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

ANALYSIS OF THE BEHAVIOR OF A CLASS OF GENETIC ADAPTIVE SYSTEMS

Kenneth Alan De Jong

August 1975

Technical Report No. 185

with assistance from:

National Aeronautics and Space Administration
Grant No. NSG-1176
Langly Research Center
Hampton, Virginia

ABSTRACT

AN ANALYSIS OF THE BEHAVIOR OF A CLASS
OF GENETIC ADAPTIVE SYSTEMS

by

Kenneth Alan De Jong

Chairman: John H. Holland

This thesis is concerned with the design and analysis of adaptive systems, particularly in the area of adaptive computer software. To that end a formalism for the study of adaptive systems is introduced and, within this framework, a means of evaluating the performance of adaptive systems is defined. The central feature of the evaluation process is robustness: the ability of an adaptive system to rapidly respond to its environment over a broad range of situations. To provide a concrete measure of robustness, a family E of environmental response surfaces was carefully chosen to include a wide variety of surfaces, including multimodal and discontinuous ones. The performance of an adaptive system is evaluated over E by computer simulation by monitoring two distinct performance curves: on-line and off-line performance. With on-line performance every response of the adaptive system is evaluated, reflecting situations in which an adaptive system is used to dynamically improve the performance of a system. With off-line performance only responses which improve performance are evaluated, reflecting situations

in which testing can be done independently of the system being controlled.

Within this evaluation framework, a class of genetic adaptive systems is introduced for analysis and evaluation. These artificial genetic systems, called reproductive plans, generate adaptive responses by simulating the information processing achieved in natural systems by means of the mechanisms of heredity and evolution. This is accomplished internally by maintaining a population of individuals whose "genetic" material specifies a particular point on the response surface. New individuals (responses) are produced by simulating population development via mating rules, production of offspring, mixing of genetic material, and so on.

Even the most elementary genetic adaptive plan is shown to produce performance on E which is superior to pure random search of the response surfaces. However, these elementary genetic plans were shown to be easily affected by stochastic side-effects resulting from internal random processes. By suitable adjustments in parameters and modifications to the basic genetic plan, a considerable improvement in the performance was achieved on E.

As a final point of comparison, the performance of two standard function optimization techniques was evaluated on E. Their performance is shown to be superior on the continuous quadratic-like functions for which they were

designed. However, the genetic plans are shown to be superior on the discontinuous and multimodal surfaces, suggesting that genetic plans hold a valid position between specialized local adaptive techniques and pure random search.

AN ANALYSIS OF THE BEHAVIOR OF A CLASS OF GENETIC ADAPTIVE SYSTEMS

by
Kenneth Alan De Jong

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer and Communication Sciences)
in the University of Michigan
1975

Doctoral Committee:

Professor John H. Holland, Chairman
Associate Professor Larry K. Flanigan
Associate Professor Richard A. Volz
Associate Professor Bernard P. Zeigler

ACKNOWLEDGEMENTS

I would like to thank those persons who made this research not only possible, but also worthwhile and enjoyable. A special note of appreciation to John H. Holland whose enthusiasm for and insight into genetic algorithms provided a constant source of encouragement; to my committee members Larry K. Flanigan, Richard A. Volz, and Bernard P. Zeigler for their time and interest; and especially to my wife, Ruth, who chose to remain in that status even though called upon to be typist, editor, mother, housewife, teacher, and companion during this period.

This research was partially supported by the National Aeronautics and Space Administration under grant NSG 1176. The computer simulations were done on the excellent facilities provided by the University of Michigan Computing Center.

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS

Chapter 1

FORMAL ADAPTIVE SYSTEMS

1.1 Introduction

The adjective "adaptive" is frequently encountered in the highly scientific and technological age in which we live. We read of sophisticated radar guidance systems which are capable of adapting quickly to changes in terrain and thus permit high-speed low-altitude flying. The space program has focused our attention on the need for machines which are flexible enough to adapt their responses to unexpected environmental factors. Artificial intelligence research has generated complex game-playing computer programs which have learned by experience to play better than their authors. Biologists continue to study the fascinating adaptive capabilities of organisms as simple as bacteria and as complex as man.

