DOCUMENT RESUME

ED 082 486 EM 011 455

AUTHOR	Stinaff, Russell D.
TITLE	Application of Computer Techniques to Instructional Research. Final Report.
INSTITUTION	Clemson Univ., S.C. Coll. of Engineering.
SPONS AGENCY	National Center for Educational Research and Development (DHEW/OE), Washington, D.C.
BUREAU NO	BR-1-0539
PUB DATE	May 73
GRANT	OEG-4-71-0036
NOTE	97p.
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	Algorithms; *Computer Oriented Programs; Computer Science; Educational Research; *Learning Theories; Mathematical Models; *Models; *Paired Associate Learning; *Search Strategies; Sequential Approach; Stimuli; Teaching Methods; Technical Reports
IDENTIFIERS	Dynamic Programing; Exhaustive Search Method; One Element Model; Random Trial Increments Model; RTI Model; Single Operator Linear Model

ABSTRACT

The problem of structuring sequences of instructional stimuli such that learning is optimized is modelled as a sequential decision problem with an imbedded mathematical model of learning providing a criterion function. Three types of optimization methods for such a representation are investigated for the specific case of paired-associate learning using the single-operator linear model, the one-element model, or the random-trial increments (RTI) model. Globally optimal exhaustive-search methods, such as Dynamic Programing, are found to be impractical for all but the simplest problems. due to inherent dimensionality limitations. Algorithmic methods, whereby the optimal decision at each step may be specified immediately without recourse to extensive look-ahead search, appear to be sufficient for the models investigated, primarily due to the absence of stimulus interaction. An optimal algorithm is specified for a class of learning models which includes the linear, one-element, and RTI models as special cases. Certain previously reported optimal algorithms are shown to be special cases of this algorithm. Finally, a heuristic search technique is outlined as a possible optimization method for problems too large for exhaustive-search solution and too complex for algorithmic solution. (Author)

Final Report

Project No. 10539
Grant No. OEG-4-71-0036

Russell D. Stinaff
College of Engineering
Clemson University
Clemson, S. C. 29631

APPLICATION OF COMPUTER TECHNIQUES TO INSTRUCTIONAL RESEARCH

May 1973

U. S. DEPARTMENT OF HEALTH, EDUCATION, AND WELFARE

Office of Education

National Center foi Educational Research and Development

Abstract

The problem of structuring sequences of instructional stimuli such that learning is optimized may be modelled as a sequential decision problem with an imbedded mathematical model of learning providing a criterion function. Three types of optimization methods for such a representation have been investigated for the specific case of paired-associate learning using the single-operator linear model, the one-element model, or the random-trial increments model. Globally optimal exhaustive-search methods, such as Dynamic Programming, have been found to be impractical for all but the simplest problems, due to inherent dimensionality limitations. Algorithmic methods, whereby the optimal decision at each step may be specified immediately without recourse to extensive look-ahead search, appear to be sufficient for the models investigated, primarily due to the absence of stimulus interaction. An optimal algorithm is specified for a class of learning models which includes the linear, one-element, and RTI models as special cases. Certain previously reported optimal algorithms are shown to be special cases of this algorithm. Finally, a heuristic search technique is outlined as a possible optimization method for problems too large for exhaustive-search solution and too complex for algorithmic solution.

Final Report

Project No. 10539
Grant No. OEG-4-71-0036

APPLICATION OF COMPUTER TECHNIQUES TO INSTRUCTIONAL RESEARCH

Russell D. Stinaff
College of Engineering
Clemson University
Clemson, S. C. 29631

May 1973

The research reported herein was performed pursuant to a grant with the Office of Education, U.S. Department of Health, Education, and Welfare. Contractors undertaking such projects under Government sponsorship are encouraged to express freely their professional judgement in the conduct of the project. Points of view or opinions stated do not, therefore, necessarily represent official Office of Education position or policy.

U.S. DEPARTMENT OF
HEALTH, EDUCATION, AND WELFARE

Office of Education
National Center for Educational Research and Development

Table of Contents

List of Tables

List of Figures

Figure Page

Chapter I - Introduction

This investigation is concerned with the problem of structuring sequences of instructional stimuli such that learning is optimized. The particular type of learning considered is that of "paired-associates", where one "trial" of a stimulus-response pair consists of the presentation of the stimulus member of the pair, followed by the subject's response, followed by presentation of the response member for reinforcement. Learning a list of foreign language vocabulary pairs in this manner can be thought of as an example of paired-associate learning. The optimization of learning, in the sense considered in this investigation, can take the form either of maximizing learning for a specified number of trials, or of minimizing the number of trials necessary to achieve a specified level of learning. The quantitative evaluation of "level of learning" takes the form, in most cases, of an expected value of a test score obtained after learning has taken place.

The optimization problem, for the purposes of this investigation, was considered abstractly as a sequential decision process with an imbedded mathematical model of learning providing a criterion function. The problem investigated can be expressed as follows: "Given a mathematical model of paired-associate learning and a set, S, of stimulus-response pairs to be learned, which element of S should be selected for presentation at each trial so that either learning is maximized for a given number of trials, m, or the number of trials necessary to achieve a given level of learning is minimized?" The sequence of presentations, $(s_1, s_2, \ldots, s_m)$, thus obtained will be referred to as the optimal presentation strategy for the given sequential decision problem.

It should be emphasized at this point that the primary orientation of the research was toward the investigation of techniques of solution, and particularly computer-oriented techniques, for the abstract optimization problem just stated, as opposed to any investigation of the psychological relevance of the processes.

A few general, but hopefully not very restrictive (in terms of psychological relevancy) assumptions are made concerning the framework of the sequential decision problem. First, it is assumed that the presentation strategy can be either response-insensitive or response-sensitive, depending on the model of learning used. Secondly, it is assumed that the "state" of the model, in the form of a state vector whose components consist of the probabilities of incorrect response, appropriately quantized, for each of the elements of the stimulus set, S, can be explicitly determined at each stage of the sequential decision process. For response-sensitive strategies, this determination will, of course, depend on the actual or simulated response history of the subject. It is further assumed that the effect on the state of the model of selecting a particular stimulus for presentation can be determined at each stage of the process, in terms of either explicit or expected changes in probabilities of incorrect response. It is assumed that in the case of response-sensitive strategies, where expected values of change in state must be used, that only two responses are possible, namely "correct" and "incorrect".

Figures 1 and 2 illustrate, in terms of the framework just described, the sequential decision processes applicable to learning models which correspond to deterministic and non-deterministic state transitions, respectively. Response-insensitive strategies may correspond to deterministic or non-deterministic transitions, depending on the learning model used, while response-sensitive strategies will generally correspond to non-deterministic transitions. Although certain of the learning models used imply further restrictions, such as non-interaction of stimuli, the general framework proposed for the problem necessitates no further restrictions.

The sequential decision process for models imposing deterministic state transitions is illustrated in Figure 1. It is assumed that the model is initially in some arbitrary state, Q_{01}. State Q_{ij} is defined as follows:

$$Q_{ij} = <q_{ij}^{(1)}, q_{ij}^{(2)}, \ldots , q_{ij}^{(n)}> \qquad (1)$$

where

$$q_{ij}^{(r)} = \text{probability of incorrect response to rth stimulus}$$

The n transitions emanating from this initial state (Node 01) indicate that there are n possible stimuli to choose from for the first trial and, in general, n different possible state transitions, depending on the choice. The transitions will be determined by the imbedded model of learning. Although most of the particular learning models used imply independence of stimuli (i.e., each component of the state vector is a function only of the presentation of the corresponding stimulus), the decision process has been deliberately formulated to allow dependence of each component on all presentations and, by implication, on time (to include memory effects). Stage 1 illustrates the n new states which can result, corresponding to each of the n stimuli, if chosen for presentation. In general, these states will be different, although one or more of them could conceivably be identical with the initial state. Stage 2 continues the process definition by illustrating all possible state transitions from each of the possible states at Stage 1. A dotted transition is shown between Q_{1n} and Q_{2i} to illustrate that, in general, any node in the graph below the initial node need not have a unique predecessor. The number of states at any stage will be less than or equal to the number of states in the

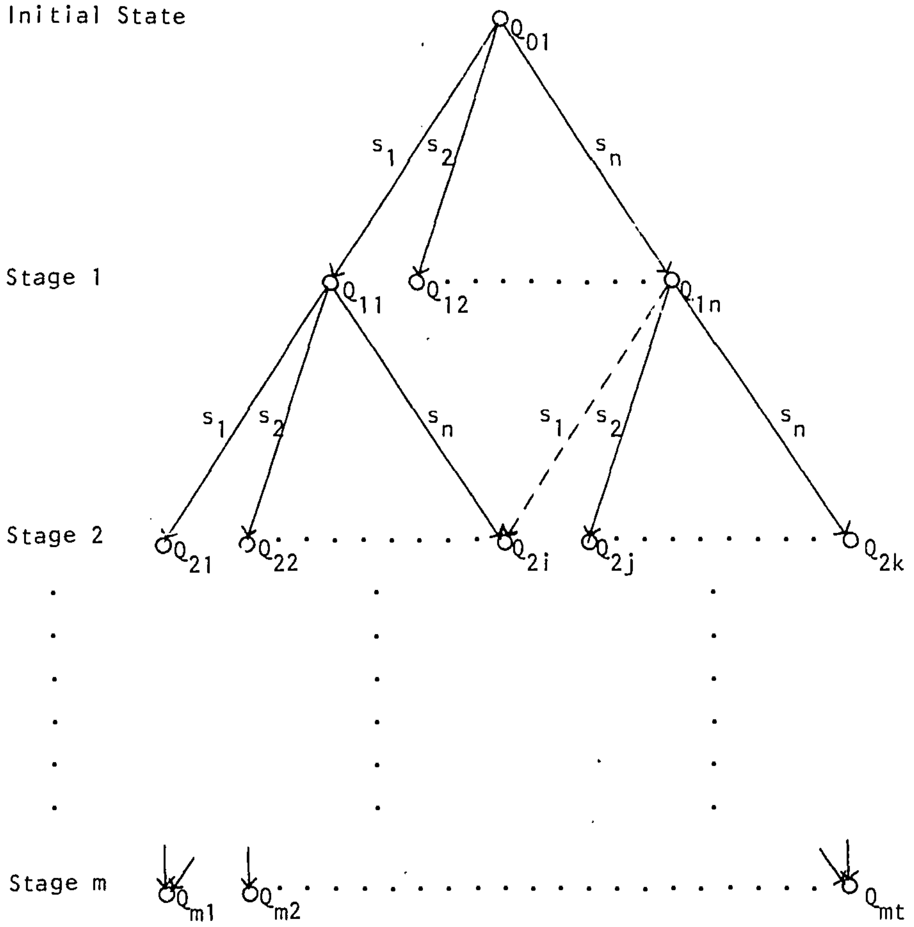


Figure 1 Sequential Decision Graph: Deterministic Transitions

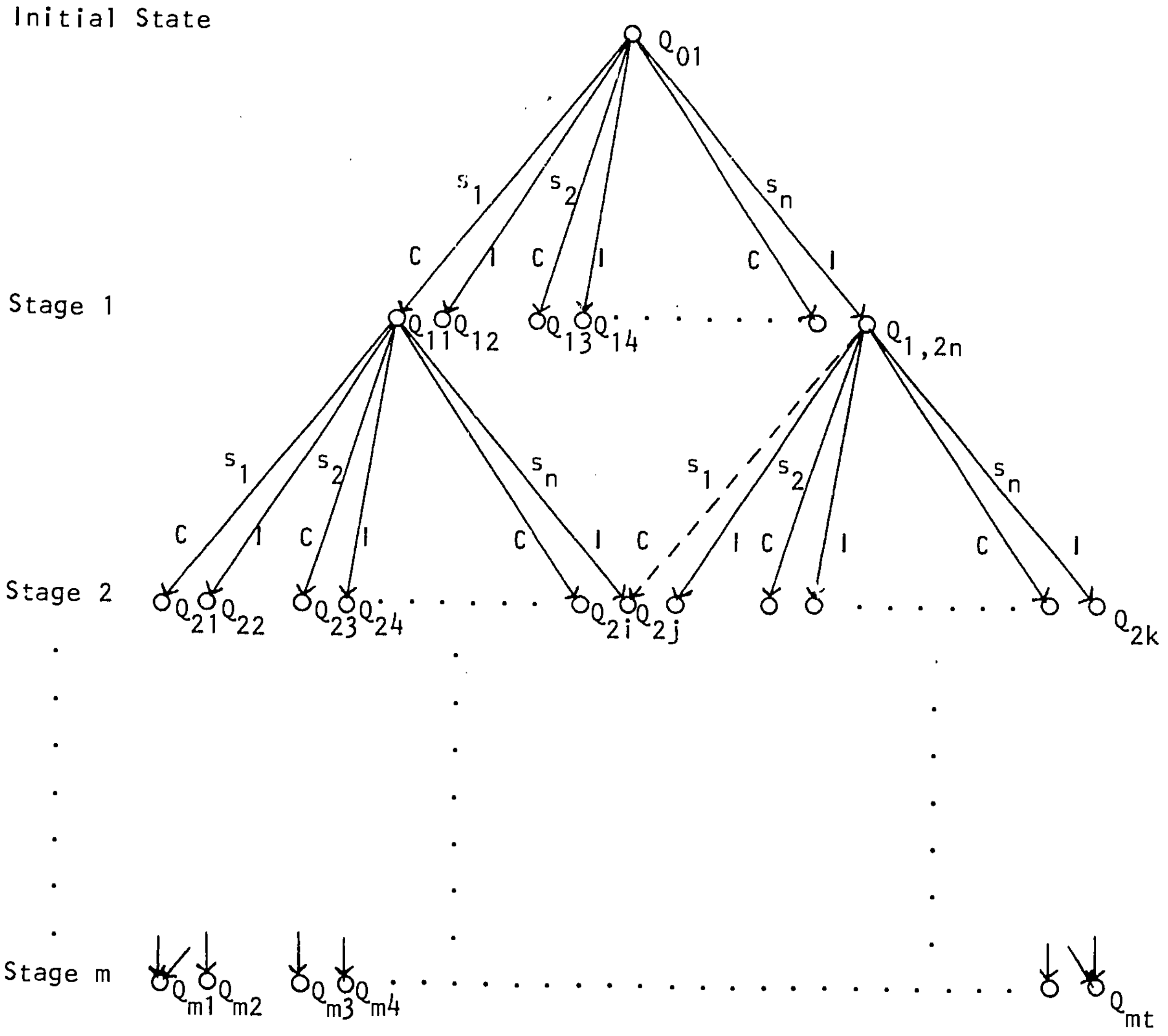


Figure 2 Sequential Decision Graph: Non-deterministic Transitions

state space, namely z^n, where z is the number of quantization levels.

An optimal presentation strategy for the process illustrated in Figure 1 is defined as any sequence of presentations, $(P_1,P_2,\ldots,P_m)$, where each P_i is chosen from the set $(s_1,s_2,\ldots,s_n)$, which maximizes an expected test score after m trials. The expected test score in this case is defined as follows:

$$E\{T_{mj}\} = (100/n)\sum_{k=1}^{n}(1-q_{mj}^{(k)}) = 100 - (100/n)\sum_{k=1}^{n} q_{mj}^{(k)} \qquad (2)$$

where the $q_{mj}^{(k)}$ are the components of the jth state of stage m.

As illustrated in Figure 2, the sequential decision process is essentially the same when the state transitions are non-deterministic, except that more than one transition is possible as the result of the presentation of a particular stimulus. The particular framework illustrated, with two possible transitions for each presentation, is applicable either to dichotomous response-sensitive models, such as the one-element model with correct and incorrect responses, or to two-valued stochastic-increment models, such as the random-trial increments (RTI) model. The labelling of Figure 2 denotes correct (C) and incorrect (I) responses corresponding to the presentation of each possible stimulus. It is, of course, true that P{C}+P{I}=1. The labelling could just as well correspond to "increment" (I) and "no increment" (C) for the RTI model. The other difference inherent in the non-deterministic process illustrated by Figure 2 is that, in the normal usage, optimization makes sense only in the context of an expected value of $E\{T_{mj}\}$. In other words, for the deterministic process, given the same initial state and the same model of learning, the same presentation strategy will always be optimal; for the non-deterministic process, on the other hand, the best that can be done, a priori, is to specify an algorithm which will guarantee, on the average, the optimal value of $E\{T_{mj}\}$.

The remainder of this report will be concerned with the investigation of different approaches to the problem of optimal instruction sequencing formulated as one of the sequential decision processes of the type illustrated by Figures 1 and 2. This formulation includes as special cases several previous investigations reported by other authors. These investigations will be commented upon at the appropriate point in the report. The Methods section of the report discusses the theoretical and experimental approaches taken to the problem and generally follows the organization of the Results section. Part A of the Results section (Chapter III) is concerned with exhaustive (globally optimal) optimization methods, such as Dynamic Programming, and includes comments on previous investigations involving this approach. Chapter III-B discusses algorithmic methods, including the specification of an optimal algorithm applicable to a class of learning models which includes the single-operator linear model, the one-element model, and the RTI model as special cases. Also included in this section are the results of a number of Monte Carlo simulations designed to determine the efficacy of the optimal algorithm relative to standard cyclical or random presentation strategies. Chapter III-C outlines a possible heuristic approach to the optimization problem for more complex learning models which cannot be optimized algorithmically. A primary advantage of this heuristic state-space search approach is that it provides a means of overcoming the difficulties of dimensionality inherent in methods such as Dynamic Programming for problems of even moderately realistic complexity. The Conclusions section includes an evaluation of the optimization methods proposed and suggestions for further research.

Chapter II - Methods

The starting point for the initial research plan was an investigation of the applicability and limitations of Dynamic Programming (Bellman, 1957, 1961; Bellman & Dreyfus, 1962) in the solution of the general optimization problem described in the Introduction. Dynamic Programming approaches to problems of this type have been taken or suggested by various researchers, including Smallwood (1962), Matheson (1964), Groen and Atkinson (1966), Smallwood (1971), and Calfee (1970). The approach taken in this investigation was to attempt to determine general criteria of applicability of Dynamic Programming to problems of the type described, and to outline practical limitations of this method. In addition, dimensionality reduction techniques and modified forms of Dynamic Programming, such as State-increment Dynamic Programming (Larson, 1968), were investigated for their potential in increasing the practical applicability of this form of solution.

In the second phase of the research, algorithmic optimization methods were investigated (i.e., methods by which the optimal strategy can be specified outright, rather than reconstructed by means of search techniques). Monte Carlo simulations of the instructional process for several such algorithmic methods were conducted for the purpose of determining the theoretical effectiveness of these methods. Included in this work was a simulation of the experiment conducted by Dear, et al. (1965), which was designed to test an optimal presentation strategy based on the one-element model of learning. The purpose of the simulation was to answer some questions brought out by their study and to attempt to obtain more substantive verification of their conclusions. The method used was straightforward repetitive stochastic simulation with sample sizes determined in part by tolerance criteria on the variance of the sample means. The simulations were conducted on a PDP-15/40 computer using an additive pseudo-random number generation scheme, and on an IBM 370/155 computer using the SSP power residue method. Common runs of certain cases were made using both machines to detect bias in the results. As the simulation programs used were fairly short and straightforward, representative examples are included for reference in the Appendix, along with verification data for the pseudo-random number generation schemes and programs to determine confidence levels on the sample means.

The final phase of the research was concerned with the investigation of optimization methods suitable for problems not apparently amenable to algorithmic or Dynamic Programming solution. It is anticipated that one source of such problems will be learning models which allow for general stimulus interaction (and by implication, memory of a sort). The approach taken was to investigate the applicability of certain heuristic state-space search methods developed in the field of Artificial Intelligence (cf. Nilsson, 1971; Dreyfus, 1969; Hart, et al., 1968; Pohl, 1969). Since presently there are apparently no generally accepted learning models of the type necessitating such an approach, an attempt was made to formulate the heuristic methods in terms of a general class of learning models which would contain certain anticipated interactive features. As such, the heuristic methods proposed serve primarily an illustrative purpose. The heuristic solution paradigm will need to be refined by further research as more appropriately complex learning models are developed.

Chapter III - Results

A. Globally Optimal Search Techniques

Several investigators have proposed the use of Dynamic Programming in the solution of optimization problems similar to the problem outlined in the Introduction of this report. Dynamic Programming is the pricipal globally optimal search technique discussed in this section, but is by no means the only such technique. Smallwood (1962) is generally credited with making the first application of a Dynamic Programming type of solution to an instructional sequencing problem, although the problem examined is somewhat different and more specific than the problem considered in this report. A later investigation by Smallwood (1971) involved the application of Dynamic Programming to the solution of an instruction sequencing problem which included cost of instruction as an additional criterion. This problem falls more within the framework of classical Dynamic Programming than does the simple sequence optimization problem based solely on quantized learning as a criterion, and would seem to constitute a more valid application of this technique. Applications of Dynamic Programming principles in contexts similar to that of the present investigation are described by Dear (1964), Matheson (1964), Karush & Dear (1966), and Calfee (1970). The Dynamic Programming aspects of these investigations will not be discussed directly here, since the comments to follow regarding another report will generally apply to these as well.

The formulation which is perhaps most relevant to the present discussion is that of Groen and Atkinson (1966). In addition, this reference appears to be the most widely accepted and oft-cited general application of Dynamic Programming principles to optimal instruction-sequencing problems. The example chosen by the authors to illustrate solution by Dynamic Programming is that of a sequence of three presentations from a set of two instructional stimuli, using the single-operator linear model to specify state transitions. The "decision tree" used by the authors to illustrate the decision process is shown in Figure 3. The following comments regarding the application of Dynamic Programming to problems in optimal instruction-sequencing will stem largely from this example, but will not be restricted to it, since in all important respects, the general structure of Figure 1 is embodied in this example.

The first observation is that while the tree structure of Figure 3 was apparently chosen to more lucidly illustrate the branching characteristics of the decision process, it is, nevertheless, not the customary framework for a Dynamic Programming formulation of the problem. A Dynamic Programming formulation is ordinarily given in terms of a mapping of the state space of the process into itself at each

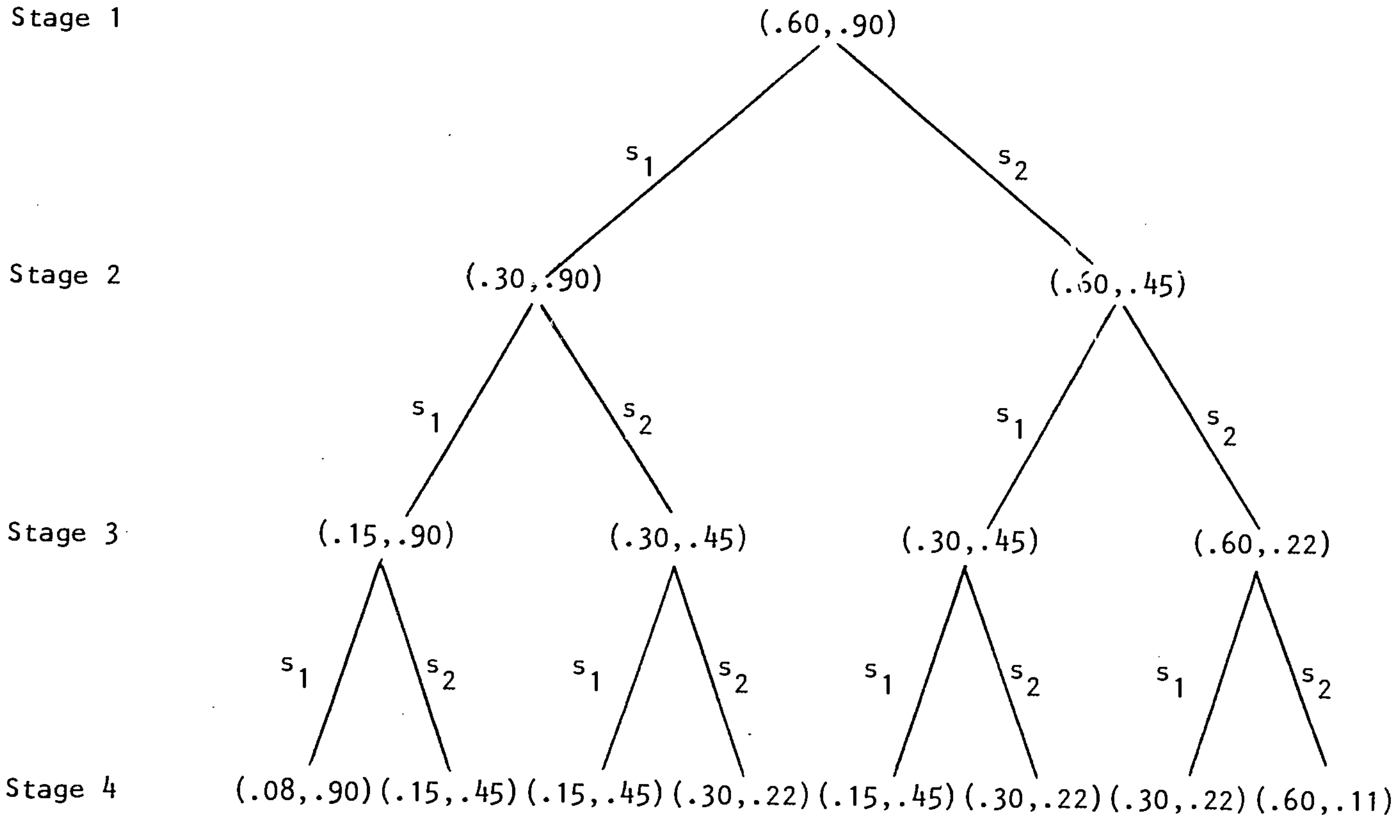


Figure 3 - Sequential Decision Process (Tree Formulation)

succeeding stage. In the customary formulation, the mappings stemming from the particular initial state shown would be illustrated as in Figure 4. In a cursory sense, of course, Figure 4 merely results from a consolidation of common states at each stage in Figure 3. In terms of practical considerations imposed by computer implementation, however, the implications are more far-reaching. First of all, forward Dynamic Programming, which could have advatages over backward Dynamic Programming for applications of this type, is not directly suited to the problem as implemented literally as shown in Figure 3. The reason for this is that common states in the forward direction have been separated and are treated in memory no differently from states which are actually distinct. It would be necessary to effect a search through the whole list of successors at each stage in order to identify common states so that recursive optimization could be performed, a task which grows exponentially in magnitude with the number of stages. In the customary Dynamic Programming formulation, the computational task at each stage is independent of the number of stages, since the entire state space is represented systematically in memory at each stage. The size of the state space depends, of course, on the quantization accuracy of the components of the state vector, as well as on the number of components, in contrast to the size of the final stage of the tree representation, which depends on the number of stages and the number of components (it is assumed that the number of components in the state vector is the same as the number of instructional stimuli in the set, S). Note that the quantization accuracy, i.e., the accuracy with which the magnitudes of the components of the state vector are represented, does not affect memory requirements for the tree representation insofar as increased accuracy does not require extended precision arithmetic in the computer.

For very small problems, the tree representation is less restrictive in terms of computer memory requirements. In the example of Figure 3, the last two stages impose a fast-memory requirement of 12 "nodes" (the term "node" is used as the measure of memory requirement, rather than a decomposition into a more detailed specification in terms of the number of words or bytes required to contain the information represented by each node, since this is less obfuscative and provides a greater degree of generality). A standard Dynamic Programming formulation, on the other hand, would impose a fast-memory requirement of $2x10^4$ nodes, assuming a quantization interval of 0.01. These requirements are determined from the fact that, in general, sufficient fast storage must be available to contain all information relating to any two successive stages in order to implement a practically feasible Dynamic Programming solution. Since the amount of storage required is stage-dependent in the tree formulation, the limitation is obtained from the storage required to contain the last two stages, since these are the largest. Hence, the 12-node figure. The requirement is the same for any two successive stages in the standard formulation, and is numerically equal to twice the size of the state space, which in this

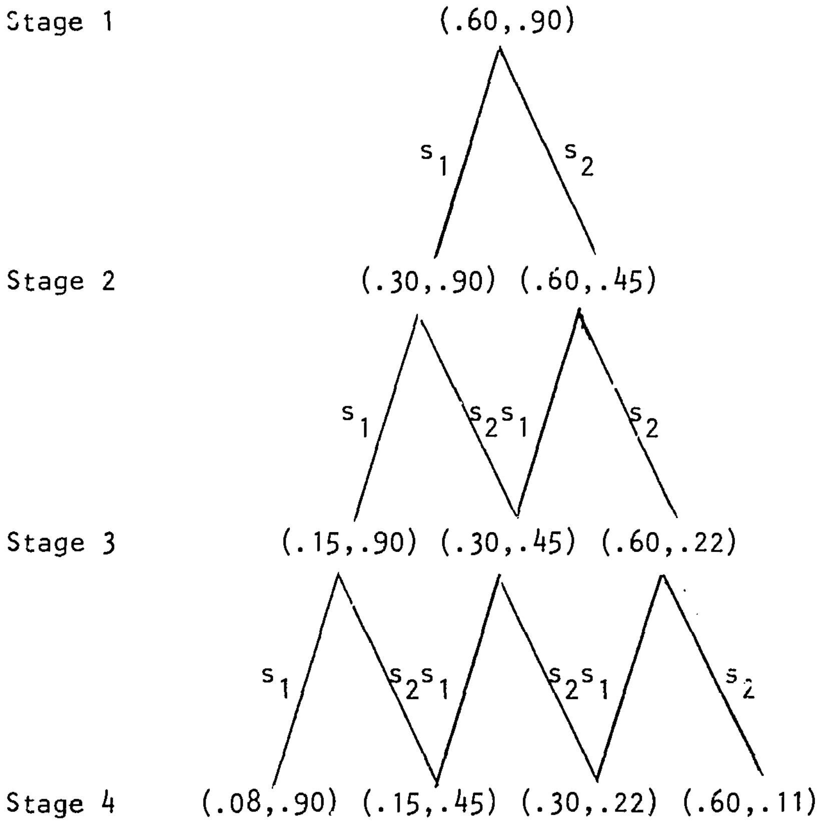


Figure 4 - Sequential Decision Process
(Standard Formulation)

case is $(100)^2$ or 10^4.

Again, the requirement is independent of the number of stages for the standard formulation. If the example process were continued to 21 stages (not seemingly unreasonable) the fast-memory requirement for the standard formulation would remain at 2×10^4, while the requirement for the tree formulation would increase to more than 1.5×10^6. In addition, direct comparisons such as this must be tempered with the fact that the tree representation provides an optimal solution to a sequential decision problem beginning at one particular initial state, while the standard formulation provides the solutions to a family of sequential decision problems beginning at any initial state in the state space. If optimal sequences are to be obtained for a number of different initial states, these sequences would, in effect, be obtained with a single recursive optimization pass through the stages using the standard Dynamic Programming formulation, while a separate complete optimization would be required for each initial state using the tree formulation.

The important point regarding fast-memory and computation-time limitations is that for any implementation involving straightforward Dynamic Programming techniques, the size of the problem which may be treated is severely limited. For example, for a state vector with only five components (five paired-associate items to be learned) and a quantization interval of 0.01, the fast-memory requirement would be 2×10^{10} nodes. Bearing in mind that the minimum conceivable associated byte requirement would be 4×10^{10}, and that even the largest present-day computers have fast-memory sizes on the order of only 10^6 or 10^7 bytes, this requirement is clearly prohibitive. For the tree formulation, five components would impose a fast-memory requirement of approximately 2×10^{10} nodes after only 16 stages. Table 1 is a compilation of fast-memory requirements in terms of number of nodes for the standard formulation over a range of state vector size and quantization interval size, while Table 2 shows the fast-memory requirements for the tree formulation over a range of state vector size and number of stages. The barriers imposed by fast-memory limits of present-day computers are illustrated by dashed lines in the tables. A straightforward Dynamic Programming solution would thus be implementable only for values of the parameters corresponding to points in the upper portion of each table.

Modified Dynamic Programming techniques, such as the State-increment Dynamic Programming of Larson (1968) can, in some cases, effect reductions of fast-memory requirements by two or three orders of magnitude. It would seem, however, that even with this great a reduction, the requirements for most cases of practical interest would still be prohibitive in general.

