 6.3.1 PHASE II COMPUTATIONS (PART 1)

MEAN NUMBER OF TRIALS REQUIRED TO ESTIMATE SUB-UTILITY

DEPTH = 4

CONFIDENCE LEVEL	PERCENT ESTIMATION ERROR 50.0	25.0	10.0	5.0	1.0
0.500	11646.	46586.	291168.	1164672.	29116808.
0.750	33873.	135494.	846841.	3387365.	84684112.
0.900	69265.	277063.	1731644.	6926579.	*********
0.950	98344.	393379.	2458623.	9834494.	*********
0.990	169849.	679398.	4246236.	16984944.	*********

DEPTH = 5

CONFIDENCE LEVEL	PERCENT ESTIMATION ERROR 50.0	25.0	10.0	5.0	1.0
0.500	93173.	372695.	2329345.	9317382.	*********
0.750	270989.	1083956.	6774730.	27098920.	*********
0.900	554126.	2216505.	13853158.	55412632.	*********
0.950	786759.	3147037.	19668988.	78675952.	*********
0.990	1358796.	5435184.	33969888.	*********	*********

TABLE 6.3.1 PHASE II COMPUTATIONS (PART 2)

concerns higher level adaptors, follows the same pattern proceeding from the general result to the particular application.

6.5 The Level One Adaptor vs Depth One MEs

Recall that the probability P_{ij} that a given parameter P_j will assume a given value V_i is given by the row selection algorithm of the level one adaptor. It depends only upon the elements of the j^{th} column of the X array and is given by the following expression:

$$P_{ij} = \frac{x_{ij} + a_2}{\sum_{i=1}^{v}(x_{ij}+a_2)}$$

Let us look more carefully at a specific element of the X array x_{ij}. The value of this element reflects the average utility returned by all of the parameters value vectors which include value i for parameter j. Because, at this point, we have limited our concern to depth one MEs, we know that the utility of a parameter value vector is obtained by summing p sub-utility values. These sub-utility values are obtained by evaluating the individual parameters values in the component functions. This is stated more simply in the following expression:

$$f(P_1, P_2, \ldots, P_p) = f_1(P_1) + f_2(P_2) + \ldots + f_p(P_p).$$

In our consideration of the element x_{ij}, we note that all of the parameter value vectors that are relevant to this element have one common feature. They all include value i for parameter j so that the sums that make up their utility values all must include the element $f_j(v_i)$. Thus, we may write:

$$x_{ij} = Q + R \text{ where } Q = f_j(v_i)$$

where Q includes the term(s) peculiar to the particular element x_{ij} and R is a remainder term necessary because utility is associated with entire

PVV and not individual parameter tuples.

Let us explore R in more detail for the depth one case. For any individual parameter value vector, it consists of the values of the other p-1 sub-utility functions (all of those except the j^{th}) evaluated for the particular values of the parameters to which they correspond. Since we have assumed that the X array remains constant (Phase III) and the form of the generation algorithm is known, we can compute the proportion of trials for which each particular value will be present. More concretely: consider a subfunction $f_k(P_k)(k \neq j)$. P_k will assume a value v_m with a probability equal to:

$$\frac{x_{mk} + a_2}{\sum_{\ell=1}^{v}(x_{\ell k}+a_2)}$$

and when it does, its contribution to the total utility value will be just $f_k(v_m)$.

Weighting each of these values by the probability of its occurrence and summing, we may obtain the average contribution of the k^{th} sub-utility function to the total utility value:

$$\overline{f_k(P_k)} = \sum_{m=1}^{v} \frac{f_k(v_m^k)x_{mk}+a_2)}{\sum_{\ell=1}^{v}(x_{\ell k}+a_2)}$$

The value of R is given by summing the average contributions of all of the remaining sub-utility functions:

$$R = \sum_{\substack{k=1 \\ k\neq j}}^{p} \overline{f_k(P_k)} \ .$$

Combining the above results, we may write out the complete expression for x_{ij}:

$$x_{ij} = f_j(v_i) + \sum_{\substack{k=1 \\ k\neq j}}^{p} \sum_{m=1}^{v} \frac{f_k(v_m^k)(x_{mk}+a_2)}{\sum_{\ell=1}^{v}(x_{\ell k}+a_2)} \quad .$$

At this point, let us examine what we have and where we are going. We would like to be able to predict the number of trials for a level one adaptor to maximize given depth one ME. We have assumed that the X array is in Phase III and have obtained a set of equations relating the values of the elements of that X array. The right hand side of the equation includes terms of the form $f_j(v_i)$ which are just numbers whose values may be computed from the definition of the given ME. This leaves us with a set of p·v equations in p·v unknowns. When these have been solved, we shall know the contents of the X array and using the generation algorithm, we may then compute the probability of any particular PVV being generated and in particular those PVV's which elicit maximum utility from the ME. Thus given any particular depth one ME, we may indeed, using the procedure outlined here, compute the probability that a level one adaptor will maximize its utility. But this is not really very satisfying, so let us introduce the special case of our simulated meta-environments to see if we can obtain more specific results.

Recall that p = v = 8 so that we are dealing with 64 equations in 64 unknowns. Then note that each column was generated in a uniform manner so that there are no statistical differences in the columns. This enables us to replace the expression for R by the following:

$$R = 7 \sum_{m=1}^{8} \frac{f_k(v_m)(x_{mk}+a_2)}{\sum_{\ell=1}^{8}(x_{\ell k}+a_2)}$$

where the k is a constant value (from one to eight), the columns are uniform. Note that R is now independent of i and j so that:

$$x_{ij} = f_j(v_i) + R(X)$$

This brings out very clearly the relationship between the values of the X array of the adaptor and the values of the ME array of the simulated ME (i.e., $f_j(v_i)$). In fact, there can be only as many distinct values of x_{ij} as there are of $f_j(v_i)$. But by the manner in which the ME array has been constructed, there are only 4 distinct values of the depth one ME array (1, 2, 3 and 4) so that there can be only four distinct values of x_{ij} and we have reduced the size of the problem to four equations in four unknowns.

We may now simplify our notation to reflect the simplified situation by letting u_s (s = 1, ..., 4) to be equal to the value of x_{ij} corresponding to $f_j(v_i)$ when $f_j(v_i)$ is equal to s. This enables us to write:

$$u_1 = 1 + R$$

$$u_2 = 2 + R = 1 + u_1$$

$$u_3 = 3 + R = 2 + u_1$$

$$u_4 = 4 + R = 3 + u_1$$

$$\text{and} \quad R = \left(\sum_{s=1}^{4} \frac{s(u_s+a_2)}{u_1 + u_2 + u_3 + u_4 + 4a_2}\right)$$

Substituting into the expression for R and solving for u_1, we obtain:

$$u_1 = \frac{(17-a_2) + \sqrt{(17-a_2)^2 + 146 + 74a_2}}{2}$$

Having obtained the values of u_s, we may compute the probability that in any trial, the value of a sub-utility function will be k. This is given by the row selection algorithm:

$$q_k = \frac{u_k}{u_1 + u_2 + u_3 + u_4} \qquad (6.5.1)$$

The probability that a maximum utility value (32) will be attained in any one trial is just $(q_4)^8$ since all of the sub-utility values must be equal to 4. The mean number of trials required to obtain the maximum is just the reciprocal of this number or $\frac{1}{(q_4)^8}$.

We should like to be able to examine the effect of variations of a_2 upon the success of the level one adaptor. We have chosen as a measure of success the mean number of trials required to reach a maximum value. A somewhat more detailed picture of the effects of a_2 may be obtained if we compute a density function, that is a plot of utility values versus the probability of attaining them. In order to do this, we note that the distribution function of utility values is identical with the distribution function of a random variable formed in the following manner. Suppose that we are given 8 urns each of which contains a number of balls. Each ball has one of the numbers 1, 2, 3 or 4 inscribed on it and each urn contains balls in such a proportion that the probability of withdrawing a ball with the number k on it is just q_k. Let one ball be drawn from each urn and the values inscribed upon the balls be summed. The distribution of the sums is given by Feller [1957, p. 266]. We have devised an equivalent algorithm which is computationally more suitable and have computed the distribution and mean number of trials to attain all utility values for various values of a_2. The results of these computations are given in Tables 6.5.1 through 6.5.7.

The dependence of the results upon the value of a_2 can be seen in the preceding results. Let us now determine the limiting-value which a_2 may assume. Clearly $u_1 < u_2 < u_3 < u_4$ and accordingly $q_1 < q_2 < q_3 < q_4$. From the form of the equation, it may be seen that u_1 is a monatonic function of a_2 and as a_2 decreases so does u_1. But if $u_1 + a_2$ becomes

RESULTS FOR A2= 35.00

SCORE	PROBABILITY	EXPECTED TRIALS TO FIRST ENCOUNTER	EXPECTED ENCOUNTERS IN 64K TRIALS
8	0.00001223	81709.	0.
9	0.00009973	10026.	6.
10	0.00045712	2187.	29.
11	0.00155192	644.	101.
12	0.00424116	235.	277.
13	0.00976537	102.	639.
14	0.01950757	51.	1278.
15	0.03439010	29.	2253.
16	0.05416166	18.	3549.
17	0.07686837	13.	5037.
18	0.09884601	10.	6477.
19	0.11561767	8.	7577.
20	0.12327955	8.	8079.
21	0.11988775	8.	7856.
22	0.10628218	9.	6965.
23	0.08570370	11.	5616.
24	0.06261733	15.	4103.
25	0.04122747	24.	2701.
26	0.02424975	41.	1589.
27	0.01258761	79.	824.
28	0.00566878	176.	371.
29	0.00215093	464.	140.
30	0.00065696	1522.	43.
31	0.00014862	6728.	9.
32	0.00001891	52876.	1.

AVERAGE VALUE = 20.181

TABLE 6.5.1 CA(1) VS ME(1) WITH A2 = 35

RESULTS FOR A2= 20.00

SCORE	PROBABILITY	EXPECTED TRIALS TO FIRST ENCOUNTER	EXPECTED ENCOUNTERS IN 64K TRIALS
8	0.00001125	88826.	0.
9	0.00009238	10823.	6.
10	0.00042643	2345.	27.
11	0.00145774	685.	95.
12	0.00401124	249.	262.
13	0.00929946	107.	609.
14	0.01870414	53.	1225.
15	0.03319910	30.	2175.
16	0.05264253	18.	3449.
17	0.07522088	13.	4929.
18	0.09738466	10.	6382.
19	0.11468085	8.	7515.
20	0.12310849	8.	8068.
21	0.12053057	8.	7899.
22	0.10757295	9.	7049.
23	0.08732874	11.	5723.
24	0.06423357	15.	4209.
25	0.04257532	23.	2790.
26	0.02521018	39.	1652.
27	0.01317354	75.	863.
28	0.00597215	167.	391.
29	0.00228107	438.	149.
30	0.00070131	1425.	45.
31	0.00015969	6261.	10.
32	0.00002045	48893.	1.

AVERAGE VALUE = 20.248

TABLE 6.5.2 CA(1) VS ME(1) WITH A2 = 20.

RESULTS FOR A2= 4.00

SCORE	PROBABILITY	EXPECTED TRIALS TO FIRST ENCOUNTER	EXPECTED ENCOUNTERS IN 64K TRIALS
8	0.00000917	108967.	0.
9	0.00007662	13050.	5.
10	0.00035977	2779.	23.
11	0.00125073	799.	81.
12	0.00349961	285.	229.
13	0.00824933	121.	540.
14	0.01686878	59.	1105.
15	0.03043904	32.	1994.
16	0.04906556	20.	3215.
17	0.07126744	14.	4670.
18	0.09378568	10.	6146.
19	0.11225633	8.	7356.
20	0.12247936	8.	8026.
21	0.12187346	8.	7987.
22	0.11054348	9.	7244.
23	0.09119819	10.	5976.
24	0.06816643	14.	4467.
25	0.04591175	21.	3008.
26	0.02762332	36.	1810.
27	0.01466598	68.	961.
28	0.00675481	148.	442.
29	0.00262096	381.	171.
30	0.00081851	1221.	53.
31	0.00018927	5283.	12.
32	0.00002461	40631.	1.

AVERAGE VALUE = 20.410

TABLE 6.5.3 CA(1) VS ME(1) WITH A2 = 4

RESULTS FOR A2= 0.00

SCORE	PROBABILITY	EXPECTED TRIALS TO FIRST ENCOUNTER	EXPECTED ENCOUNTERS IN 64K TRIALS
8	0.00000829	120627.	0.
9	0.00006982	14321.	4.
10	0.00033062	3024.	21.
11	0.00115901	862.	75.
12	0.00326987	305.	214.
13	0.00777124	128.	509.
14	0.01602114	62.	1049.
15	0.02914481	34.	1910.
16	0.04736001	21.	3103.
17	0.06934508	14.	4544.
18	0.09198936	10.	6028.
19	0.11098787	9.	7273.
20	0.12206141	8.	7999.
21	0.12242317	8.	8023.
22	0.11192160	8.	7334.
23	0.09306372	10.	6099.
24	0.07010763	14.	4594.
25	0.04758874	21.	3118.
26	0.02885536	34.	1891.
27	0.01543884	64.	1011.
28	0.00716552	139.	469.
29	0.00280157	356.	183.
30	0.00088153	1134.	57.
31	0.00020536	4869.	13.
32	0.00002689	37179.	1.

AVERAGE VALUE = 20.489

TABLE 6.5.4 CA(1) VS ME(1) WITH A2 = 0

RESULTS FOR A2= -4.00

SCORE	PROBABILITY	EXPECTED TRIALS TO FIRST ENCOUNTER	EXPECTED ENCOUNTERS IN 64K TRIALS
8	0.00000712	140283.	0.
9	0.00006082	16441.	3.
10	0.00029165	3428.	19.
11	0.00103511	966.	67.
12	0.00295616	338.	193.
13	0.00711117	140.	466.
14	0.01483733	67.	972.
15	0.02731517	36.	1790.
16	0.04491656	22.	2943.
17	0.06654809	15.	4361.
18	0.08932220	11.	5853.
19	0.10903757	9.	7145.
20	0.12132078	8.	7950.
21	0.12309922	8.	8067.
22	0.11384601	8.	7461.
23	0.09575775	10.	6275.
24	0.07296679	13.	4781.
25	0.05009613	19.	3283.
26	0.03072127	32.	2013.
27	0.01662300	60.	1089.
28	0.00780161	128.	511.
29	0.00308416	324.	202.
30	0.00098111	1019.	64.
31	0.00023100	4328.	15.
32	0.00003057	32710.	2.