The question as to what constitutes an adaptive system has been widely debated, most recently in Tsypkin's survey of control theory (1971). This debate will not be continued here; rather, a broad view of what constitutes an adaptive system will be adopted, a view succinctly stated by Tsypkin (1971, p. 45):

> ... the most characteristic feature of adaptation is an accumulation and slow usage of the current information to eliminate the uncertainty due to insufficient a priori information and for the purpose of optimizing a certain selected performance index.

Tsypkin has focused his attention on artificial adaptive systems used in control theory. Holland (1975), on the other hand, has been studying the characteristics of both natural and artificial adaptive systems from this broad viewpoint. Out of this work has come a formal framework for describing, analyzing, and comparing adaptive systems. This framework is the basis for the formal definition of adaptive systems used in this thesis. However, before presenting the formalism, let us consider some examples of problems which are candidates for an adaptive solution.

1.2 Some Problems for Adaptation

I am particularly interested in the application of adaptive system theory to the problem of adaptive software design. This bias will show itself in the choice of examples and applications discussed in this thesis. The reader is reminded that the adaptive system theory presented here is limited in its application only by the imagination, and he is encouraged to consider examples from his own experience.

1.2.1 Data Structure Design

Suppose we are faced with designing a data structure for a generalized information-retrieval system. If it is really intended to be general purpose, the characteristics of input data sets are unknown at design time. A standard approach is to assume random input and choose the data

structure which minimizes some performance criterion for a standard set of data structure operations (e.g. search, delete, insert). Unfortunately, many applications consist of distinctly non-random input resulting in sub-optimal performance. An adaptive approach would explore the possibility of deferring the choice of a specific data structure until the characteristics of a particular data set are available in order to enhance on-line performance.

1.2.2 Algorithm Design

Suppose we are faced with designing a sophisticated time-sharing system which supports a large number of batch and terminal users simultaneously. The heart of such a system is a supervisor program which is responsible for sharing limited system resources among competing processes. The performance of a time-sharing system (usually specified in terms of terminal response time, batch throughput, and system overhead) is directly affected both by the algorithms chosen for resource sharing and by the demand characteristics for system resources. Unfortunately, the demand characteristics can vary widely from day to day and are often difficult to predict. A standard approach is to base resource sharing algorithms on average demand characteristics and hence obtain good performance "on the average". An adaptive approach would explore the possibility of modifying resource sharing algorithms in response to current demand characteristics (see, for example,

Bauer (1974)).

1.2.3 Game-playing Programs

Some of the most fascinating aspects of software design have arisen in the area of game-playing programs. Credible systems have been developed for playing games as complex as checkers and chess. The difficulty in designing such programs lies in our inability to specify a winning strategy in a precise algorithmic way. The standard approach has been to specify as precisely as possible the strategies used by expert players. Unspecified parameters (of which there are many) are externally "tuned" during development by observing their effects on performance. This approach has led to the development of several good chess-playing programs. An adaptive approach would explore not only the possibility of self-tuning programs, but also the possibility of strategy-generating systems.

1.2.4 Two-armed Bandits

Two-armed bandit problems arise in the context of statistical decision theory, but have considerable bearing on the problem of adaptation. In its simplest form, the problem is stated as follows: you are presented with two slot-machines, one of which pays better than the other. If you are unaware of which is the better-paying machine, what strategy would you use to minimize your expected losses over N trials? The optimal (but alas, non-realiz-

able) strategy is to play the better-paying machine all the time. Lacking this a priori information, the problem becomes one of minimizing the expected number of trials to the lower-paying machine. Each trial yields more information about the relative performances of the two machines. The goal is to exploit this information as quickly and efficiently as possible. It should be clear by now that two-armed bandit problems capture adaptation in its simplest form: the dynamic gathering and exploitation of information to reduce uncertainty and improve performance.