Dynamic Programming solutions, including those described above, as ordinarily implemented, are, in effect, breadth-first state-space search techniques, which impose a fundamental limitation in terms of memory requirements, as has been seen. It is possible, for the instruction

Table 1 - Fast-memory Requirements (in nodes) for Standard Formulation

	Quantization Interval		
n	0.1	0.05	0.01
2	10^2	4×10^2	10^4
3	10^3	8×10^3	10^6
4	10^4	1.6×10^5	10^8
5	10^5	3.2×10^6	10^{10}
6	10^6	6.4×10^7	10^{12}
7	10^7	1.3×10^9	10^{14}
8	10^8	2.7×10^{10}	10^{16}
9	10^9	5.1×10^{11}	10^{18}
10	10^{10}	1.0×10^{13}	10^{20}

Table 2 - Fast-memory Requirements (in nodes) for Tree Formulation

	m			
n	5	10	15	20
2	48	1.5×10^3	4.9×10^4	1.6×10^6
4	1.2×10^3	1.3×10^6	1.3×10^9	1.4×10^{12}
6	9.0×10^3	7.0×10^7	5.5×10^{11}	4.3×10^{15}
8	3.7×10^4	1.2×10^9	3.9×10^{13}	1.3×10^{18}
10	1.1×10^5	1.1×10^{10}	1.1×10^{15}	1.1×10^{20}
12	2.7×10^5	6.7×10^{10}	1.7×10^{16}	4.2×10^{21}
14	5.7×10^5	3.1×10^{11}	1.7×10^{17}	9.0×10^{22}
16	8.1×10^5	1.2×10^{12}	1.2×10^{18}	1.3×10^{24}
18	2.0×10^6	3.8×10^{12}	7.1×10^{18}	1.3×10^{25}
20	3.4×10^6	1.1×10^{13}	3.4×10^{19}	1.1×10^{26}

sequence optimization problem under consideration, to use a form of depth-first search to partially alleviate this space restriction. The tree formulation is more convenient for visualization of this method, although it is not requisite. To illustrate depth-first search, suppose that in the example of Figure 3, available fast memory was restricted to 3 nodes. This would prohibit application of standard Dynamic Programming, which would require 12-node storage. Instead of "searching" through the entire last stage, as would be done in the first step of a conventional Dynamic Programming formulation, the search could be conducted one node at a time from Stage 3. For example, the first node at the left of Stage 3 (.15,.90) could be stored, along with its successors, (.08,.90) and (.15,.45). On the basis of this information, it could be concluded that the optimal decision from node (.15,.90) would be s_2. The next step would be to attempt to "back up" to the parent node of (.15,.90) at Stage 2, which would be (.30,.90), and determine the optimal decision at that node. This cannot be done, however, until the optimal decision at node (.30,.45) (second from left) at Stage 3 is known, so this is determined next. This process is continued until the optimal decision at the initial node can be determined. For each successive step in the optimization, only three nodes at a time need be considered. In general, it is not necessary that the nodes contained in the portion of the tree being optimized be restricted to two successive stages. The only limitation is the size of the subtree which can be contained in memory at any given time. Even though this method may alleviate storage requirement restrictions in some cases, computation time restrictions do not permit any significant practical extension of the size of the problem which may be solved by globally optimal techniques of this type. In the previously cited example involving a state vector with 5 components, a subtree of 10 stages would be the largest that could realistically be optimized at one time. This means that the total optimization for 16 stages would involve $5^{(16-10)}=5^6 \approx 1.5\times10^4$ separate steps. If each step required one minute of computation time (a very conservatively low estimate), the entire optimization would take 250 hours, and each succeeding stage would multiply this figure by a factor of 5.

The final observation with regard to the application of Dynamic Programming to the solution of optimal instruction-sequencing problems is that in most cases, at least for the paired-associate framework, it would appear to be unnecessary. The reason for this is that path costs, which are ordinarily independent of the states of the process in the sense that they need not be a function of the states joined, are in fact a direct function of the states joined for processes of the type illustrated in Figure 1, of which the process of Figure 3 is a special case. The result is that with the process of Figure 3, for example, once the state with the minimum value of $q_4^{(1)}+q_4^{(2)}$ is located, the problem is effectively solved, since reconstruction of any path from any such state back to the initial state constitutes optimization. Recursive optimization is unnecessary. For general processes of the type illustrated in Figure 1, all that is necessary to effect optimization is to search the final stage for the state, Q_{mj}, for which $\sum_{r=1}^{n} q_{mj}^{(r)}$ is minimum and reconstruct any path of m segments back to the initial state. This path constitutes an optimal presentation strategy. It may be noted that since the models considered do not allow for stimulus interaction, the order of presentation is immaterial. All that is required to contruct an optimal presentation strategy is a count of the number of times each element of the stimulus set is to be presented. It may be observed, for example, that the three optimal presentation strategies ($s_1s_2s_2$, $s_2s_1s_2$, and $s_2s_2s_1$) derived by Groen and Atkinson for the problem of Figure 3 are merely all possible distinguishable permutations of the set (s_1,s_2,s_2). It is not even necessary that the fast-memory capacity be sufficient to contain the entire last stage. The search may be broken into as many segments as necessary, since the only information which must be transferred between segments is a single value of $\sum_{r=1}^{n} q_{mj}^{(r)}$ corresponding to the current optimal state and a state identifier. In theory, then, this simplification in optimization procedure entirely eliminates the fast-memory constraint inherent in a Dynamic Programming formulation. Unfortunately, the computation-time constraint prevents any significant extension of the size of the problem which may be treated. In the previous 5-component example, for instance, the state space size was 2×10^{10}. This means that the final stage must be divided into at least 2×10^{4} segments, allowing for a fast-memory capacity of 10^{6} nodes. If each segment could be searched in 10 seconds (a conservatively low estimate) the entire search would take approximately 55 hours, and each added component would increase this figure by two orders of magnitude.

The conclusion which apparently must be reached is that the practical applicability of globally optimal search techniques to the problem at hand, even for the simplest cases, is inherently and rather severely limited by dimensionality constraints.

B. Algorithmic Optimization Methods

Algorithmic methods of optimization for several special cases of the general optimal instuction-sequencing problem described in the Introduction have been proposed by various researchers. The term "algorithmic" is used here to indicate methods by which the optimal decision at each stage in succession may be determined directly from the present state of the process together with the learning model and possibly a response history. In other words, an algorithmic method is taken here to mean one by which each element of an optimal sequence may be determined directly as the sequence proceeds, as opposed to methods like Dynamic Programming, where the entire recursive optimization must be performed before the sequence can be determined.

The primary purpose of this section of the report will be to describe an optimization algorithm which applies to a broad class of learning models which includes as special cases the single-operator linear model (Bush & Sternberg, 1959), the one-element model (Bower, 1961; Estes, 1960) and the random-trial increments (RTI) model (Norman, 1964). The linear and RTI models are described in Figures 5-1 and 5-2, respectively, while the one-element model is described in Figure 6. The reason for discussing the algorithm in terms of these particular models is that they appear from the literature to be the most widely accepted and analyzed models of paired-associate learning. In addition to a description of the algorithm and its application to paired-associate learning processes based on these three models, some Monte Carlo simulation results will be discussed which compare the effectiveness of the optimal presentation strategy, as specified by the algorithm, with standard cyclical or random strategies.

Prior to the statement of the algorithm, a few definitions are in order. First, let the learning process to be optimized be represented by either Figure 1 (deterministic) or Figure 2 (non-deterministic), where the states Q_{ij}, and the function to be maximized, $E\{T_{ij}\}$, are as defined by equations (1) and (2), respectively, in the Introduction. Next, let the <u>gain function</u>, ΔQ_{ijk}, be defined as the increase in $E\{T\}$ produced by a transition from the jth state of Stage i to the kth state of Stage i+1. Thus,

$$\Delta Q_{ijk} = \sum_{r=1}^{n} (q_{ij}^{(r)} - q_{i+1,k}^{(r)}) \qquad (3)$$

If stimulus actions are treated as being independent by the learning model used in optimization, as is the case with the linear, one-element, and RTI models, then ΔQ_{ijk} is a function only of the component of the state vector corresponding to the stimulus presented. Thus, if stimulus r is presented at Stage i, then ΔQ_{ijk} is simply the difference

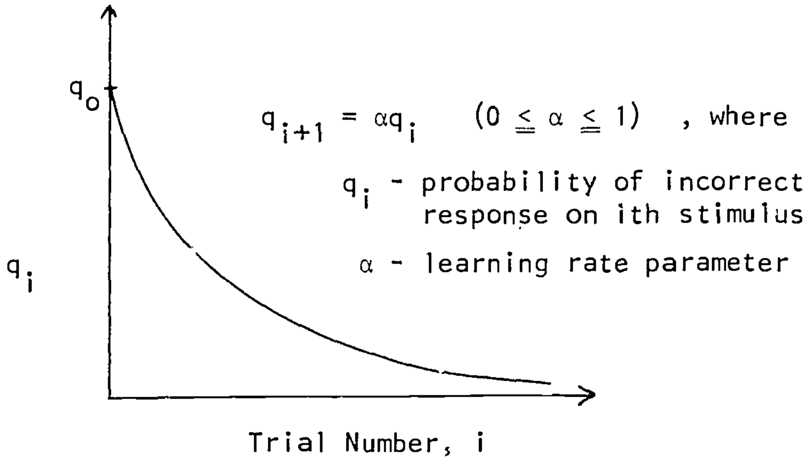


Figure 5-1 Single Operator Linear Model

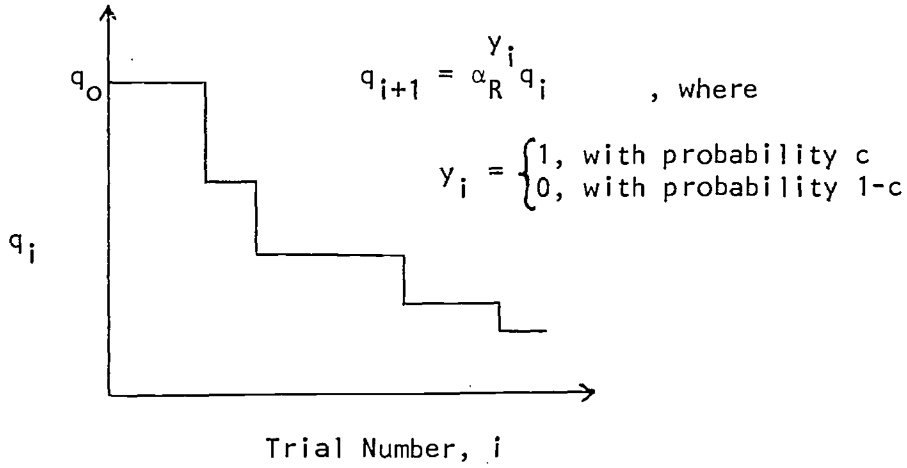
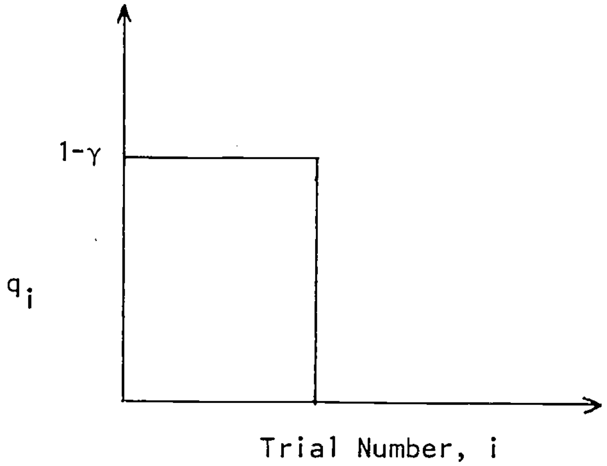


Figure 5-2 Random-trial Increments Model



$$\begin{array}{ccc} & C & \overline{C} \\ C & 1 & 0 \\ \overline{C} & \theta & 1-\theta \end{array} \qquad \begin{array}{ccc} & E & \overline{E} \\ E & 1 & 0 \\ \overline{E} & \gamma & 1-\gamma \end{array}$$

where,

C - conditioned state

$\overline{C}$ - unconditioned state

E - correct response

$\overline{E}$ - incorrect response

γ - "guessing" probability

θ - probability of transition from $\overline{C}$ to C

Figure 6 One-element Model

between the value of the rth component before presentation of the corresponding stimulus and the value after presentation:

$$\Delta Q_{ijk} = q_{ij}^{(r)} - q_{i+1,k}^{(r)} \qquad (4)$$

The class of learning models for which the optimization algorithm to be specified applies is defined as follows:

1) The model must be applicable to paired-associate learning as specified by the structure and associated descriptions of Figures 1 and 2.
2) Stimulus interaction must be negligible, i.e., ΔQ_{ijk} must be as specified by Equation (4)
3) ΔQ_{ijk} must be a non-negative monotonic non-decreasing function of i.

Conditions 2) and 3) imply that ΔQ for each individual component must be non-negative monotonic non-decreasing, i.e., the learning model itself must have this property.

At each stage, i, of a process described by Figure 1 or Figure 2, the process will be in a given state, Q_{ij}. Presentation of a given stimulus, s_r, will cause a transition to a new state, $Q_{i+1,k}$, at Stage i+1. If the imbedded learning model satisfies the three conditions specified above, then an algorithm which specifies an optimal presentation strategy for the process is as follows:

A: Choose for presentation at each stage, i, that stimulus which produces the largest value of ΔQ_{ijk} (for deterministic models) or the largest value of $E\{\Delta Q_{ijk}\}$ (for non-deterministic models).

This algorithm is, in effect, a more general version of the Largest Immediate Gain (LIG) algorithm of Calfee (1970) (A was arrived at independent of the work of Calfee. This fact would tend to support the validity of both algorithms).

The reasoning behind the claim of optimality for algorithm A is as follows: since there is negligible stimulus interaction, the values of ΔQ produced by each stimulus, s_r, at any given point in the process, can be treated individually for that stimulus. Since these values of

ΔQ are also non-negative and monotonic non-decreasing with i, the value of ΔQ produced by a current presentation of s_r will be at least as great as that produced by any subsequent presentation of s_r. In other words, only current values of ΔQ for all s_r need be considered at each stage of the process, since each of these values will be at least as large as subsequent values for the same stimulus. In more picturesque terms, the values of ΔQ for the process may be viewed as blocks whose size is proportional to the magnitude of the corresponding ΔQ, with the blocks arranged in n stacks, corresponding to the n different stimuli. Presentation of a given stimulus corresponds to removing a block from the top of the corresponding stack. An optimal presentation strategy for m presentations (trials) corresponds to removing the m largest blocks from the tops of the stacks. Note that this analogy makes very clear the fact that the order in which the m largest blocks are removed is immaterial, subject only to the restriction that only the top block of any given stack may be removed at each step. The optimality of the procedure of removing at each step the block which is currently largest among the top blocks is apparent from the observation that this block is the largest anywhere in any stack, since each top block is as large or larger than any other block in its stack.

The block analogy applies directly only to deterministic models in the sense that the block sizes are fixed. The reasoning is essentially the same for non-deterministic models except that expected values of ΔQ must be used (these may or may not depend on Bayesian corrections, depending on whether or not the model is response-sensitive. Deterministic models are inherently response-insensitive). Choosing at each step the stimulus for which $E\{\Delta Q\}$ is greatest produces, on the average, the largest value of $E\{T\}$.

Note that algorithm A does not require that the same model parameters, or even the same model, be used for all items in the paired-associate list. In terms of the block analogy, the algorithm obviously holds, regardless of the relative sizes of the stacks, as long as the blocks in each stack are arranged in decreasing order of size. Algorithm A is thus more general than certain other algorithms which have been proposed to apply only to cases where the same model with the same parameters is applied to each paired-associate item.

As mentioned previously, algorithm A will be applied to the linear, one-element, and RTI models of paired-associate learning. Conditions 1) and 2) of this section are obviously satisfied by each of these three models. In order to show that A holds for each model, therefore, it is necessary only to show further that Condition 3) is satisfied.

Calfee (1970) and Atkinson and Paulson (1972) have correctly observed that the optimal presentation strategy for the special case of the linear model with the same value of α and the same initial values of probability of incorrect response ($q_{01}^{(r)}$) for all items is the same as the standard cyclical strategy commonly employed in paired-associate experiments. Since the order of arrangement of stimuli in the stimulus sequence is immaterial, this strategy is equivalent to presentation of each stimulus k_1 times, where k_1 is the largest integer less than or equal to m/n, followed by presentation of any k_2 different stimuli, where

$$k_2 = m - k_1 n.$$

Algorithm *A*, however, is more general than this algorithm in that the value of α and the initial value of q associated with each stimulus are arbitrary (within the unit interval). For equal values of α and q, of course, *A* produces the standard cyclical strategy, in effect.

The fact that Condition 3) is satisfied for the linear model can be seen from the fact that q_i satisfies Condition 3), since

$$q_{i+1} = \alpha q_i$$

and $0 \leq \alpha \leq 1$. Hence,

$$\begin{aligned}\Delta Q &= q_i - q_{i+1} \\ &= q_i - \alpha q_i \\ &= q_i(1-\alpha)\end{aligned}$$

also satisfies Condition 3) since q_i satisfies Condition 3) and $0 \leq (1-\alpha) \leq 1$. Therefore, *A* holds for the linear model.

In order to determine the effectiveness of *A* under conditions other than uniform parameters, Monte Carlo simulations were conducted for two different cases. In the first case, all initial values of q were set equal to the complement of the guessing probabilities, but the values of α for the individual items were chosen randomly according to a truncated Gaussian distribution. Mean values were chosen at 0.4, 0.6, 0.7, 0.8, 0.85, 0.90, 0.95 and 0.97 to permit coordination with Calfee's results (1970). The standard deviation was specified to be the difference between the mean value and the closest boundary of the unit interval, and the distribution was truncated at this boundary and at the symmetric point on the other side of the mean value. The optimal strategy specified by *A* for this case will not, in general, be the same as the standard cyclical strategy, so a simulation was conducted to compare the effectiveness of the two (details of the simulation are contained in the Appendix), using a simulated paired-associate list of 10 items.

The results of the simulation are shown in Figures 7-1 through 7-8. As can be seen from these figures, the maximum advantage