AVERAGE VALUE = 20.604

TABLE 6.5.5 CA(1) VS ME(1) WITH A2 = -4

RESULTS FOR A2= -20.00

SCORE	PROBABILITY	EXPECTED TRIALS TO FIRST ENCOUNTER	EXPECTED ENCOUNTERS IN 64K TRIALS
8	0.00000005	18808716.	0.
9	0.00000071	1394599.	0.
10	0.00000523	190849.	0.
11	0.00002747	36402.	1.
12	0.00011399	8772.	7.
13	0.00039362	2540.	25.
14	0.00116647	857.	76.
15	0.00302468	330.	198.
16	0.00695392	143.	455.
17	0.01430675	69.	937.
18	0.02650405	37.	1736.
19	0.04439533	22.	2909.
20	0.06739862	14.	4417.
21	0.09280729	10.	6082.
22	0.11583851	8.	7591.
23	0.13076782	7.	8570.
24	0.13298022	7.	8714.
25	0.12108781	8.	7935.
26	0.09784707	10.	6412.
27	0.06926405	14.	4539.
28	0.04215220	23.	2762.
29	0.02140326	46.	1402.
30	0.00863311	115.	565.
31	0.00252008	396.	165.
32	0.00040610	2462.	26.

AVERAGE VALUE = 23.380

TABLE 6.5.6 CA(1) VS ME(1) WITH A2 =-20

RESULTS FOR A2= -24.00

SCORE	PROBABILITY	EXPECTED TRIALS TO FIRST ENCOUNTER	EXPECTED ENCOUNTERS IN 64K TRIALS
8	0.00000000	3883822431076351.	0.
9	0.00000000	320623922954 23.	0.
10	0.00000000	583697827327.	0.
11	0.00000000	17687244855.	0.
12	0.00000000	797781761.	0.
13	0.00000001	50464992.	0.
14	0.00000023	4309855.	0.
15	0.00000207	482820.	0.
16	0.00001445	69171.	0.
17	0.00008080	12375.	5.
18	0.00036922	2708.	24.
19	0.00140238	713.	91.
20	0.00448352	223.	293.
21	0.01217333	82.	797.
22	0.02823077	35.	1850.
23	0.05606836	17.	3674.
24	0.09534391	10.	6248.
25	0.13833868	7.	9066.
26	0.17004516	5.	11144.
27	0.17493623	5.	11464.
28	0.14777177	6.	9684.
29	0.09938044	10.	6512.
30	0.05053711	19.	3312.
31	0.01756423	56.	1151.
32	0.00325580	307.	213.

AVERAGE VALUE = 26.366

TABLE 6.5.7 CA(1) VS ME(1) WITH A2 =-24

negative, then q_1 is also negative and we may no longer interpret it as a probability. The minimum allowable value of a_2 may be computed by setting u_1 equal to $-a_2$ in the expression for u_1 given previously. Solving for a_2 yields:

$$a_2 = -24\ ^1/_3\ .$$

Recall that a_2 may be interpreted as a scaling factor which determines the amount of reliance which the adaptor places upon the information contained in the X array. The larger (more positive) a_2 is, the more it dilutes the effects of the X array and the more the results resemble random sampling. On the other hand, the smaller (more negative) a_2 is, the more skewed the results are because more weight is given to accumulated experience.

A natural extension of the correlation algorithm as given is to apply feedback to the situation so that as more experience is recorded in the X array, more reliance is placed upon it. Since the values in the X array should rise as experience is accumulated, one way of implementing feedback is to let a_2 decrease (become more negative) by letting a_2 be slightly greater than the negative of the smallest x_{ij} value in the array. This permits the maximum feedback while avoiding negative values of q. The effect is easily computed:

$$\text{set} \qquad a_2 = (-u_1) + \varepsilon \qquad \text{where} \quad \varepsilon > 0$$

$$\text{then } p_4 = \frac{u_1 + a_2 + 3}{4_{u_1} + 6 + 4_{a_2}} = \frac{3 + \varepsilon}{6 + 4\varepsilon}\ .$$

The results in terms of number of trials required to reach a maximum are given in Table 6.5.8. The use of such feedback clearly cuts across our neat definitions of phase boundaries since it allows for Phase III PVV generation to occur while Phase II experience is being gathered.

ε	$(\frac{1}{P_4})^8$
0	256
1	1526
2	3778

Table 6.5.8 Trials Required for CA(1) to Maximize ME(1) as a Function of ε

The basic Phase III assumption is that the values in the X array are constant and yet the feedback principle is predicated on the assumption that these values do indeed change. In practice, the feedback should not be applied until Phase II has been completed because its effect is to severely limit trials of tuples which have previously done poorly. But the apparently poor performance history of a tuple may be only a consequence of its having been sampled in the company of other inherently poor tuples. Only when the EA has proceeded sufficiently far into Phase II can we be confident that the tuple has been sampled adequately to allow its worth to be estimated. The feedback loop can then be closed to discriminate against tuples which are inefficient in gathering utility. This feedback principle will be applied extensively in the succeeding sections, but the results obtained must be considered to be lower bounds on Phase III performance.

6.6 The Level One Adaptor vs Higher Depth MEs

Starting with the depth two case we proceed in a manner which is precisely analogous with the depth one case. Let the elements of the ME function be denoted as follows:

$$f(P_1, P_2, \ldots, P_p) = f_{1,2}(P_1,P_2) + f_{1,3}(P_1,P_3) + \ldots + f_{p-1,p}(P_{p-1},P_p)$$

We may again divide the expression for x_{ij} into two parts one of which

(Q) consists of the contributions of the subfunctions which have value i for parameter j and the other (R) in which the contributions of the remaining subfunctions which are included:

$$x_{ij} = Q(i,j) + R$$

where:

$$Q = \sum_{\ell=1}^{v}\left(\sum_{k=1}^{j-1}\frac{f_{kj}(v_\ell,v_i)(x_{\ell k}+a_2)}{\sum_{m=1}^{v}(x_{mk}+a_2)} + \sum_{k=j+1}^{p}\frac{f_{jk}(v_i,v_\ell)(x_{\ell k}+a_2)}{\sum_{m=1}^{v}(x_{mk}+a_2)}\right)$$

and

$$R = \sum_{\substack{k=1\\k\neq j}}^{p-1}\sum_{\substack{\ell=k+1\\\ell\neq j}}^{p}\left(\sum_{m=1}^{v}\sum_{n=1}^{v}\frac{f_{k\ell}(v_m,v_n)(x_{mk}+a_2)(x_{n\ell}+a_2)}{\sum_{p=1}^{v}(x_{pk}+a_2)\sum_{q=1}^{v}(x_{qk}+a_2)}\right)$$

At this point, since the general pattern is apparent, the reader should not doubt that it would require a page and a half to contain the complete expression for the depth three meta-environment and that there would be little profit in the exercise.

We can now take a large step and develop an expression for the X values of level one adaptor immersed in a general depth d meta-environment. This consists essentially of an exapansion of the R component of the expressions previously obtained. It holds, of course, for meta-environments of all depths but does not take account of the simplifications which can be introduced for the more constrained forms of meta-environment.

$$x_{ij} = \sum_{k_1=1}^{v}\sum_{k_2=1}^{v}\cdots\sum_{k_{j-1}=1}^{v}\sum_{k_{j+1}=1}^{v}\cdots\sum_{k_p=1}^{v}\left\{f\left(v^1_{k_1}, v^2_{k_2}, \ldots, v^{j-1}_{k_{j-1}}, v^j_i, \ldots, v^p_{k_p}\right)\left(\prod_{\substack{\ell=1\\\ell\neq j}}^{p}\frac{(x_{k_\ell\ell}+a_2)}{\sum_{m=1}^{v}(x_{m\ell}+a_2)}\right)\right\} \qquad (6.6.8)$$

We next proceed with an attempt to obtain some more concrete results for the special case of depth two simulated meta-environment. We shall follow the same steps as in the depth one case and solve for the values in the X array introducing simplifying assumptions as needed. We shall follow this same procedure in subsequent sections.

First of all, $p = v = 8$ but there are only 8 (as opposed to 28) non-zero columns in the ME array (although there are 64 rows). Recall that the columns in the ME array have been chosen in such a manner that each parameter is represented in precisely two columns.

We then plunge right ahead and evaluate the Q part of the expression for x_{ij} since it is here where the differences between the elements of the X array are to be found. Recall that Q is composed of those elements of the ME array which are consulted by the ME because the PVV has value i for parameter j. In the depth one case, this led to a single element $(f_j(v_i))$ in the ME array. In the depth two case, it leads to many elements in the ME array since the precise value returned by a sub-utility function depends not only upon the fact that v_i is present for P_j but also upon the value of the accompanying parameter. In particular, there are precisely sixteen elements of the ME array which are relevant. The sixteen are obtained in the following manner: There are two columns in which P_j occurs (by the manner in which the ME array is generated) and in each of these, v_i is present in eight rows in an element $(f_{kj}(v_\ell, v_i)$ or $f_{jk}(v_i, v_\ell))$ in which it is accompanied by one of the eight values of the other parameter of that column. Since our aim is to simplify and obtain specific results, let us determine which assumptions we may make about these sixteen elements. The most natural assumption, since it is a depth two ME, is that there is no reason for these particular sixteen

elements to be different from any other sixteen elements. Then the weighted average of the elements which goes to make up Q_{ij} would be equal to the column average (which is just 2.5). But this same assumption holds for all i and j so that Q is invariant over i and j as is R and thus we have an X array which is completely uniform. This turns the correlation adaptor into a random adaptor (i.e., it chooses values for each parameter at random from a uniform distribution). All of which is not really very surprising, for it all follows from the assumption that there was no depth one influence whatsoever in the ME. It is at least comforting to realize that in this rather extreme instance of mismatch, the adaptor does no worse than random. This is true only in terms of the number of trials required to find a maximum. It does do worse in the sense that the cost of the level one adaptor (measured by the amount of storage it requires, the time per trial or some other criterion) is probably greater than that of the random adaptor. In that sense, the performance of the overmatched correlation adaptor is worse than random.

On the other hand, we may wish to specify values for Q_{ij}, but in doing so, we shall be in effect making assumptions about the ME array which will be reflected back into the depth of the ME itself. The most rash assumption is that the sixteen elements which form a single Q_{ij} all have the same value which makes our computation much easier since there will be only four distinct values for Q_{ij} (1, 2, 3 or 4). But a careful examination will reveal that we have reverted to a depth one situation and although the ME array has depth two form, the function it describes is actually of depth one.

In essence, there are no general assumptions that can be made which will simplify the equations given above and still yield results valid

for any meaningful class of level one adaptors immensed in depth two meta-environments.

6.7 Implications for Real-World Meta-Environments

In our definition of meta-environmental depth, we made no allowance for mixed MEs — if an ME has a single pair of parameters which interact while all of the rest are orthogonal, it is classified as depth two. It is now evident that this is too rigid a classification and that the cases of mixed MEs must be treated. Suppose that it could be determined that half of the parameters interacted in pairs while the rest were orthogonal. For the latter, the u values would be as given above for the depth one case as would the value of q_4, while for the former the effect of random sampling would be to give $q_4 = 0.25$ so the expected number of trials to reach a maximum is just

$$N = \left(\frac{1}{q_4}\right)^4 (4)^4 . \qquad (6.7.1)$$

Similarly, if the ME is Z percent depth one and (100-Z) percent depth two, the result is simply:

$$N = \left(\frac{1}{q_4}\right)^{\left(\frac{Z}{100} - 8\right)} (4)^{\left(1 - \frac{Z}{100}\right)8} \qquad (6.7.2)$$

The remarks given here may easily be generalized to apply to an adaptor of any level which is immensed in a ME of greater depth.

However, even with the above concession to reality, we are still some distance from a "natural" meta-environment. In previous sections, we have assumed that if parameters interact at a depth d, the best that an adaptor of level d-1 can do is to sample at random. But this is not true since the parameters may be "almost" of depth d-1. That is, they must be formally described as being of depth d, but for most values,

they act as if they had only d-1 dependence. This situation may be reflected in the computation by adding additional terms which describe the degree to which parameters interact at various depths. The effect of this will be a change in the value of q_4 which will vary from a maximum value given by the matched case to a minimum (0.25) for the completely mismatched case. This may be expressed by a modification of the exponent Z given in the previous equation.

Recall that Z describes the degree of match with a level 1 adaptor. We may extend its meaning by defining a whole spectrum of exponents $Z_1 \leq Z_2 \leq \ldots 1$ which may be defined in a manner analogous with Z_1 whose definition is as follows:

Compute the x_{ij} values using the general equation 6.6.8 or any of its simplified versions which are applicable to the ME at hand and an appropriate value of a_2. Then using these values, compute the probability of attaining a maximum using the obvious extension of equation 6.5.1. Use the reciprocal of that probability for N and solve equation 6.7.2 for Z. The value obtained is Z_1. In a similar manner, Z_2, Z_3, etc. may be computed and the result, the vector $\overline{Z}$ is a much more accurate description of the hostility of a meta-environment than the scalar depth.

6.8 Adaptors with Level Greater than One

We shall begin with a consideration of the level two adaptor. Formally, writing the equations for higher level adaptors is a straightforward extension of the work of level one adaptors with the inclusion of a term to account for the process of column selection.

We first consider the case of the level two adaptor immensed in a depth one ME. We begin by examining the column selection function.

As indicated in the description of the correlation algorithm, the column selection function is dependent not only upon the elements of the X array which are in the column to which it refers, but also to the remainder of the X array. Accordingly, let $\overline{X}$ be the matrix consisting of the elements of the X array. We may then represent the column selection algorithm by the family of probabilities which it generates in the following manner:

$C_{ij}(\overline{X})$ is the probability that column ij (corresponding to the parameters P_i and P_j) will be selected given the state of the X array - $\overline{X}$. The extension of this notation to levels greater than two is obvious.