1.3 A Formal Framework

With these examples in mind, we now ask what are the essential characteristics of adaptation. It has already been suggested that a problem in adaptation arises out of a lack of a priori information which prevents one from choosing between competing alternative solutions to the problem. Implicit in the idea of competing solutions is a measure of performance used to compare alternative solutions. The performance of a solution is a function both of its own characteristics and the particular environment in which it is tested. Adaptation consists of a strategy for generating better-performing solutions to the problem by reducing the initial uncertainty about the environment via feedback information made available during the evaluation of particular solutions.

Holland has been studying the properties of both natural and artificial systems. Out of these studies has come a formalism for representing problems in adaptation which will be used in this thesis. Briefly, a problem in adaptation is formally represented as:

E: the set of environments to be faced.

A: a set of structures describing alternative solutions to the problem.

U: a performance measure for evaluating solutions in a particular environment, i.e.

$U: A \times E \rightarrow R$ (R representing the real line)

I: a feedback function providing dynamic information to the adaptive system about the performance of a particular solution in a particular environment, i.e.

$I: A \times E \rightarrow R^n$

S: the collection of adaptive strategies under study. Each $s \in S$ is a strategy for generating better-performing solutions based on feedback information from previous trial solutions, i.e.

$$s: \left[\left\{ (A(t), I(t)) \right\}_{t=1}^{T} \right] \rightarrow A$$

X: the criterion used for comparing the performances of adaptive strategies, i.e.

$X: S \rightarrow R$

As an example of the formalism, consider how one might formally represent the previously discussed problem of

choosing data structures for an information retrieval system:

E: the set of all possible input data sets.

A: the set of alternative data structures.

U: the performance of the information-retrieval system on a particular data set.

I: data structure performance statistics (e.g. search, insert, delete timings).

S: alternative strategies for changing data structures based on input data set characteristics.

X: usually U averaged over random samples from E.

1.4 The Problem of Function Optimization

In this section we will consider the close relationship between the problems of adaptation and function optimization. Function optimization is a well-studied problem in applied mathematics and is briefly stated as follows: given a function f: $A \rightarrow R$, find those points in A on which f takes its maximum (minimum) values. To see its relationship to the problem of adaptation, consider again the formalism discussed above. The performance measure U: $A \times E \rightarrow R$ is more precisely the composition of two functions, a behavioral function B: $A \times E \rightarrow R^n$ specifying the behavioral characteristics of a particular solution in a particular environment, and a metric function M: $R^n \rightarrow R$ specifying the performance rating associated with behavioral characteristics. We

can further emphasize the role of the environment by considering a family of behavioral functions $\{b_e\}_{e \in E}$, where each b_e is simply the restriction of B to A x $\{e\}$. In this way adaptation can be viewed as attempting to optimize the performance measure $u_e : A \rightarrow R$ associated with a particular environment $e \in E$ and defined by $u_e(a) = M(b_e(a))$. The difficulty of the problem of adaptation (i.e. the initial uncertainty) can then be expressed in terms of the richness of the set $\{u_e\}_{e \in E}$ of performance measures. Because of this close relationship between the two problems it is worth considering the applicability of function optimization theory to the problem of adaptive system design.