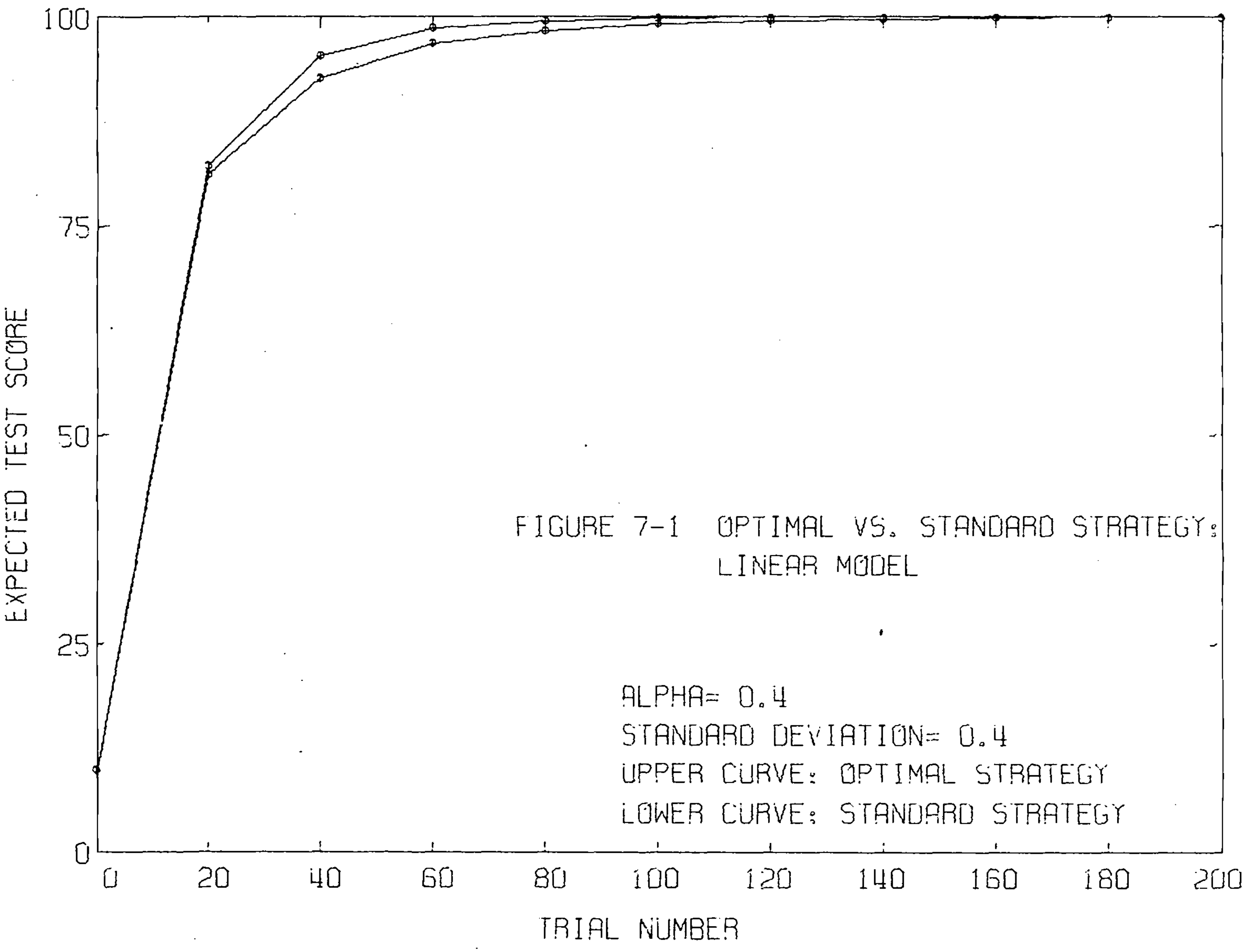


FIGURE 7-1 OPTIMAL VS. STANDARD STRATEGY: LINEAR MODEL

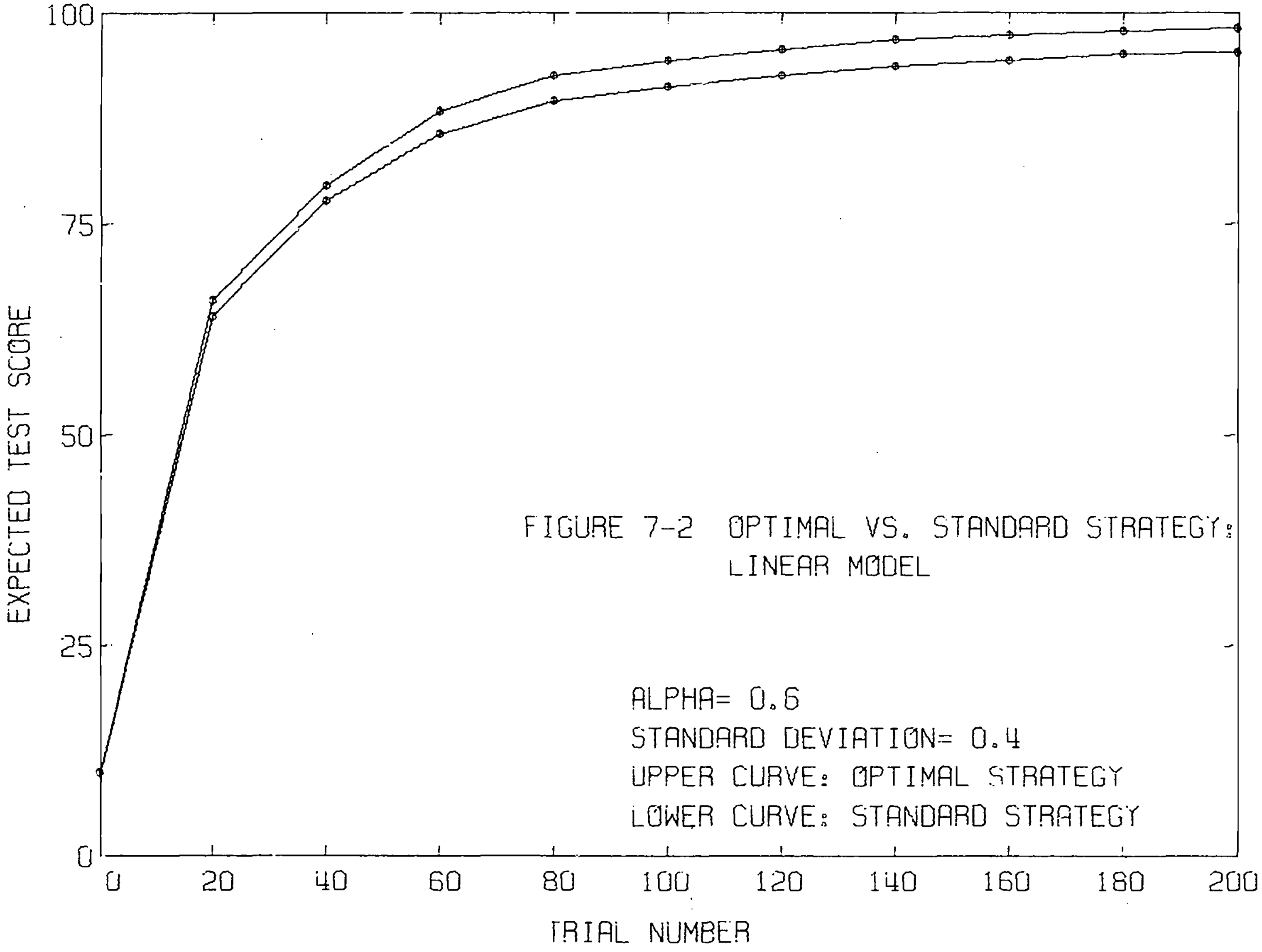


FIGURE 7-2 ØPTIMAL VS. STANDARD STRATEGY:
LINEAR MØDEL

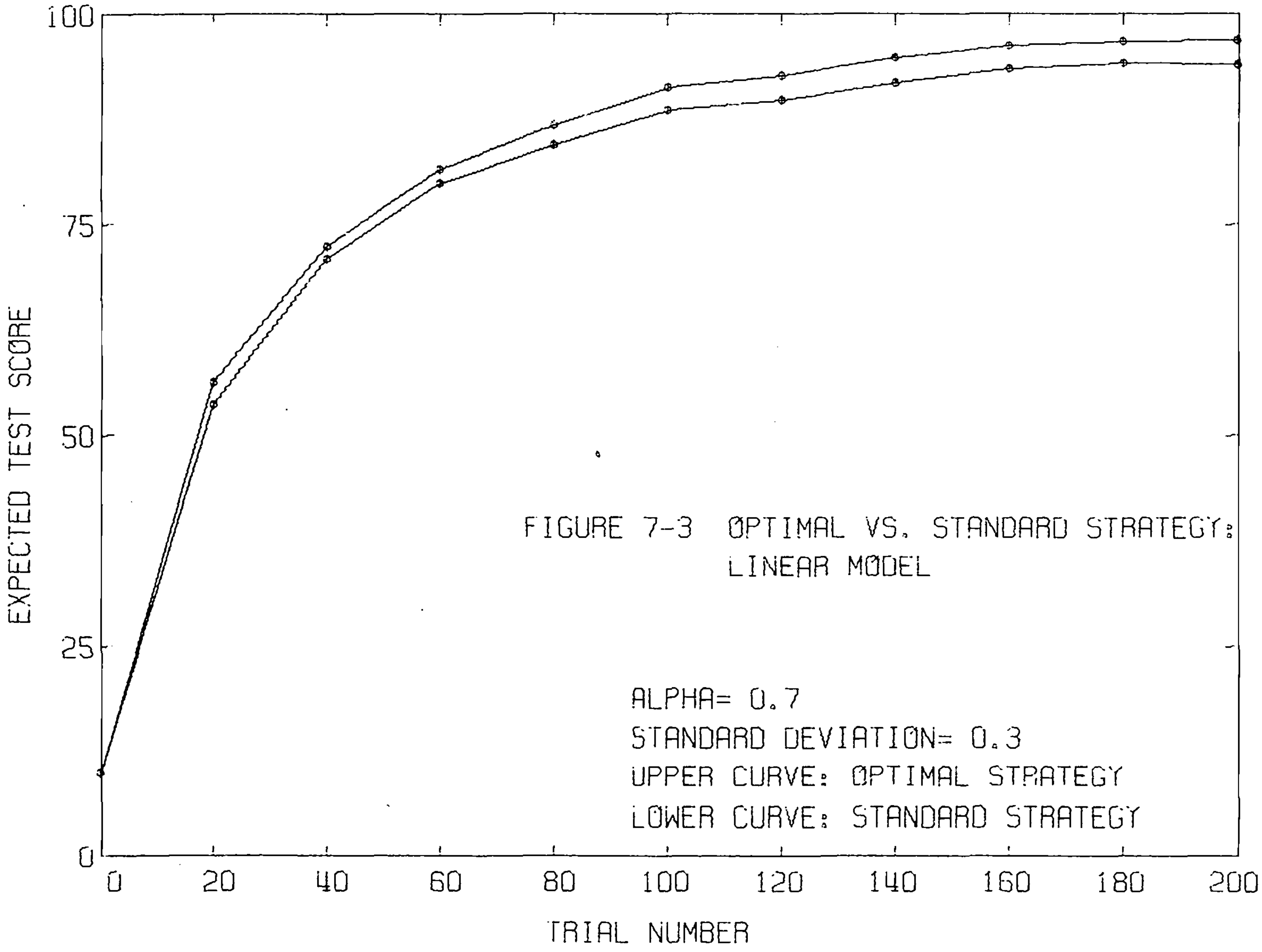


FIGURE 7-3 ØPTIMAL VS. STANDARD STRATEGY:
LINEAR MØDEL

ALPHA= 0.7
STANDARD DEVIATIØN= 0.3
UPPER CURVE: ØPTIMAL STRATEGY
LØWER CURVE: STANDARD STRATEGY

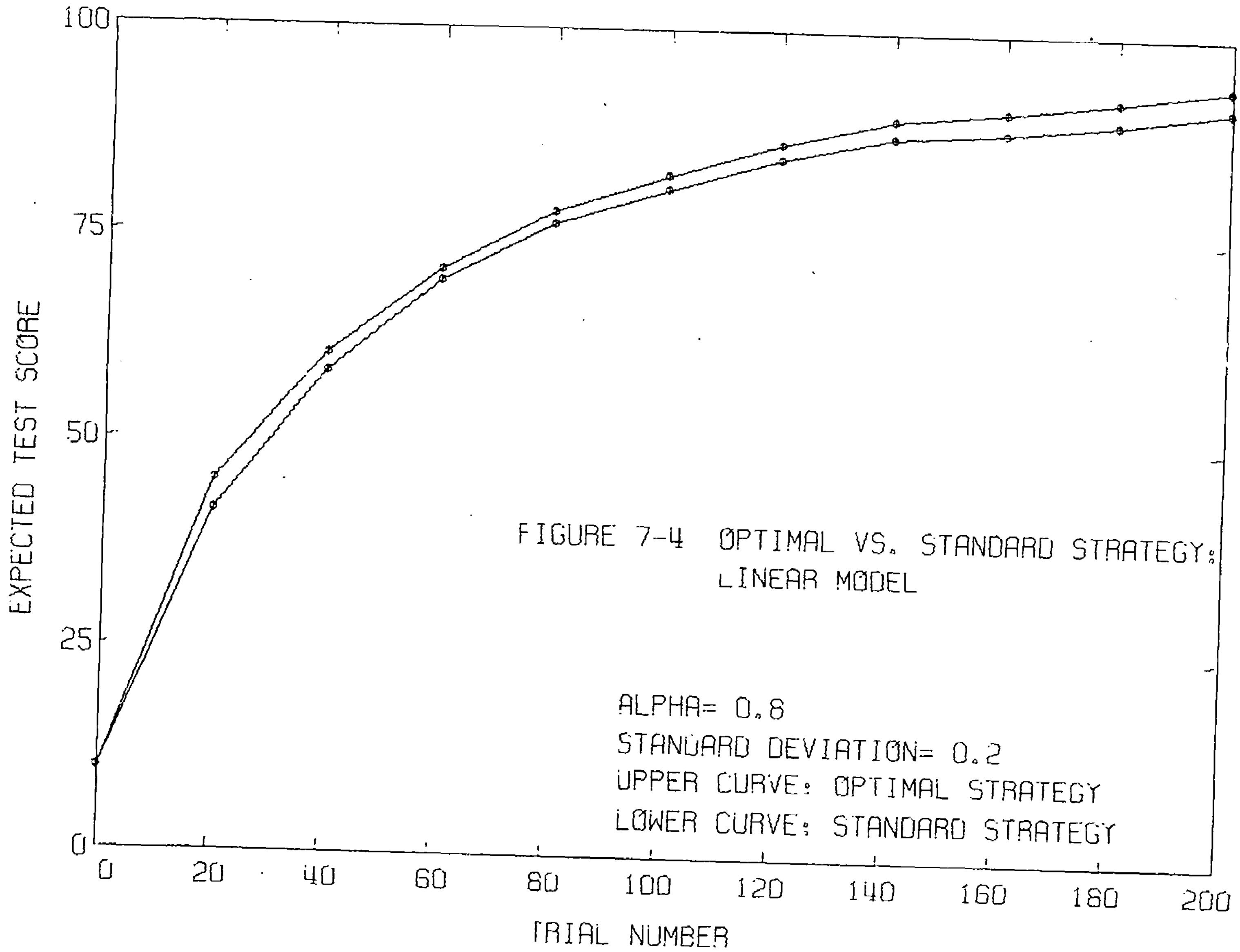


FIGURE 7-4 ØPTIMAL VS. STANDARD STRATEGY: LINEAR MODEL

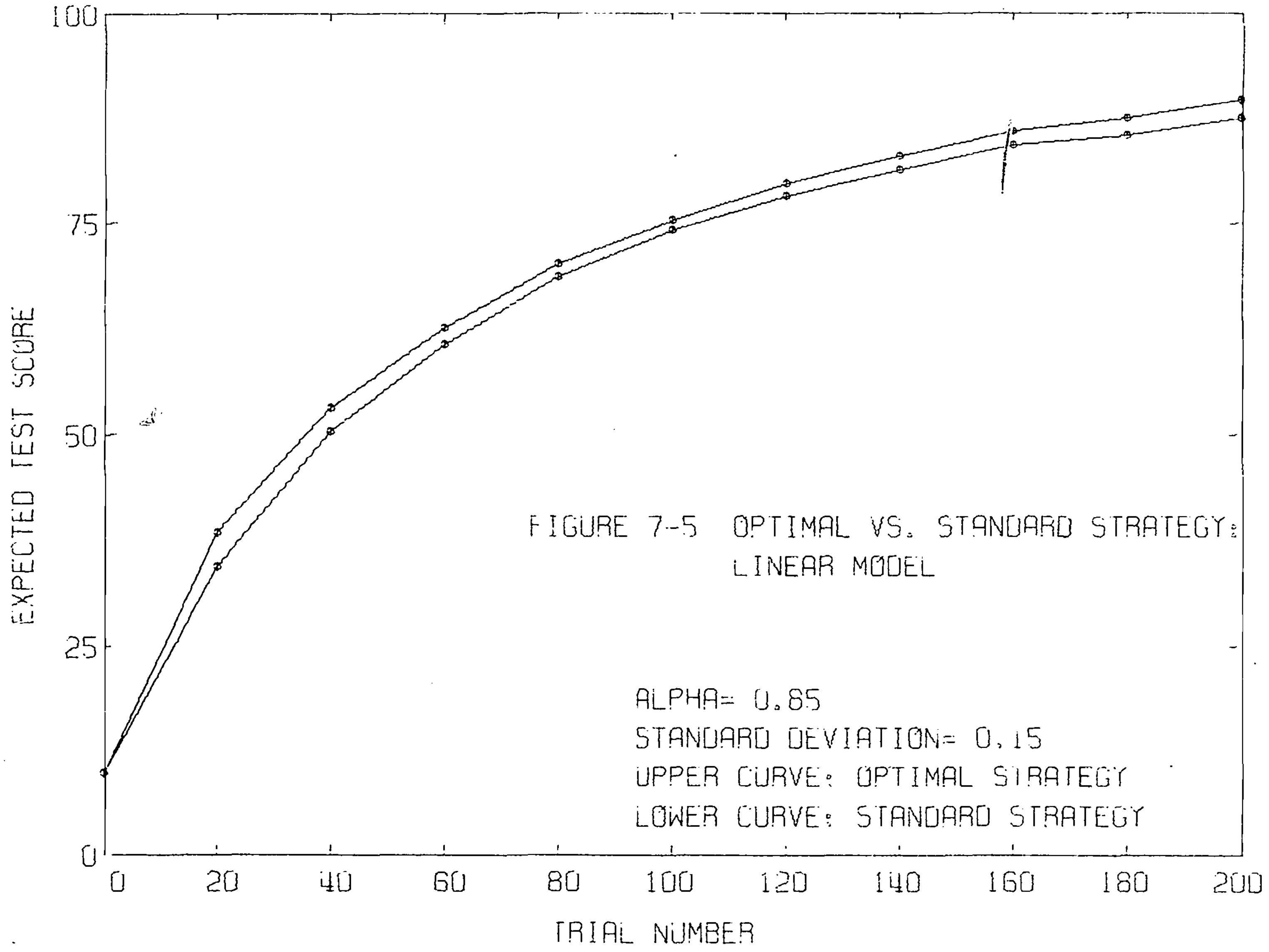


FIGURE 7-5 OPTIMAL VS. STANDARD STRATEGY:
LINEAR MODEL

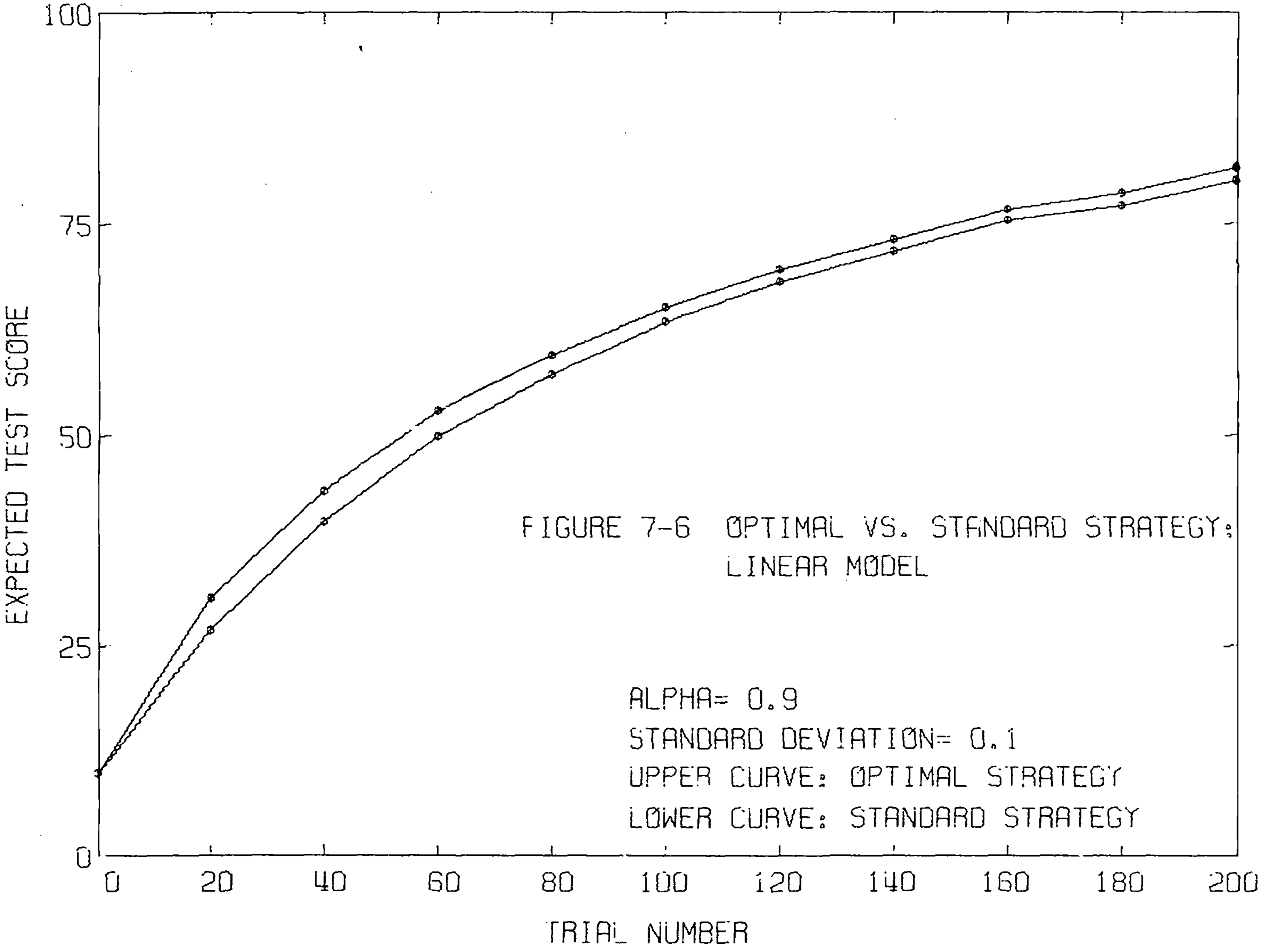


FIGURE 7-6 ØPTIMAL VS. STANDARD STRATEGY:
LINEAR MØDEL

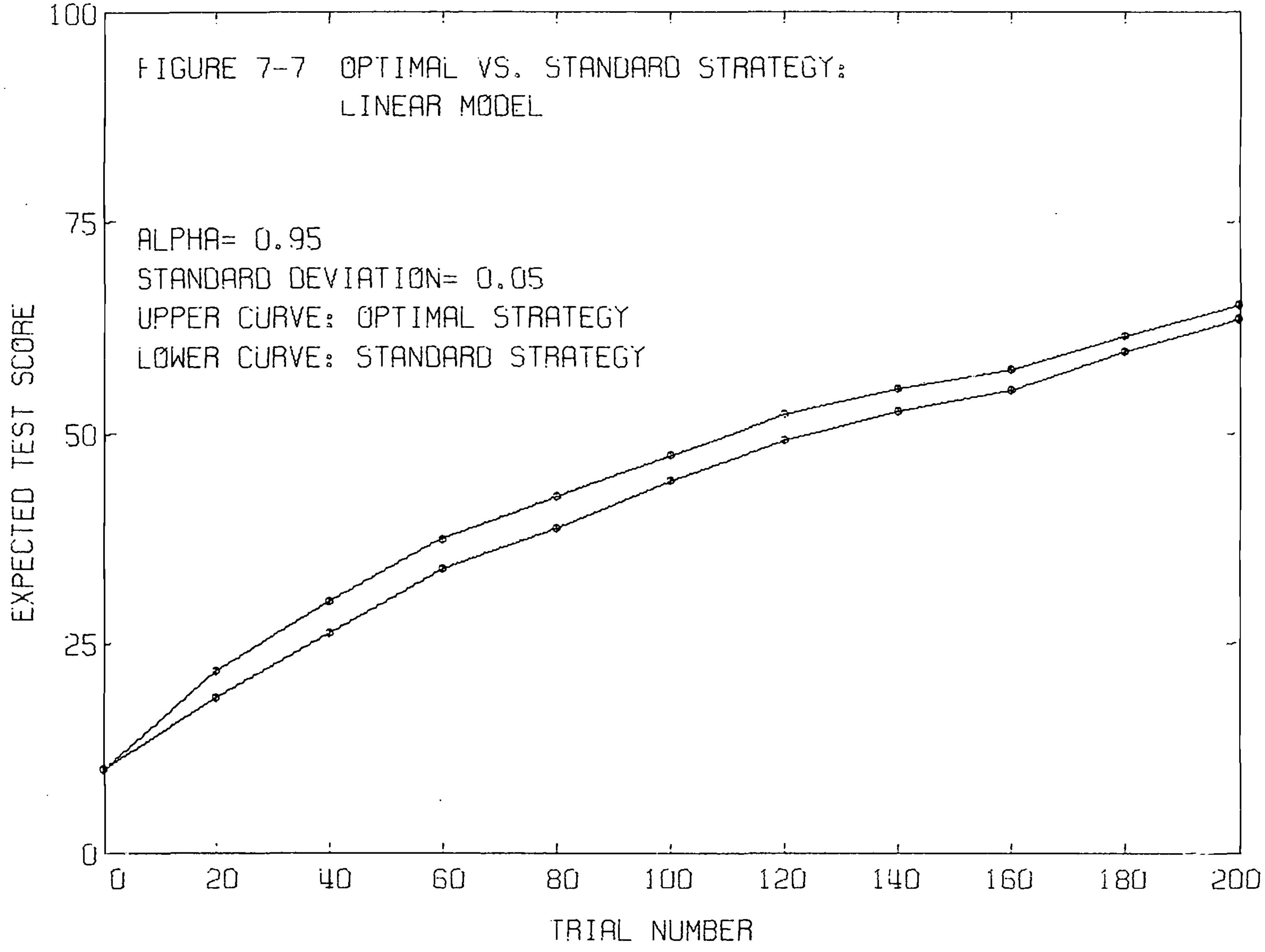


FIGURE 7-7 OPTIMAL VS. STANDARD STRATEGY:
LINEAR MØDEL

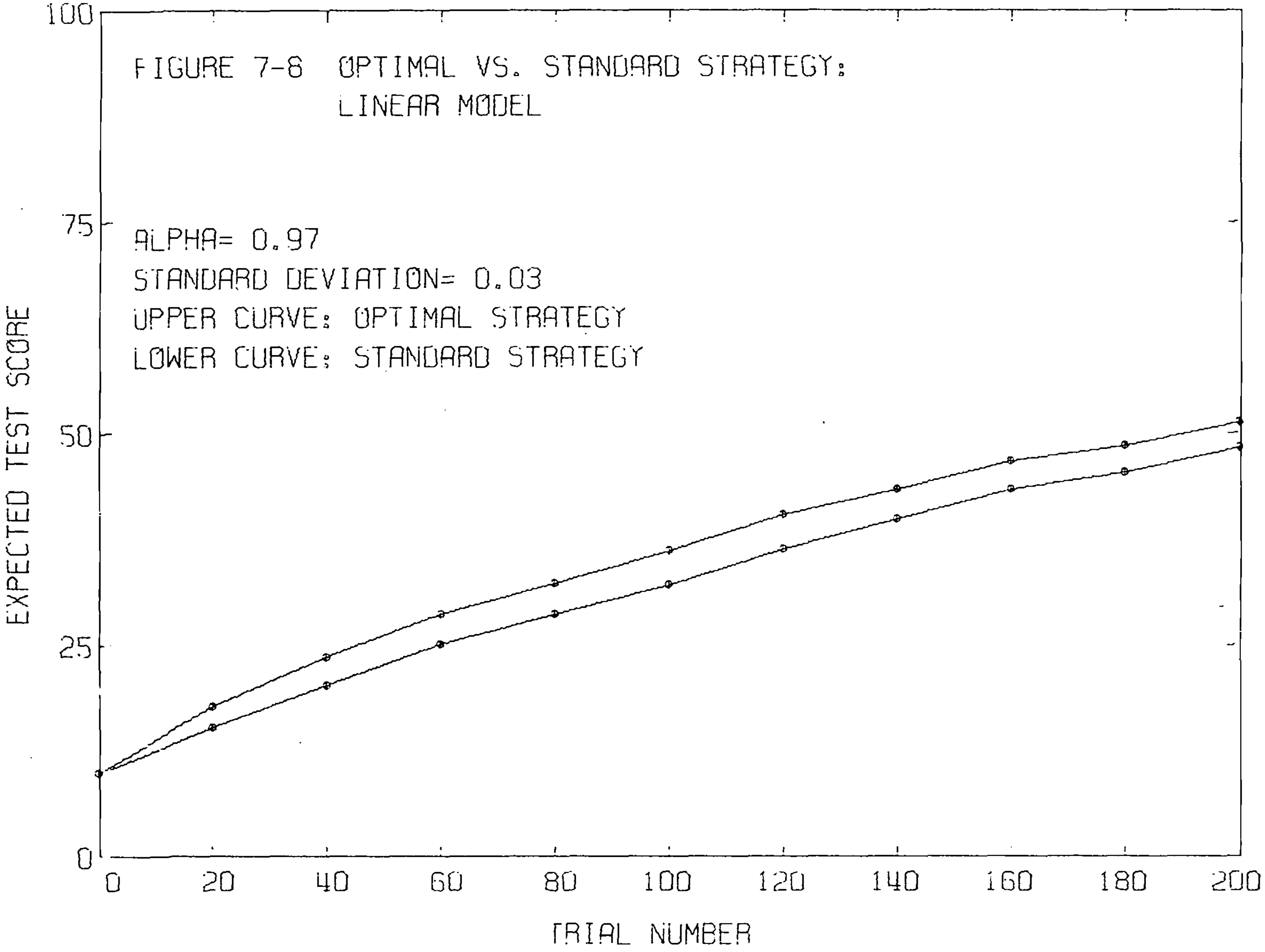


FIGURE 7-8 OPTIMAL VS. STANDARD STRATEGY: LINEAR MODEL

provided by A under any conditions is on the order of 5%. The curves shown represent average values for a sample size of 200. In other words, the simulated process was run 200 times, each time with a different set of α values chosen as described above (Note: the fact that a value of 200 trials was chosen as the terminating value for the simulation is coincidental; the number of trials and the sample size are not necessarily related).

The second case involving the linear model was a comparison of the strategy specified by A for uniform parameters (the cyclical strategy) versus a uniform random strategy. Note that these two strategies are asymptotically equivalent for large values of m (in a sense, all strategies are equivalent for large m, since E{T} asymptotically approaches 100% for any strategy when the learning model satisfies the three conditions for A), but for finite values of m, the random strategy, on the average, will produce a lower value of E{T}. The results of this simulation are shown in Figures 7-9 through 7-16. Again, the maximum advantage produced by A is on the order of 5%. Note that, for most values of α, the values of E{T} for the two strategies appear to converge faster than do the corresponding values for the two strategies in the first case. This is apparently due to the fact that the effects of values of α distributed about some mean value are not, in general, symmetric. It might be noted that Figures 7-1 through 7-16 apply, as well, to the RTI model used in response-insensitive mode under the same conditions, with the only difference being that the ordinates represent average values of E{T}, since this model is nondeterministic. An equivalence between the parameters of the two models results from the fact that, for the RTI model in response-insensitive mode:

$$E\{q_{i+1}\} = q_i(1-c) + \alpha_r q_i c$$
$$= q_i(1-c+\alpha_r c)$$

Hence, the same values of $E\{q_{i+1}\}$ are obtained for the RTI model with parameters α_r and c, as values of q_{i+1} are obtained for the linear model with

$$\alpha = 1 - c + \alpha_r c.$$

Karush and Dear (1966) obtained an optimal presentation strategy for the special case of the one-element model with uniform values of θ and γ for all items, as well as equal initial values of q. The strategy is to choose at each stage that item for which the current probability of being in the learned state is least. This algorithm