Proceeding in a manner which parallels that used in the level one situation we now write the equations for the level two adaptor immersed in a depth one meta-environment. (We may assume without loss of generality that $j_2 > j_1$.)

$$x_{i_1 i_2 j_1 j_2} = f_{j_1}(v_{i_1}) + f_{j_2}(v_{i_2}) + \sum_{k_1=1}^{p-1} \sum_{k_2=k_1+1}^{p} \left(\sum_{m_1=1}^{v} \sum_{m_2=1}^{v} \xi(\overline{X}) \right)$$
$$k_1 \neq j_1 \quad k_2 \neq j_2$$

where

$$\xi(\overline{X}) = (f_{k_1}(v_{m_1}) + f_{k_2}(v_{m_2})) \frac{(x_{m_1 m_2 k_1 k_2} + a_2)}{\sum_{\ell_1=1}^{v} \sum_{\ell_2=1}^{v} (x_{\ell_1 \ell_2 k_1 k_2} + a_2)} C_{k_1 k_2}(\overline{X}) \quad .$$

In a like manner, we proceed to the depth two case.

$$x_{i_1 i_2 j_1 j_2} = f_{j_1 j_2}(v_{i_1}, v_{i_2}) + \sum_{k_1=1}^{p-1} \sum_{k_2=k_1+1}^{p} \left(\sum_{m_1=1}^{v} \sum_{m_2=1}^{v} \Delta(\overline{X}) \right)$$

where

$$\Delta(\overline{X}) = f_{k_1 k_2}(v_{m_1}, v_{m_2}) \frac{(x_{m_1 m_2 k_1 k_2} + a_2)}{\sum_{\ell_1=1}^{v} \sum_{\ell_2=1}^{v} (x_{\ell_1 \ell_2 k_1 k_2} + a_2)} \cdot C_{k_1 k_2}(\overline{X}) \quad .$$

At this point, the pattern should be clear and we may proceed to the straightforward extention to the general depth d case for the level two adaptor.

$$x_{i_1 i_2 j_1 j_2} = \sum_{k_1=1}^{v} \cdots \sum_{k_{j_1-1}=1}^{v} \sum_{k_{j_1+1}=1}^{v} \cdots \sum_{k_{j_2-1}=1}^{v} \sum_{k_{j_2+1}=1}^{v} \cdots \sum_{k_p=1}^{v} \Gamma(\bar{X})$$

where

$$\Gamma(\bar{X}) = f(v_{k_1}, \ldots, v_{k_{j_1-1}}^{j_1-1}, v_{i_1}^{j_1}, v_{k_{j_1+1}}^{j_1+1}, \ldots, v_{k_{j_2-1}}^{j_2-1}, v_{i_2}^{j_2}, v_{k_{j_2+1}}^{j_2+1}, \ldots, v_{k_p}^{p}) \cdot z(\bar{X})$$

where

$$z(\bar{X}) = \prod_{\substack{\ell_1=1 \\ \ell_1 \neq j_1}}^{p-1} \prod_{\substack{\ell_2=\ell_1+1 \\ \ell_2 \neq j_2}}^{p} \frac{(x_{k_{\ell_1} k_{\ell_2} \ell_1 \ell_2}^{+a_2}) \, C_{\ell_1 \ell_2}(\bar{X})}{\sum_{m_1=1}^{v} \sum_{m_2=1}^{v} (x_{m_1 m_2 \ell_1 \ell_2}^{+a_2})}$$

The extension to higher level adaptors is quite straightforward and it seems pointless to use any more paper to bore the reader with more unwieldy expressions. Briefly, reviewing we find that we have displayed sets of equations which when solved would enable us to compute the Phase III state of the X array of levels one and two adaptors which immersed in MEs of any depth. Once the contents of the X array is known, it is a simple (if lengthy) task to compute the mean number of trials required for the adaptor to elicit a maximum values of utility from the meta-environment.

6.9 Applications of Higher Level Adaptors

Following the pattern of previous sections, we shall now apply the results just developed to the special case of the simulated meta-environment. Our goal is to develop an expression for the probability that a correlation adaptor will generate a successful PVV (one which will elicit a maximum utility) when immersed in a simulated

meta-environment. We make the usual assumptions that p = v = 8 and that the ME array has been randomly selected as described above.

The first complication that we are faced with is that of column selection, which was casually circumvented in the previous section by the use of an appropriate notation. In order to attack column selection, we will introduce two additional concepts — freedom and relevance. They will be given a precise definition later on and column selection (and eventually the probability of success) will be expressed in these terms.

Recall that the adaptor must make a selection from among the columns of its X array because there are more columns than there are parameter ℓ-tuples (for levels greater than one). The generation section of the adaptive algorithm works in the following manner: First a column is selected and then values are selected for the parameters represented in that column. Then another column is selected. But because of the fact that a parameter may appear in many columns, certain complications arise. The second column selected may contain some parameters whose values were already chosen as the result of the first column selection. The row selector algorithm must then confine its choices to those value tuples which include the previously chosen value(s) for the parameters that were represented in the first column selection. In the sections that follow, we shall use the notion of freedom to describe this phenomenon. That is, a column which has been selected by the column selector and which contains no parameters which have not appeared in columns selected previously will be said to have zero free parameters while a column which contains a single parameter which has not been previously selected will be said to have one free parameter and so on.

Next, we shall introduce the concept of relevance. Relevance concerns the number of columns of the ME array that are reflected in

the makeup of a column of the X array. Consider the level one adaptor immersed in a depth one ME. In that situation, to each column of the X array, there corresponds a single column in the ME array and if we define R(r) to be the probability that the number of ME array columns relevant to a given X array column is r, then R(1) = 1. However, this situation is no longer true for the level two adaptor because there are two columns in the depth one ME array which correspond to each column in the level two X array and R(2) = 1. On the other hand, for the level 2 adaptor immersed in a depth 2 ME since there are only p columns in the depth two ME array, but $\binom{p}{2}$ combinations of parameters, either zero or one columns of the ME array may be relevant to a randomly selected column of the depth two X array and $R(1) = p/\binom{p}{2}$ with R(0) = 1 - R(1). This concept may easily be extended for various depths and levels in the obvious manner.

Our plan is to give more constructive definitions to the two concepts of freedom and relevance and use them to compute the mean number of trials required for correlation adaptors of various levels to maximize meta-environments of various depths. To do so, we shall first compute QT(F), the average number of times that the column selector will choose a column with F free parameters. Next, we shall compute the relevance distribution R(r) as defined above. Then, we shall develop an expression for P(S/r,F), the probability that the row selector will choose a parameter value tuple that contributes to the attainment of an optimizing PVV given that the column selector has chosen a column with relevance r and freedom F. Finally, these will be combined to give P(S), the probability that a successful PVV will be generated on any given trial. The reciprocal of this value is the desired expected number of trials required to maximize the ME.

6.10 Random Column Selection

First of all, let us assume random column selection and attempt to compute the average number of column selections which must be made by a level ℓ adaptor to insure that a complete set of parameters has been represented.

Define $P_{t,k}$ to be the probability that after t trials at column selection, k distinct parameters will have been represented (or p-k free parameters remain). Recall that in the level ℓ adaptor, there are $\binom{p}{\ell}$ different columns from which the choice may be made.

Our goal here is to compute a — the mean number of column selections required to complete the generation process. The value of a is given by the following equation:

$$a = \sum_{t=0}^{\binom{p}{\ell}} P_{t,p} \cdot t \quad .$$

The following may be noted:

$P_{t,k} = 0$ for $t \leq 0$ and for all k

and $P_{1,\ell} = 1$, $P_{1,k} = 0$ $k \neq \ell$ } This is an initial condition which is true since all the parameters chosen at the first trial must be free.

and $P_{t-1,p} = 0$ for all $t > 1$ } This is merely a formal statement of the condition that column selection ceases when all of the parameters have been represented.

Now that we have specified the boundary conditions for $P_{t,k}$, we may write the recursion equations which will enable us to compute its values. Consider the following situation: after t trials, k parameters are represented. This situation must have arisen in one of the following $\ell + 1$ ways:

1) k parameters were represented at the $t-1^{th}$ trial and the t^{th} trials brought no new parameters.

2) k-1 parameters were present .... 1 new parameter ...

3) k-2 parameters were present .... 2 new parameters...

ℓ+1) k-ℓ parameters were present .... ℓ new parameters...

Let $P'(\alpha/\beta)$ be the probability that α parameters will be represented after t trials given that β are represented after t-1 trials.

Let $P'(\alpha \cdot \beta)$ be the joint probability that α parameters will be present after t trials and that β are represented after t-1 trials.

Then:

$$P_{t,k} = P'(k \cdot k) + \sum_{n=1}^{\ell} P'(k \cdot k-n)$$

where

$$P'(k \cdot k) = P_{t-1,k} \cdot P'(k/k)$$

and

$$P'(k \cdot k-n) = P_{t-1,k-n} \cdot P'(k/k-n) \ .$$

Recall that there are a total of $\binom{p}{\ell}$ columns and that there have been (t-1) choices from among them (drawings without replacement) so that there are

$$d = \binom{p}{\ell} - (t-1)$$

columns left from which to choose.

Let us now compute the conditional probability P'(k/k):

There are precisely $\binom{k}{\ell}$ columns which contain k different parameters and of these, (t-1) have already been chosen. Thus, if we let $c = \binom{k}{\ell} - (t-1)$, then $P'(k/k) = c/d$ if c is positive, otherwise $P'(k/k) = 0$.

Next, we shall compute the conditional probability P'(k/k-n).

There are $\binom{k-1}{\ell-1}\binom{p - (k-1)}{1} = f$ columns which contain a single new parameter so that the probability of selecting one of these is just f/d.

The emerging pattern should be clear and we may write that

the probability that n new parameters have been chosen is just

$$\binom{k-n}{\ell-n}\left(\binom{p}{n} - \binom{(k-n)}{n}\right)/d = P'(k/k-n) \ .$$

The above text may be translated into the following equation:

Let $\delta(a) = a$ if $a > 0$

$= 0$ otherwise

then

$$P_{t,k} = \frac{\delta\left(\binom{k}{\ell}-(t-1)\right)P_{t-1,k} + \sum_{n=1}^{\ell}\binom{k-n}{\ell-n}\left(\binom{p}{n} - \binom{(k-n)}{n}\right)P_{t-1,k-n}}{\binom{p}{\ell} - (t-1)} \ .$$

Here, we note the fact that it is possible, using the difference equation method and a good deal of labor to obtain an expression for $P_{t,k}$ in a closed form. However, for our present purposes a numerical solution will suffice. Accordingly, the above relations and boundary conditions were programmed for a small digital computer and the results shown in Table 6.10.1 obtained.

Level	Mean No. of Selections
1	8.00
2	8.67
3	6.22
4	4.52
5	3.37
6	2.55
7	2.0
8	1.0

Table 6.10.1 Number of Column Selections Per Trial

We have now computed, for an adaptor of any level, the average number of column selection trials required to select a parameter value vector. Next, we must determine in more detail the nature of the columns selected. In particular, we must determine the average number of trials which will result in a column being selected all of whose

parameters appear in previously chosen columns (i.e., no free parameters), the average number of columns which contain one free parameter and so on.

Let Q(t,F) be the probability that a column with F free parameters will be chosen on that t^{th} trial.

Clearly:

$$Q(t,F) = \sum_{k=0}^{p} P'(k,k-F) \quad .$$

If we sum the values of Q(t,F) over a large number of trials (in practice until additional trials add no appreciable amount to the sum) we obtain, QT(F), the average number of trials at which a column with F free parameters was selected by the generation algorithm.

$$QT(T) = \sum_{t=1}^{n} Q(t,F) \qquad \text{where } n > p - \ell \quad .$$

Table 6.10.2 is a tabulation which shows the average number of trials at which columns with various numbers of free parameters are chosen for a range of levels for the simulated ME. The discrepancies between this table and the Table 6.10.1 represent trials at which columns with zero free parameters were chosen.

Average Number of Trials at which a Column with F Free Parameters was Chosen

Level	F=1	F=2	F=3	F=4	F=5	F=6	F=7	F=8
1	8.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	3.692	2.154	0.000	0.000	0.000	0.000	0.000	0.000
3	2.319	1.014	1.217	0.000	0.000	0.000	0.000	0.000
4	1.592	0.800	0.250	1.014	0.000	0.000	0.000	0.000
5	1.145	0.655	0.182	0.000	1.000	0.000	0.000	0.000
6	0.889	0.556	0.000	0.000	0.000	1.000	0.000	0.000
7	1.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000
8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000

Table 6.10.2 QT(F) as a Function of F for Various Levels

Next, we confront the problem of relevance. The general expression for R(r) — the probability that the number of relevant columns is r — is best developed by analogy with the following ball and urn problem:

Assume that an urn contains a total of n balls of which m are red and the rest are white. A sample of k balls is withdrawn (without replacement) and we are required to determine the probability that there are r red balls in the sample (r = 1, 2, ..., k).

In our situation, n is the number of potential columns in the ME array ($n = \binom{p}{d}$) and m is the number actually in the ME array (m = p). The value K is the maximum number of ME array columns that can be relevant to an X array column ($k = \binom{\ell}{d}$) and r is the number of ME array columns actually relevant to a randomly chosen X array column.

The complete expression for R(r) in terms of the original problem (not balls and urns) is given below:

$$R(r) = \frac{\binom{p}{r}\left(\binom{p}{d} - p\right)}{\binom{p}{d}\binom{\ell}{d}} \left.\right\} \ell \geq d$$

$$\left.\begin{aligned} R(0) &= 1 \\ R(k) &= 0,\ k > 0 \end{aligned}\right\} \ell < d$$

Next, we shall assume that level, depth, freedom and relevance are known and examine their effect upon the contents of the X array and thus on the row selection algorithm.

First of all, we shall assume that a_2 has been adjusted by means of positive feedback as outlined in an earlier section so that R portion of the expression for x_{ij} is near zero and the total contribution to the X

value comes as a result of the Q portion. Consider a column of the X array which is relevant to zero columns in the ME array. Consequently, there is no reason for any of its elements to be different from any of the others on the average so that a_2 may be adjusted so that each element of such a column has, on the average, the same small value ε. Next consider another column in the same array (i.e., with a_2 set to the same value) which is relevant to exactly one column in the ME array. In that column, the values of X will be $1 + \varepsilon$, $2 + \varepsilon$, $3 + \varepsilon$ or $4 + \varepsilon$ depending upon the contents of the relevant column in the ME array. And the probability that the row selector will choose the correct value for a single free parameter is just $\frac{4 + \varepsilon}{10 + 4\varepsilon}$ approximately 0.4 for small values of ε. On the other hand, if the row selector were required to select two parameters from the given column, the second would have to be chosen essentially at random so that the probability of a successful choice of values for two free parameters is just $\frac{1}{4}(\frac{4 + \varepsilon}{10 + 4\varepsilon})$. Next, let us consider a column which is relevant to two columns in the ME array. This column contains values of x_{ij} equal to $2 + \varepsilon$, $3 + \varepsilon$, ..., $7 + \varepsilon$, $8 + \varepsilon$ in appropriate ratios. The probability that the maximizing values for two parameters will be chosen by the row selector is just:

$$\frac{8 + \varepsilon}{\sum_{i=1}^{4}\sum_{j=1}^{4}(i+j+\varepsilon)} = \frac{8 + \varepsilon}{80 + 16\varepsilon} .$$

Next, let us consider the situation when freedom is less than relevance. This means in effect that the X array has more information contained in it than it is required to use so that the row selection algorithm must base its choice upon a part of the column that has been selected. Since our object is to compute the probability that a successful PVV (that is one which will elicit a maximum utility value from

the ME in question) will be generated, the row selector may assume that the previous choices (i.e., the choices of values for the relevant columns which are not free) have been such as to contribute to the generation of a successful PVV. This effectively restricts the row selector to the region of the selected column which contains the highest values. Stating the same thing in a different way — the row selector assumes that the unfree parameters of the X array were given values such that the relevant columns in the ME array contributed a maximum value to the utility sum.

Consider the application of the above principle to a column which has a relevance of two but a freedom of one. Applying the above implies that the choice will be restricted to those parameter value-tuples which have corrected X values of $5 + \varepsilon$, $6 + \varepsilon$, $7 + \varepsilon$, and $8 + \varepsilon$ and that the probability of a successful choice is just:

$$8 + \varepsilon/26 + 4\varepsilon \ .$$

Next, we shall give substance to the above remarks by defining $P(S/r,F)$, to be the probability of a successful choice of a value by the row selector given a column with relevance equal to r and freedom equal to F. Reflection upon the above remarks and examples will reveal that they are generalized and extended by the following formula:

$$P(S/r,F) = \frac{8r}{(8r-3F)4^F} \quad \text{for } F \leq r > 0$$

$$= \left(\frac{1}{4}\right)^{F-r} \frac{8r}{5r \cdot 4^r} = \frac{1.6}{4^F} \quad \text{for } F > r > 0$$

$$= \left(\frac{1}{4}\right)^F \quad \text{for } r = 0$$

There are two exceptions to the above formulas which are treated below:

Level equal one . In this case, all of the columns are relevant

and a_2 may be adjusted so that ε the minimum value in the corrected X array is derived from a component function which returns an incremental utility value of one rather than from an irrelevant column which returns a value of zero. This has the effect of changing the above expressions to the following:

$$P(S/r,F) = \frac{6r}{(6r-3F)4^F} \qquad F \leq r > 0$$

$$= (\frac{1}{4})^F \qquad r = 0$$

Level equal eight. This case really doesn't fit our method for generating simulated meta-environments, but we include it for completeness. There is only one column in the ME array and we shall assume that utility values (8 to 32) are assigned at random and in the same proportion as they appear in MEs of lower depth. Since there is only one column in the X array, there is no column selection. That single column has corrected (by a_2) values ranging from ε to $\varepsilon + 24$ and the probability of selecting a successful PVV is just

$$\frac{24 + \varepsilon}{(\frac{4^8}{2})24 + 4^8_\varepsilon} \approx \frac{2}{4^8} .$$

We now defined all of the components required to compute the probability of generating a successful PVV. The only task remaining is to put the pieces together in a systematic manner.

Let P(S) be the probability of generating a successful PVV at any one trial. Then 1/P(S) is the mean number of trials required to generate a successful PVV.

Define P(S/F) to be the probability that the row selector will choose the F parameters which maximize the component sub-utility function given that freedom is F. Clearly then:

$$P(S) = \prod_{F=1}^{\ell} P(S/F)^{QT(F)}$$ where QT(F) is as defined previously.

P(S/F) may be easily computed from previously derived quantities as given below:

$$P(S/F) = \sum_{r=0}^{\ell} R(r) : P(S/r,F) \quad .$$

Where R(r) and P(S/r,F) have been previously defined. At this point, we are able to compute the results for various levels and depths assuming random column selection. Table 6.10.3 gives those results. There are two points that should be noted. First, for cases where the depth was greater than the level, the computation was carried out in full in order to provide a check both on the logic of the program and the accuracy of the arithmetic routines. In these instances, the mean number of trials required to reach a point of maximum utility should have been $4^8 = 65,536$. As can be seen from the figures, the arithmetic error is less than .01 percent. The error present occurs mainly because the exponentiation and combination computation routines truncate results instead of rounding. Another checkpoint occurs at level equal to one and depth equal to one. The result should be $(2)^8 = 256$ and can be seen to be accurate (the input-output routines also truncate rather than round).

6.11 Non-Random Column Selection

In a preceding chapter, we described several alternate methods of column selection. At this point, we will investigate their effect upon the results we have just obtained. A careful survey of the preceding sections will reveal that the only area in which the assumption of random column selection was used was in the computation of the relevance probability R(r). The effect of non-random column selection will be

EXPECTED NUMBER OF TRIALS REQUIRED TO ATTAIN A MAXIMUM UTILITY VALUE

LEVEL	DEPTH=1	DEPTH=2	DEPTH=3	DEPTH=4	DEPTH=5	DEPTH=6	DEPTH=7	DEPTH=8
1	255.9	65535.9	65535.9	65535.9	65535.9	65535.9	65535.9	65535.9
2	5090.8	25986.9	65535.6	65535.6	65535.6	65535.6	65535.6	65535.6
3	12239.0	16522.6	45075.2	65534.3	65534.3	65534.3	65534.3	65534.3
4	18525.9	18316.4	27642.2	51421.1	65534.6	65534.6	65534.6	65534.6
5	23519.5	25708.7	23507.2	32549.4	51282.7	65534.1	65534.1	65534.1
6	27407.5	35409.5	29322.3	27716.0	31883.9	44515.1	65535.9	65535.9
7	30427.4	43796.9	38822.8	36542.2	34628.6	32748.9	25599.9	65535.9
8	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9

TABLE 6.10.3 CA WITH RANDOM COLUMN SELECTION

simply to alter the distribution of R(r) in such a way that it becomes more likely that relevant columns are selected. R(r) as previously defined may be interpreted as that proportion of all of the columns of the X array which have a relevance value of r. But after a non-random column selection, the columns presented to the row selection algorithm will have a different proportion of relevance values. Let us define R'(r) to be the probability that a column selected by the column selector algorithm has relevance r. The preceding expressions will then all be valid with R'(r) substituted for R(r) whenever it occurs. This introduces no change in the results presented previously since random column selection doesn't alter the distribution of relevance so that for that case R'(r) = R(r).

1) Column Selection Based Upon Maximum Element

It is quite obvious that in any column, the maximum element has, on the average, a corrected value of 4r + ε. This implies that:

$$R'(r) = \frac{R(r)(4r+\varepsilon)}{\sum_{m=0}^{\ell}(R(m)(4m+\varepsilon))} \approx \frac{R(r)r}{\sum_{m=0}^{\ell}R(m)m}$$

Computations made incorporating this expression for R'(r) on a digital computer and the results are given in Table 6.11.1. Note that these results show a marked advantage over random column selection.

2) Column Selection Based Upon Column Sum

The sum of the elements of any corrected column in the X array is equal to the column average (2.5r) + ε times the number of elements in the column (8^{ℓ}). In this case, R'(r) is as follows:

$$R'(r) = \frac{(\varepsilon+2.5r)8^{\ell}R(r)}{\sum_{m=1}^{\ell}(\varepsilon+2.5m)8^{\ell}R(m)} \approx \frac{R(r)r}{\sum_{m=1}^{\ell}R(m)m}$$

which yields the same results as were obtained for column

EXPECTED NUMBER OF TRIALS REQUIRED TO ATTAIN A MAXIMUM UTILITY VALUE

LEVEL	DEPTH=1	DEPTH=2	DEPTH=3	DEPTH=4	DEPTH=5	DEPTH=6	DEPTH=7	DEPTH=8
1	255.9	65535.9	65535.9	65535.9	65535.9	65535.9	65535.9	65535.9
2	5090.8	4199.1	65535.6	65535.6	65535.6	65535.6	65535.6	65535.6
3	12239.0	10194.7	7719.3	65534.3	65534.3	65534.3	65534.3	65534.3
4	18525.9	18126.8	13511.3	11750.0	65534.6	65534.6	65534.6	65534.6
5	23519.5	26491.6	21397.4	17954.2	16137.0	65534.1	65534.1	65534.1
6	27407.5	33695.0	30669.2	27033.0	23630.8	20773.9	65535.9	65535.9
7	30427.4	38333.0	37833.2	37043.0	35717.2	33351.2	25599.9	65535.9
8	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9

TABLE 6.11.1 CA WITH MAXIMUM ELEMENT COLUMN SELECTOR

selection based upon maximum element.

3) Column Selection Based Upon Maximum Relative Deviation

The minimum corrected value contained in a column is just 1r + ε while the maximum corrected value is 4r + ε. The maximum deviation is then (4r-r) = 3r and since the mean corrected value is just 2.5r + ε, the maximum relative deviation is zero for r equal to zero and a constant for r not equal to zero. Translated to a formula for R'(r), we obtain:

$R'(0) = 0$

$$R'(r) = \frac{1.2R(r)}{\sum_{m=0}^{\ell} 1.2(R(m))} = \frac{R(r)}{\sum_{m=0}^{\ell} R(m)} \qquad \text{for } r > 0$$

The results of computation using this definition of R'(r) are given in Table 6.11.2. Note that these results are slightly better than those obtained for other methods of column selection.

4) Column Selection Based Upon Total Deviation

As previously noted, $\bar{x}_j = 2.5r$. The following expression give the total deviation, D, of the elements of a column in the X array in terms of the values of level and relevance.

$$D(r) = 4^{\ell-r} \sum_{K_1=1}^{4} \cdots \sum_{K_r=1}^{4} \left| K_1 + \cdots + K_r - 2.5r \right|$$

and $$R'(r) = \frac{R(r)D(r)}{\sum_{m=1}^{\ell} R(m)D(m)}$$

The results of a computation using this value of R'(r) is given in Table 6.11.3. Note that this method of column selection yields results for the below diagonal cases and depth greater than one which are not quite so good as the preceding method.

EXPECTED NUMBER OF TRIALS REQUIRED TO ATTAIN A MAXIMUM UTILITY VALUE

LEVEL	DEPTH=1	DEPTH=2	DEPTH=3	DEPTH=4	DEPTH=5	DEPTH=6	DEPTH=7	DEPTH=8
1	255.9	65535.9	65535.9	65535.9	65535.9	65535.9	65535.9	65535.9
2	5090.8	4199.1	65535.6	65535.6	65535.6	65535.6	65535.6	65535.6
3	12239.0	9128.9	7719.3	65534.3	65534.3	65534.3	65534.3	65534.3
4	18525.9	16035.4	12708.9	11750.0	65534.6	65534.6	65534.6	65534.6
5	23519.5	24582.1	19462.2	17130.5	16137.0	65534.1	65534.1	65534.1
6	27407.5	32944.9	28749.9	24964.9	22423.5	20773.9	65535.9	65535.9
7	30427.4	38249.8	37544.5	36315.9	34311.0	31459.8	25599.9	65535.9
8	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9

TABLE 6.11.2 CA WITH MAXIMUM RELATIVE DEVIATION COLUMN SELECTOR

EXPECTED NUMBER OF TRIALS REQUIRED TO ATTAIN A MAXIMUM UTILITY VALUE

LEVEL	DEPTH=1	DEPTH=2	DEPTH=3	DEPTH=4	DEPTH=5	DEPTH=6	DEPTH=7	DEPTH=8
1	255.9	65535.9	65535.9	65535.9	65535.9	65535.9	65535.9	65535.9
2	5090.8	4199.1	65535.6	65535.6	65535.6	65535.6	65535.6	65535.6
3	12239.0	9451.1	7719.3	65534.3	65534.3	65534.3	65534.3	65534.3
4	18525.9	16883.1	12941.8	11750.0	65534.6	65534.6	65534.6	65534.6
5	23519.5	25538.2	20205.9	17374.0	16137.0	65534.1	65534.1	65534.1
6	27407.5	33350.0	29711.3	25824.4	22817.3	20773.9	65535.9	65535.9
7	30427.4	38293.4	37702.3	36714.3	35031.3	32286.1	25599.9	65535.9
8	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9	32767.9

TABLE 6.11.3 CA WITH TOTAL DEVIATION COLUMN SELECTOR

6.12 Phase III Summary

It must be emphasized that the numbers obtained here were based on the assumption of a_2 feedback which tends to give a very optimistic bias (i.e., lower numbers). In general, however, the results given above confirm our original hypothesis that the matching case (where the level of the adaptor is equal to the depth of the meta-environment) is optimal. The choice of column selection method turned out to have a relatively small effect on the eventual outcome. This was contrary to our intuition prior to the analysis.

6.13 Analysis Summary

Phase III results indicate that the level must not be less than the depth. Phase II results indicate that the depth cannot be greater than three or four. Combining these tells us that the correlation adaptor is not useful for levels greater than three or four. The Phase I results confirm this. Another object of our study of correlation algorithms is to provide a standard against which we may compare the behavior of other classes of algorithms. Table 6.13.1 gives a summary of our correlation adaptor analysis results for depths one and two and for levels one and two. The consensus is based partially upon enlightened intuition.

		Level = 1	Level = 2
Depth 1	Phase I	32	449
	Phase II	541	541
	Phase III	256	5091
	Consensus	500-800	5000-6000
Depth 2	Phase I	32	449
	Phase II	4329	4329
	Phase III	65536	4199
	Consensus	65536	5000-8000

Table 6.13.1 Summary of Results for the Correlation Adaptor

6.14 Some Practial Considerations

In choosing "the number of trials required to reach a point of maximum utility" as our figure of merit for adaptors, we have, in effect, made the assumption that the cost of adaptation is negligible compared to the cost of taking samples (trials) of the meta-environment. In the real world, this may not necessarily be true and so it may be useful at this point to give an indication of the comparative cost of adaptation as determined by simulation.

The experience array of the level one adaptor requires 256 words of core storage in the IBM 1130 computer used in the simulation. The simulation proceeded at a rate of 285 trials per minute. The same simulation with the correlation adaptor replaced by a random adaptor ran at a rate of 1616 trials per minute. The level two experience array would have required 7168 words of storage. Since the computer contains only 8192 words, the simulation was not attempted. If column selection time is neglected, the execution time may be considered to be inversely proportional to the size of the experience array. Under that assumption, the level two adaptor can be expected to operate at a rate of about 10.2 trials per minute. We will have occasion to refer to these values in a later chapter.

6.15 Guidepost

We have now concluded our consideration of the class of correlation algorithms. In the next chapter we will describe an entirely distinct class of adaptive algorithms. These are the genetic algorithms. In a subsequent chapter, we will present quantitative results which may be compared with the results derived in this chapter.