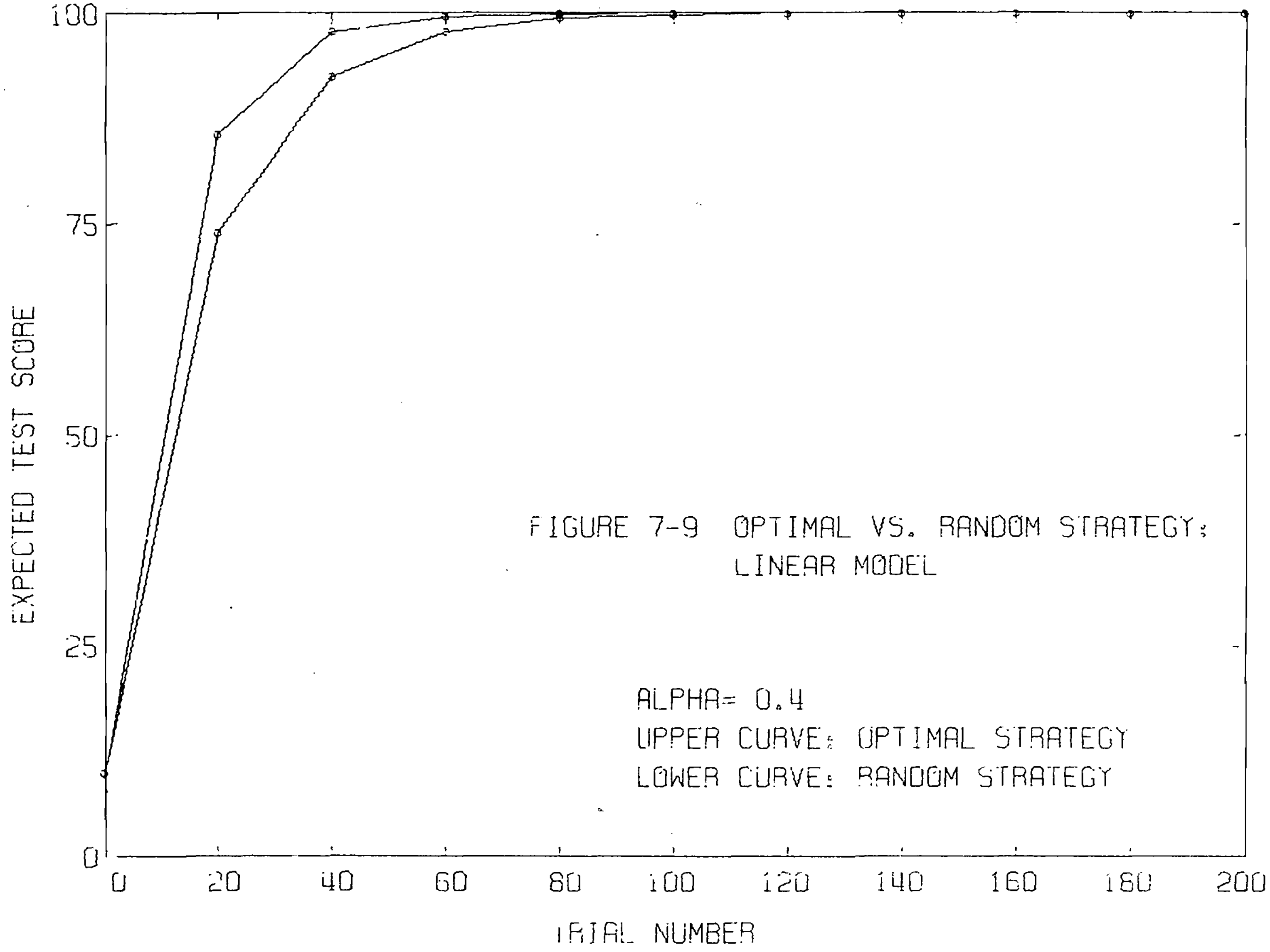


FIGURE 7-9 OPTIMAL VS. RANDOM STRATEGY;
LINEAR MODEL

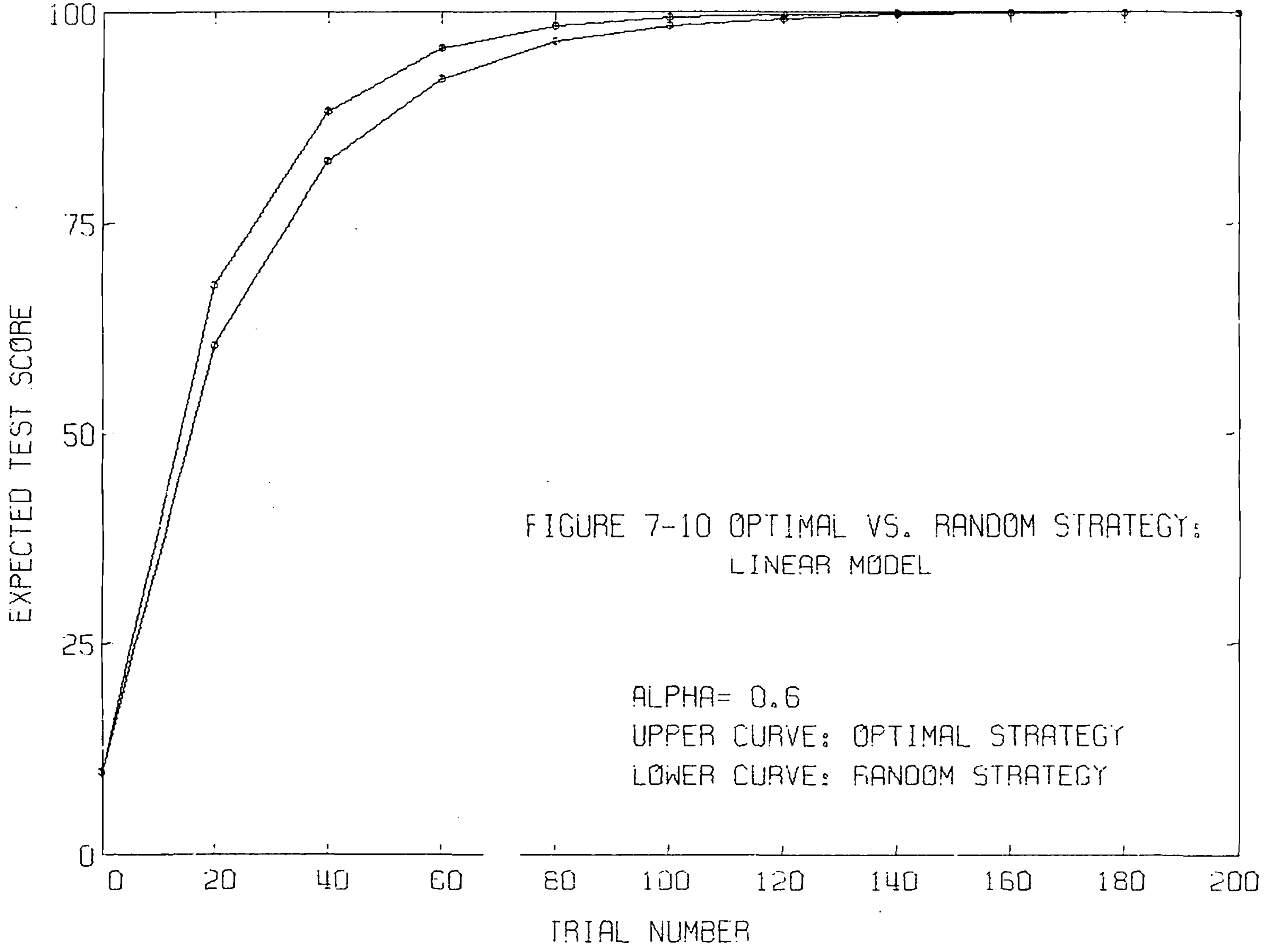


FIGURE 7-10 OPTIMAL VS. RANDOM STRATEGY:
LINEAR MØDEL

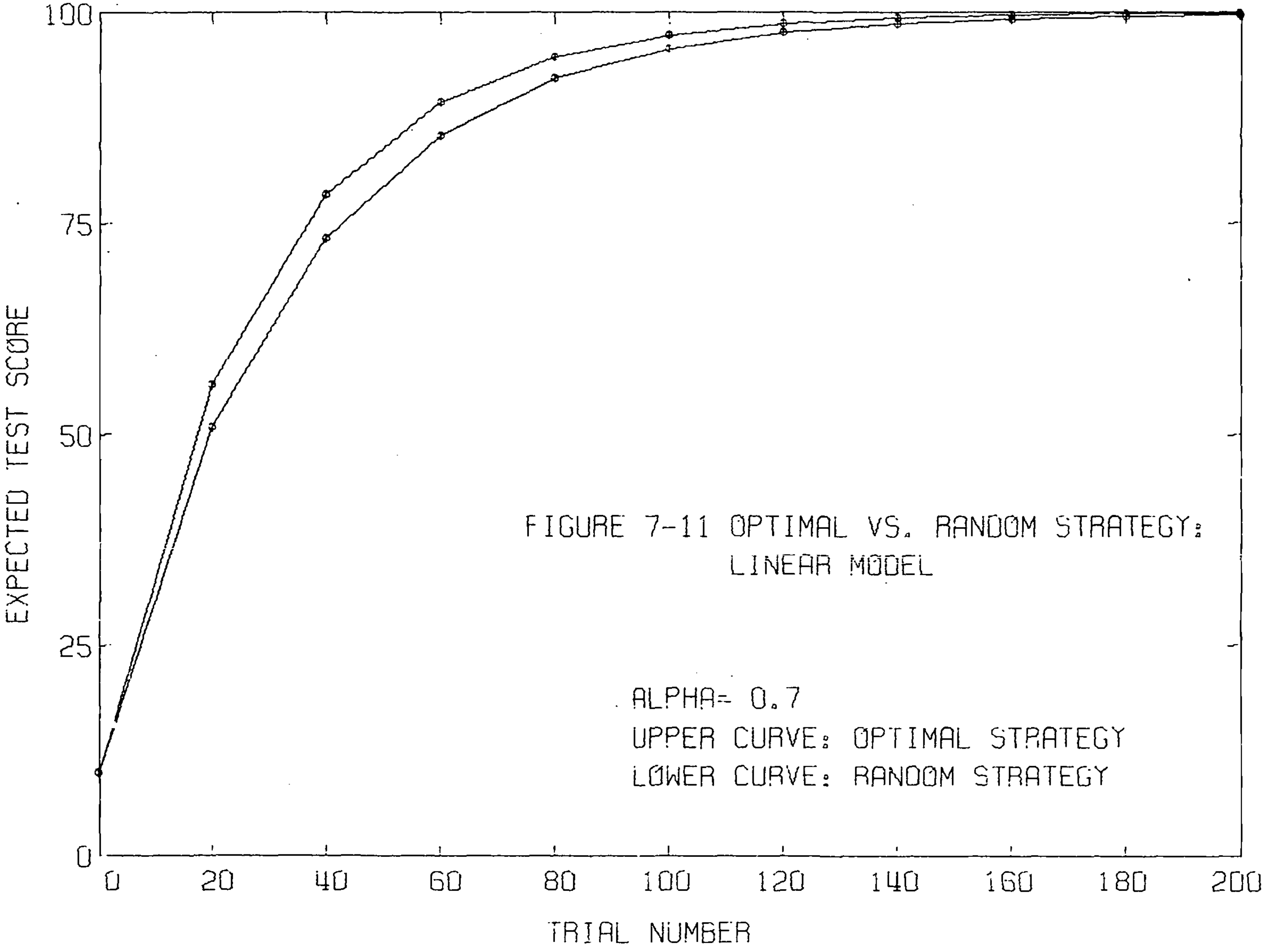


FIGURE 7-11 OPTIMAL VS. RANDOM STRATEGY: LINEAR MODEL

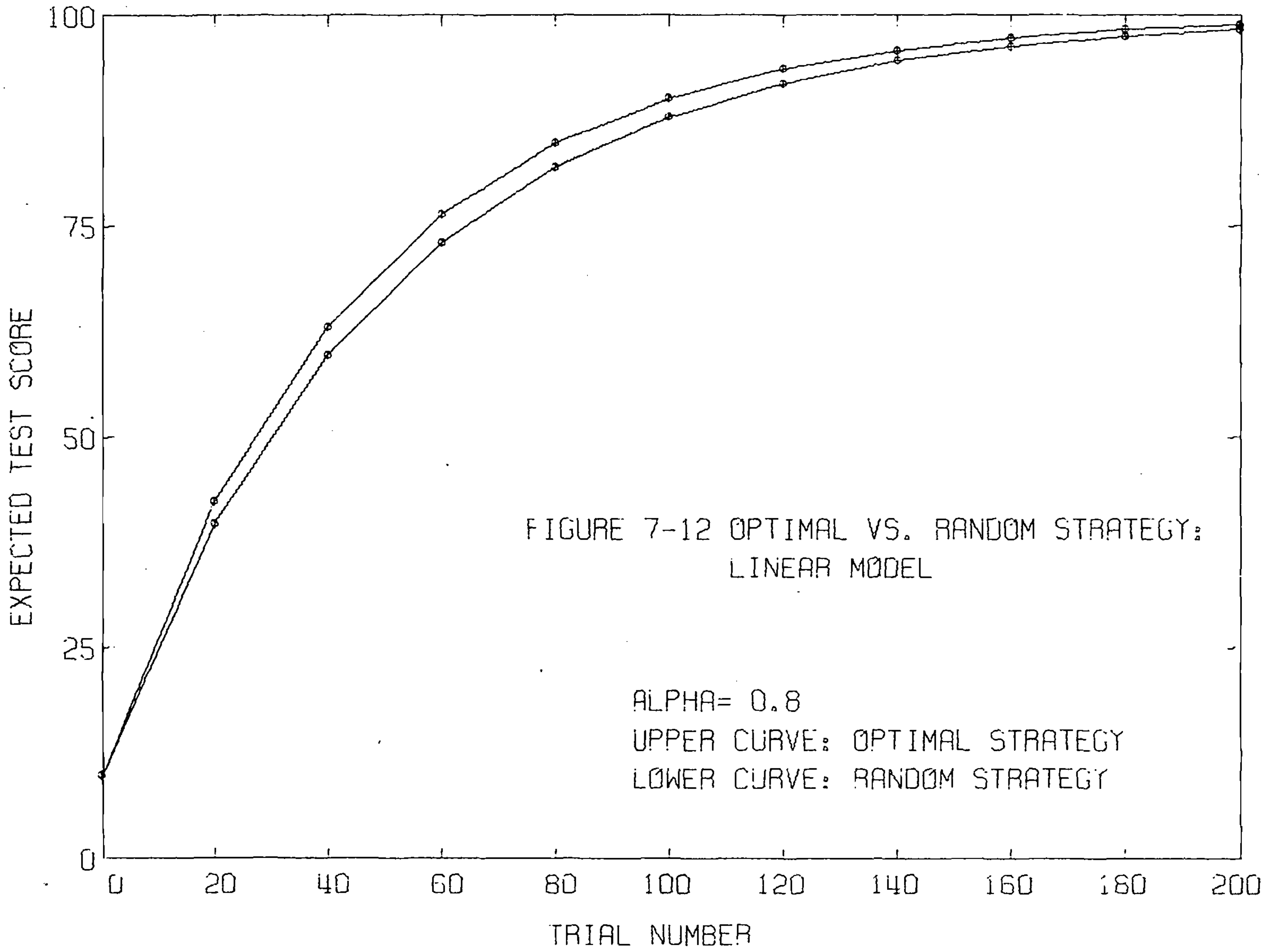


FIGURE 7-12 OPTIMAL VS. RANDOM STRATEGY:
LINEAR MØDEL

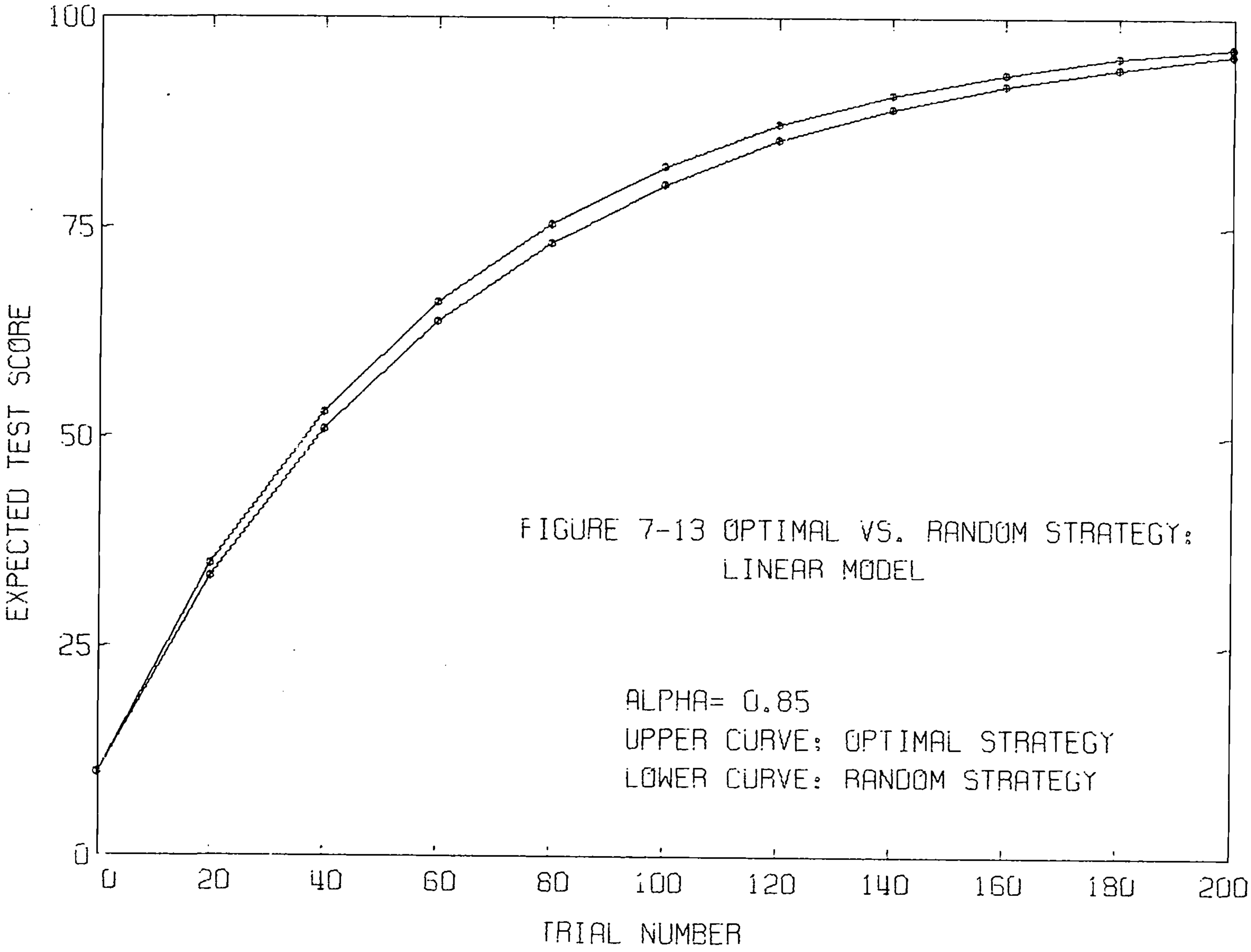


FIGURE 7-13 OPTIMAL VS. RANDOM STRATEGY:
LINEAR MODEL

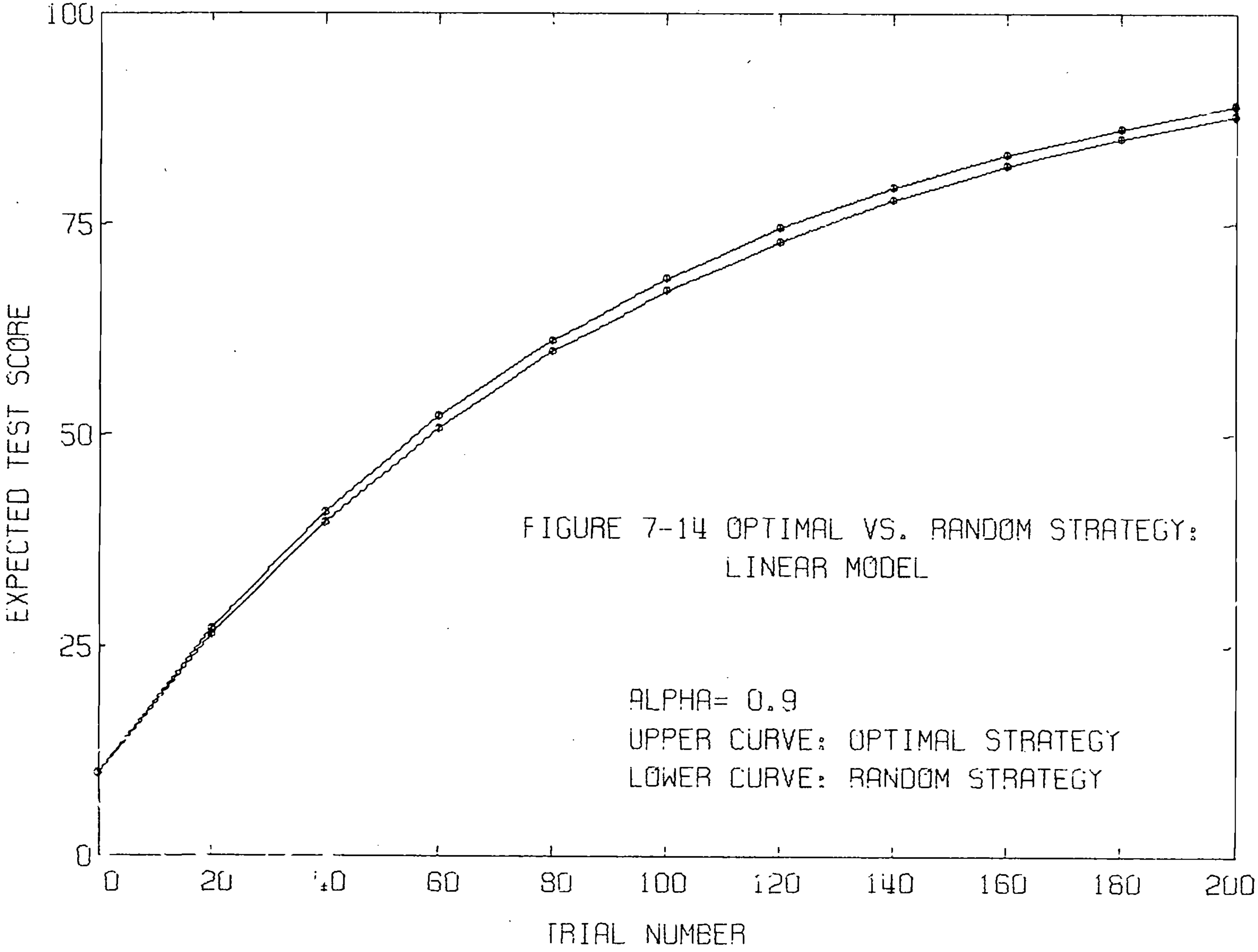


FIGURE 7-14 OPTIMAL VS. RANDOM STRATEGY: LINEAR MODEL

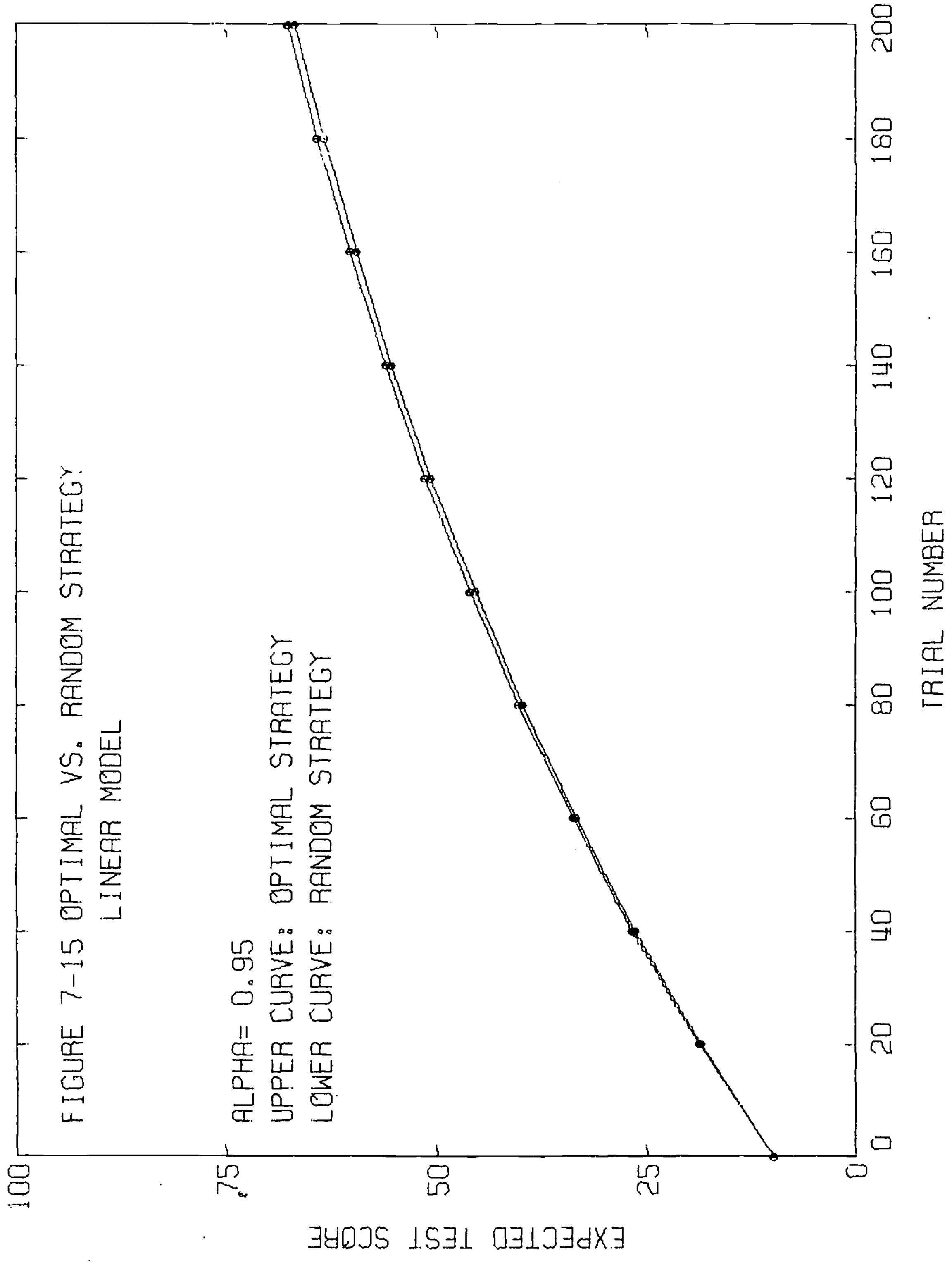


FIGURE 7-15 ØPTIMAL VS. RANDØM STRATEGY
LINEAR MØDEL

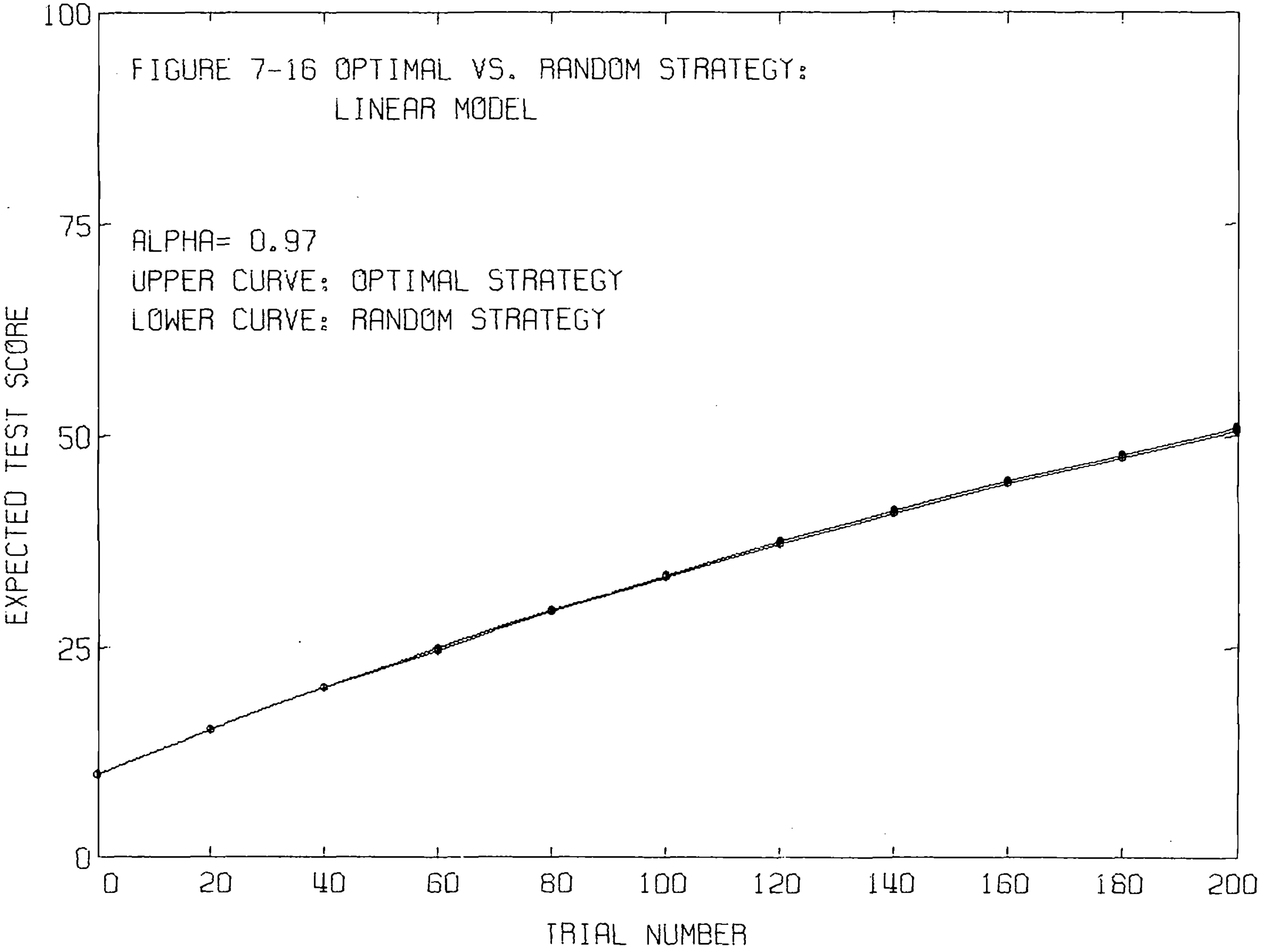


FIGURE 7-16 OPTIMAL VS. RANDOM STRATEGY:
LINEAR MODEL

was later abstracted (cf. Atkinson, 1968; Calfee, 1970; Atkinson and Paulson, 1972) to a form based on counts of responses, where the item chosen for presentation is the one with the fewest consecutive correct responses since the last incorrect response. This algorithm is a special case of algorithm *A*, since choosing the item for which ΔQ is largest under conditions of uniform parameters is equivalent to choosing the item with the fewest consecutive correct responses. This can be shown as follows. Karush and Dear (1966) have shown that λ_i, the probability of being conditioned to a given item at the outset of trial i, is a monotonic non-decreasing function of i, given correct responses at each trial. Following any incorrect response, λ_i is reset to θ, the probability of transition to the conditioned state, given that the model was in the unconditioned state prior to reinforcement. Following the notation of Karush and Dear (E for correct response, $\bar{E}$ for incorrect response, C denoting the conditioned state, $\bar{C}$ denoting the unconditioned state, and $p^*=1-p$ for any probability, p), then

$$\begin{aligned} E\{\lambda_{i+1}\} &= P\{E\}P\{C_{i+1}|E\} + P\{\bar{E}\}P\{C_{i+1}|\bar{E}\} \\ &= (\lambda_i+\gamma\lambda_i^*)\{(\lambda_i+\gamma\theta\lambda_i^*)/(\lambda_i+\gamma\lambda_i^*)\} + \gamma^*\lambda_i^*\theta \\ &= \lambda_i + \lambda_i^*\theta \end{aligned}$$

and

$$\begin{aligned} E\{\lambda_{i+1}^*\} &= 1 - \lambda_i - \lambda_i^*\theta \\ &= \lambda_i^*\theta^* \end{aligned}$$

Hence,

$$\begin{aligned} \Delta Q &= q_i - q_{i+1} \\ &= \gamma^*(\lambda_i^* - \lambda_{i+1}^*) \end{aligned}$$

and

$$\begin{aligned} E\{\Delta Q\} &= \gamma^*(\lambda_i^* - E\{\lambda_{i+1}^*\}) \\ &= \gamma^*(\lambda_i^* - \lambda_i^*\theta^*) \\ &= \gamma^*\lambda_i^*\theta \qquad (5) \end{aligned}$$

which is monotonic non-increasing, since λ_i is monotonic non-decreasing. Therefore, choosing at each step the item for which $E\{\Delta Q\}$ is largest is equivalent to choosing the item, under conditions of uniform model parameters, for which the number of consecutive correct responses is least. Equation (5) also shows that the one-element model satisfies Condition 3) for $\underline{A}$, quaranteeing that $\underline{A}$ holds for this model with arbitrary parameters.

Monte Carlo simulations were conducted for the one-element model for three cases. In all three cases, separate simulations were run for values of θ (or θ-mean) equal to 0.025, 0.05, 0.1, and 0.3, again, to allow for coordination with Calfee's data (1970). In the first two cases, the strategy specified by $\underline{A}$ was compared with the standard cyclical strategy for fixed uniform values of θ, and for randomly determined values of θ (by the method specified for the linear model), respectively. The results of these simulations are shown in Figures 8-1, 8-3, 8-5, and 8-7 for the uniform values, and in Figures 8-2, 8-4, 8-6, and 8-8 for the random values. The curves show, first of all, that greater advantages are predicted for the optimal strategy by the one-element model than by the linear model. Secondly, the simulations show that greater gains can be expected, on the average, for the one-element model with non-uniform parameters than for the one-element model with uniform parameters.

The third case involving the one-element model was a comparison of the optimal strategy given by $\underline{A}$ with a uniform random strategy. Again, the results show greater predicted gains using the one-element model than using the linear model. This is not surprising, certainly, since it would be expected that an optimal response-sensitive strategy should be able to use the additional information supplied by the responses to advantage.

Dear, et al. (1967), have reported an experiment designed to test the performance of an optimal presentation strategy for the one-element model with uniform parameters (Karush & Dear, 1966). The strategy to be tested was the special case of $\underline{A}$ just discussed. The experiment was designed to compare this strategy with the standard cyclical strategy for effectiveness. The primary result of the experiment was that no statistically significant difference between the two strategies was detectable in terms of post-test scores. As there was no evidence that a full simulation of the instructional process had been conducted to determine theoretically predicted differences between the two strategies, and since such a simulation involved only minor modifications to the programs used in the one-element simulations represented by Figures 8-1 through 8-12, it was decided to include this simulation in this investigation. The program used to conduct the simulation is listed in the Appendix, and serves to illustrate the techniques used in the one-element simulations as well.

Figure 9-1 illustrates the expected sum-of-correct-responses scores versus number of trials for the two-choice response experiment, using the value of θ assumed by Dear, et al., while Figure 10-1