7. THE GENETIC ALGORITHM — DESCRIPTION

7.1 Introduction

When man builds machines in order to aid him in his pursuits, he often looks to nature for inspiration. In attempting to fabricate an adaptive system, one is almost inevitably led to consider evolutionary processes and the mechanisms which exist in nature to implement them. See, for example, Holland [1962]. It is with these thoughts that we have formulated what we have termed the genetic algorithm.

We will use terminology borrowed from the biological sciences where it is convenient to express concepts borrowed from that area. Most of this terminology and many of the judgements as to the relative magnitude of various effects is based upon information gleaned from various sources particularly Dobzhansky [1951], Herskowitz [1965], and Sager and Ryan [1962]. However, we have no desire to evade the responsibility for the number of rather arbitrary decisions it was necessary to make in order to insure that the algorithm fits our paradigm and is relatively simple to implement. Accordingly the algorithm considered as a model of its biological counterpart is certainly a first order model.

7.2 Biological Preliminaries

The inheritable characteristics of an individual are controlled by a set of elements known as chromosomes. A chromosome is a string of fixed length composed of a linear array of genes. Each gene is thought to exert control over one or more biochemical processes within the cell. The geographical positions along the chromosomes are called sites. Each site may be occupied by a locus which controls a particular biochemical process which in turn manifests itself as a particular observable trait of the individual. For example, there may be a locus (or several) which

controls some aspect of eye color, another which controls hair texture, etc. Each locus may be occupied by any one of a set of genes each of which is an allele of that locus. For example, the eye color locus may have an allele which produces red eyes, another which produces white eyes, etc. The site which a locus occupies usually has relatively little effect on the expression of the allele occupying the locus, however exceptions do exist and will be treated to some extent in later sections. These exceptions are called position effects.

The chromosomes of a diploid individual may be grouped into pairs where each chromosome of the pair contains identical loci. These functionally corresponding loci are called homologous. These homologous loci do not necessarily contain identical alleles and in cases where they contain different alleles, the effect of the pair on the individual or zygote is determined by the relative dominance of the two alleles. For example, the homologous loci for eye color may contain a dominant red allele and a recessive white allele so that the zygote would have red eyes. Such an individual with different alleles for a particular locus is said to be heterozygous with respect to the trait controlled by that locus (i.e., the individual in the example is heterozygous with respect to eye color).

When a diploid zygote begets offspring, it does so by first generating a special type of cell called a gamete in a process called gametogenesis. The gametes of a diploid individual are usually haploid, that is they contain half the usual number of chromosomes and in fact precisely one member of each homologous pair so that the gamete contains only one locus for each trait. A diploid individual is formed by the union of two gametes in the process of fertilization or conjugation.

Recall that the memory of the correlation algorithm is contained in

an array of numbers and that experience is recorded by making changes in the numbers in the array. In contrast, the memory of an evolutionary system is contained in a population and experience is recorded by making changes in the composition of that population. The population involved might equally well be a population of zygotes or a population of gametes. Nature prefers the former but, for technical reasons, our genetic algorithm will center about a population of gametes.

The usual discrete model of the evolutionary process makes the idealization of non-overlapping generations (see for example Kimura [1958]). In such a model, an entire population is first generated, then tested against the environment, next allowed to reproduce and finally to die. The process is then repeated with the progeny. In computer parlance this amounts to parallel or batch processing (the batch is the population). In contrast to this, our model employs a serial or sequential approach. A single individual is generated from the population of gametes, tested against the environment, and then, allowed to contribute offspring (gametes) to the population in proportion to its relative success in the environment.

7.3 Algorithmic Preliminaries

A chromosome is a string of sites, each of which is occupied by a locus. The loci are of two types: 1) Active loci which correspond directly to the parameters. There are exactly p active loci-each one identified with one and only one parameter. The alleles for these loci are the values which the parameters may assume. 2) Quiet loci, which do not correspond to parameters and, in fact, have no direct effect on the determination of the PVV. Their function is to control (both directly and indirectly) the process by which the chromosomes are

generated. The function of the quiet loci will be outlined in detail in the section entitled "gametogenesis". Sites containing active loci will be called active sites.

7.4 Conjugation

Fertilization takes place by the most elementary of processes. Two gametes are selected at random (uniform probability) from the population and mated. This is a considerable simplification over the natural process. It eliminates sex (a potential source of variability) and the consideration of mating rules which may have considerable effect on the population (for example, kinship mating taboos undoubtedly increase the frequency of recessive lethal alleles in the population). We have chosen to ignore these factors simply because we are interested in concentrating our attention on other areas.

7.5 Phenotype Expression

The object of this section of the algorithm is to generate a single parameter value vector from a homologous pair of chromosomes. Each active locus contains, besides the information which identifies the parameter to which it is associated and the particular parameter value, a dominance value. At each locus the algorithm simply selects the allele having the highest dominance value. Unlike the biological case where partial dominance may be permissible (resulting, for example, in speckled eyes), our interpretation demands that only one of the alleles of the homologous loci be chosen. The decision process in the case of ties (equal dominance values) involves position effects and is somewhat complicated so that it will be necessary to outline the process in some detail.

One of the chromosomes is arbitrarily selected to be the "key"

chromosome and its sites are examined in turn proceeding from left to right. Each time an active locus is discovered, the contents of its homologous locus are retreived from the other chromosome. The dominance values are compared and the allele which is associated with the highest dominance value is selected. If the dominance values are identical, dominance follows the key chromosome. That is, the nearest active site on the key chromosome to the left of site under examination is checked. If the locus at that site was dominant, the present locus on the key chromosome is set to be dominant, otherwise the homolog dominates. If the locus under examination happens to occupy the initial active site on the key chromosome, the key locus dominates.

Dobzhansky [1951] states that there is some evidence that dominance is partially position dependent which accounts for our choice of tie breaking scheme. Other than this, we have completely disregarded position effects. Again, it's a case of choosing to concentrate our efforts on the aspects of the biological situation which suit our purposes while ignoring other aspects.

7.6 Environmental Selection

The function of envionmental selection is to determine the amount of influence the progeny (gametes) of the given diploid zygote are to have on the population. The PVV generated from the zygote by the process described above is fed into the meta-environment which responds by assigning an estimated utility value, $O_{\bar{U}}(t)$. The idea of environmental selection is that if $O_{\bar{U}}(t)$ is approximately the same as the average estimated utility $\overline{O_{\bar{U}}(t)}$ encountered thus far, then two gamete progeny will be returned to the population. If $O_{\bar{U}}(t)$ is comparatively larger or smaller, the zygote will be permitted to contribute a greater or

lesser number of progeny to the population. More formally, the zygote being considered at time t will be allowed to generate precisely E(t) gametes where:

$$E(t) = 2.\frac{(\overline{O_U}(t)+B2)}{(\overline{O_U}(t)+B2)}$$

and where

$$\overline{O_U}(t) = \frac{\overline{O_U}(t+1)(B1-1) + O_U(t)}{B1} \qquad t > 1$$

$$= 20.0 \qquad t = 1$$

and B2 is a control coefficient which is analogous to the a_2 control coefficient of the correlation algorithm. B1 is a control coefficient which modifies the average estimated utility in the following manner:

t < B1: more weight given to earlier trials

t = B1: true average

t > B1: more weight given to later trials

7.7 Gametogenesis

Gametogenesis produces gametes derived from the parental diploid zygote. The number of gametes produced (actually the number of times the gametogenesis sub-algorithm is executed) is determined by process of environmental selection outlined above. During gametogenesis a great deal of recombination takes place so that there is a very small probability that the gamete produced will be identical to either of the parental chromosomes. This section is broken down into four subsections dealing with 1) Crossover, 2) Chromosome Breakage, 3) Point Mutation, and 4) Self Contained Controls.

1) Crossover is a recombination method in which portions of the parental chromosomes are interchanged so that the resulting strings

are composed of sections of each original string. Figure 7.7.1 illustrates the result of a single crossover.

Basically, the crossover mechanism is very simple. Crossover may take place independently at any site boundary. The probability of crossover at each boundary is a constant B6. This basic mechanism is modified by a set of rules which permit crossover only between regions which are sufficiently locus homologous. This means that crossover is prohibited between regions that do not have essentially the same loci at the same sites.

The reasons for these restrictions are two-fold: 1) Nature seems to form crossovers much more readily between locus homologous regions (there is a good deal of evidence which links crossover with the physical attraction that occurs during gametogenesis). 2) Neglecting these restrictions leads to the generation of gametes which are deficient in some loci and have duplicates of others. Such gametes have no natural interpretation in our paradigm.

There exists evidence that the presence of a crossover at one site on the chromosome string inhibits crossover at neighboring sites. This is the phenomenon of "interference". There is also evidence of the contrary effect "negative interference". In specifying that the probabilities of crossover are independent, we have neglected both of these effects.

Perhaps at this point we should pause to consider the consequences of crossover as a recombination method. Although we shall allow point mutations, crossover remains the primary recombination mechanism. Let us first note the phenomenon of gene linkage. Since crossover occurs with equal probability along the length of the chromosome, it follows that two alleles of loci which occupy neighboring sites are much less

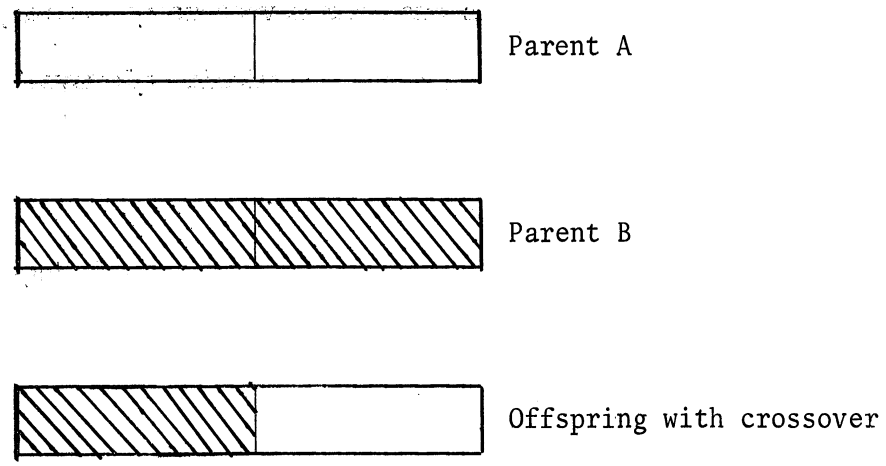

Fig. 7.7.1 Offspring Produced as a Result of a Single Crossover

likely to be separated by crossover than are alleles which are on widely spaced sites. Conversely, widely spaced alleles are much more likely to be separated (i.e., they are less tightly linked).

For example, suppose that there are two alleles of different loci which when present in combination are more advantageous than the presence of either individually would warrant. If the loci are closely linked, then the probability that the pair will survive intact is high. On the other hand, if they are loosely linked they are likely to be separated so that the probability of that advantageous combination becoming established is reduced. The quiet loci assume an important role in recombination. They serve to decouple the active loci which they separate and thus increase the variety of gene linkages. It should be clear that gene linkages play much the same role as the columns play in the experience array of the correlation algorithm.

It should be noted that, as a result of the homologous neighborhood restrictions, the crossover mechanism does nothing to change the linkages of the strings upon which it operates. The next subsection deals

with methods of altering gene likages.

2) Chromosome Breakage in nature takes many forms and has many consequences. The most important of these which affects a system of the type that we have postulated is inversion. An inversion occurs when, after a chromosome has been broken in two places, restitution takes place with the intermediate region inverted as shown in Figure 7.7.2.

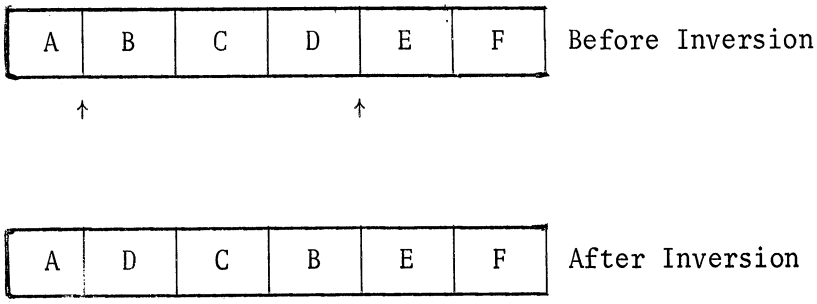

Fig. 7.7.2 Consequences of a Single Inversion

In our algorithm, each chromosome (only one chromosome survives crossover) is selected for a single inversion with probability B7. If an inversion is to take place, two site boundaries are selected at random and the loci on intermediate sites are inverted.

Biologically, this is a first order model. We have neglected the effects of single breaks as well as those of more than two breaks. As a recombination method, its effects are to introduce different gene linkages and to suppress crossovers (since crossovers will not occur within the inverted regions of inversion heterozygotes). It is a simple matter to show that the inversion operation is complete in the sense that starting from an arbitrary initial string of loci, it is able to produce any other string.

3) Point Mutations occur when one allele is substituted for another at a single locus. Such events are induced in nature by radiation and by chemical mutagens. In our algorithm, each of the sites is selected independently for mutation with a probability B8.

Biologically, the assumption of site independent of mutation seems well justified. We have chosen to neglect the evidence of regions along the chromosome that appear to be more or less susceptible to mutations than neighboring regions. Our use of a uniform mutation rate is a natural idealization and seems reasonable especially since crossover and not mutation is our main source of recombination.

As in the case of the correlation algorithm, we could have introduced a neighbor relation between the alleles (values) of a given locus (parameter) so that mutations between "close" alleles were more likely. This can be accomplished with a neighbor array similar to that of the correlation algorithm. Such an approach does have a biological counterpart based on the assumption that the chromosome is a large DNA molecule. It is clear that mutations are more likely to occur between alleles which differ in a single neucleotide pair than they are between alleles that differ in many pairs. Experimental work on the fine structure of the gene confirms this expectation. As discussed previously, we shall not consider in a quantitive manner the question of allele (or value) neighbors.

4) Self Contained Controls offer a very natural possibility. Recall that we have introduced a number of control coefficients B1, B2, B6, B7, B8 which are to be set by some external mechanism (A'). In the biological case, these are (in the main) the parameters which control the reproductive activities of the organism. But, they themselves are genetically determined.

This situation may be imitated in our algorithm by allowing the quiet loci to control the coefficients B1 through B8. The loci are still quiet in the sense that they don't directly specify parameter values, but they do help to specify the process by which the parameter values are determined. It's not at all clear that the coefficients B1 and B2 should be genetically controlled since they really deal with the selection process. However the gametogenesis coefficients B6 through B8 are prime candidates for self contained control. There is abundant evidence, for example, that in fact the mutation rate B8 is regulated by mutagenic genes. Herskowitz [1965] states "Some strains (of Drosophila) are known to contain mutation genes in whose presence the general point mutation rate is increased as much as tenfold."