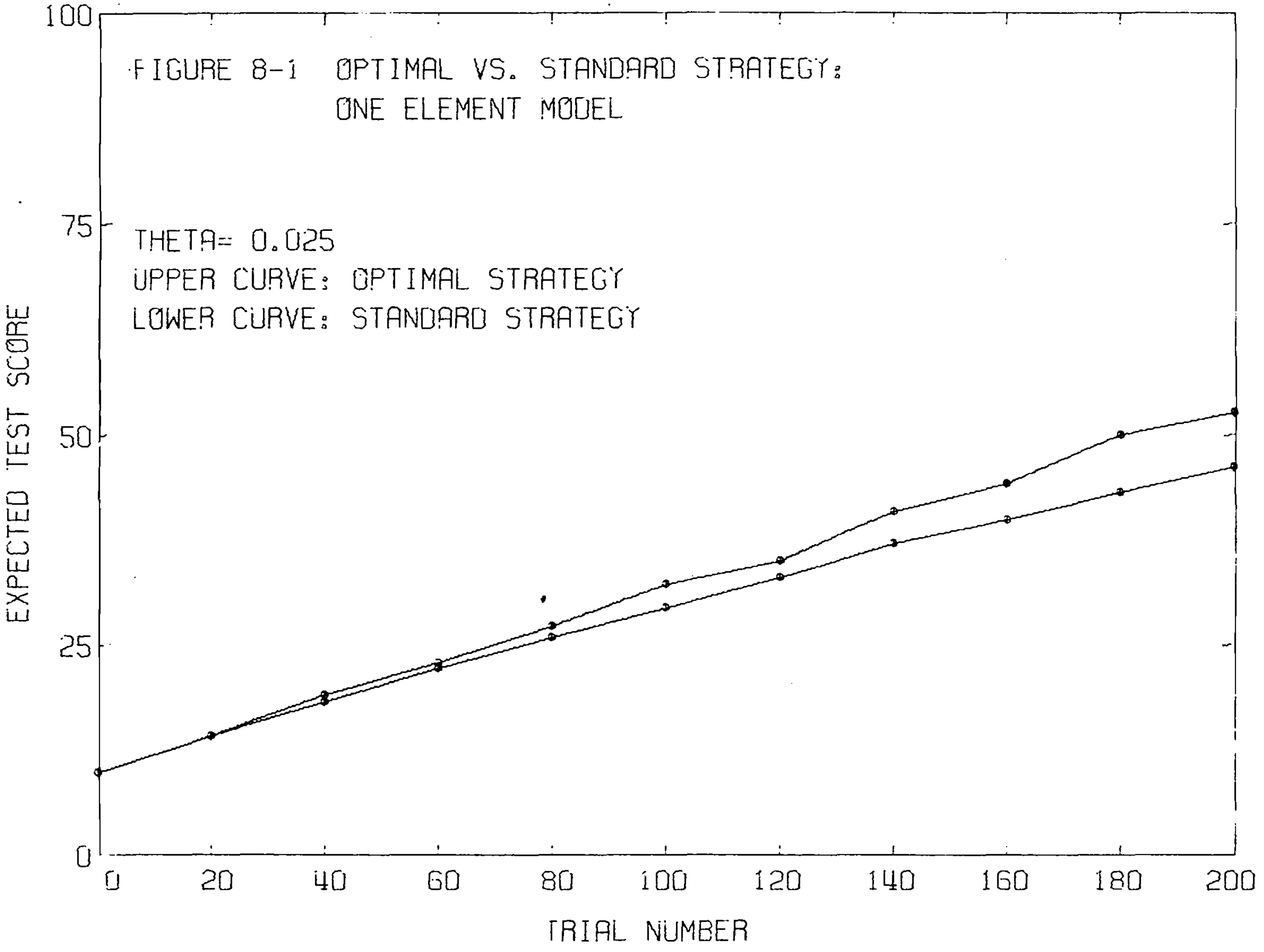


FIGURE 8-1 OPTIMAL VS. STANDARD STRATEGY: ONE ELEMENT MODEL

THETA= 0.025
UPPER CURVE: OPTIMAL STRATEGY
LOWER CURVE: STANDARD STRATEGY

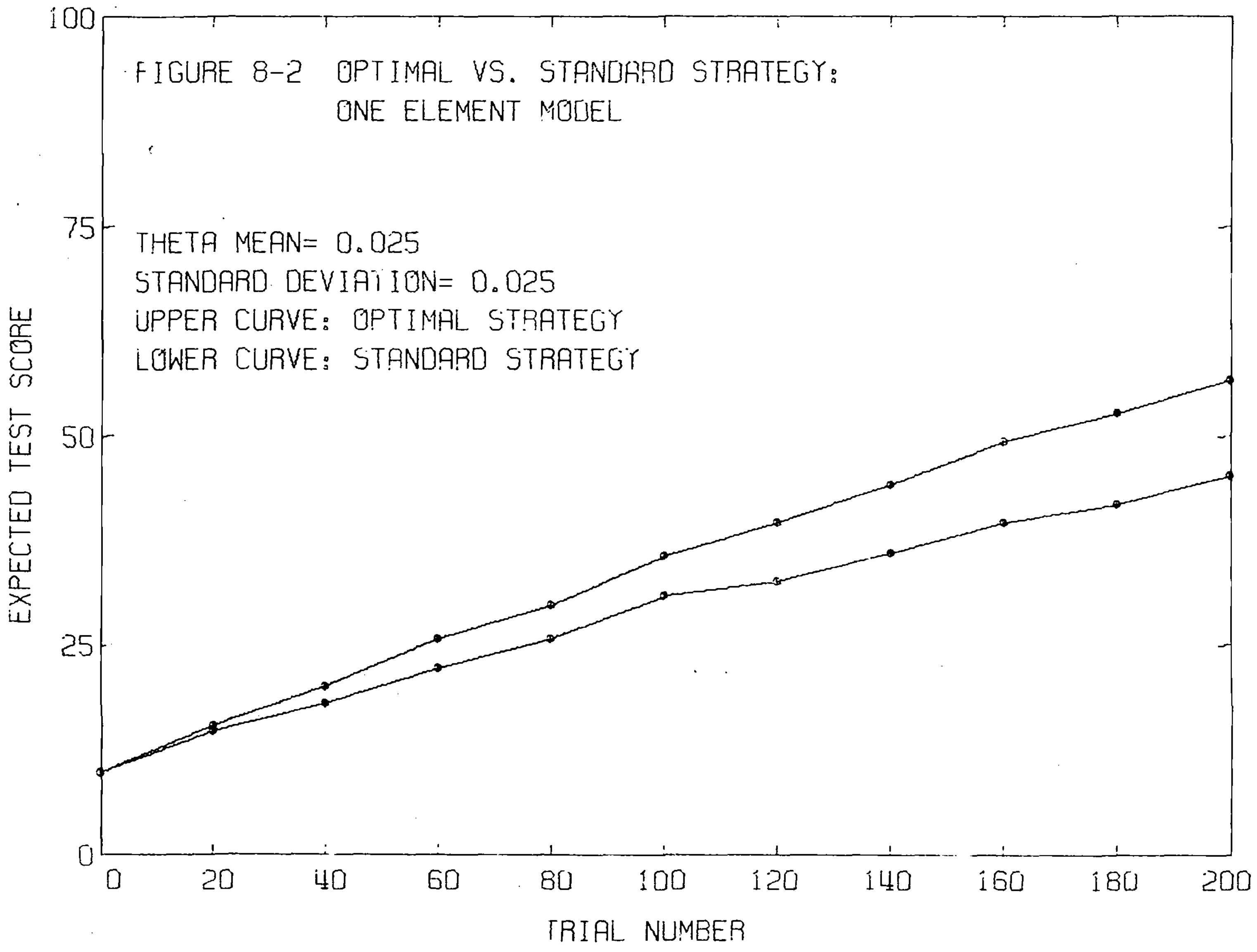


FIGURE 8-2 OPTIMAL VS. STANDARD STRATEGY:
ONE ELEMENT MODEL

THETA MEAN= 0.025
STANDARD DEVIATION= 0.025
UPPER CURVE: OPTIMAL STRATEGY
LOWER CURVE: STANDARD STRATEGY

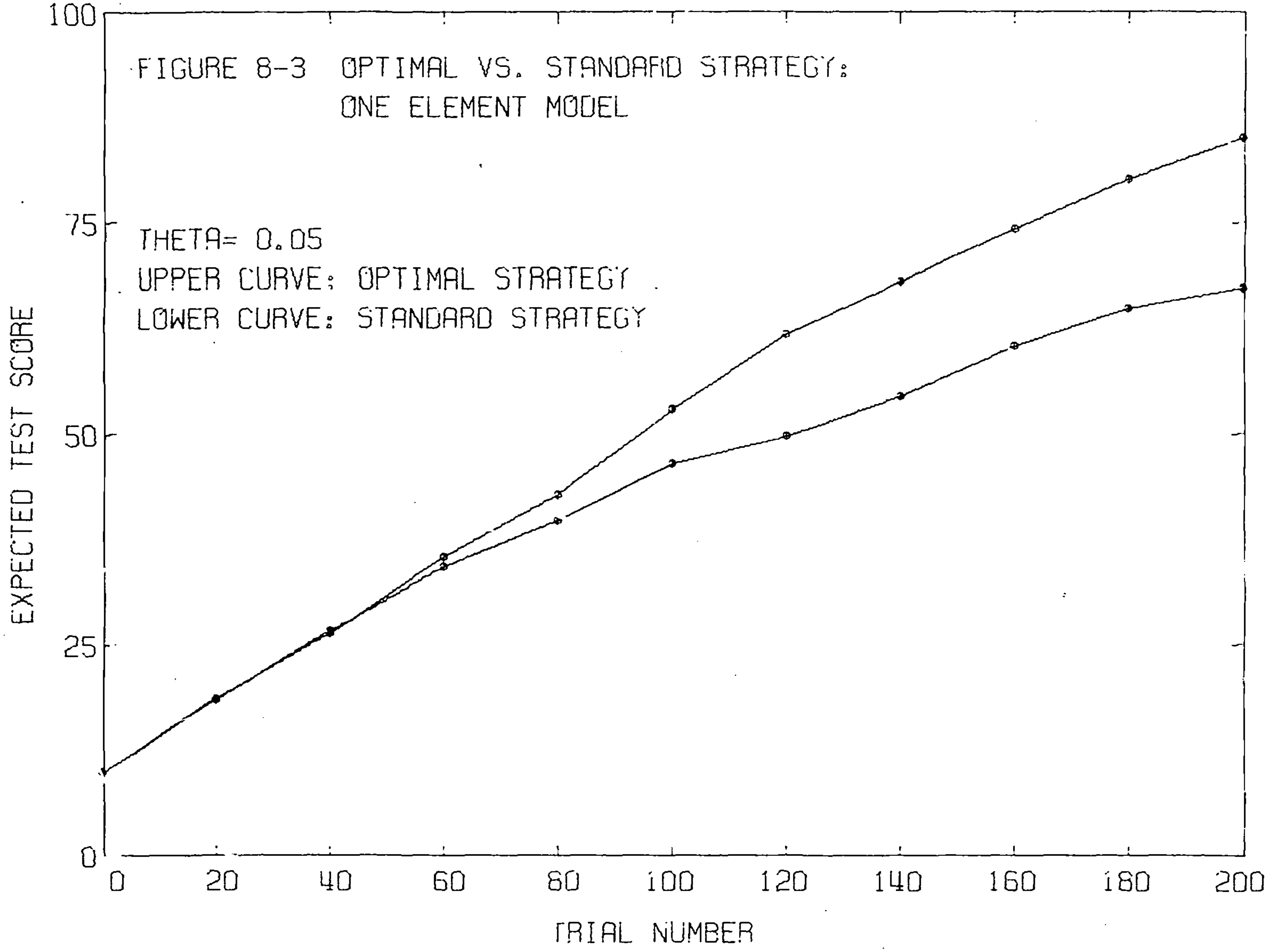


FIGURE 8-3 OPTIMAL VS. STANDARD STRATEGY:
ONE ELEMENT MODEL

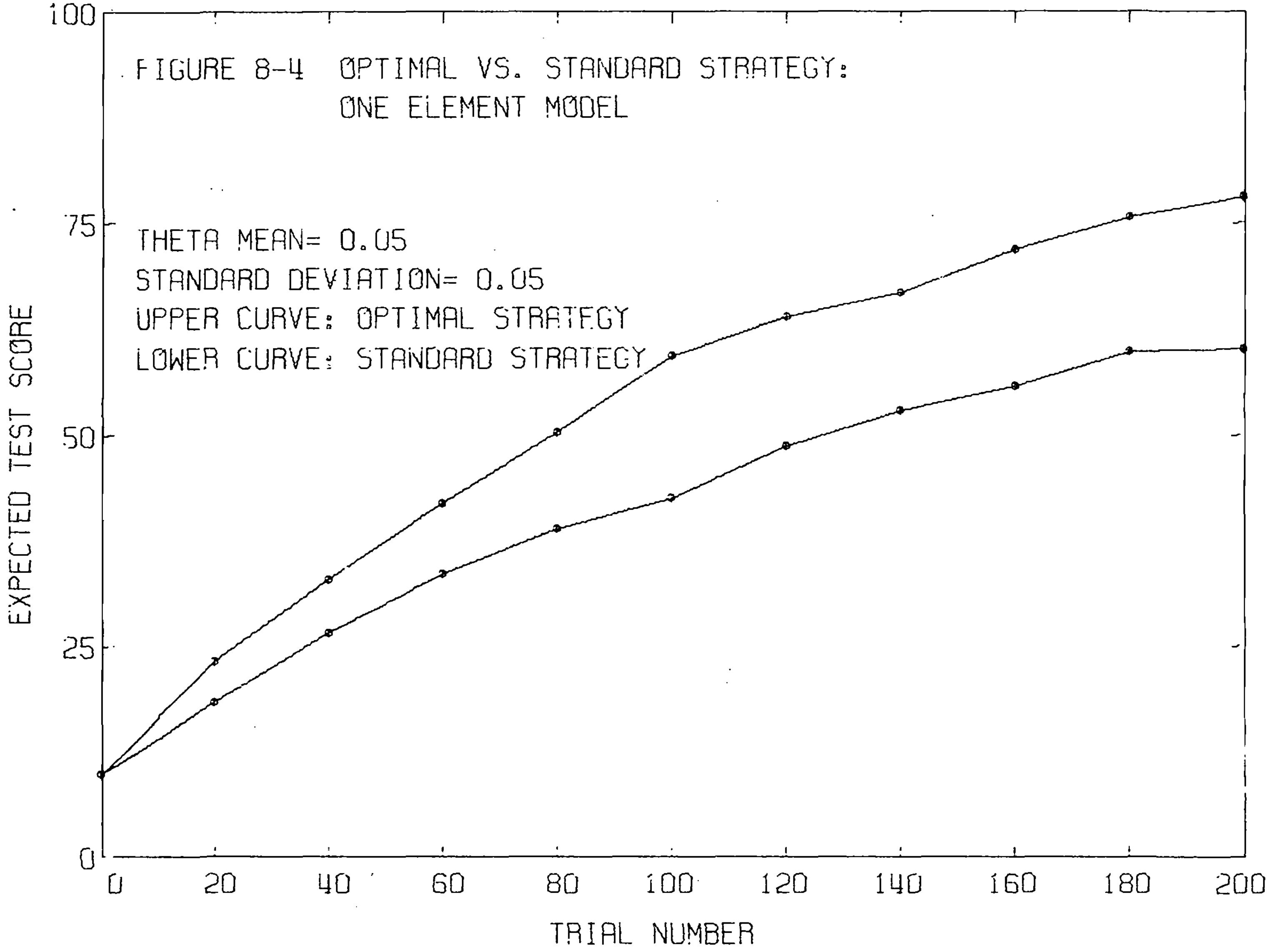


FIGURE 8-4 OPTIMAL VS. STANDARD STRATEGY:
ONE ELEMENT MODEL

THETA MEAN= 0.05
STANDARD DEVIATION= 0.05
UPPER CURVE: OPTIMAL STRATEGY
LOWER CURVE: STANDARD STRATEGY

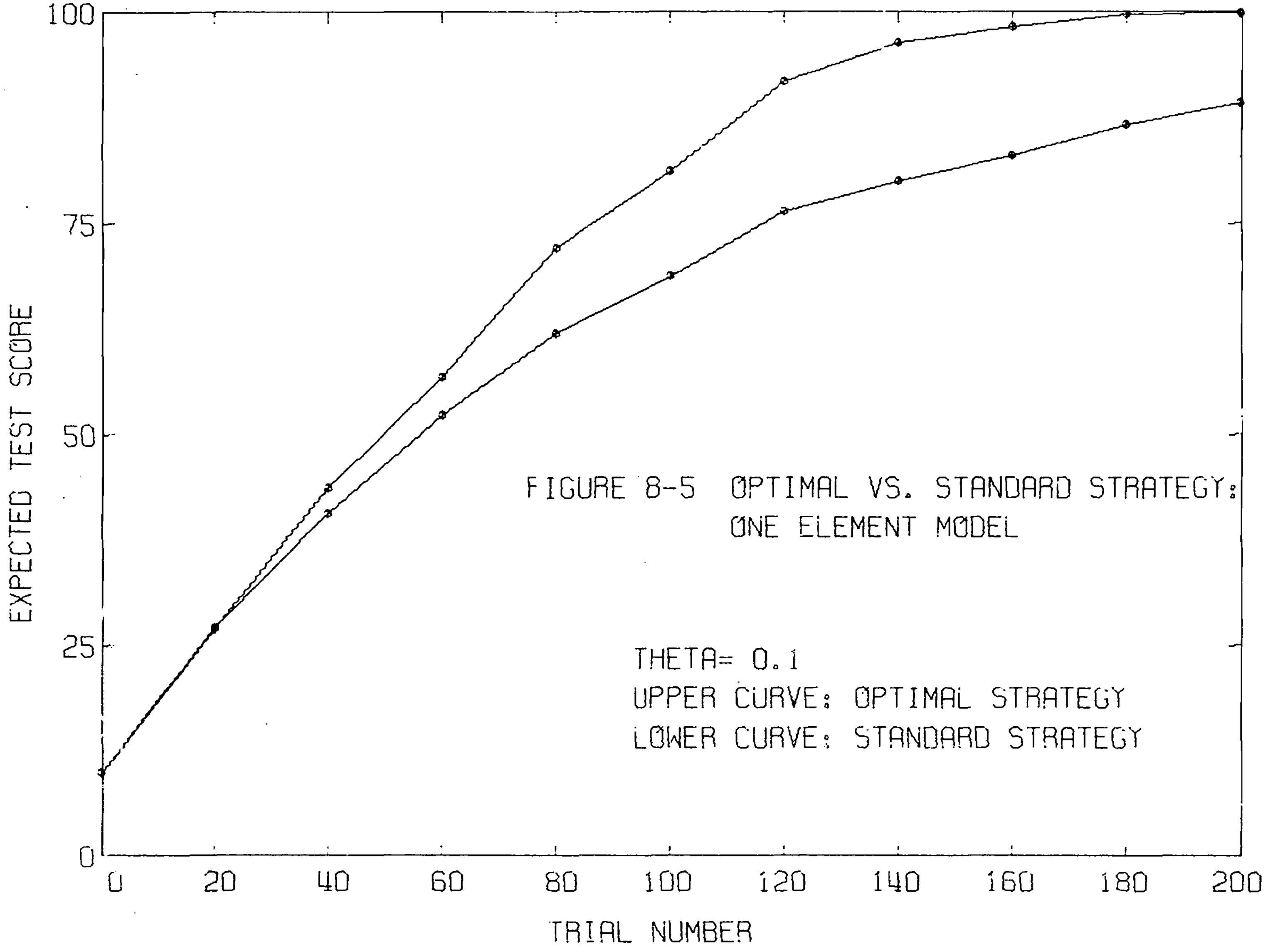


FIGURE 8-5 ØPTIMAL VS. STANDARD STRATEGY:
ØNE ELEMENT MØDEL

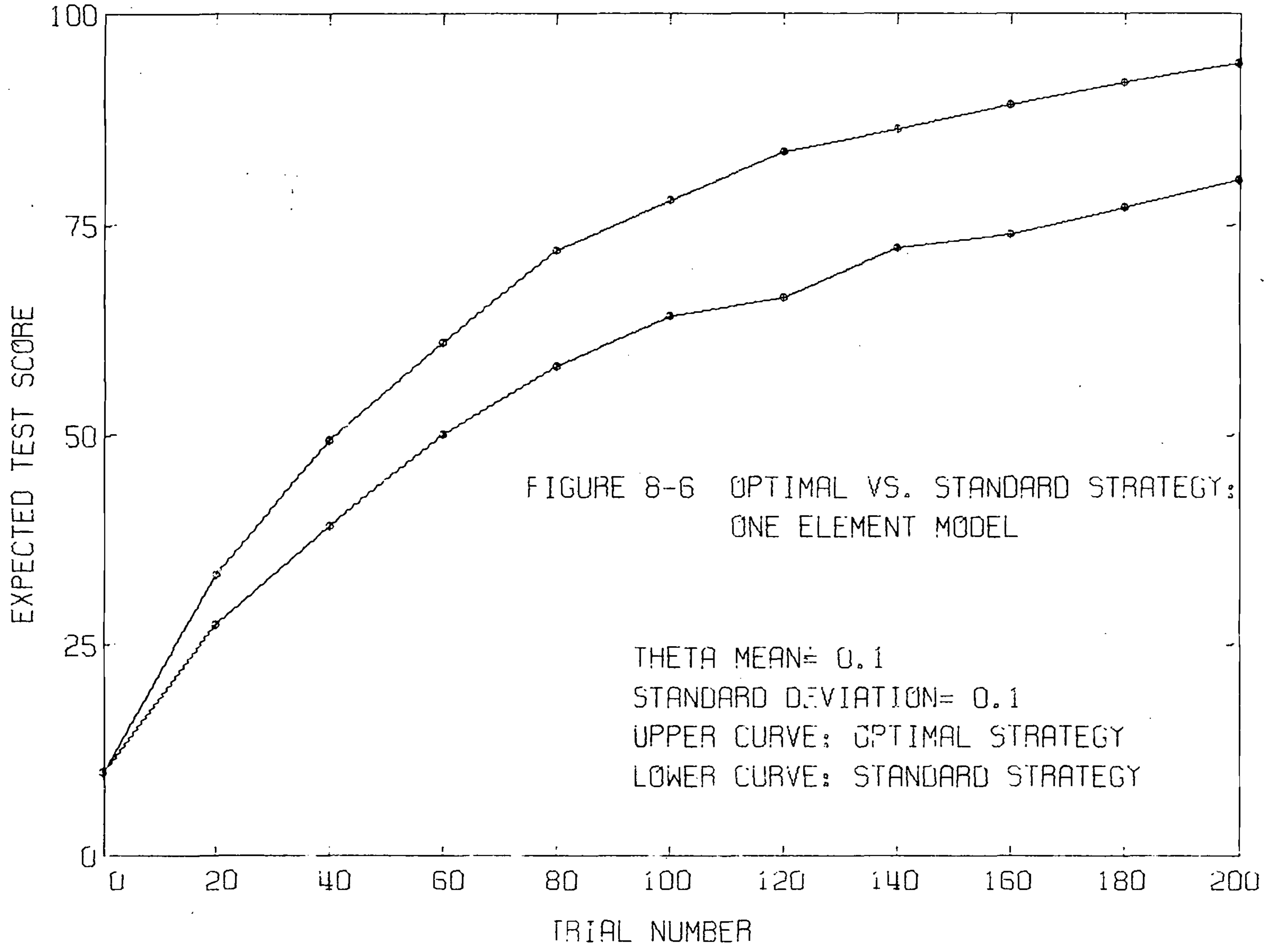


FIGURE 8-6 OPTIMAL VS. STANDARD STRATEGY: ONE ELEMENT MODEL

THETA MEAN= 0.1
STANDARD D:VIATION= 0.1
UPPER CURVE: CPTIMAL STRATEGY
LOWER CURVE: STANDARD STRATEGY

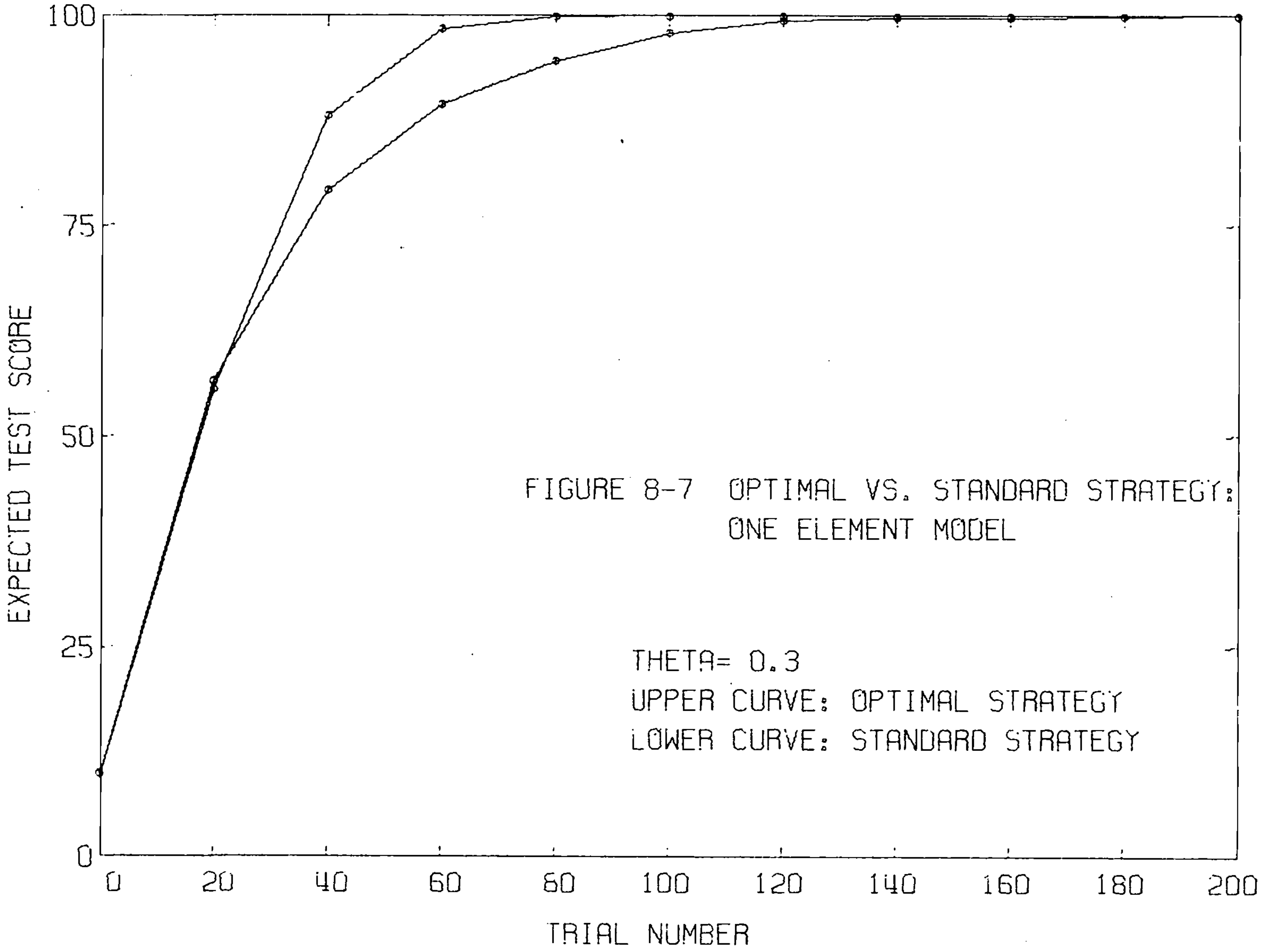


FIGURE 8-7 OPTIMAL VS. STANDARD STRATEGY: ONE ELEMENT MODEL

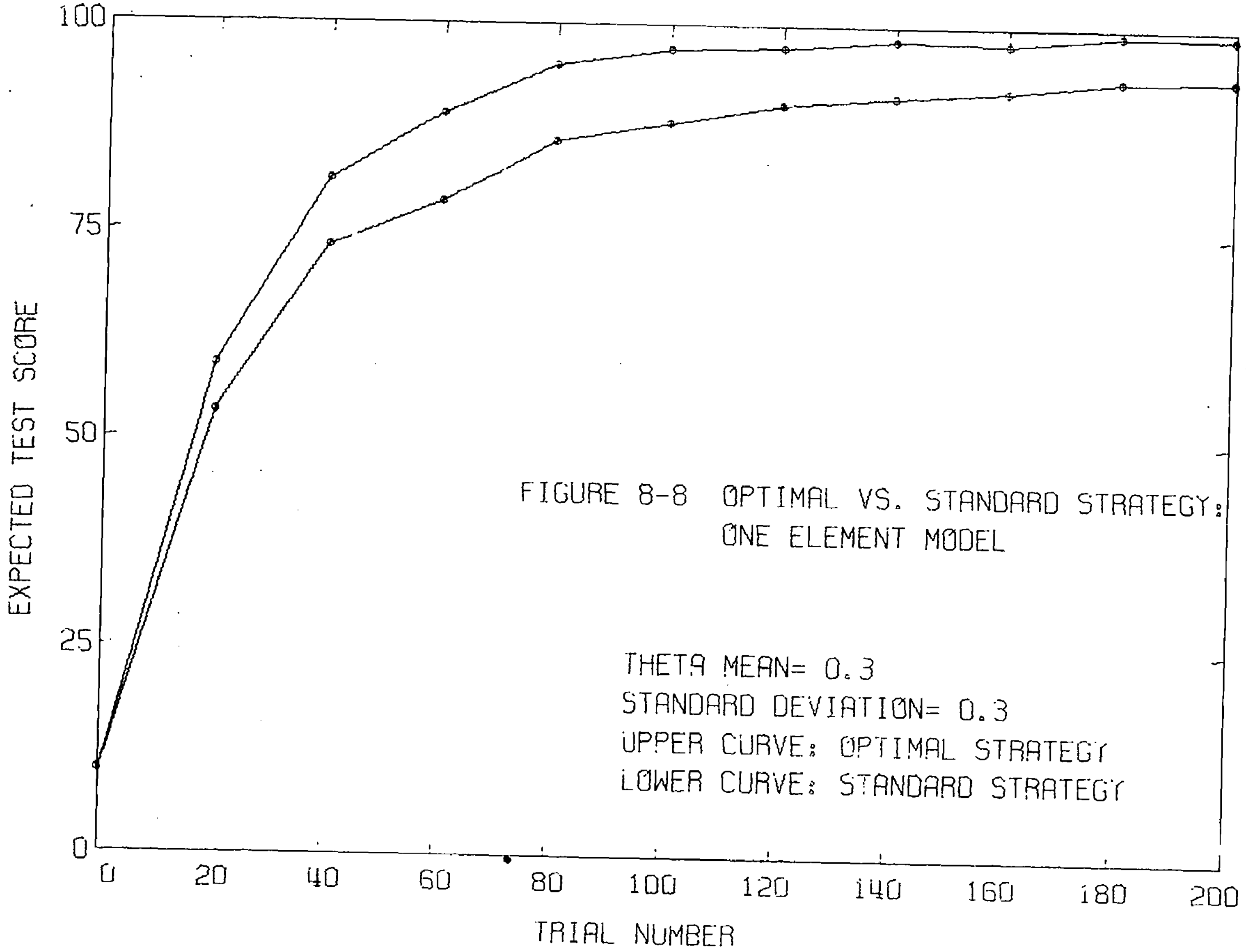


FIGURE 8-8 ØPTIMAL VS. STANDARD STRATEGY: ØNE ELEMENT MØDEL

THETA MEAN= 0.3
STANDARD DEVIATIØN= 0.3
UPPER CURVE: ØPTIMAL STRATEGY
LØWER CURVE: STANDARD STRATEGY

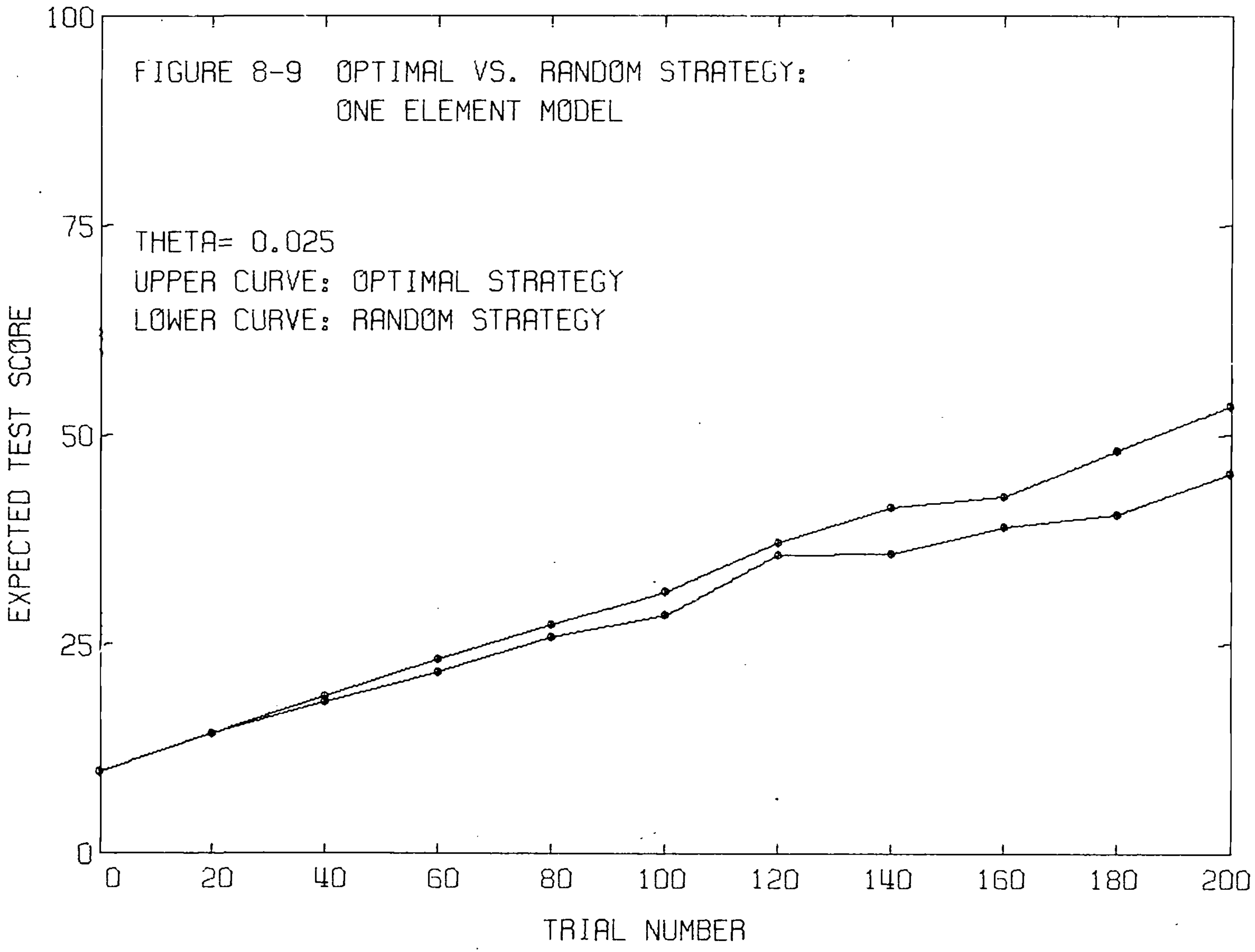


FIGURE 8-9 ØPTIMAL VS. RANDØM STRATEGY:
ØNE ELEMENT MØDEL

THETA= 0.025
UPPER CURVE: ØPTIMAL STRATEGY
LØWER CURVE: RANDØM STRATEGY

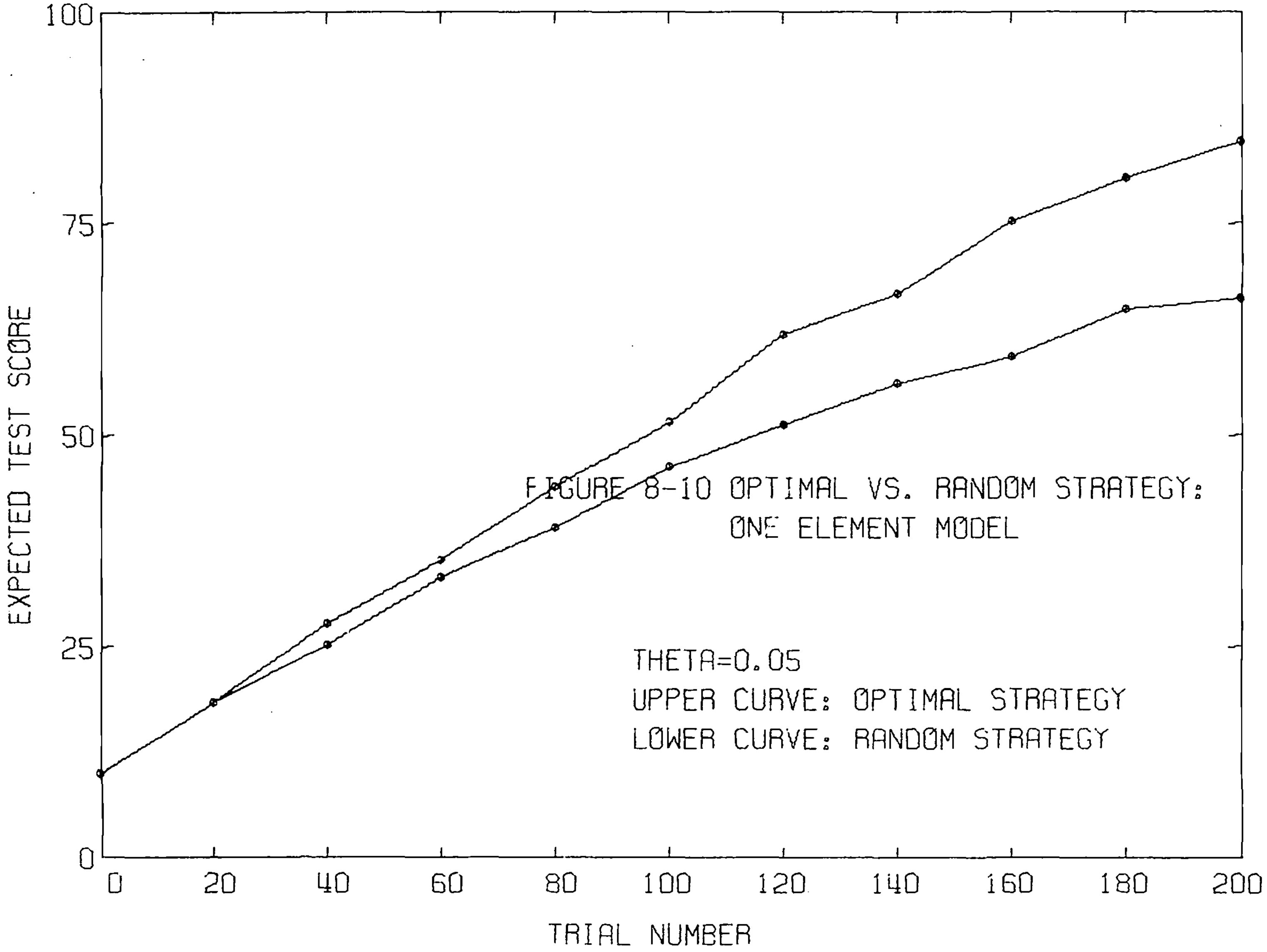


FIGURE 8-10 ØPTIMAL VS. RANDØM STRATEGY:
ØNE ELEMENT MØDEL

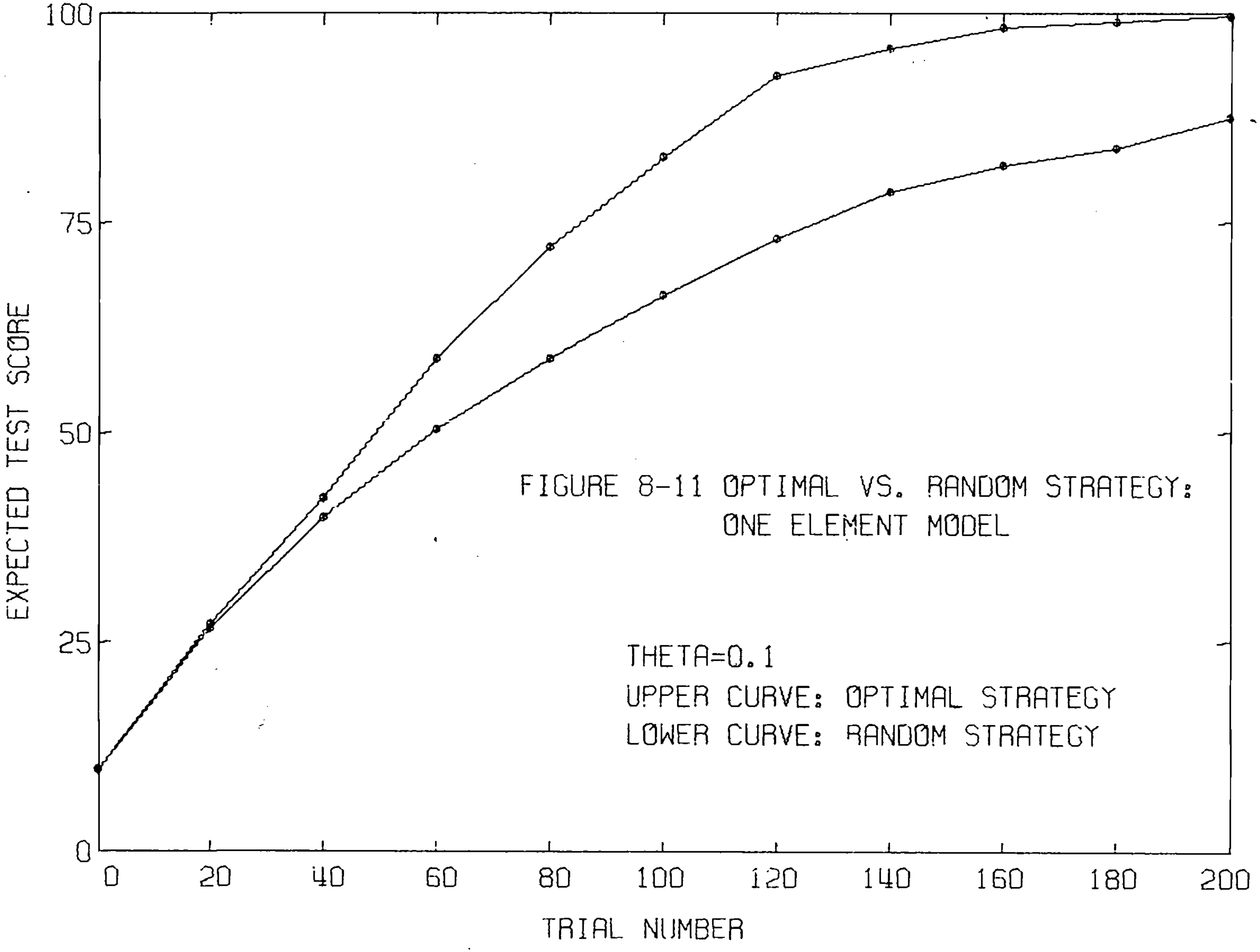


FIGURE 8-11 OPTIMAL VS. RANDOM STRATEGY: ONE ELEMENT MODEL

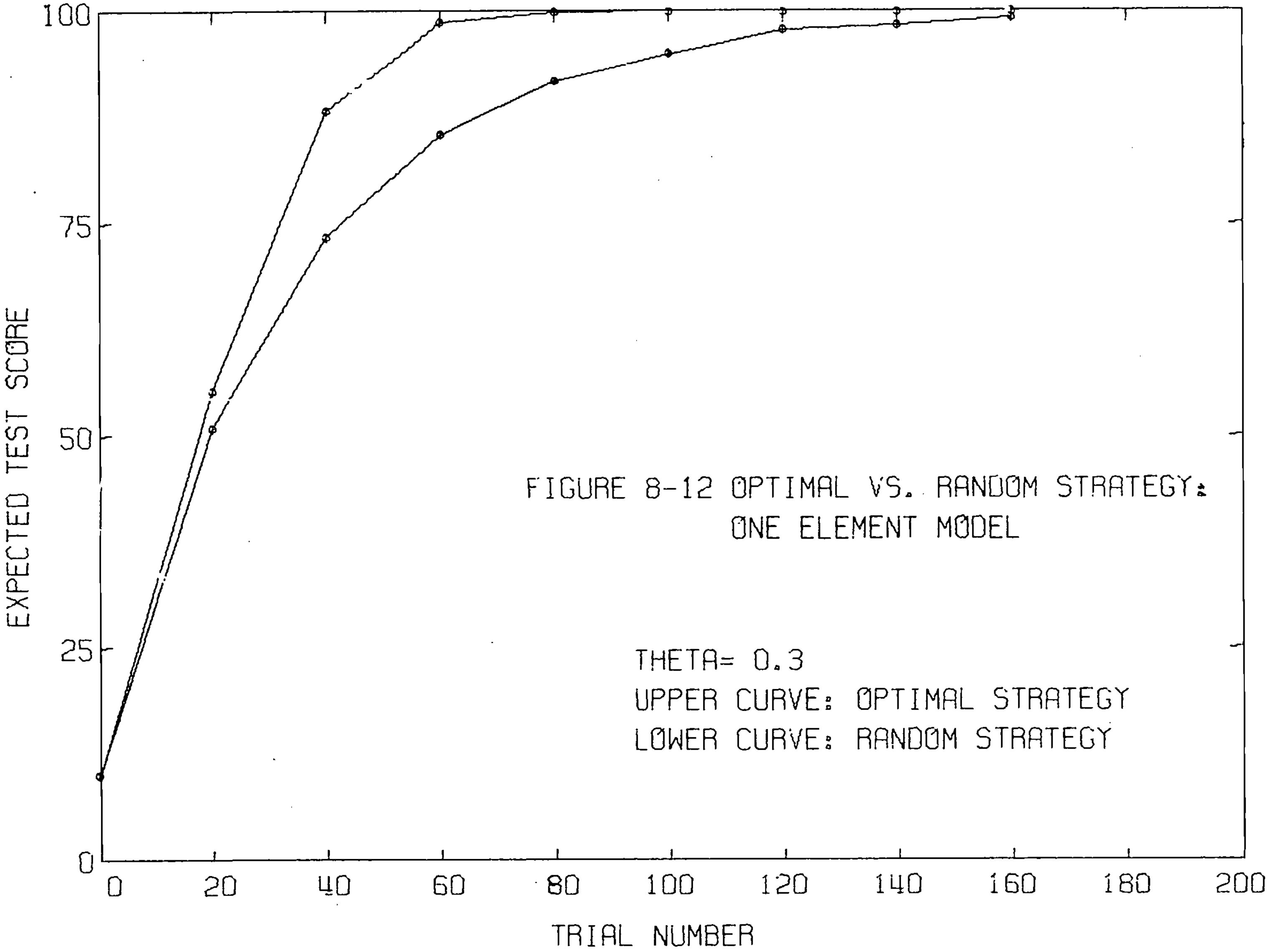


FIGURE 8-12 OPTIMAL VS. RANDOM STRATEGY:
ONE ELEMENT MØDEL

illustrates the corresponding perfect-performance scores. Figures 9-3 and 10-3 illustrate the corresponding information for the four-choice response experiment. Figures 9-1 and 9-3 illustrate the fact that, for both experiments, higher scores are predicted for both strategies than were actually observed, in addition to the fact that larger differences between the strategies were predicted than were observed, although the predicted differences are not as great as might have been expected. It may also be noted that the number of trials chosen by Dear, et al. (320) appears to be past the point of predicted maximum difference between the two strategies. This means that a greater difference might have been observed in the experiment if a smaller number of trials (between 200 and 240) had been chosen, although it is unlikely this would have been the case, since the maximum difference in either case is not a great deal larger than the difference at 320 trials. Figures 10-1 and 10-3, illustrating the mean perfect-performance scores for the two experiments, also shows that a different choice for the number of trials probably would not have produced a significantly different result, in terms of the difference between the two strategies.

Figures 9-2, 9-4, 10-2, and 10-4 show the results of the same simulations as were run for 9-1, 9-3, 10-1, and 10-3, respectively, except for the values of θ (0.05 for the two-choice response experiment and 0.044 for the four-choice response experiment). These values were chosen so that the expected sum-of-correct-responses scores for the standard strategy would be approximately the same as those obtained experimentally. The reason for doing this was to observe the corresponding predicted results for the optimal strategy for these values of θ. Figures 9-2, 9-4, 10-2, and 10-4 show that the predicted differences for the two strategies are the same or greater than those corresponding to a θ of 0.1. It will be noted, however, that the maximum predicted differences occur in this case for values of m <u>greater</u> than 320. The basic conclusion drawn by Dear, et al., that the one-element model is shown to be inadequate for accurately representing the learning process involved, nevertheless appears to be justified on the basis of the simulation data.

The final learning model to be discussed in the context of optimal instruction-sequencing is the Random-trial Increments (RTI) model (Matheson, 1964). It is well known that, in a mathematical sense, the linear and one-element models are merely special cases of the RTI model corresponding to certain choices of parameters (c=1.0 and α_R=0.0, respectively). As has already been mentioned, when the RTI model is used in response-insensitive mode, the values of E{T} produced by a given strategy, on the average, correspond to those produced by the same strategy using a linear model with

$$\alpha = \alpha_R + 1 - c \qquad (6)$$

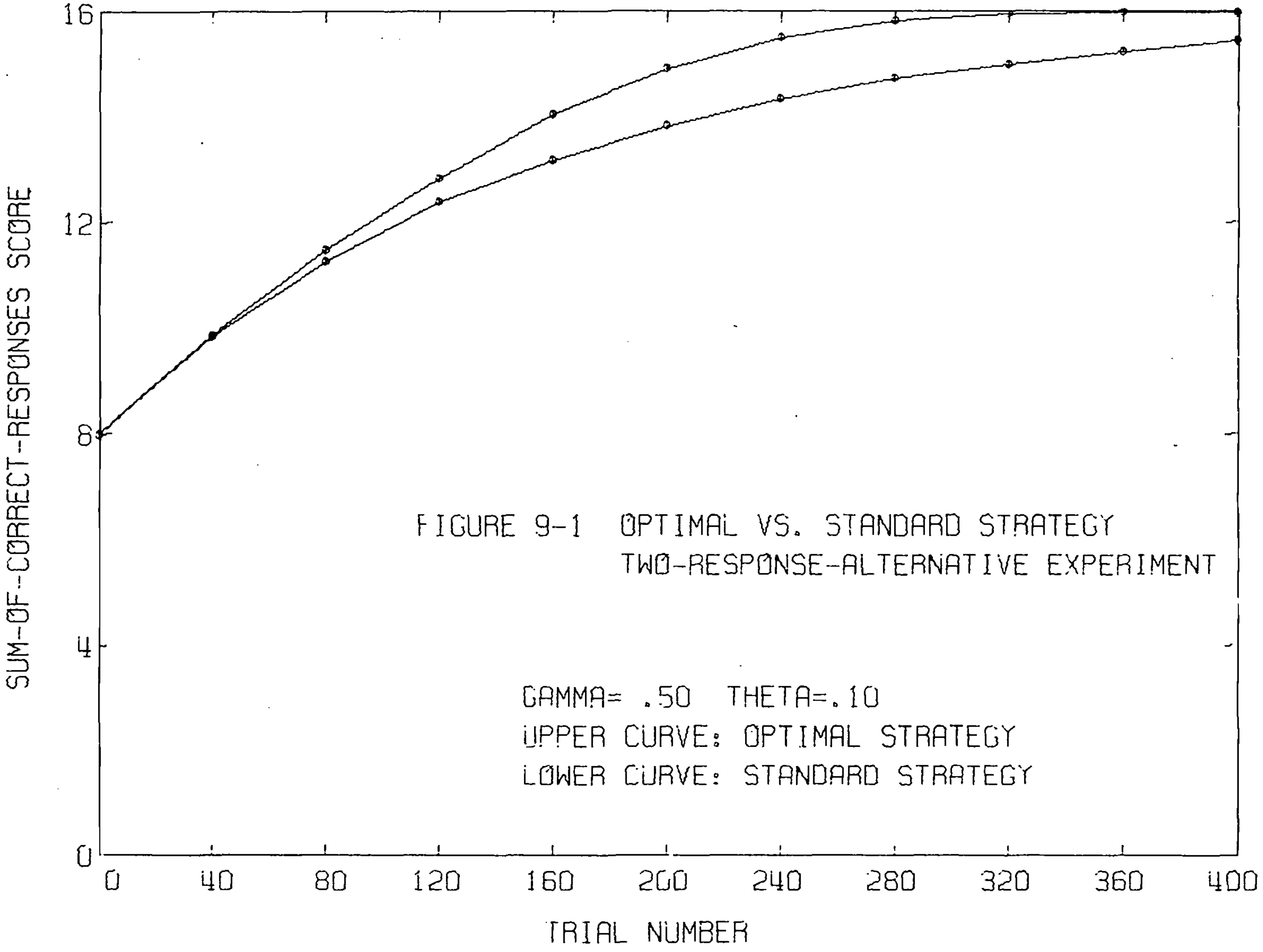


FIGURE 9-1 OPTIMAL VS. STANDARD STRATEGY
TWO-RESPONSE-ALTERNATIVE EXPERIMENT

GAMMA= .50 THETA=.10
UPPER CURVE: OPTIMAL STRATEGY
LOWER CURVE: STANDARD STRATEGY

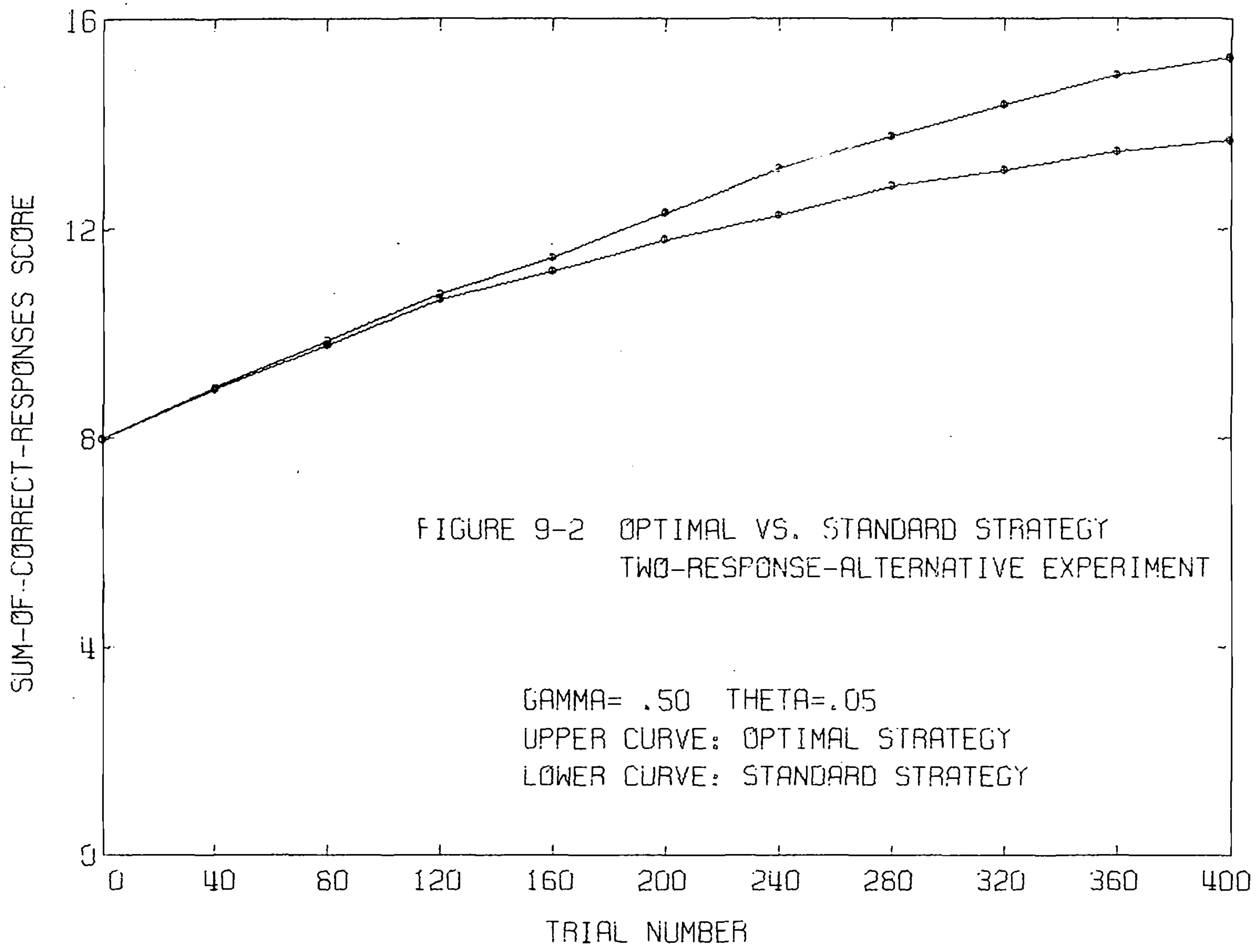


FIGURE 9-2 OPTIMAL VS. STANDARD STRATEGY
TWO-RESPONSE-ALTERNATIVE EXPERIMENT

GAMMA= .50 THETA=.05
UPPER CURVE: OPTIMAL STRATEGY
LOWER CURVE: STANDARD STRATEGY

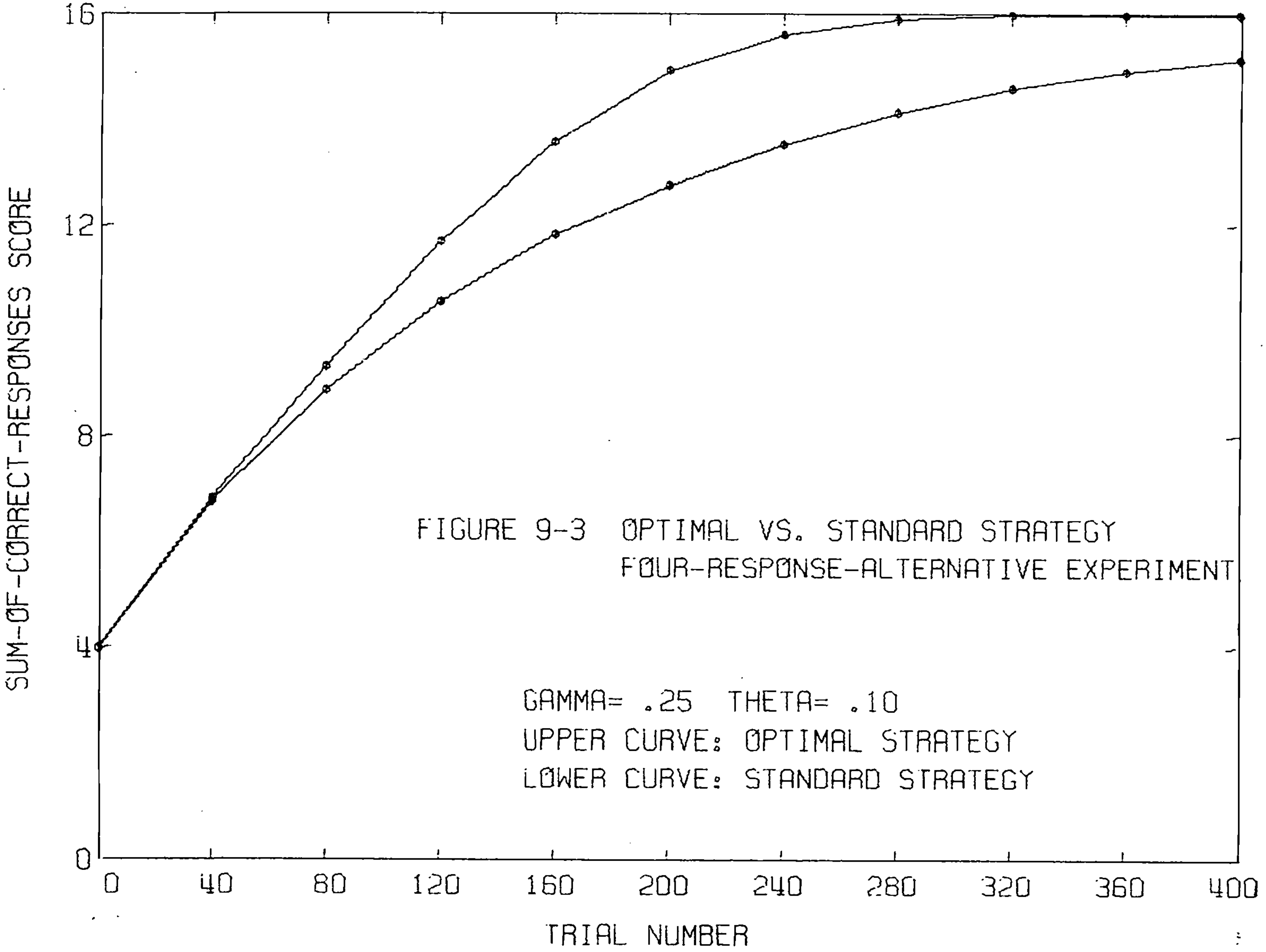


FIGURE 9-3 ØPTIMAL VS. STANDARD STRATEGY
FØUR-RESPØNSE-ALTERNATIVE EXPERIMENT

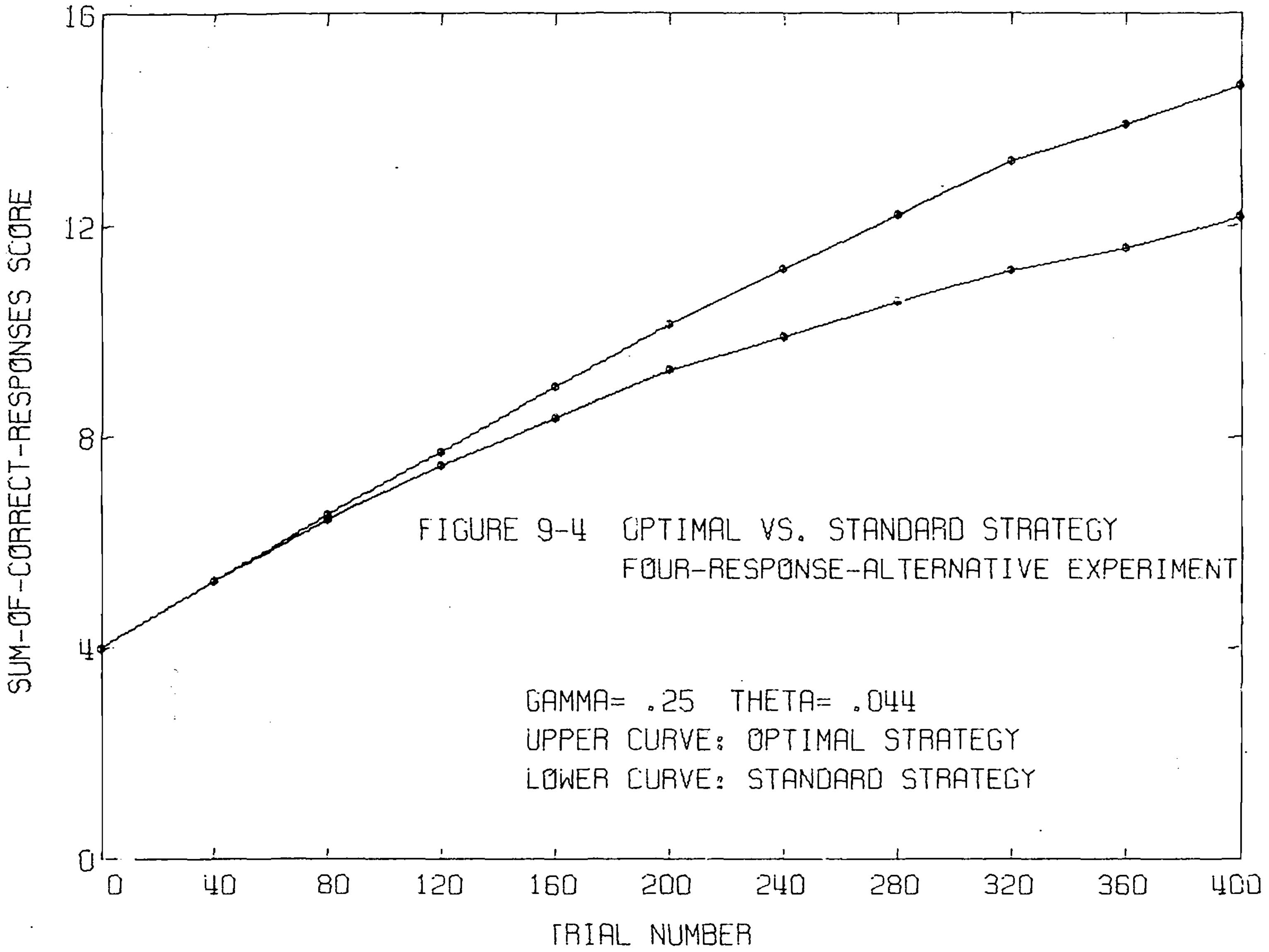


FIGURE 9-4 ØPTIMAL VS. STANDARD STRATEGY
FØUR-RESPØNSE-ALTERNATIVE EXPERIMENT

GAMMA= .25 THETA= .044
UPPER CURVE: ØPTIMAL STRATEGY
LØWER CURVE: STANDARD STRATEGY

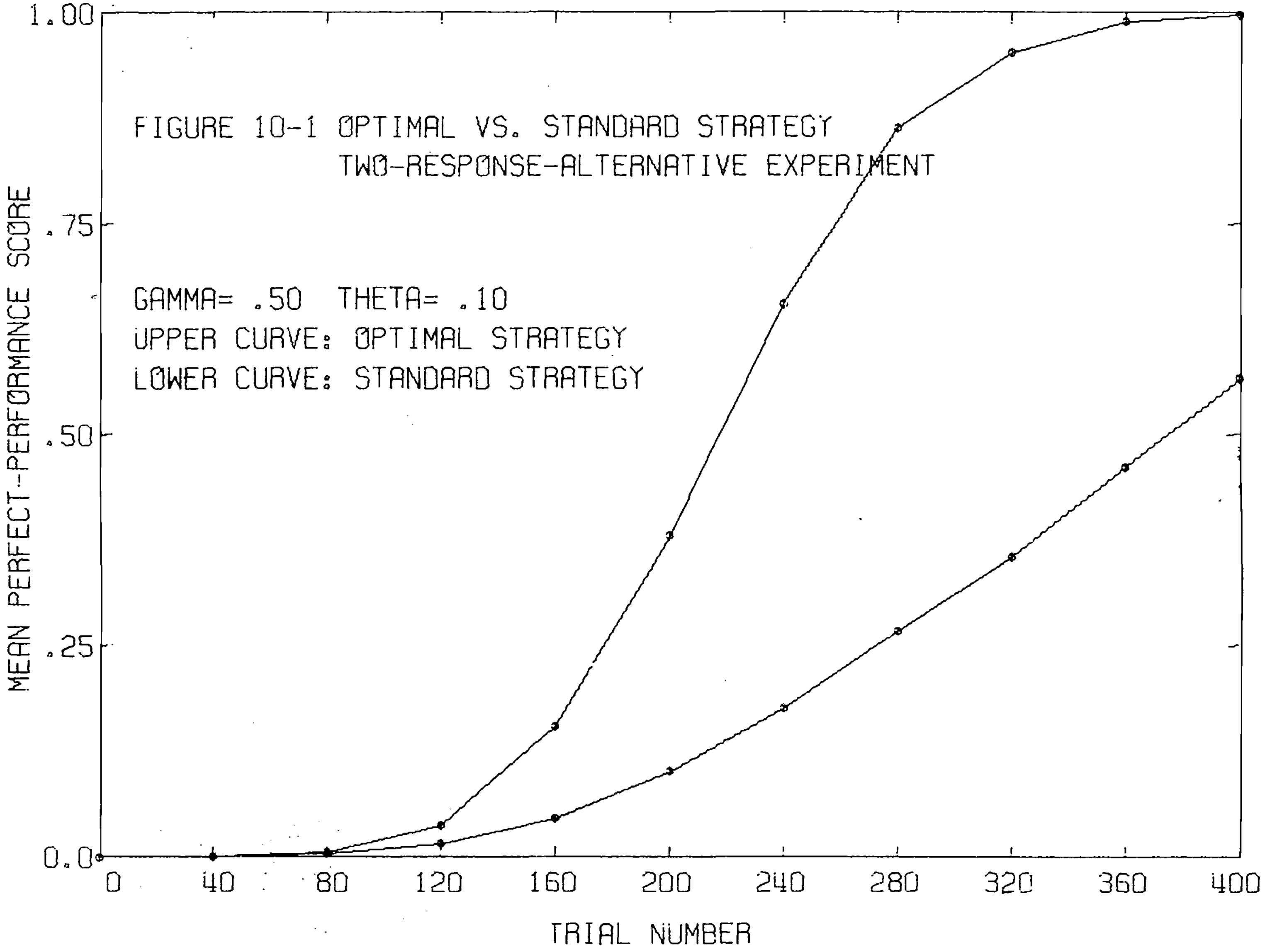


FIGURE 10-1 ØPTIMAL VS. STANDARD STRATEGY
TWØ-RESPØNSE-ALTERNATIVE EXPERIMENT

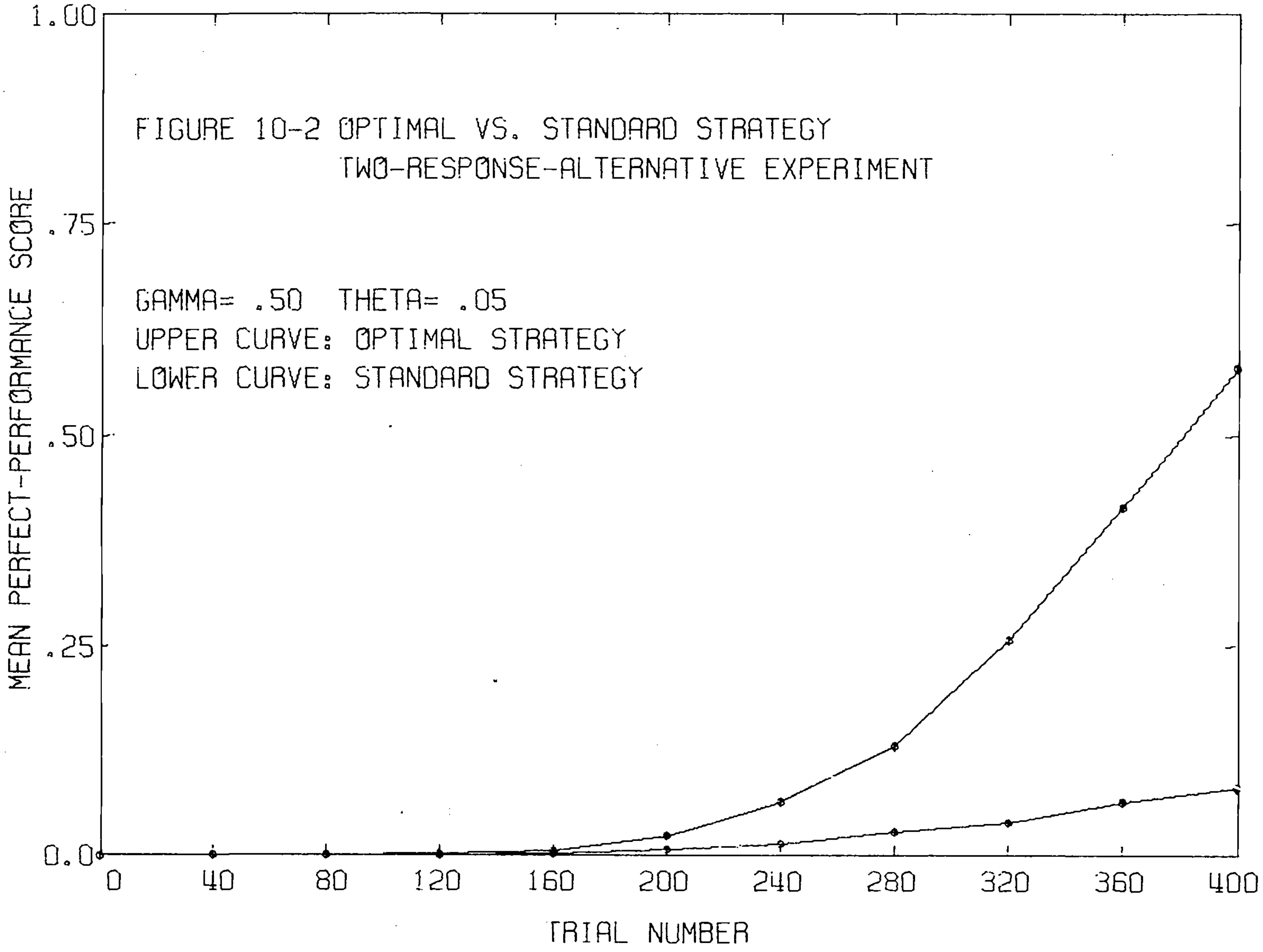


FIGURE 10-2 OPTIMAL VS. STANDARD STRATEGY
TWO-RESPONSE-ALTERNATIVE EXPERIMENT

GAMMA= .50 THETA= .05
UPPER CURVE: OPTIMAL STRATEGY
LOWER CURVE: STANDARD STRATEGY

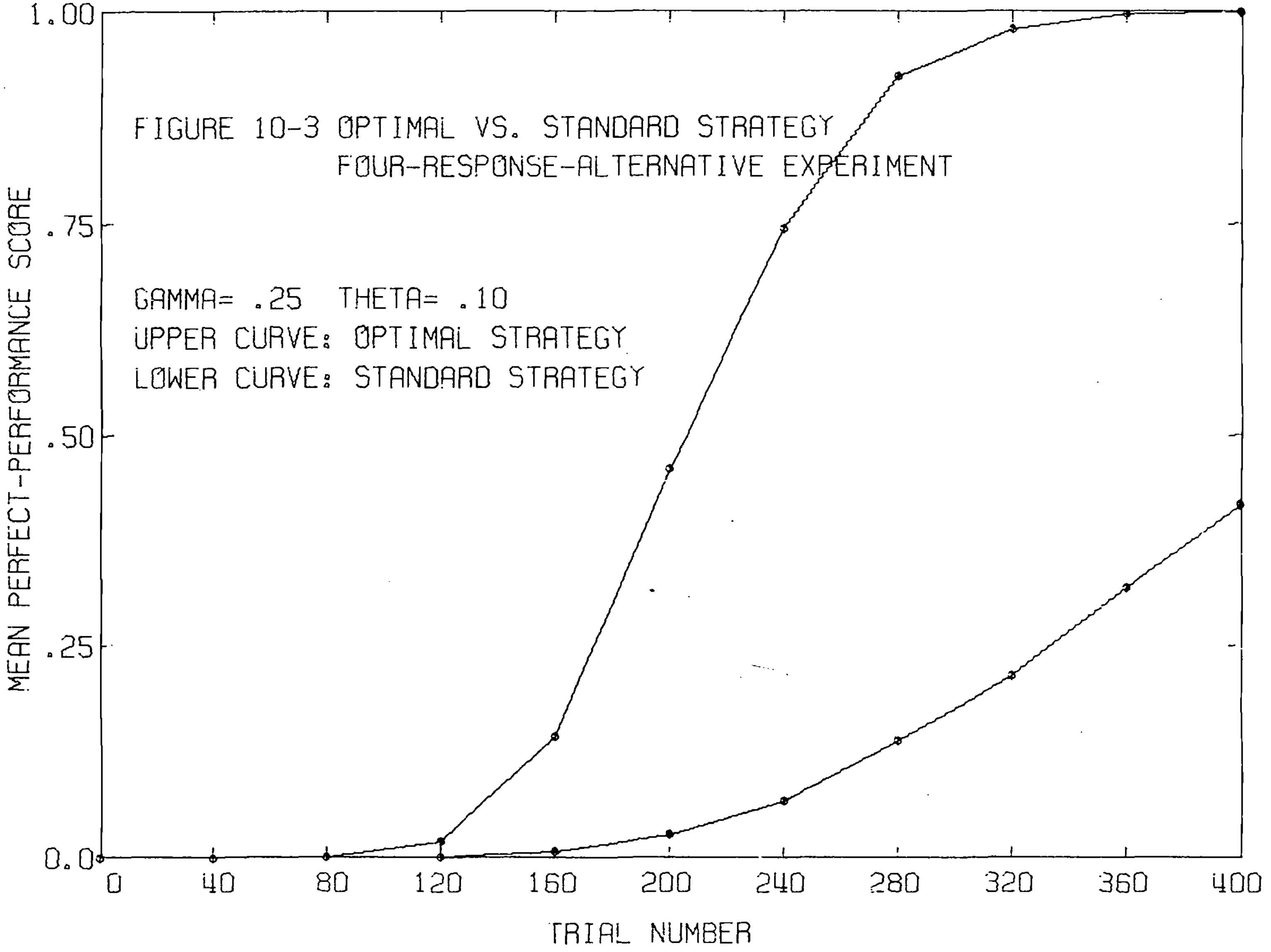


FIGURE 10-3 ØPTIMAL VS. STANDARD STRATEGY
FØUR-RESPØNSE-ALTERNATIVE EXPERIMENT

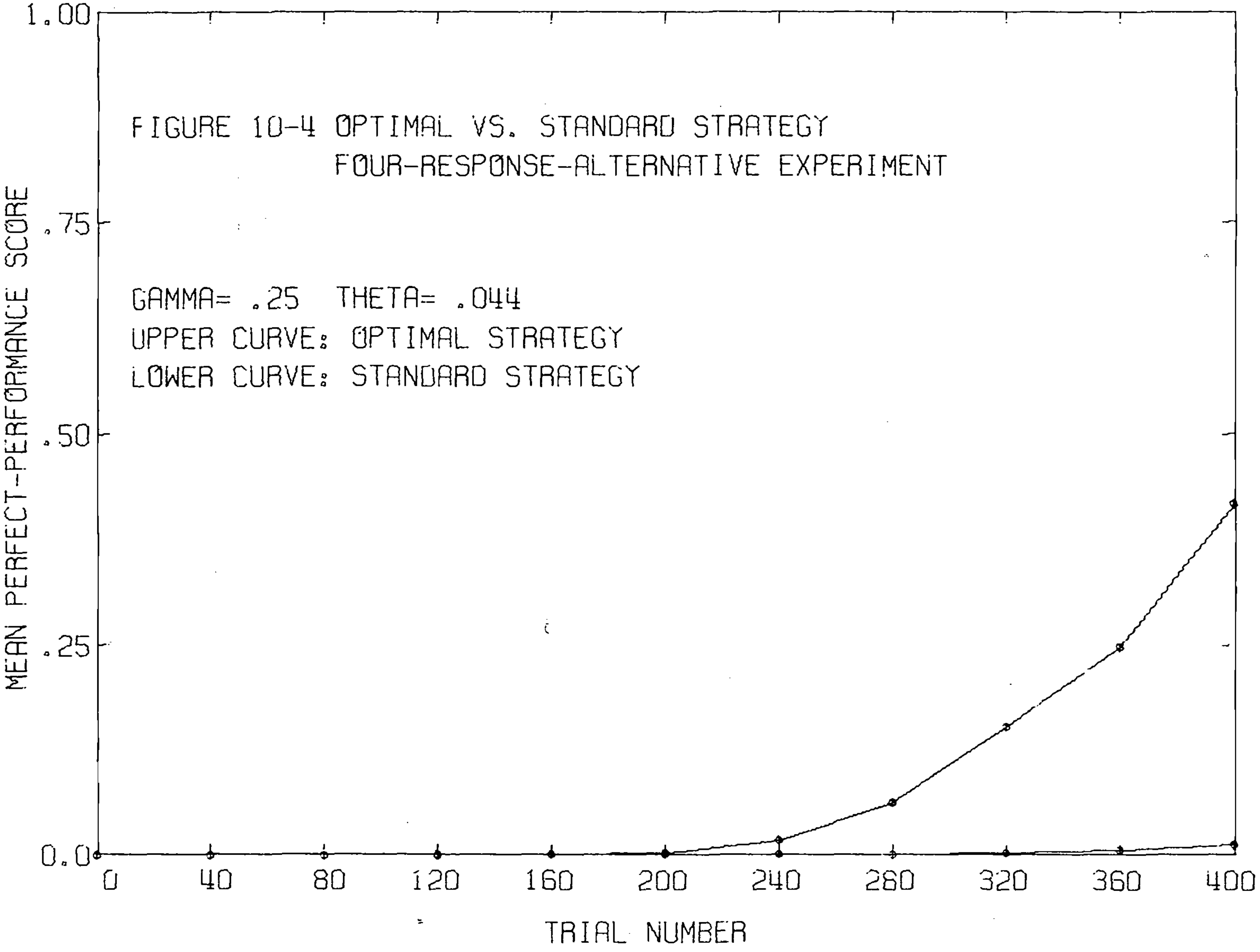


FIGURE 10-4 OPTIMAL VS. STANDARD STRATEGY
FOUR-RESPONSE-ALTERNATIVE EXPERIMENT

It appears that the RTI model might also be used in a response-sensitive mode by employing some statistical means of differentiating expected values of q following a correct response from those following an incorrect response, such as the Bayesian estimator used by Karush & Dear (1966). In order to obtain such an estimator for the RTI model, one might first consider the distribution of q values for sequential trials. Following Atkinson, et al. (1965, pp. 123-128), the joint probability of incorrect responses on trials i and i+1 is:

$$P\{\overline{E}_i \& \overline{E}_{i+1}\} = \sum_{\nu} c\alpha q^2_{\nu,i} P\{S_{i-1}=\nu\} + \sum_{\nu}(1-c)q^2_{\nu,i}P\{S_{i-1}=\nu\}$$

$$= (\alpha_R c + 1 - c)V_{2,i} \qquad ,$$

where

$$V_{r,i} = \sum_{\nu} q^r_{\nu,i} P\{S_{i-1}=\nu\} = (\alpha^r_R c + 1 - c)^{i-1} q^r_o$$

is the r-th moment of the distribution of q values on trial i. Now,

$$P\{\overline{E}_{i+1}|\overline{E}_i\} = P\{\overline{E}_i \& \overline{E}_{i+1}\}/P\{\overline{E}_i\}$$

$$= (\alpha_R c+1-c)V_{2,i}/V_{1,i}$$

$$= (\alpha_R c+1-c)q_o \xi^{i-1} \qquad (7)$$

where

$$\xi = (\alpha^2_R c+1-c)/(\alpha_R c+1-c).$$

Correspondingly, the joint probability of a correct response on trial i and an incorrect response on trial i+1 is:

$$P\{E_i \& \overline{E}_{i+1}\} = \sum_\nu c\alpha q_{\nu,i}(1-q_{\nu,i})P\{S_{i-1}=\nu\} + \sum_\nu (1-c)q_{\nu,i}(1-q_{\nu,i})P\{S_{i-1}=\nu\}$$

$$= (\alpha_R c+1-c)(V_{1,i}-V_{2,i})$$

$$= \alpha(V_{1,i}-V_{2,i})$$

Hence,

$$P\{\overline{E}_{i+1}|E_i\} = \alpha(V_{1,i}-V_{2,1})/(1-V_{1,i})$$

$$= \{V_{1,i}/(1-V_{1,i})\}(\alpha-P\{\overline{E}_{i+1}|\overline{E}_i\}) \qquad (8)$$