As in all feedback control systems, a question of stability arises. For example, if the mutagenic gene controls its own mutation rate, will oscillations occur? Most of the questions of inner loop stability may be safely disregarded because there is an outer control loop — that provided by natural selection — which serves to stabilize the entire algorithm.

7.8 Guidepost

Having described the operation of the genetic algorithm, we shall next be concerned with evaluating it. In particular, we shall concern ourselves with the performance of the GA immersed in simulated meta-environments of various depths. Recall that the correlation algorithm performed best when the level of the adaptor was matched to the depth of the ME and that it did poorly in unmatched situations. We expect the GA to be much more versatile because, in effect, it rewards not only single parameter values but at the same time parameter value pairs,

triples, etc. to the extent to which they have proven to be advantageous. A reasonable hypothesis is that the GA will perform slightly poorer than the matched CA, but very much better than the unmatched CA. In the chapter that follows, we shall examine the behavior of the genetic adaptor experimentally in order to be able to test this hypothesis.

8 THE GENETIC ALGORITHM-QUANTITATIVE CONSIDERATIONS

8.1 Introduction

In this chapter, we shall be concerned with the performance of genetic adaptors with various control coefficients immersed in meta-environments of depths one and two. The method employed is experimental as opposed to analytical. That is, the genetic adaptors and the meta-environments and the interaction between them have been simulated on a digital computer. From the viewpoint of the theoretician, this is certainly less satisfactory than an analytical approach and so it is appropriate at this point to comment on our decision to undertake experimentation.

In previous chapters we have presented very detailed descriptions of the genetic algorithms and the simulated meta-environments and the way in which they interact. (The fact that we are able to simulate them is evidence that the descriptions are constructively given.) Our problem is to be able to obtain the global consequences of these locally effective rules. In particular we would like to know the expected number of trials required for the adaptor to generate a PVV which maximizes the (simulated) ME. In the case of the correlation adaptor, it was possible to make certain global assumptions about the asymptotic behavior of the algorithm from which we were able to proceed to the desired result. Proceeding in an analogous manner, the key to the analysis of the genetic algorithm is a set of useful global assumptions concerning the interaction which neither prejudice the result nor ignore the factors whose effects we wish to study. (Indeed, the art in any branch of applied mathematics lies in formulating just such assumptions.) The logical first place to look for aid in formulating the type of

global assumptions required is the literature of quantitative genetics. Here we observe that the problem most often treated is the inverse one; namely that of deducing the local mechanism given the observed global behavior. Where the global effects of selection are explored, the results most often obtained concern the survivial of individual alleles in systems with limited means of recombination (see for example Dobzhansky [1951, p. 82]). Other authors have explored the intermediate effects of various recombination methods. Thus, Feller [1957, p. 269] obtains the distribution for the number of crossovers or mutations on a single chromosome.

Fisher [1958], Kimura [1958] and others have concerned themselves with the global effects of selection, but in such a way that the subtleties of the recombination methods and the utility (or ME) function do not appear directly. Kimura [1958], for example, computes the rate of change of population fitness. This value along with the variance in fitness as a function of time would enable one to compute the mean time (number of trials) required for an individual of maximum utility to be generated. However, Kimura's results are obtained in terms of dominance, epistatic, and mating effects which depend on the recombination methods and utility function but which cannot be derived from them directly. In a sense, Kimura's results are analogous to the general results for the correlation algorithm obtained in Chapter 6 in which both the column selection mechanism and the utility function are contained in expressions whose evaluation is, for practical purposes, impossible. Our attempts to develop an analysis which explicitly includes the effects of the recombination mechanisms and the ME depth have been unsuccessful and we have turned to experimental methods to evaluate the genetic algorithm. Simulations of genetic systems of a more elementary type have

been carried out by Fraser, Burnell and Miller [1966] and others in order to test hypotheses arising in the science of genetics.

8.2 The Simulation

A simulation experiment, like any other, may be thought of as process of taking samples of a random variable. The precision of the conclusions which can be inferred on the basis of the experiment is primarily a function of the sample size. The experiments described below do have a very small sample size, however, the conclusions drawn are also quite limited and, we believe, so evident as to preclude the necessity for elaborate experimental analysis.

The simulations were conducted at a very basic level in the sense that there was in the computer a representation of a population of gametes (chromosomes) with identifiable genes. Each chromosome contains 16 sites each of which contains a gene. Each gene contains four nuggets of information as follows:

1) Activity: Each gamete contains exactly 8 active sites.

2) Locus: The locus specifies the ME parameters controlled by the gene. Each gamete contains exactly one locus of each type.

3) Dominance: A gene may be dominant or recessive.

4) Allele Value: Each gene may have one of eight allele values.

A trial consists of the following sequence of events:

1) Conjugation: Two gametes are selected at random for mating.

2) Phenotypic Expression: A phenotype (PVV) is produced from the gamete pair based upon their dominance values.

3) Selection: The PVV is tested in the ME.

4) Gametogenesis: Offspring gametes are produced and added to the population.

The details of each step of this process may be monitored and the results printed out under control of the computer console sense switches.

A block consists of a number of trials. In these experiments, the block size was set to one half of the population size so that a block corresponds to a generation since each trial produces two gametes on the average. Summary printouts at the end of each block gave the distribution of utility values obtained during the block.

A run consists of a number of consecutive blocks. A run was terminated either when a maximum utility value was reached or when it was determined that it was impossible for a maximum utility value to be obtained from the present state of the experiment. The result of a run is the number of trials required to reach a maximum utility value which is either a positive integer or infinity (recorded as "INF" in the printed output).

An experiment consists of a number of runs. In all of our experiments, the conditions of each run were identical except for different random initial gamete populations. The same five initial populations were used for every experiment.

As we have indicated, the experiments were monitored at various levels of detail (the trial level, the block level, etc.) and a prodigious amount of data has been collected, however only the results at the run level will be tabulated herein. The data obtained on a more microscopic level proved to be of great value in debugging, planning experiments and interpreting results and trends. We shall use the insight obtained from this data where it is useful in explaining the run results. Table 8.2 contains a summary of all of the experiments

EXPT NO	ME NO	POP	B1	B2	XOVR PROB	MUT PROB	INV PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
1	1	200	T	-15	.125			INF	INF	INF	INF	152
2	1	200	10	-15	.125			860	348	INF	709	INF
3	1	200	T	-15	.25			704	INF	1024	INF	INF
4	1	200	10	-15	.25			348	1235	809	272	473
5	1	200	10	-15	.25		.125	1108	INF	1535	1976	376
6	1	200	50	-15	.25			396	INF	780	INF	320
7	3	200	50	-15	.25			INF	INF	690	INF	757
8	1	100	25	-15	.125	.0125		508	4316	249	8411	4926
9	3	200	50	-15	.25	.0625		1515	4156	4391	6716	191
10	3	300	10	-15	.25			INF	1045	INF	2127	1637
11	3	300	10	-15	.25		.125	INF	1557	INF	INF	INF
12	3	200	T	-15	.125			INF	INF	INF	INF	INF
13	3	200	10	-15	.125			INF	INF	720	708	INF
14	3	200	T	-15	.25			INF	INF	INF	INF	401
15	3	200	10	-15	.25			INF	INF	748	INF	INF
16	3	300	T	-15	.125			650	567	INF	INF	INF
17	3	300	10	-15	.125			INF	774	INF	1011	INF
18	3	300	T	-15	.25			INF	INF	INF	2544	1729
19	3	400	T	-15	.125			1348	1705	2272	2633	INF
20	3	400	10	-15	.125			INF	INF	1574	1765	1634
21	3	400	T	-15	.25			INF	1305	1640	1058	2088
22	3	400	10	-15	.25			INF	2126	INF	2143	1118
23	3	400	100	-15	.25			INF	3028	INF	1885	2589
24	3	400	T	0	.25			INF	INF	INF	INF	INF
25	3	400	T	-15	.0625			INF	3122	INF	INF	2512
26	3	400	T	-15	.125	.0125		12881	2007	1184	2122	969

TABLE 8.2 LIST OF ALL EXPERIMENTAL RESULTS (PART 1)

EXPT NO	ME NO	POP	B1	B2	XOVR PROB	MUT PROB	INV PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
27	3	400	T	-15	.125	.00625		3915	4142	1965	1554	1453
28	3	400	T	-10	.125			2322	3772	INF	2532	408
29	3	400	T	-18	.125			612	1741	968	698	INF
30	3	400	T	-20	.125			INF	INF	INF	INF	INF
31	3	400	T	-22	.125			INF	INF	INF	INF	INF
32	3	400	T	-19	.125			INF	798	476	INF	INF
33	3	400	10	-18	.125			INF	2840	1444	1556	717
34	3	400	10	-19	.125			INF	INF	855	4650	INF
35	3	400	100	-18	.125			INF	INF	1538	INF	4204
36	3	400	100	-19	.125			1184	948	INF	INF	INF
37	3	400	T	5	.125			1992	1781	2322	1507	INF
38	3	400	100	5	.125			864	INF	2897	INF	INF
39	3	400	T	3	.125			INF	INF	INF	INF	INF
40	3	400	100	3	.125			INF	636	577	INF	2479
41	3	400	T	7	.125			INF	1799	1711	INF	1787
42	3	400	100	7	.125			1531	2423	734	2148	INF
43	3	400	T	5	.125	.0125		1425	9059	14213	2359	840
44	3	400	T	5	.125	.00625		41624	3432	1007	644	6332
45	3	400	100	7	.125	.0125		1202	2115	2023	1165	4945
46	3	400	100	7	.125	.00625		3923	1663	2545	1767	7170
47	3	400	100	-15	.125	.0125		1414	3691	406	2821	1903
48	3	400	100	-15	.125	.00625		49214	2036	21977	2422	923
49	3	400	100	-15	.25	.0125		7996	1389	1851	3099	3234
50	3	400	100	-15	.25	.00625		7131	3035	8970	2663	6674
51	3	400	100	-12	.125	.0125		4077	2078	2216	3039	10862
52	3	400	100	-18	.125	.0125		456	27291	869	688	309

TABLE 8.2 LIST OF ALL EXPERIMENTAL RESULTS (PART 2)

EXPT NO	ME NO	POP	B1	B2	XOVR PROB	MUT PROB	INV PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
53	3	400	T	-12	.125	.0125		6854	1265	4278	3151	6004
54	3	400	T	-18	.125	.0125		4490	935	961	2153	5869
55	4	400	100	-15	.125	.0125		1535	10581	1528	4062	14633
56	1	400	100	-15	.125	.0125		1975	814	711	665	1221
57	3	400	100	-15	.0625	.0125		4405	14950	1601	7442	5704
58	3	400	100	-15	.125	.0125	.125	2858	35875	16977	19329	5667
59	3	400	100	-15	.125		.125	1418	INF	INF	INF	INF
60	5	400	100	-15	.125	.0125		3551	1556	2306	2817	4119
61	1	400	100	-15	.125			854	693	1670	INF	834
62	1	300	75	-15	.125			878	INF	337	887	1287
63	1	200	50	-15	.125			913	364	1460	INF	INF
64	3	300	75	-15	.125	.0125		4700	5224	740	794	4453
65	3	200	50	-15	.125	.0125		7696	17120	273	5383	9214
66	6	400	100	-15	.125	.0125		763	1226	1066	12018	24371
67	7	400	100	-15	.125	.0125		2884	1032	2921	3798	4622
68	1	300	75	-15	.125	.0125		836	958	572	1161	2316
69	1	200	50	-15	.125	.0125		212	516	649	1192	794
70	1	400	100	-18	.125	.0125		991	844	941	269	910
71	3	400	T	-12	.125			INF	INF	1351	1826	1180
72	3	400	100	-12	.125			INF	1440	1791	2426	3347
73	3	400	100	-15	.125			1536	1853	1905	INF	1347
74	1	400	100	-12	.125	.0125		765	1036	1202	490	2937
75	3	400	T	7	.125	.0125		1603	3618	5165	5239	8601
76	3	400	100	5	.125	.0125		1309	939	1962	2403	3036
77	3	400	100	3	.125	.0125		630	2038	331	122872	1356

TABLE 8.2 LIST OF ALL EXPERIMENTAL RESULTS (PART 3)

performed. The meaning of the column headings is as follows:

1) EXPT NO ----- The sequential number of the experiment.

2) ME NO --------- The number of the meta-environment used. ME number one was of depth one and numbers three through seven were distinct depth two MEs.

3) POP ---------- The size of the gamete population.

4) B1 ---------- The control coefficient B1 which determines the way in which the estimated average utility is computed. A value of "T" indicated that the run average is used as the estimated average.

5) B2 ---------- The control coefficient B2 which determines the degree of selection. Negative values apply to the standard selection method while positive values apply to the alternative selection method.

The next three columns contain the gametogenesis controls:

6) XOVR PROB ----- The probability of crossover at any site boundary along a chromosome.

7) MUT PROB ------ The probability that a point mutation will occur at any site.

8) INV PROB ------ The probability that a single inversion will occur during gametogenesis.

The remaining columns contain the results of the experiment.

8.3 Principle Experimental Results

Our primary objective in undertaking experimentation was to compare the behavior of the genetic adaptor with that of the correlation adaptor and to test our hypothesis that the genetic adaptor would do well in meta-environments with differing depths. Therefore, we will first examine

the results of experiments involving a single genetic adaptor with fixed coefficients immersed in meta-environments of depths one and two. The coefficients used in the adaptor were selected, as the result of limited experimentation, to do well in depth two meta-environments and although the results are sufficient to demonstrate the validity of our hypothesis there is little doubt that they can be improved upon. As we shall see in the next sections, in which the effects of the control coefficients are explored, many combinations were tried which did indeed produce results superior to those reported here. However, because of the amount of testing time involved, the versatility of those genetic adaptors was not as fully tested as was that of the genetic adaptor whose behavior is reported in this section.

Table 8.3.1 contains the results of an experiment in which the genetic adaptor was immersed in a depth one ME. The average number of trials in the five runs shown is 1077 trials. This may be compared to the concensus of 500-800 trials for the level one correlation adaptor and 5000-6000 trials for the level two CA. Thus it can be seen that for the depth one case, the genetic adaptor does slightly poorer than the matched (level one) CA, but very much better than the mismatched (level two) CA.