Using Equations (7) and (8) as the basis for providing updated estimates of q values, given the previous response, a simulation was conducted to compare the results of applying algorithm A for three different sets of RTI parameters. All values were chosen so that α, as given by Equation (6), corresponded to one of the eight values used in the previous simulations involving the linear model. Each set of α_R and c values consisted of three pairs, two of which corresponded to the special cases of the linear and one-element models (c=1. and α_R=0., respectively), with the third pair corresponding to an intermediate RTI model. The results of the simulation are shown in Figures 11-1 through 11-8, corresponding to the eight different α values. As would be expected, the E{T} values obtained for the linear cases (lower curves) were almost identical (subject only to minor statistical variation) to the values previously obtained for the actual linear model simulation. That the RTI model with c=1. and the linear model are equivalent in this sense can be seen from the fact that both $P\{\overline{E}_{i+1}|\overline{E}_i\}$ and $P\{\overline{E}_{i+1}|E_i\}$ reduce to $\alpha^i q_o$ (q_{i+1} for the linear model) meaning that the model is response-insensitive. The performance of A is seen to improve for values of α_R and c giving the same equivalent value of α, but corresponding to RTI models lying in between the

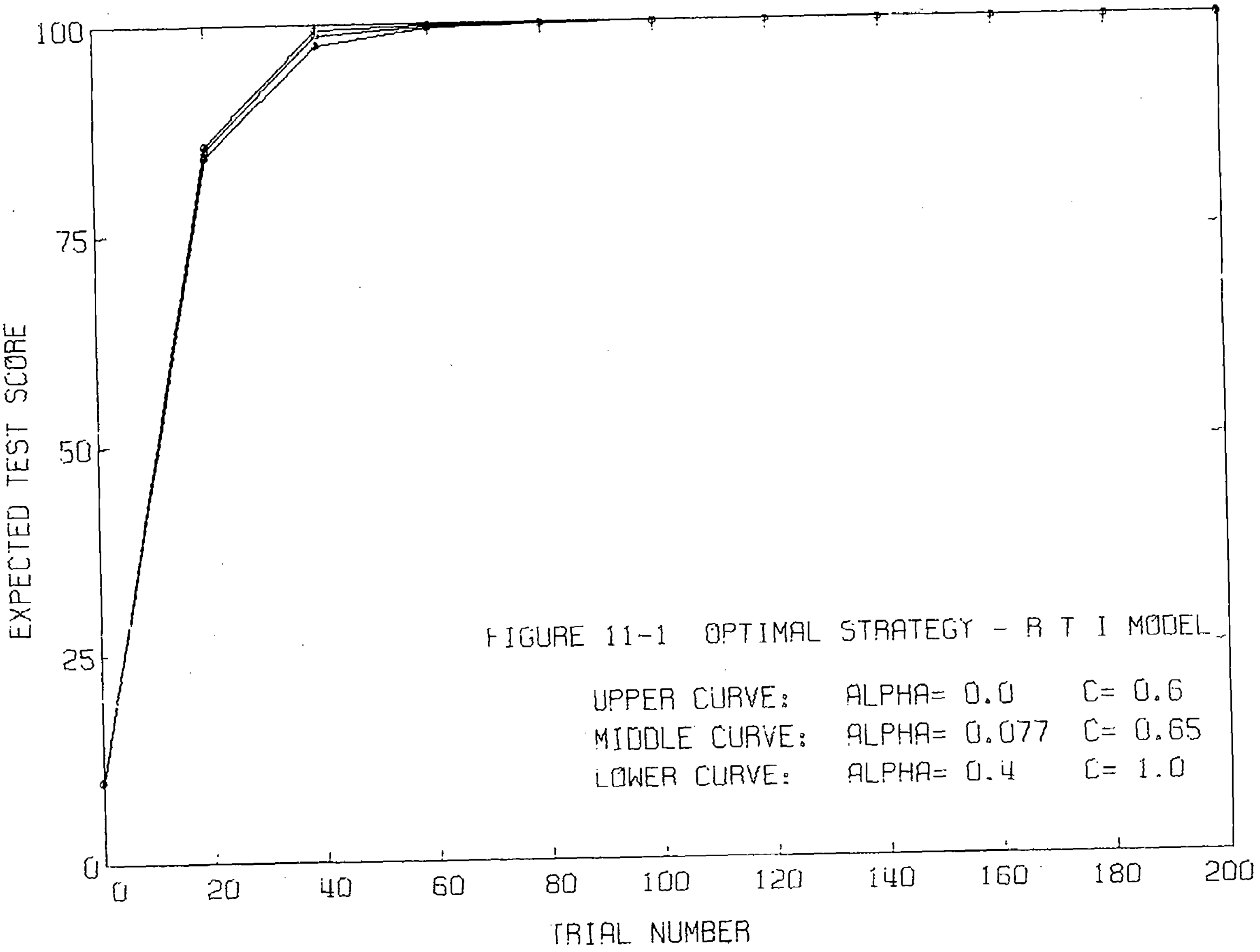


FIGURE 11-1 OPTIMAL STRATEGY - R T I MODEL

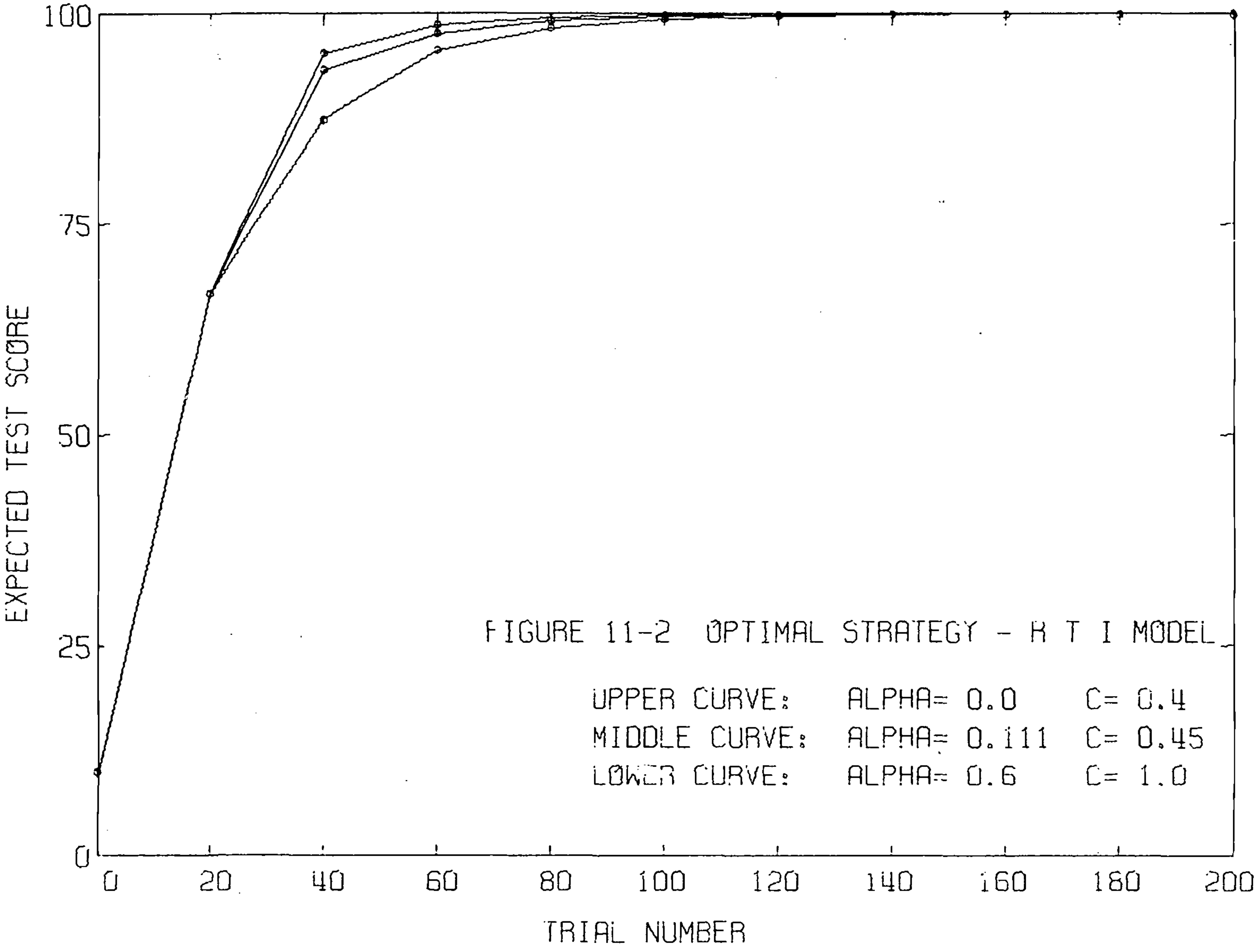


FIGURE 11-2 OPTIMAL STRATEGY - R T I MODEL

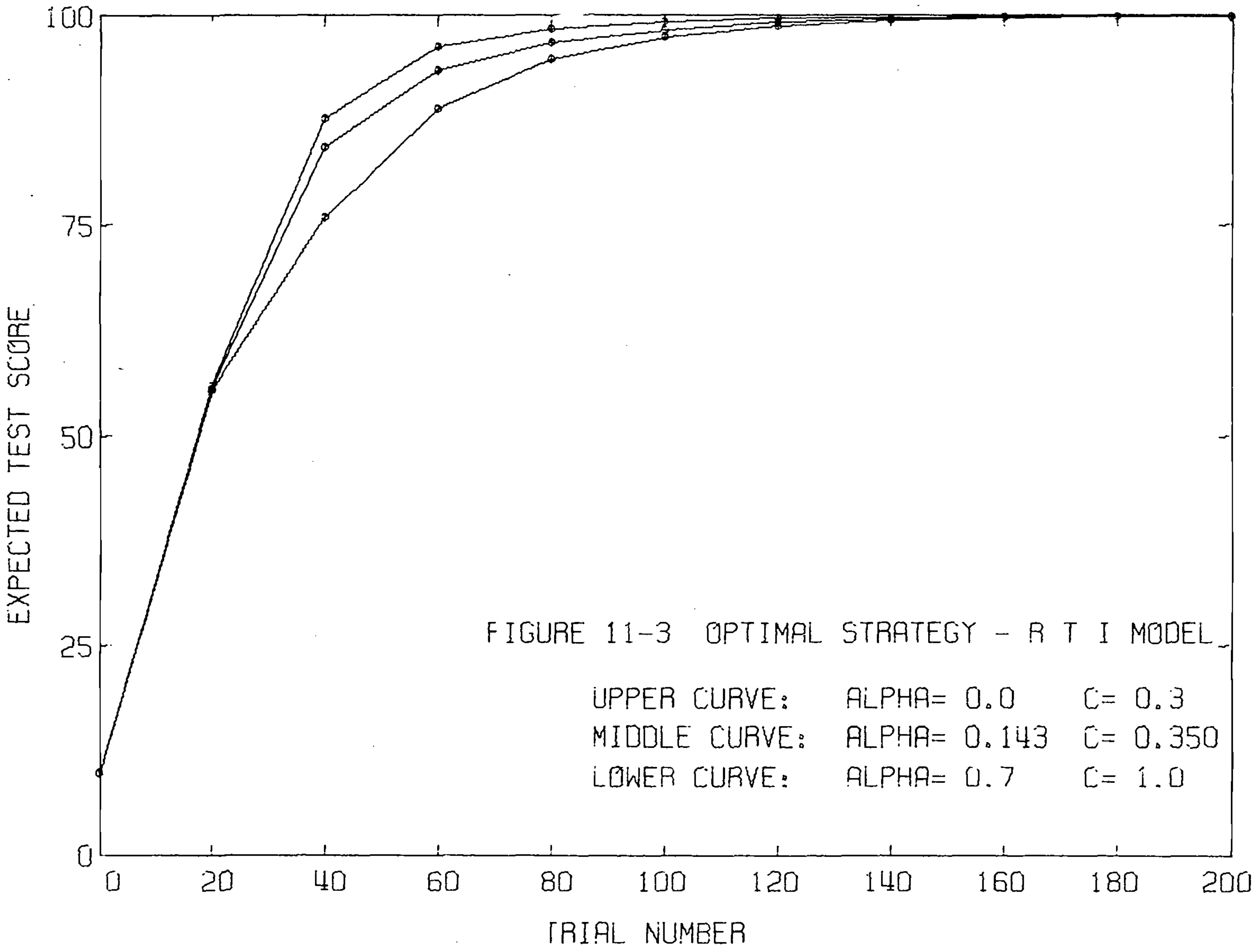


FIGURE 11-3 OPTIMAL STRATEGY - R T I MODEL

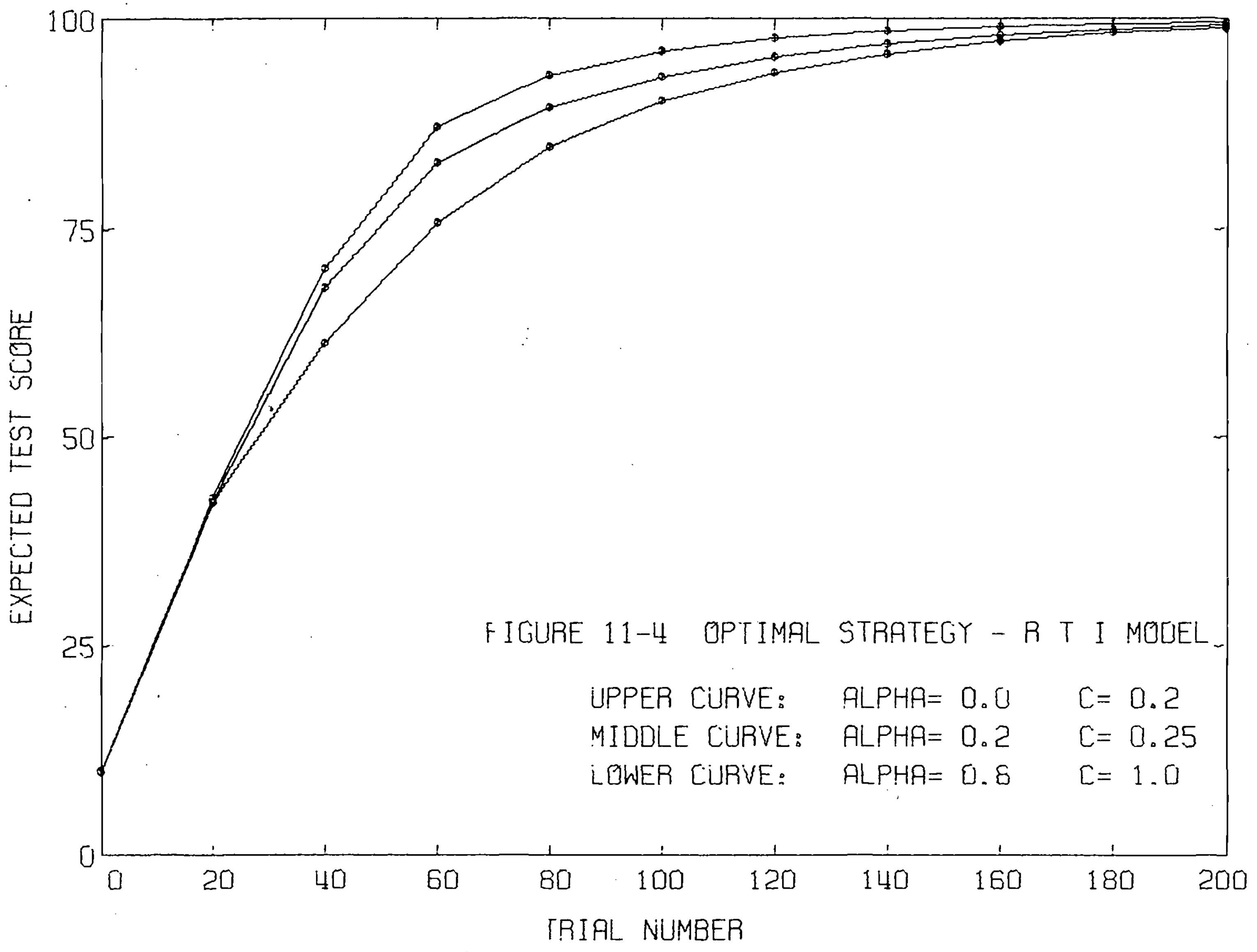


FIGURE 11-4 OPTIMAL STRATEGY - R T I MODEL

UPPER CURVE: ALPHA= 0.0 C= 0.2
MIDDLE CURVE: ALPHA= 0.2 C= 0.25
LOWER CURVE: ALPHA= 0.8 C= 1.0

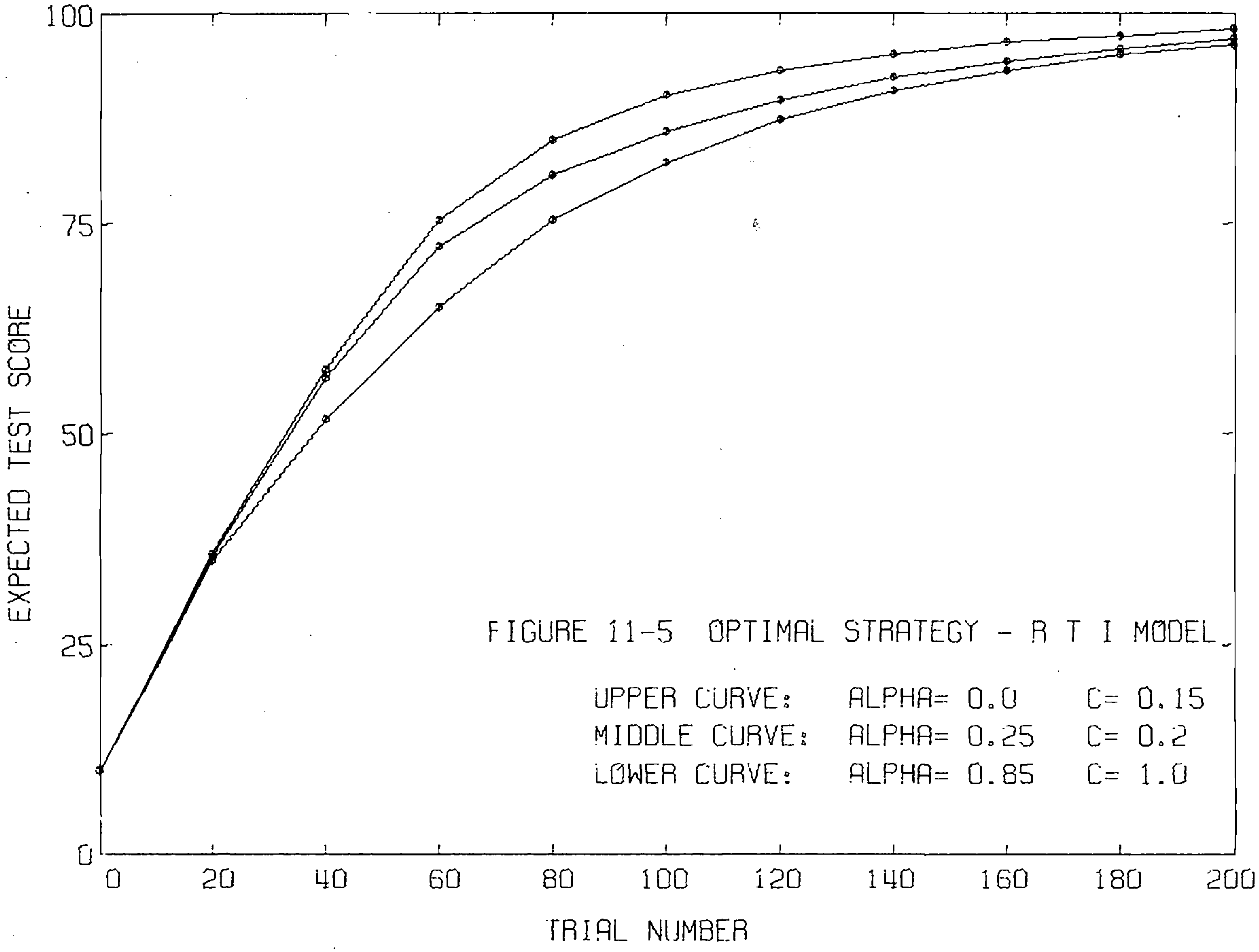


FIGURE 11-5 OPTIMAL STRATEGY - R T I MODEL

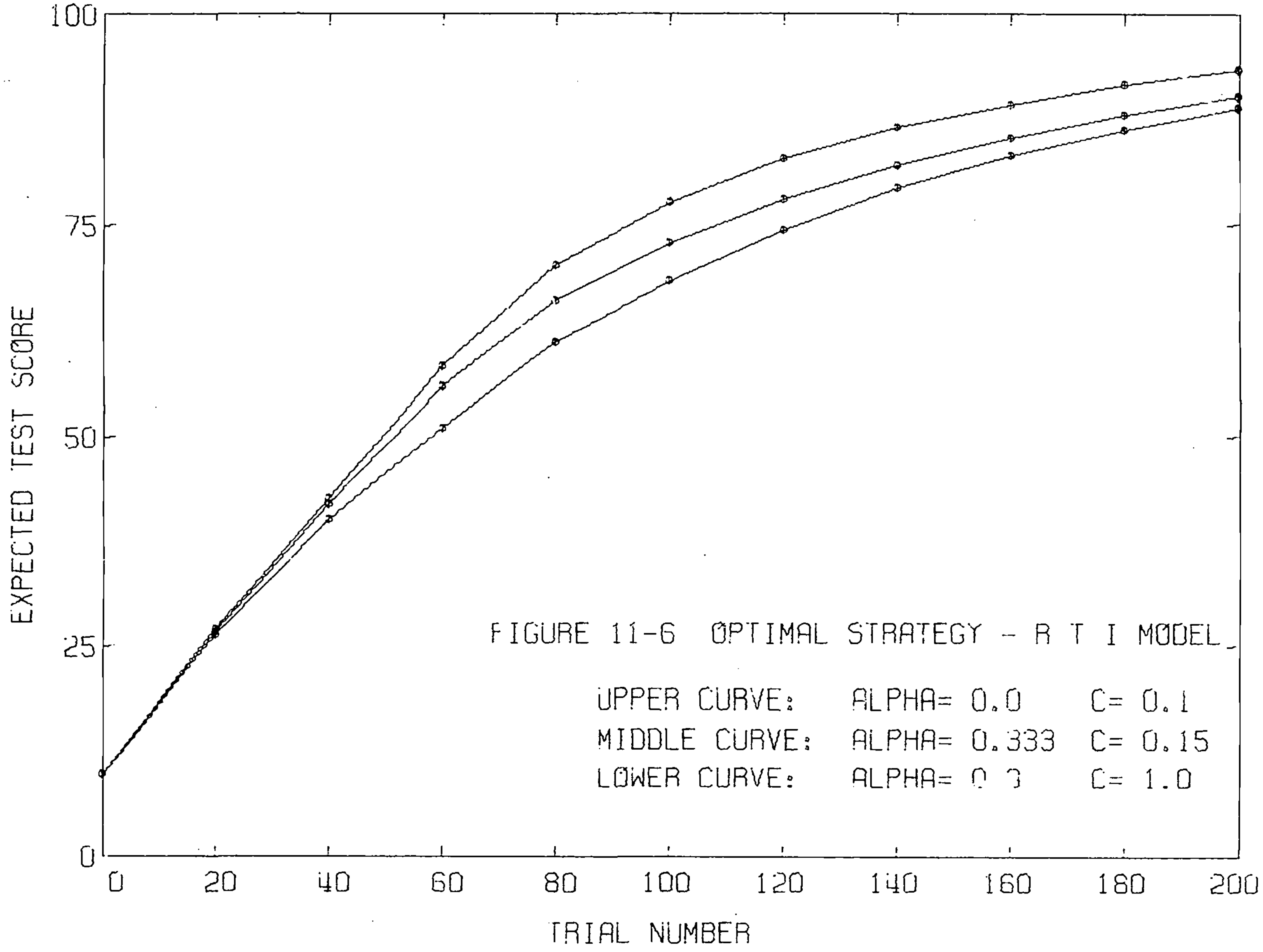


FIGURE 11-6 OPTIMAL STRATEGY - R T I MODEL

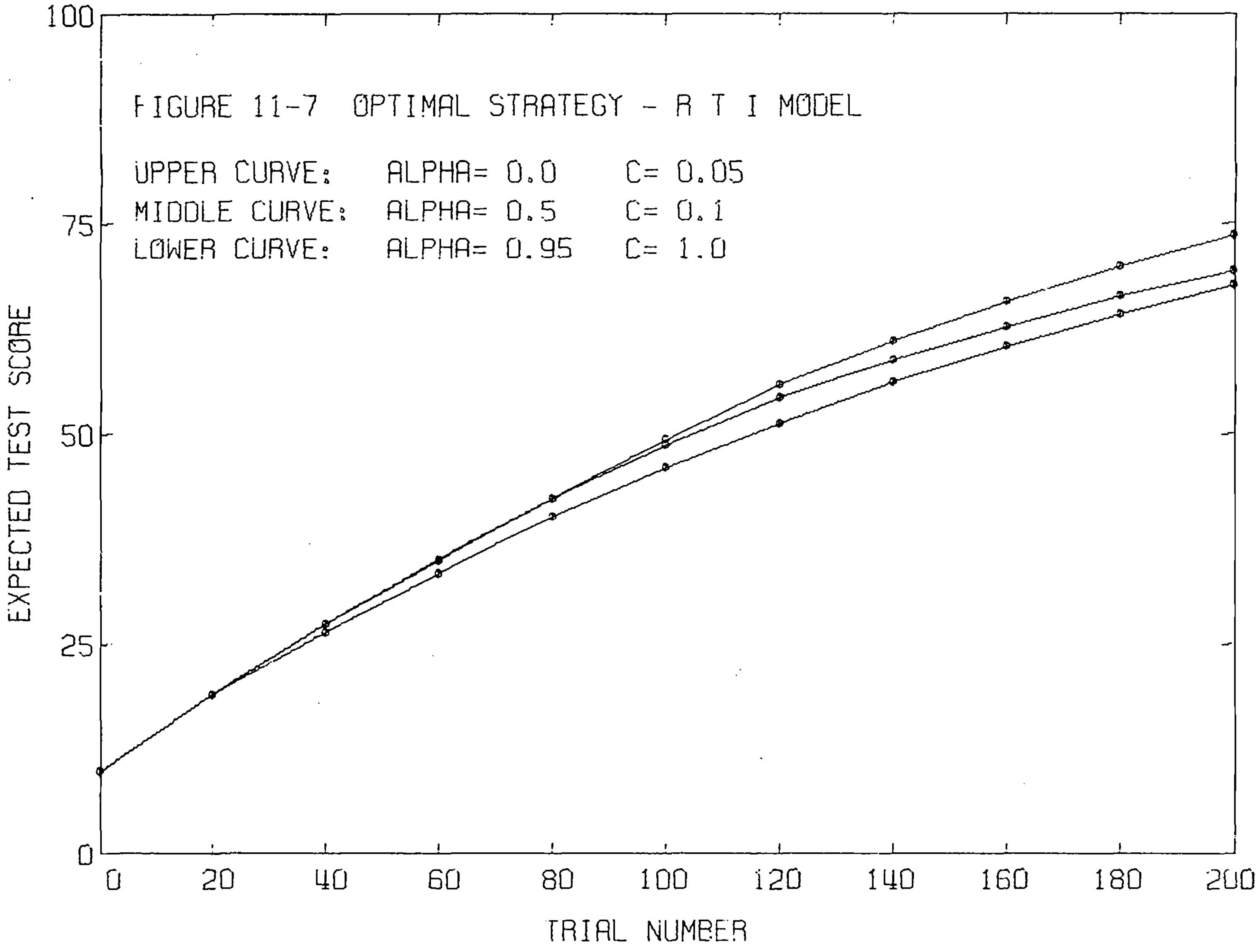


FIGURE 11-7 ØPTIMAL STRATEGY - R T I MØDEL

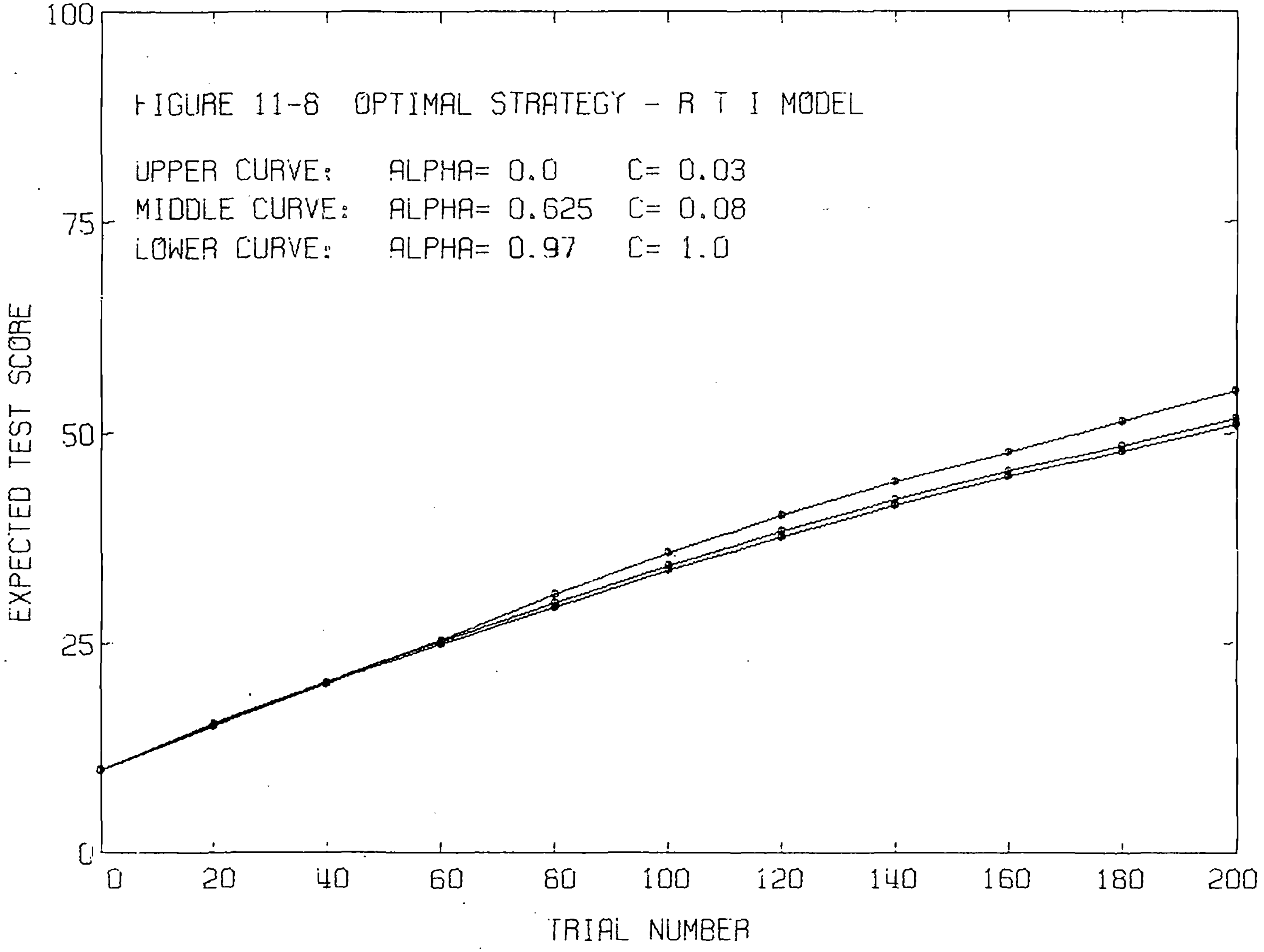


FIGURE 11-8 OPTIMAL STRATEGY - R T I MODEL

UPPER CURVE: ALPHA= 0.0 C= 0.03
MIDDLE CURVE: ALPHA= 0.625 C= 0.08
LOWER CURVE: ALPHA= 0.97 C= 1.0

extremes of the linear and the one-element models (middle curves). The best performance is obtained for equivalent values of α_R and c corresponding to the one-element model (upper curves). The results seem to indicate that the RTI model, using the estimators of Equations (7) and (8), becomes "more response-sensitive", in a sense, as the parameter values range from those corresponding to the linear model, on the response-insensitive extreme, to those corresponding to the one-element model on the response-insensitive extreme. Since each pair of α_R and c values for a given Figure (11-1 through 11-8) yields the same α value, the lower curve represents the results of applying the standard response-insensitive cyclical strategy for any of the three models.

C. Heuristic Search Techniques

It appears that optimal instruction-sequencing problems employing present models of learning may be fairly easily handled by the methods of Section III-B. Since exhaustive-search techniques such as Dynamic Prc ramming appear to be impractical, due to dimensionality constraints, the question arises as to approaches to the problem for models which cannot be handled algorithmically. In this regard, certain of the heuristic techniques developed in the field of Artificial Intelligence would appear to be applicable to the optimal instruction-sequencing problem. These techniques were developed specifically to provide an approach to search problems much too large to be handled by conventional search techniques. The particular heuristic method to be discussed with regard to possible application to the optimal instruction-sequencing problem is basically the Ordered-search Algorithm of Nilsson(1971, p.59). The search process will be described in the context of a state-space graph similar to that of Figure 4 of Section III-A. The process involves the concept of searching for a "goal node" in the graph, which in the present context could be taken either as any node at a depth m (m trials removed from the initial state, or "start node"), or as any node corresponding to a level of learning or conditioning at or above a certain criterion. If the search is conducted subject to the latter specification of "goal node", then "incremental path length" (the "cost", in a sense, of the path connecting two nodes) would be defined as 1, and the search for a minimum-length path to a goal node would correspond with the determination of the shortest sequence of stimuli necessary to achieve a specified level of learning. If a goal node is defined as any node at a depth corresponding to a specified number of trials, m, then incremental path length would be defined as some form of complement (such as n-ΔQ) of the incremental amount of learning which takes place between two nodes, and the search for a minimum-length path to a goal node in this case would correspond with the determination of the sequence of m stimuli which produces the highest level of learning. In either of these cases, the heuristic ordered-search algorithm illustrated in Figure 12 can be applied to determine a minimum-length path to a goal node.