Table 8.3.2 contains the results of five experiments which matched the same genetic adaptor against five different depth two meta-environments. An average of 4465 trials was required to maximize the depths two MEs in the twenty-five runs shown. Actually, the majority of the runs did much better than this, but the average was substantially increased by four very high runs. In these runs, crucial combinations of alleles were lost from the population in the early stages and a maximum wasn't attained until they were re-introduced by point mutation.

EXPT NO	ME NO	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
56	1	1975	814	711	665	1221

TABLE 8.3.1 GENETIC ADAPTOR VS DEPTH 1 ME.

EXPT NO	ME NO	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
47	3	1414	3691	406	2821	1903
55	4	1535	10581	1528	4062	14633
60	5	3551	1556	2306	2817	4119
66	6	763	1226	1066	12018	24371
67	7	2884	1032	2921	3798	4622

TABLE 8.3.2 GENETIC ADAPTOR VS FIVE DISTINCT DEPTH 2 MES.

In spite of this, the average value is better than the expected value of at least 5000 trials for the level two CA and very much better than the expected value of 65,536 trials for the level one CA. Thus, the performance of the genetic adaptor is superior to that predicted by our original hypothesis.

To understand why this happened, we must examine more carefully the nature of the two algorithms and the MEs in which they were tested. Recall, first of all, that the simulated MEs are rich in the sense that they contain not just one point of maximum utility, but many. The correlation algorithm operates in a very cautious manner in that it saves information about all alleles good and bad. Consequently, in Phase III when the X array has become stabilized, any one of the points of maximum utility is equally likely to be attained. The CA, in effect,

converges upon all points of maximum utility simultaneously. If the CA were allowed to continue to operate past the point when it generated its first maximizing PVV, there is no reason to suppose that the next maximizing PVV would be identical to the first. On the other hand, the genetic adaptor is much more expeditious. It continually throws away information by replacing members of its population. The GA, in effect, converges to a region about a single point of high (and hopefully maximum) utility. If the GA were allowed to continue to operate after it had generated its initial maximizing PVV, in all likelihood, the next maximizing PVV it generated will be identical to the first. And, in fact, it would continue to generate copies of that maximizing PVV with ever-increasing frequency until it reaches an equilibrium point determined by the recombination noise level. This expediency is not without its dangers and the GA will occasionally become "hung up" in the region of a suboptimal point. This usually results in a prolonged run. Minimizing this effect requires careful adjustment of the coefficients controlling selection and recombination. This adjustment is the task of the meta-adaptor A' whose role in these simulations was played by the human experimenter. The meta-meta-environment seen by A' whose parameters are the control coefficients of the GA proved to have a high depth characteristic. In the sections that follow, we will examine the effects of these control coefficients in the light of the experiments carried out in pursuit of our primary objective. We should emphasize that we do not claim to have made thorough study of the control coefficients since such an effort would clearly be beyond the scope of this thesis. The evident success of the genetic adaptor would seem to justify a much more thorough and carefully controlled study of the control coefficients than we were able to afford.

8.4 Population Size

In the genetic adaptor, the gamete population serves both as the memory of the process and the repository of its variability. In the absence of mutation, the population is the sole source of new alleles. If the population is too small, it will often happen that crucial alleles will become extinguished before their potential merit can be exploited by the selection mechanism. In fact, in the depth one ME, many GAs with no mutation and populations of less than 200 were tested and none of them succeeded in generating an optimum PVV before suboptimal stagnation set in. On the other hand, if the population is very large, each trial has proportionally less effect on the composition of the population and although there is less chance of losing crucial alleles, more trials will be required before good genotypes begin to predominate.

The depth one experiments without mutation shown in Table 8.4.1 are really not too conclusive although the tendency for suboptimal stagnation to occur less frequently in larger populations is evident. When mutation is added (Table 8.4.2), the picture becomes a bit clearer. A population of 100 is definitely too small and 200 seems to be about right. Note how mutation by adding variability does offset the effect of low population. As we might expect, as the population is increased, there is less variation between the results of runs in that both very long and very short runs are less likely to occur. The same trends are evident in the corresponding result for the depth two ME shown in Table 8.4.3 although in this case, a population of 400 and possibly even higher seems to be optimal.

8.5 Selection Factors

First, let us note that both extremes of selection will lead to

EXPT NO	ME NO	POP	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
61	1	400	854	693	1670	INF	834
62	1	300	878	INF	337	887	1287
63	1	200	913	364	1460	INF	INF

TABLE 8.4.1 POPULATION TESTS IN DEPTH 1 ME WITHOUT MUTATION

EXPT NO	ME NO	POP	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
56	1	400	1975	814	711	665	1221
68	1	300	836	958	572	1161	2316
69	1	200	212	516	649	1192	794
8	1	100	508	4316	249	8411	4926

TABLE 8.4.2 POPULATION TESTS IN DEPTH 1 ME WITH MUTATION

EXPT NO	ME NO	POP	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
47	3	400	1414	3691	406	2821	1903
64	3	300	4700	5224	740	794	4453
65	3	200	7696	17120	273	5383	9214

TABLE 8.4.3 POPULATION TESTS IN DEPTH 2 ME WITH MUTATION

poor results. In the case of overselection, the population very rapidly becomes composed exclusively of the descendents of individuals which did well in the first few blocks. The danger is that the population may "home in" on suboptimal genotypes and the variability that is essential for further improvement is extinguished. In the case of underselection, most trials result in the generation of exactly two offspring and selection becomes insensitive to small differences in utility. This also leads to long runs because the generation utility average increases slowly. In addition there is an increased danger that crucial alleles may be lost due to random genetic drift.

The selection function makes use of an estimated average utility value $\overline{\bar{Ou}}$ computed as follows:

$$\overline{\bar{Ou}(T)} = \frac{\overline{\bar{Ou}(T-1)}(B1-1) + \bar{Ou}(T)}{B1}$$

where T is the trial number. This weighted average is used as the standard against which the results of the current trial are judged. The control coefficient B1 affects this standard by giving disproportionate weight to more recent trials. Very high values of B1 lead to a stable but possible obsolete standard since each new trial contributes proportionally less weight. On the other hand, very small values of B1 lead to a more responsive standard, but one which is subject to short term fluctuations. In the special case B1 = T, the average utility for the entire run is used as a standard. Ideally, the standard should reflect the average utility attained by the parents of the gametes which make up the current population. This average is essentially the block or generation average and, if the algorithm is at all successful, it will increase much faster than the run average. It has been determined experimentally that a value of B1 which is equal to one quarter

of the population size produces a weighted average utility $\overline{O\bar{u}(t)}$ which very closely approximates the block average. If B1 is given a very large value or if the run average is used (B1 = T), each genotype will be judged with respect to a standard which is too low with a result that on the average more than two gametes will be generated for each trial thus effectively increasing the recombination rate. Other effects of a low value of $O_{\overline{U}}$ will be discussed below.

B2, the other control coefficient affecting selection appears explicitly in the selection function:

$$E = 2.0 \frac{O\bar{u}(T) + B2}{\overline{O\bar{u}(T)} + B2}$$

When the value of B2 is (algebraically) high, the effect of selection is minimized, most genotypes produce two offspring and the behavior of the algorithm becomes more random. When the value of B2 is (algebraically) low, the effect of selection is enhanced so that genotypes which do well produce many more offspring than those that do not.

In each of the tables in this section, the experiments are listed in the order of increasing amount of selection. Table 8.5.1 shows the results of variation of B2 with B1 = T in a depth 2 ME and no mutation. The results of equivalent experiments with mutation are given in Table 8.5.2. With no mutation, the effects of overselection are quite evident. These results, particularly with B2 = -18 or -19 and no mutation, do suggest an interesting possibility. For these values, the adaptor is definitely in a condition of overselection and a maximum is obtained either very quickly or else not at all. If the ME were such as to permit it, an optimum strategy would very likely be to start a run with a random initial population and allow it to continue until

EXPT NO	ME NO	B2	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
28	3	-10	2322	3772	INF	2532	408
71	3	-12	INF	INF	1351	1826	1180
19	3	-15	1348	1705	2272	2633	INF
29	3	-18	612	1741	968	698	INF
32	3	-19	INF	798	476	INF	INF
30	3	-20	INF	INF	INF	INF	INF
31	3	-22	INF	INF	INF	INF	INF

TABLE 8.5.1 SELECTION TESTS WITHOUT MUTATION. B1 = T.

EXPT NO	ME NO	B2	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
53	3	-12	6854	1265	4278	3151	6004
26	3	-15	12881	2007	1184	2122	969
54	3	-18	4490	935	961	2153	5869

TABLE 8.5.2 SELECTION TESTS WITH MUTATION. B1 = T.

a maximum was found or until the number of trials exceeded a fixed upper limit (e.g., 2000 trials). If the limit were exceeded, the process would be started again with a fresh random initial population.

Tables 8.5.3 and 8.5.4 contain the results with B1 = 100 both with and without mutation. Again, the same trends can be observed, but because a more accurate estimate of the population utility was used, the results are less eratic and the effects of overselection less pronounced. Experiment 52 is particularly striking in that the results are extremely good in four of the five runs while in the remaining run, they are very poor. A closer examination of the data reveals that this run did get hung up on a suboptimal plateau and that a large number of trials was required for mutation to dislodge it.

Table 8.5.5 contains the results of equivalent experiments in a depth one ME. Here it is apparent that in this less complex ME the adaptor is much less likely to become trapped on a suboptimal plateau so that more intensive selection may be tolerated.

During the course of the aforementioned selection experiments, two phenomena were observed which deserve further notice. The first, reverse selection, occurred when the value of B2 was very small (i.e., less than -19) and the results of the initial trials were such that the denominator of the selection function became negative. In that case, genotypes which did better than the weighted average were suppressed and those which did worse were encouraged. Thus the block average utility showed a steady decrease instead of an increase.

The second phenomenon might be termed decreasing selection pressure. It was observed that most of the runs began quite well in that initially the block average utility increased steadily. But, as the run progressed,

EXPT NO	ME NO	B2	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
72	3	-12	INF	1440	1791	2426	3347
73	3	-15	1536	1853	1905	INF	1347
35	3	-18	INF	INF	1538	INF	4204
36	3	-19	1184	948	INF	INF	INF

TABLE 8.5.3 SELECTION TESTS WITHOUT MUTATION. B1 = 100.

EXPT NO	ME NO	B2	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
51	3	-12	4077	2078	2216	3039	10862
47	3	-15	1414	3691	406	2821	1903
52	3	-18	456	27291	869	688	309

TABLE 8.5.4 SELECTION TESTS WITH MUTATION. B1 = 100.

EXPT NO	ME NO	B2	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
74	1	-12	765	1036	1202	490	2937
56	1	-15	1975	814	711	665	1221
70	1	-18	991	844	941	269	910

TABLE 8.5.5 SELECTION TESTS IN DEPTH 1 ME WITH MUTATION.
B1 = 100.

the incremental increase in block average utility became less and less until in the later stages of a run the adaptor seemed to flounder and selection seemed to be only weakly exercised. This may be partially explained by the fact that as the utility average of the population increases, there are fewer and fewer points in the ME which have a better utility. But this is not the whole story. Let us define the selection pressure P as the change in the number of offspring induced by a change in genotype utility. If we neglect the effect of the latest utility value $O\bar{u}$ on the weighted average utility $\overline{O\bar{u}}$, we obtain the following:

$$P = \frac{dE}{dO\bar{u}} = \frac{2.0}{\overline{O\bar{u}} + B2} .$$

Under normal circumstances as selection takes effect, $\overline{O\bar{u}}$ increases as the run progresses. But as $\overline{O\bar{u}}$ increases, the selection pressure decreases giving rise to a negative feedback effect. This is not necessarily undesirable since when the adaptor reaches a high plateau of utility the decreased selection pressure allows the population to retain a greater degree of variance. The increased variance increases the likelihood that the adaptor will jump from a suboptimal plateau to higher point.

As a consequence of the above observations we formulated an alternate selection function which precludes both reverse selection and decreasing selection pressure. That alternate selection function E' is given by the following:

$$E'(T) = 2.0 \frac{O\bar{u}(T) - \overline{O\bar{u}(T)} + B2'}{B2'}$$

Where the control coefficient B2' is a positive integer and is analogous to B2. As with B2, decreasing values of B2' cause intensified selection. E' was obtained from E by substituting B2' for $\overline{O\bar{u}} + B2$ in

the expression for E. Clearly, the denominator of E' cannot be negative and if the effect of $O\bar{u}$ on $\overline{O\bar{u}}$ is neglected as before, the selection pressure P' is given as follows:

$$P' = \frac{dE'}{dO\bar{u}} = \frac{2.0}{B2'} .$$

Thus the selection pressure P' remains constant throughout the run.

Tables 8.5.6 through 8.5.9 contain the results of experiments which were run using the genetic adaptor with the alternate selection function. These too are arranged in order of increasing selection intensity (decreasing values of B2') and include experiments both with and without mutation and with B1 values of both T and 100. The results parallel those obtained in experiments in which the standard selection function was used and, if anything, the trends observed there are more apparent here. The overselection in experiment 77 is particularly striking. In that experiment, four out of five of the runs were particularly short. The remaining run was very, very long because the adaptor became stuck on a suboptimal plateau (with a utility value of 31) and because it was very intolerant of suboptimal individuals it was unable to accumulate enough variability to escape for literally hundreds of generations.

8.6 Dominance

Although the adaptor contained no specific coefficients to control dominance, a dominance effect was observed which had a decided influence on the simulation. In our model, the dominance value is fixed to a locus and is independent of the allele value of that locus. Mutation, however, acts upon the allele value but not upon the dominance value. In addition there was a "grandfather" effect so that after ten to fifteen generations the majority of alleles at a locus were progeny of

EXPT NO	ME NO	B2'	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
41	3	7	INF	1799	1711	INF	1787
37	3	5	1992	1781	2322	1507	INF
39	3	3	INF	INF	INF	INF	INF

TABLE 8.5.6 ALTERNATE SELECTION TESTS WITHOUT MUTATION.
B1 = T.

EXPT NO	ME NO	B2'	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
75	3	7	1603	3618	5165	5239	8601
43	3	5	1425	9059	14213	2359	840

TABLE 8.5.7 ALTERNATE SELECTION TESTS WITH MUTATION.
B1 = T.

EXPT NO	ME NO	B2'	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
42	3	7	1531	2423	734	2148	INF
38	3	5	864	INF	2897	INF	INF
40	3	3	INF	636	577	INF	2479

TABLE 8.5.8 ALTERNATE SELECTION TESTS WITHOUT MUTATION.
B1 = 100.