The algorithm is self-explanatory except for a few comments regarding the heuristic function, $\hat{f}$. This function is a heuristically determined estimate of the length of a minimum-length path from the start node to a goal node constrained to pass through the node to which the $\hat{f}$ value to be calculated applies. The value of $\hat{f}$ at any node, n, is determined from

$$\hat{f}(n) = \hat{g}(n) + \hat{h}(n) \qquad (9)$$

1) Put the start node s on a list called OPEN and compute $\hat{f}(s)$

2) If OPEN is empty, exit with failure; otherwise continue

3) Remove from OPEN that node whose f value is smallest and put it on a list called CLOSED. Call this node n. (Resolve ties for minimal f values arbitrarily, but always in favor of any goal node)

4) If n is a goal node, exit with the solution path obtained by tracing back through the pointers; otherwise continue

5) Expand node n, generating all of its successors. (If there are no successors, go immediately to 2) For each successor n_i, compute $\hat{f}(n_i)$

6) Associate with the successors not already on either OPEN or CLOSED the $\hat{f}$ values just computed. Put these nodes on OPEN and direct pointers from them back to n

7) Associate with those successors that were already on OPEN or CLOSED the smaller of the $\hat{f}$ values just computed and their previous $\hat{f}$ values. Put on OPEN those successors whose $\hat{f}$ values were thus lowered, and redirect to n the pointers from all nodes whose $\hat{f}$ values were lowered

8) Go to 2

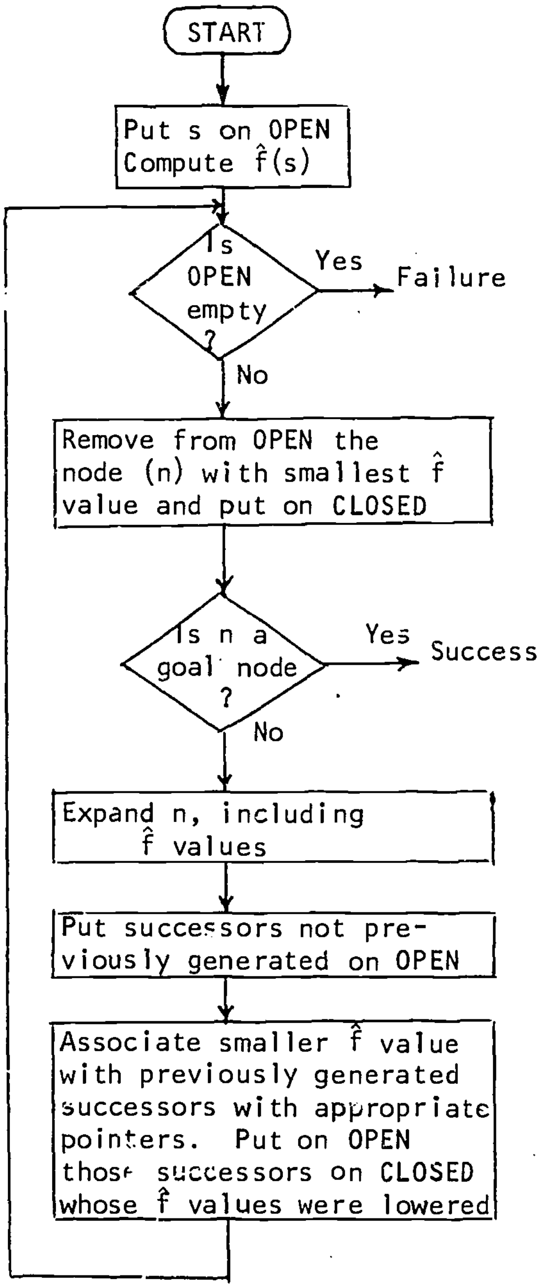

START
Put s on OPEN
Compute $\hat{f}(s)$
Is OPEN empty ?
Yes
Failure
No
Remove from OPEN the node (n) with smallest $\hat{f}$ value and put on CLOSED
Is n a goal node ?
Yes
Success
No
Expand n, including $\hat{f}$ values
Put successors not previously generated on OPEN
Associate smaller $\hat{f}$ value with previously generated successors with appropriate pointers. Put on OPEN those successors on CLOSED whose $\hat{f}$ values were lowered

Figure 12 Ordered-search Algorithm

where $\hat{g}(n)$ is an estimate of the length of a minimum-length path from the start node to node n, and $\hat{h}(n)$ is an estimate of the length of a minimum-length path from node n to a goal node. More formally, if $k(n_i,n_j)$ is the length of a minimum-length path between any two arbitrary nodes, n_i and n_j (it is possible, in general, for more than one path to exist between two nodes. There may also be no path between two nodes, in which case k is undefined), and if T is a set of goal nodes, then

$$h(n_i) = \min_{n_j \varepsilon T} k(n_i,n_j)$$

is the actual length of a minimum-length path between any node, n_i, and a goal node. The function $\hat{h}(n_i)$ is an estimate of this length. Similarly,

$$g(n_i) = k(s,n_i)$$

is the actual length of a minimum-length path between the start node, s, and any node, n_i, accessible from s. The function $\hat{g}(n_i)$ is an estimate of this length. More will be said presently about the practical determination of $\hat{g}$ and $\hat{h}$.

The expansion of nodes referred to in the algorithm of Figure 12 merely amounts to determining all possible successors to a node one step away from that node. In the context of the state-space formulation for the optimal instruction-sequencing problem, this amounts to determining all possible state vectors which can result in one step from the present state vector as a function of which stimuli are given. In order to avoid the problem of inordinately large state spaces encountered in Dynamic Programming applications, the generated successor nodes are stored in the form of a list, as indicated in the algorithm, rather than requiring storage of the entire state space. This means, of course, that the entire list must be searched for previously generated successors, as would be done in the tree formulation for a Dynamic Programming solution, but the portion of the entire graph which is searched when effective heuristics are employed is extremely small, so that computation time is not nearly as great a limitation as in a Dynamic Programming formulation.

In the progression of the algorithm, the fact that a value of $\hat{f}$ is to be determined for a given node means that at least one path has been determined between that node and the start node, Thus,

$\hat{g}(n)$ can be set equal to the length of the minumum-length path thus far determined between the start node and node n, with the guarantee that $\hat{g}(n) \geq g(n)$. It can be shown (cf. Nilsson, 1971, pp.59-65) that if

$$\hat{h}(n) \leq h(n) \qquad (10)$$

and if

$$\hat{h}(m) - \hat{h}(n) \leq k(m,n)$$

for any two nodes, m and n, and if there exists a minimum-length path between the start node and a goal node, the algorithm specified in Figure 12 will find this path. The second inequality is not actually necessary to guarantee that an optimal path will be found, but does guarantee that once the algorithm expands a node, an optimal path to that node has been found. This inequality is called the consistency assumption (the assumption that the inequality is satisfied) and is usually satisfied if the heuristic information used in determining $\hat{h}$ is applied consistently at all nodes.

The effectiveness of the search algorithm, in terms of computational power expended to find a goal node (a solution), depends critically on the heuristic function, $\hat{h}$. In effect, the requirement that (10) be satisfied guaratees that the solution is globally optimal. Selection of heuristics which give the largest value of $\hat{h}$ subject to 10 yields the solution with a minimum of computational effort (setting $\hat{h}=0$ corresponds to a complete absence of heuristic information and results in inefficient blind search). Relaxing the inequality constraint on $\hat{h}$ may yield a solution requiring even less computational effort, but forfeits the guarantee of global optimality. Nevertheless, such a solution could be useful.

As an example of the calculation of $\hat{h}$ for an optimal instruction-sequencing problem, consider the application of the ordered-search algorithm to a problem with the structure of Figure 1, using a model of learning which includes general stimulus interaction subject only to the restriction that application of any given stimulus is non-reinforcing to all but the corresponding component of the state vector. In other words, the effect of application of the r-th stimulus on any but the r-th q value would be to either increase it or leave it unchanged. Consider also that the search for a minimum-length path is taken in the context of a search for the shortest sequence of stimuli which will produce a given level of learning. Under these conditions, a possible heuristic for the determination of $\hat{h}$ would be to estimate the length of the minimum-length path without taking stimulus interaction into account. Due to the restriction on the interaction, $\hat{h}$ would certainly then constitute a lower bound on h, thus guaranteeing an optimal solution. If the properties of the learning model under these conditions were even further constrained, such as if q_i were a monotonic function, or were known to vary at less than some given rate, then the determination of $\hat{h}$ could be quite straightforward. The description in general terms of the calculation of a heuristic function is quite difficult, at best, since its determination is intrinsically related to the detailed structure of the particular problem to be solved. At thc same time, the determination of powerful heuristics is the crucial point in the effectiveness of heuristic search techniques. It must be left to further research into specific instruction-sequencing problems to determine how effective the heuristic paradigm may be in the solution of these problems.

Chapter IV - Conclusions

Of the three optimization techniques (exhaustive search, algorithmic, and heuristic search) investigated for the problem structure outlined in the Introduction, algorithmic methods seem the best suited for use with presently accepted models of paired-associate learning (specifically, the single-operator linear model, the one-element model, and the random-trial increments model). Dynamic Programming as a solution technique for problems involving these models is not only unnecessarily complex, but is fundamentally and severely limited in its applicability due to constraints of dimensionality, primarily in terms of computer fast-storage size requirements. Modified exhaustive techniques, such as State-increment Dynamic Programming, depth-first search, and last-stage search, may be used to alleviate memory-size constraints, but computation-time constraints still severely limit the size of problem which can be treated. Algorithmic methods of optimization, i.e., methods by which the optimal decision at each step may be specified immediately without the need for look-ahead search, appear to be sufficient for the optimization problem involving models with no stimulus interaction or time dependence. The optimal algorithm specified in Section III-B of the report, algorithm *A*, has been shown to be quite general in that the class of models to which it applies includes the linear, one-element, and RTI models as special cases. In addition, it is seen that the standard cyclical strategy, which is optimal for the linear model with uniform parameters, the optimal strategy of Karush & Dear (1966) for the one-element model with uniform parameters, and the Largest Immediate Gain strategy of Calfee (1970) are all special cases of algorithm *A*. An interesting by-product of the algorithm is the way in which it makes clear the fact that, for models without stimulus interaction, the actual *order* of presentation of stimuli is *immaterial*, i.e., a specification of the number of times each stimulus is to be presented is sufficient to construct an optimal sequence, and all such sequences are equivalent.

Monte Carlo simulations were conducted to determine expected test scores versus number of trials using the strategy specified by algorithm *A* in comparison with the standard cyclical strategy and a uniform random strategy for the three models used over a range of parameter values. For the linear model with uniform parameters, of course, the cyclical strategy *is* the optimal strategy, and a random variation in parameters was therefore introduced to separate the strategies. While the optimal strategy in this case did provide an advantage over the cyclical strategy, the advantage was fairly small, the maximum advantage being on the order of 5-10% in expected test score for a given number of trials. Corresponding advantages for the one-element model were much larger, the maximum advantage being on the order of 20-25% in test score, with advantages for intermediate response-sensitive RTI models lying inbetween. The conclusion to be drawn seems to be that optimal instruction sequencing strategies can be used to best advantage by response-sensitive models which make the most effective use of the additional statistical information contained in the response history of a subject. This seems only logical, but perhaps the most important point brought out by the simulations is the magnitude of the difference involved. To take a specific example, the uniform-parameter one-element model in response-insensitive mode is statistically equivalent to a uniform-parameter linear model with $\alpha=1-\theta$. The cyclical strategy applied to this one-element model is therefore the optimal *response-insensitive* strategy, and the simulations show that the optimal *response-sensitive* strategy applied to the same model can provide a maximum advantage of 20-25% in test score for a given number of trials, or 30-40% in the number of trials necessary to achieve a given score. These results would tend to indicate that the pursuit of optimal instruction-sequencing methods, including the models involved, which make effective use of the response history of a subject could be very worthwhile. In contrast with this view, the results of the experiment of Dear, et al. (1967), using the one-element model, would seem to indicate that response-sensitive optimization strategies are fairly ineffective in practice. In all likelihood, however, this is due primarily to the inadequacy of the one-element model, per se, to represent the learning process involved. The simulations of this experiment included in this investigation tend to support this conclusion. It remains to be seen whether or not more accurate models will result in effective optimal strategies in practice.

It is reasonable to expect that more complete models of the paired-associate learning process will be sufficiently complex to preclude the use of algorithmic methods such as algorithm *A*. A heuristic-search technique which has the capability of overcoming the dimensionality limitations of exhaustive-search techniques has been presented as a possible alternative approach to the optimization problem in such cases. Although its specification must necessarily be fairly general at present, particularly with regard to the heuristics involved, it is nevertheless quite likely that this approach could prove viable through future research when appropriate learning models are developed.

Chapter V - Recommendations

First of all, it is probable that further research into the application of exhaustive-search techniques, such as Dynamic Programming, to the optimal instruction-sequencing problem will prove relatively fruitless in practice, primarily due to severe inherent limitations of dimensionality. Algorithmic methods appear adequate for present learning models, but the models themselves appear to be inadequate from the standpoint of accurate representation of the learning process. Nevertheless, further research into algorithmic methods, such as algorithm A presented in Section III-B of this report, may be worthwhile, particularly from the standpoint of extending their applicability to broader classes of learning models. As more complete learning models are developed, it is likely that their complexity, particularly with regard to interrelationships among the factors in the learning process, will be sufficiently great to preclude a simple algorithmic approach to optimization. Heuristic methods similar to that presented in this report may then be the only viable approach to the problem, at least of the three approaches considered in this investigation. Further research into such heuristic methods, therefore, is definitely recommended.

Ultimately, of course, one of the most important practical applications of optimal instruction-sequencing could be automatic interactive optimization by computer in a CAI environment. Algorithmic optimization methods are normally very easily implemented in such a situation, which provides another good reason to pursue the investigation of such methods to the limits of their applicability. Even for more complex optimization tasks, heuristic methods give promise of being able to provide effective instruction-sequencing in interactive real-time CAI situations. Such methods should be pursued further in this regard, as well.

Bibliography

Atkinson, R. C., Bower, G. H., & Crothers, E. J. *An introduction to mathematical learning theory*. New York: Wiley, 1965.

Atkinson, R. C. Computerized instruction and the learning process. *American Psychologist*, 1968, *23*, 225-239.

Atkinson, R. C., & Paulson, J. A. An approach to the psychology of instruction. *Psychological Bulletin*, 1972, *78*, 49-61.

Bellman, R. *Dynamic programming*. Princeton:Princeton University Press, 1957.

Bellman, R. *Adaptive control processes*. Princeton: Princeton University Press, 1961.

Bellman, R., & Dreyfus, S. E. *Applied dynamic programming*. Princeton: Princeton University Press, 1962.

Bower, G. H. Application of a model to paired-associate learning. *Psychometrika*, 1961, *26*, 255-280.

Bush, R. R., & Sternberg, S. H. A single operator model. In R. R. Bush & W. K. Estes (Eds.), *Studies in mathematical learning theory*. Stanford: Stanford University Press, 1959. Pp. 204-214.

Calfee, R. C. The role of mathematical models in optimizing instruction. *Scientia: Revue Internationale de Synthèse Scientifique*, 1970, *105*, 1-25.

Dear, R. E. Solutions for optimal designs of stochastic learning experiments. Technical Report SP-1765/000/00. Santa Monica: System Development Corporation, 1964.

Dear, R. E., Silberman, H. F., Estavan, D. P. and Atkinson, R. C. An optimal strategy for the presentation of paired-associate items. *Behavioral Science*, 1967, *12*, 1-13.

Dreyfus, S. An appraisal of some shortest path algorithms. *Operations Research*, 1969, *17*, 395-412.

Estes, W. K. Learning theory and the new mental chemistry. *Psychological Review*, 1960, *67*, 207-223.

Groen, G., & Atkinson, R. C. Models for optimizing the learning process. *Psychological Bulletin*, 1966, *66*, 309-320.

Hart, P., Nilsson, N., & Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 1968, *SSC-4*, 100-107.

Karush, W., & Dear, R. E. Optimal stimulus presentation strategy for a stimulus sampling model of learning. *Journal of Mathematical Psychology*, 1966, *3*, 19-47.

Larson, R. E. *State increment dynamic programming*. New York: American Elsevier, 1968.

Matheson, J. Optimum teaching procedures derived from mathematical learning models. Unpublished doctoral dissertation, Stanford University, 1964.

Nilsson, N. *Problem-solving methods in artificial intelligence*. New York: McGraw-Hill, 1971.

Norman, M. F. Incremental learning on random trials. *Journal of Mathematical Psychology*, 1964, *2*, 336-350.

Pohl, I. Bi-directional and heuristic search in path problems. Doctoral dissertation, Stanford University, 1969.

Smallwood, R. D. *A decision structure for teaching machines*. Cambridge, Mass.: MIT Press, 1962.

Smallwood, R. D. The analysis of economic teaching strategies for a simple learning model. *Journal of Mathematical Psychology*, 1971, *8*, 285-301.

Appendix - Simulation Details

```
   DIMENSION Q(2,10),A(10),SC1(10),SC2(10)
   REAL AC(7)/.6,.7,.8,.85,.9,.95,.97/
   IX=7654321
   DO 3 K1=1,7
   AM=AC(K1)
   SD=0.4
   WRITE(3,20)AM,SD
   S1=0.
   S2=0.
   DO 5 J=1,200
   DO 1 I1=1,10
   DO 1 I2=1,2
1  Q(I2,I1)=0.9
   DO 6 K2=1,10
8  AB=0.
   DO 9 I=1,12
   CALL RANDU(IX,IY,YFL)
   IX=IY
9  AB=AB+YFL
   V=(AB-6.0)*SD+AM
   IF((V.GT.1.).OR.(V.LT.(2.*AM-1.)))GO TO 8
6  A(K2)=V
   IC=0
   DO 4 K=1,200
   IC=IC+1
   IF(IC.GT.10)IC=1
   IND=IC
   Q(1,IND)=Q(1,IND)*A(IND)
   DMAX=Q(2,1)-Q(2,1)*A(1)
   IMAX=1
   DO 7 I=2,10
   DQ=Q(2,I)-Q(2,I)*A(I)
   IF(DQ.LE.DMAX)GO TO 7
   DMAX=DQ
   IMAX=I
7  CONTINUE
   Q(2,IMAX)=Q(2,IMAX)*A(IMAX)
   IN=K/20
   IF(IN*20.NE.K)GO TO 4
```

Figure A-1 Simulation of Optimization for Linear Model with Normally Distributed α-values

```
      DO 5 L=1,10
      SC1(IN)=SC1(IN)+Q(1,L)
      SC2(IN)=SC2(IN)+Q(2,L)
    4 CONTINUE
    5 CONTINUE
      DO 2 I=1,10
      T1=100.-SC1(I)/20.
      T2=100.-SC2(I)/20.
    2 WRITE(3,21)M,T1,T2
    3 CONTINUE
   20 FORMAT(//5X,3HAM=,F4.2,5X,3HSD=,F7.5/)
   21 FORMAT(5X,I3,2(5X,F5.1))
      STOP
      END
```

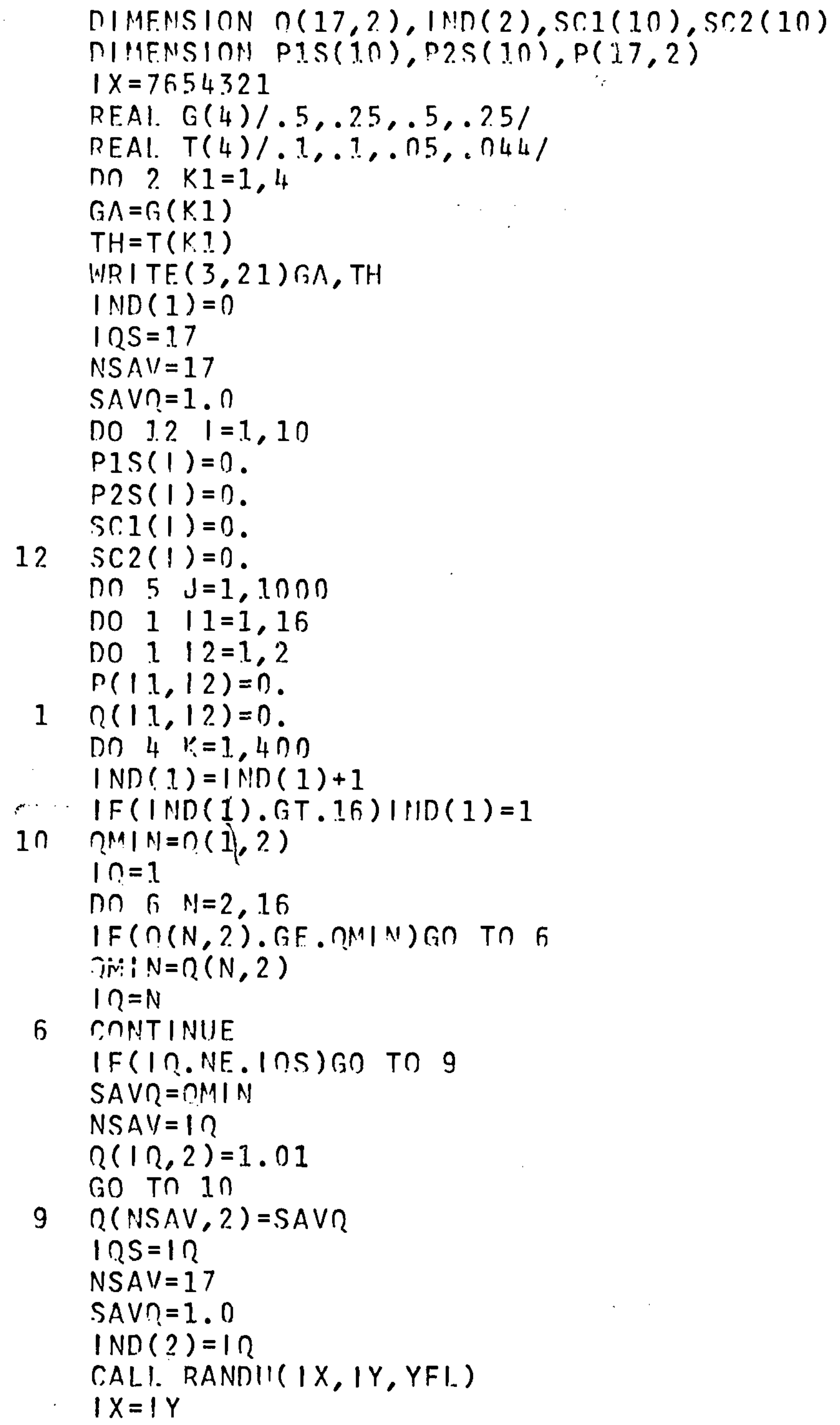
Figure A-1 (continued)

```
      DIMENSION Q1(10),Q2(10),GF(10),SC(10),F(10)
      IX=5173
      REAL APT(7)/.111,.143,.2,.25,.333,.5,.625/
      REAL CT(7)/.45,.35,.25,.2,.15,.1,.08/
      DO 10 K1=1,7
      AR=APT(K1)
      C=CT(K1)
      A=AR*C+1.-C
      G=(AR*AR*C+1.-C)/A
      V12=0.9*A
      WRITE(3,21)AR,C,A
      DO 2 I=1,10
 2    SC(I)=0.0
      DO 12 J=1,1000
      DO 1 I=1,10
      GF(I)=V12
      F(I)=V12
      Q1(I)=0.9
 1    Q2(I)=0.9
      DO 4 K=1,200
      QMAX=Q2(1)
      IMAX=1
      DO 6 I=2,10
      IF(Q2(I).LE.QMAX)GO TO 6
      QMAX=Q2(I)
      IMAX=I
 6    CONTINUE
      CALL RANDU(IX,IY,YFL)
      IX=IY
      IF(YFL.GT.Q1(IMAX))GO TO 8
      Q2(IMAX)=GF(IMAX)
      GO TO 9
 8    Q2(IMAX)=(A-GF(IMAX))*(F(IMAX)/(1.-F(IMAX)))
 9    GF(IMAX)=GF(IMAX)*G
      F(IMAX)=F(IMAX)*A
      CALL RANDU(IX,IY,YFL)
      IX=IY
      IF(YFL.GT.C)GO TO 13
      IF(Q1(IMAX).GT.1E-20)Q1(IMAX)=Q1(IMAX)*AR
 13   IND=K/20
      IF(IND*20.NE.K)GO TO 4
      DO 5 K2=1,10
 5    SC(IND)=SC(IND)+Q1(K2)
```

Figure A-2 Simulation of Optimization for RTI Model

```
 4   CONTINUE
12   CONTINUE
11   DO 10 I=1,10
     M=20*I
     TS=100.-SC(I)/100.
10   WRITE(3,20)M,TS
20   FORMAT(5X,I3,5X,F5.1)
21   FORMAT(//5X,3HAR=,F5.3,3X,2HC=,F5.3,3X,2HA=,F5.3/)
     END
```

Figure A-2 (continued)



```
      DIMENSION Q(17,2),IND(2),SC1(10),SC2(10)
      DIMENSION P1S(10),P2S(10),P(17,2)
      IX=7654321
      REAL G(4)/.5,.25,.5,.25/
      REAL T(4)/.1,.1,.05,.044/
      DO 2 K1=1,4
      GA=G(K1)
      TH=T(K1)
      WRITE(3,21)GA,TH
      IND(1)=0
      IQS=17
      NSAV=17
      SAVQ=1.0
      DO 12 I=1,10
      P1S(I)=0.
      P2S(I)=0.
      SC1(I)=0.
   12 SC2(I)=0.
      DO 5 J=1,1000
      DO 1 I1=1,16
      DO 1 I2=1,2
      P(I1,I2)=0.
    1 Q(I1,I2)=0.
      DO 4 K=1,400
      IND(1)=IND(1)+1
      IF(IND(1).GT.16)IND(1)=1
   10 QMIN=Q(1,2)
      IQ=1
      DO 6 N=2,16
      IF(Q(N,2).GE.QMIN)GO TO 6
      QMIN=Q(N,2)
      IQ=N
    6 CONTINUE
      IF(IQ.NE.IQS)GO TO 9
      SAVQ=QMIN
      NSAV=IQ
      Q(IQ,2)=1.01
      GO TO 10
    9 Q(NSAV,2)=SAVQ
      IQS=IQ
      NSAV=17
      SAVQ=1.0
      IND(2)=IQ
      CALL RANDU(IX,IY,YFL)
      IX=IY
```

Figure A-3 Simulation of Experiment of Dear, et al. (1967)

```
      IF(YFL.LT.TH)P(IND(1),1)=1.
      ID=IND(2)
      IF(P(ID,2).EQ.1.)GO TO 8
      CALL RANDU(IX,IY,YFL)
      IX=IY
      IF(YFL.LT.GA)GO TO 8
      Q(ID,2)=TH
      GO TO 7
  8   QT=Q(ID,2)
      X=GA*(1.-QT)
      Q(ID,2)=(QT+X*TH)/(QT+X)
  7   CALL RANDU(IX,IY,YFL)
      IX=IY
      IF(YFL.LT.TH)P(ID,2)=1.
      IN=K/40
      IF(IN*40.NE.K)GO TO 4
      P1=1.
      P2=1.
      DO 11 L=1,16
      P1=P1*(GA+P(L,1)*(1.-GA))
      P2=P2*(GA+P(L,2)*(1.-GA))
      SC1(IN)=SC1(IN)+(1.-P(L,1))*(1.-GA)
 11   SC2(IN)=SC2(IN)+(1.-P(L,2))*(1.-GA)
      P1S(IN)=P1S(IN)+P1
      P2S(IN)=P2S(IN)+P2
  4   CONTINUE
  5   CONTINUE
      DO 2 I=1,10
      M=40*I
      TS1=16.-SC1(I)/1000.
      TS2=16.-SC2(I)/1000.
      PS1=P1S(I)/1000.
      PS2=P2S(I)/1000.
  2   WRITE(3,20)M,TS1,TS2,PS1,PS2
 20   FORMAT(5X,I3,4(5X,F8.5))
 21   FORMAT(//5X,6HGAMMA=,F4.3,2X,6HTHETA=,F4.3/)
      STOP
      END
```

Figure A-3 (continued)

```
      SUM1=0.
      SUM2=0.
      DO 8 J=1,K3
      S1=0.
      S2=0.
      DO 1 I=1,10
      Q1(I)=1.
 1    Q2(I)=1.
      DO 4 K=1,M
      CALL RANDU(IX,IY,YFL)
      IX=IY
      IND=YFL*10.+1
      CALL RANDU(IX,IY,YFL)
      IX=IY
      IF(YFL.LT.C)Q1(IND)=Q1(IND)*ALPH
      QMAX=Q2(1)
      IMAX=1
      DO 6 I=2,10
      IF(Q2(I).LE.QMAX)GO TO 6
      QMAX=Q2(I)
      IMAX=I
 6    CONTINUE
      CALL RANDU(IX,IY,YFL)
      IX=IY
 4    IF(YFL.LT.C)Q2(IMAX)=Q2(IMAX)*ALPH
      DO 5 L=1,10
      S1=S1+Q1(L)
 5    S2=S2+Q2(L)
      SUM1=SUM1+S1
      SUM2=SUM2+S2
      S(J,1)=100.-S1*10.
      S(J,2)=100.-S2*10.
 8    CONTINUE
      SM1=100.-SUM1/K4
      SM2=100.-SUM2/K4
      X1=0.
      X2=0.
      DO 9 I=1,K3
      X1=X1+(SM-S(I,1))**2
 9    X2=X2+(SM-S(I,2))**2
      SIG1=SQRT(X1/(FK3*(FK3-1)))
      SIG2=SQRT(X2/(FK3*(FK3-1)))
      PS1=SIG1*100/SM1
      PS2=SIG2*100/SM2
      WRITE(3,25)SIG1,SIG2
      WRITE(3,26)PS1,PS2
```

Figure A-4 Routine for Determining Confidence Levels

Table A-1 Frequency Distribution for Pseudo-random Number Generator (RANDU - IBM 370/155) 100,000 Samples

0.0 - 0.01	980	0.50 - 0.51	1034
0.01 - 0.02	1030	0.51 - 0.52	984
0.02 - 0.03	1045	0.52 - 0.53	998
0.03 - 0.04	994	0.53 - 0.54	974
0.04 - 0.05	974	0.54 - 0.55	990
0.05 - 0.06	1053	0.55 - 0.56	981
0.06 - 0.07	986	0.56 - 0.57	1006
0.07 - 0.08	977	0.57 - 0.58	1036
0.08 - 0.09	1021	0.58 - 0.59	1003
0.09 - 0.10	980	0.59 - 0.60	1004
0.10 - 0.11	1031	0.60 - 0.61	980
0.11 - 0.12	1028	0.61 - 0.62	1018
0.12 - 0.13	1047	0.62 - 0.63	976
0.13 - 0.14	993	0.63 - 0.64	1040
0.14 - 0.15	1019	0.64 - 0.65	1036
0.15 - 0.16	1019	0.65 - 0.66	1009
0.16 - 0.17	930	0.66 - 0.67	1004
0.17 - 0.18	1041	0.67 - 0.68	957
0.18 - 0.19	958	0.68 - 0.69	1008
0.19 - 0.20	974	0.69 - 0.70	1002
0.20 - 0.21	973	0.70 - 0.71	1021
0.21 - 0.22	1037	0.71 - 0.72	977
0.22 - 0.23	1008	0.72 - 0.73	963
0.23 - 0.24	1015	0.73 - 0.74	994
0.24 - 0.25	949	0.74 - 0.75	926
0.25 - 0.26	1029	0.75 - 0.76	1027
0.26 - 0.27	956	0.76 - 0.77	986
0.27 - 0.28	1006	0.77 - 0.78	985
0.28 - 0.29	1007	0.78 - 0.79	979
0.29 - 0.30	980	0.79 - 0.80	1029
0.30 - 0.31	1004	0.80 - 0.81	1043
0.31 - 0.32	1029	0.81 - 0.82	974
0.32 - 0.33	985	0.82 - 0.83	1016
0.33 - 0.34	1014	0.83 - 0.84	1011
0.34 - 0.35	1046	0.84 - 0.85	1074
0.35 - 0.36	1011	0.85 - 0.86	990
0.36 - 0.37	983	0.86 - 0.87	997
0.37 - 0.38	1045	0.87 - 0.88	1029
0.38 - 0.39	962	0.88 - 0.89	974
0.39 - 0.40	1019	0.89 - 0.90	969
0.40 - 0.41	1014	0.90 - 0.91	969
0.41 - 0.42	1024	0.91 - 0.92	942
0.42 - 0.43	999	0.92 - 0.93	1015
0.43 - 0.44	981	0.93 - 0.94	1013
0.44 - 0.45	945	0.94 - 0.95	991
0.45 - 0.46	976	0.95 - 0.96	975
0.46 - 0.47	924	0.96 - 0.97	986
0.47 - 0.48	997	0.97 - 0.98	987
0.48 - 0.49	1067	0.98 - 0.99	993
0.49 - 0.50	1041	0.99 - 1.00	1019