EXPT NO	ME NO	B2'	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
45	3	7	1202	2115	2023	1165	4945
76	3	5	1309	939	1962	2403	3036
77	3	3	630	2038	331	122872	1356

TABLE 8.5.9 ALTERNATE SELECTION TESTS WITH MUTATION.
B1 = 100.

an identical successful ancestor and had the same dominance value. Consequently, after a time, all of the alleles of a given locus came to have the same dominance value although the allele value may have been modified by mutation. Thus each member of the population developed the same invariant dominance pattern which together with the operation of the dominance tie-breaking mechanism rendered the dominance ineffective. One of the advantages of dominance is that it acts to conserve variability by preserving, in recessive loci, alleles which are suboptimal with respect to the dominant genotype. This advantage was lost in the later stages of most of our simulation runs.

8.7 Crossover

Tables 8.7.1 to 8.7.3 contain the results which show the effects of variation in the crossover probability. The three values of crossover probability tested produced an average of one-half, one, and two crossovers per gamete generated. The lowest value (0.0625) does lead to results which are inferior to those obtained using higher values. There appears to be no substantial difference between the results produced by the two higher values (0.125 and .25). This is reasonable since multiple crossovers in the generation of a single gamete do not really add anything to recombination. The intermediate value (0.125) was used in the majority of the experiments because it required less real time to simulate.

8.8 Mutation

Mutation may be used to partially compensate for the effects of a small population or overselection. If the mutation rate is low and the adaptor gets on the right track, it quickly converges to an optimum value since there is little "noise" introduced into the system. On

EXPT NO	ME NO	XOVR PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
63	1	.125	913	364	1460	INF	INF
6	1	.25	396	INF	780	INF	320

TABLE 8.7.1 CROSSOVER TESTS IN DEPTH 1 ME WITHOUT MUTATION.

EXPT NO	ME NO	XOVR PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
25	3	.0625	INF	3122	INF	INF	2512
19	3	.125	1348	1705	2272	2633	INF
21	3	.25	INF	1305	1640	1058	2088

TABLE 8.7.2 CROSSOVER TESTS IN DEPTH 2 ME WITHOUT MUTATION.

EXPT NO	ME NO	XOVR PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
57	3	.0625	4405	14950	1601	7442	5704
47	3	.125	1414	3691	406	2821	1903
49	3	.25	7996	1389	1851	3099	3234

TABLE 8.7.3 CROSSOVER TESTS IN DEPTH 2 ME WITH MUTATION.

the other hand if the adaptor gets stuck on a suboptimal plateau it becomes dislodged only with great difficulty. If the mutation rate is too high, it adds so much noise that it counteracts the selection process. Tables 8.8.1 and 8.8.2 contain the results of experiments with different mutation rates. The value of 0.00625 is certainly too low while the value of 0.0125 which was used in the majority of the experiments seems to be a good compromise. It might have been made a bit larger, but that possibility was not explored in depth since crossover and not mutation was intended to be our principle recombination method.

8.9 Inversion

The inversion results shown in Tables 8.9.1, 8.9.2 and 8.9.3 were some what disappointing. The most obvious effect of inversion is a large increase in the length of the runs. Recall that one of the consequences of inversion is a decrease in the effective crossover rate since crossover is inhibited between parts of chromosomes which are not locus homologous.

EXPT NO	ME NO	MUT PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
19	3		1348	1705	2272	2633	INF
27	3	.00625	3915	4142	1965	1554	1453
26	3	.0125	12881	2007	1184	2122	969

TABLE 8.8.1 MUTATION TESTS WITH B1 = T.

EXPT NO	ME NO	MUT PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
73	3		1536	1853	1905	INF	1347
48	3	.00625	49214	2036	21977	2422	923
47	3	.0125	1414	3691	406	2821	1903

TABLE 8.8.2 MUTATION TESTS WITH B1 = 100.

EXPT NO	ME NO	INV PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
4	1		348	1235	809	272	473
5	1	.125	1108	INF	1535	1976	376

TABLE 8.9.1 INVERSION TESTS IN DEPTH 1 ME WITHOUT MUTATION

EXPT NO	ME NO	INV PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
73	3		1536	1853	1905	INF	1347
59	3	.125	1418	INF	INF	INF	INF

TABLE 8.9.2 INVERSION TESTS IN DEPTH 2 ME WITHOUT MUTATION

EXPT NO	ME NO	INV PROB	TRIALS REQUIRED TO ATTAIN MAXIMUM RUN 1	RUN 2	RUN 3	RUN 4	RUN 5
47	3		1414	3691	406	2821	1903
58	3	.125	2858	35875	16977	19329	5667

TABLE 8.9.3 INVERSION TESTS IN DEPTH 2 ME WITH MUTATION.

As we have seen, a lower crossover rate has an adverse effect on the simulation and it is this effect that was predominant in our results. The desirable consequence of inversion, which is the appearance in the population of gametes whose geographical gene linkages reflect combinations which are rewarded by the depth two ME, was not observed. There are a number of reasons for this. First of all, as we noted in the Chapter 4, the requirement that simulated MEs of all depths have uniform characteristics under random sampling necessitated a dilution of the depth characteristic which was accomplished by including only a fraction of the possible parameter combinations in the ME array. Next, the sub-utility range (1-4) was very restricted. Finally, there were a number of potentially rewarding linkages competing for favor. To summarize, it seems, in retrospect, that our depth two simulated MEs did not offer a sufficient reward to linkages to permit them to show up in these simulations. It is also true that selection coupled with the other recombination methods proved to be more powerful than had been anticipated.

8.10 Practical Considerations

In general, the results of this chapter have implications for two general areas — problem solving and the study of genetic systems. We shall treat these separately.

With respect to problem solving, these experiments have demonstrated that the genetic adaptor is a powerful tool for dealing with a range of difficult meta-environments whose performance exceeds that of even the idealized CA. As presently constituted, the gamete population requires eight words of storage per chromosome or 3200 words for a population of 400 chromosomes. If inversion (which proved to be ineffective) were eliminated, the inactive sites could have been suppressed and

the space required for population storage cut in half so that the 400 chromosomes would have required only 1600 words. Recall that the level one correlation adaptor required 256 words and the level two correlation adaptor 7168 words for array storage. The computation speed of the simulation is independent of the population size and for long runs with large (400) populations (where the initialization and input-output time is a small fraction of the total) the simulation proceeds at an average rate of 167 trials per minute. This compares quite favorably with the speed of the level one CA (285 trials per minute) and is vastly superior to the estimated speed of the depth two CA (10.2 trials per minute). In addition, the control coefficients of the genetic algorithm allow the experimentor wide flexibility for adjustment of the behavior of the adaptor to suit the requirements of the ME or to his own personality.

The practical consequences to the individual interested in the behavior of genetic systems, can be stated quite simply. Simulations of the type described here give him the ability to adjust the basic parameters of the genetic system (e.g., crossover and mutation rate) and to study their effect on a population of 200 diploid individuals over a span of 50 generations in one hour at a cost of about eight dollars.

8.11 Guidepost

This concludes the exposition of our study of the genetic algorithm. The final chapter concludes this thesis with a summary of its contents and a look to the future.

9 CONCLUSION

9.1 Summary

At the outset, we indicated that our goal was to contribute to the development of a theory of Adaptive Systems. Our approach was to limit our attention to some rather specialized aspects of the general field and explore these in depth so that we could obtain concrete results. In particular we have accomplished the following:

1) Developed a paradigm for Adaptive System which effectively separates the learning algorithm from the heuristic aspects which lie in a meta-environment.
2) Demonstrated how the work of other workers in the field can be fit into the above paradigm.
3) Defined a concept of meta-environmental depth which was intended to reflect the degree of interaction among the parameters of the meta-environment. We presented an example of meta-environment which includes the game of Hexapawn and showed how its depth could be made to depend upon the strategy chosen by a fixed opponent.
4) Presented a correlation adaptor which is a prototype and extension of adaptive algorithms which have appeared in the literature.
5) Developed expressions which can be used to determine the results of interaction of correlation adaptors of arbitrary level with meta-environments of arbitrary depth.
6) Evaluated these expressions for a special case of simulated meta-environments and computed the mean number of trials required for their maximization under a variety of conditions. These results indicate that the level of the correlation

adaptor must be closely matched to the depth of the meta-environment and that it is of limited use for meta-environments of depth greater than three.

7) Formulated an alternative algorithm based upon the mechanism of natural genetic systems.

8) Demonstrated by means of simulation experiments that the genetic adaptor does well in both depth one and depth two meta-environments when compared to the correlation adaptor. The price paid for this versatility is quite small and, under certain conditions, the genetic adaptor is superior even to the matched correlation adaptor.

9) Determined the effects of a limited range of variation of the control coefficients upon the optimizing behavior of the genetic adaptor.

9.2 What Next?

The work reported in the previous chapters has suggested a number of avenues for future research. We conclude by listing some of these. They deal almost exclusively with the genetic algorithm since, once the premises which underlie the paradigm and the depth characterization of meta-environments have been accepted, the analysis contained herein pretty well exhausts the possibilities of the correlation algorithm.

Many theoretical and practical aspects of the genetic adaptor which are relevant both to the science of genetics and that of problem solving remain unexplored. The primary need is for a systematic study of the effects of the various control coefficients so that the statements and inferences made in this paper may be given a more precise formulation. Suggestions for specific experiments are given below:

1) Presently each mating (trial) produces an average of two offspring. Varying the mean number of offspring produced per trial should effectively vary the overall recombination rate in a uniform manner. Once a proper balance between the different recombination methods has been determined, this coefficient should prove useful.

2) Meta-environments which provide a stronger incentive for gene linkages should be studied in order to evaluate the inversion mechanism. The effects of inactive sites should be explored.

3) The effects of much larger population and the relationships between mutation rate and population size need to be determined. By eliminating inactive sites, it is a simple matter to double the maximum allowable population to a value of 800. The simulation programs are now coded extensively in FORTRAN. By recoding them in assembly language it should be possible to both decrease the execution time and free more core storage in order to allow for populations as large as 1000. In addition a recently announced computer model with increased storage capacity will allow simulation of populations as large as 8000 using these same simulation programs.

4) The effects of changes to the mutation mechanism which would allow mutations of the dominance value and simultaneous mutations to more than one site should be investigated.

5) The dominance tie-breaking mechanism should be changed so dominance will remain effective throughout the run and recessive loci can become repositories of variance.

6) The "grandfather" effect can be tested by using the inactive sites to label the adjacent active sites and modifying

the crossover mechanism in such a way that the inactive sites remain attached to the active sites which they identify.

7) The potential of self contained controls contained in the inactive sites should be explored.

8) The simulations may be extended to include both different chromosome structures (e.g., more sites each containing fewer alleles) and multiple chromosome pairs per individual.

9) The genetic algorithm should be tested against some "real" meta-environment — perhaps in the fields of pattern recognition or process control.

10) Finally, the search for analytical tools suitable to deal with genetic adaptors should not be abandoned. To the contrary, we are confident that the insight gained as the result of simulation experiments such as outlined here will eventually lead to the development of a more rigorous theory.

BIBLIOGRAPHY

Andrews, P. B. [1962], "A Survey of Artificial Learning and Intelligence," IBM Research Report RC-612.

Bledsoe, W. W., and Browning, I. [1959], "Pattern Recognition and Reading by Machine," Proceedings of Eastern Joint Computer Conference, pp. 225-232.

Chow, C. K. and Liu, C. N. [1966], "An Approach to Structure Adaptation in Pattern Recognition," IBM Research Paper RC-1600.

Dobzhansky, T. [1951], Genetics and the Origin of Species, Columbia University Press, New York.

Fein, L. [1964], "The Artificial Intelligensia," IEEE Spectrum 1-2, pp. 74-87.

Feller, W. [1957], An Introduction to Probability Theory and Its Applications, Vol. 1, Second Edition, John Wiley and Sons, New York, N.Y.

Fisher, R. A. [1958], The Genetical Theory of Natural Selection, Dover Publications, New York.

Fraser, A., Burnell, D., and Miller, D. [1966], "Simulation of Genetic Systems - X. Inversion Polymorphism," J. Theoret. Biol. 13, pp. 1-14.

Friedberg, R. M. [1958], "A Learning Machine, Part I," IBM J. Res. Develop., 2, 2.

Friedberg, R. M., Dunham, B., North, J. H. [1959], "A Learning Machine, Part II," IBM J. Res. Develop., 3, p. 282.

Gardner, M. [1962], "Mathematical Games," Scientific American, 206, pp. 138-144, March.

Herskowitz, I. H. [1965], Genetics, Second Edition, Little, Brown and Co., Boston, Mass.

Holland, J. H. [1962], "Information Processing in Adaptive Systems," in Information Processing in the Nervous System, III of Proc. of the International Union of Physiological Sciences, pp. 330-338, XXII International Congress, Leiden, Netherlands.

Kimura, M. [1958], "On the Change of Population Fitness by Natural Selection," Heredity 12, 2, pp. 145-167, May.

Mayes, C. H. [1963], "Adaptive Threshold Logic," Stanford Electronics Laboratory Report No. SEL-63-027.

Mealy, G. H. [1955], "A Method for Synthesizing Sequential Circuits," Bell Sys. Tech. Jour., 34, 5, pp. 1045-1079, Sept.

Minsky, M. [1961], "Steps Toward Artificial Intelligence," Proc. IRE, 49, pp. 8-30, January.

Mood, A. M. and Greybill, F. A. [1963], Introduction to the Theory of Statistics, Second Edition, Mc Graw-Hill, New York, N. Y.

Moore, E. F. [1956], "Gedanken Experiments on Sequential Machines," in Automata Studies, Annals of Mathematics Studies No. 34, pp 129-153, Princeton University Press, Princeton, N.J.

Newell, A. and Simon, H. A. [1964], "Problem Solving Machines," International Science and Technology, 36, pp. 48-62, December.

Rosenblatt, F. [1962], Principles of Neurodynamics, Sparton Books, Washington, D.C.

Sager, R. and Ryan, F. J. [1962], Cell Heredity, John Wiley and Sons, New York, N.Y.

Samuel, A. L. [1959], "Some Studies in Machine Learning Using the Game of Checkers," IBM J. Res. Develop., 3, 3, pp. 210-229, July.

Solomonoff, R. J. [1966], "Some Recent Work in Artificial Intelligence," Proc. IRE, 54, 12, pp. 1687-1697, December.

Uhr, L., and Vossler, C. [1961], "A Pattern Recognition Program That Generates, Evaluates and Adjusts Its Own Operators," Proc. Western Joint Computer Conf., 555-569.

Widrow, B. [1962], "Generalization and Information Storage in Networks of Adaline Neurons," in Self-Organizing Systems - 1962, pp. 435-462, Sparton Books, Washington,D.C.