Function optimization theory is generally divided into two areas: constrained and unconstrained problems. The tractability of a constrained problem is often highly dependent on the complexity of the constraints; finding the maximum is often eclipsed by the problem of staying within the constraints. From an adaptive systems point of view, the problem of constraints can be subsumed in the definition of the representation space A and the performance measures $\{u_e\}$. For example, a complexly constrained space H can be embedded in a simply constrained space A with u_e defined to take on its minimal value on A-H. For these reasons we will restrict our attention to unconstrained problems which, as far as any implementation is concerned, are really linearly constrained problems where

the constraints are of the form

$$1(i) \leq x(i) \leq h(i),\ i = 1, \ldots, n.$$

A second observation restricts our attention even further. It is the case that the performance measure u_e is almost never available in analytic form. Recall from above that u_e is really the composition of b_e and M. While M is often explicitly expressed in analytic form, b_e, the behavior function, is generally only a "black box" representing the complexity of the problem under adaptation. This observation immediately rules out classical analytic techniques and those iterative techniques which depend on exact expressions for first and possibly second order partial derivatives.

A third observation, and perhaps the most critical as far as the applicability of function optimization theory is concerned, is the fact that for problems of any complexity the behavioral function b_e (and hence in general u_e) is a high-dimensional, non-linear, multimodal function. As a consequence standard optimization techniques which assume linearity or unimodality, or techniques whose computation time grows rapidly with dimensionality are generally inapplicable to the adaptation problem.

With these constraints we are left with only a few alternative optimization techniques. The most commonly proposed search technique for multimodal functions is to run one's favorite local (unimodal) optimizer repeatedly using random starting points, the assumption being

that each local maximum will be encountered after a sufficient number of trials. Alternate approaches perform some type of patterned search over the whole space looking for likely areas in which the local optimizer should be employed. Finally, for problems of high dimensionality, several authors (see, for example, Rastrigin (1963) or Schumer & Steiglitz (1968)) have recommended reverting to various forms of random search.

Whether or not these techniques produce the kind of adaptive performance we would like is at this point an open question which will be explored further in this thesis. Comparisons of function optimizers center around the number of function evaluations required to find the optimum within a certain tolerance. The emphasis here is on convergence. In contrast, adaptation is also concerned with the quality of interim performance, the criterion often involving the integral of the performance curve.

1.5 A Reduction in Scope

Having stated and explored the general framework for problems in adaptation, we will now focus our attention on a specific class of adaptive systems which will be the object of this study.

In the first place, we will consider only discrete time-scale adaptive systems. A time step generally consists of generating, testing, and receiving feedback about a particular solution to the problem. The inter-

pretation of a time step is, of course, application-dependent.

Secondly, we will be concerned with the design of adaptive systems in which the only available feedback is the value of the performance measure u_e. Such systems are usually termed "first-order" feedback systems in the sense that the very minimal feedback information available about the behavior of a particular solution is its performance rating.

Finally, we will restrict our attention to two adaptive system performance criteria (X and X^* defined below). The motivation for these criteria arises from the concept of robustness. We say that an adaptive system is robust if it is able to generate and maintain acceptable solutions to a problem across a wide variety of environments. In order to formalize this concept, consider first the definition of local robustness, i.e. the ability of a strategy to generate and maintain acceptable solutions to a problem in a particular environment. Two such measures will be used in this thesis: local on-line performance and local off-line performance. On-line performance $x_e : S \rightarrow R$ will be defined as follows:

$$x_e(s) = \frac{1}{\sum_{t=1}^{T_e} c_t} * \sum_{t=1}^{T_e} c_t \cdot u_e(a_t)$$

That is, the performance of strategy s in environment e is a weighted average of the performances $u_e(a_t)$ of the generated solutions a_t over a time period T_e. On-line performance measures are motivated by situations in which adaptive systems are being used to dynamically improve the overall performance of an on-line system such as a time-sharing system. In such situations every new solution generated by the adaptive system for testing is included in the overall performance rating of the system.

In contrast, local off-line performance $x^* : S \rightarrow R$ will be defined as:

$$x_e^*(s) = \frac{1}{\sum_{t=1}^{T_e} c_t} * \sum_{t=1}^{T_e} c_t \cdot u_e^*(a_t)$$

where $u_e^*(a_t) \stackrel{d}{=} \min \left\{ u_e(a_1), \ldots, u_e(a_t) \right\}$. Off-line performance is motivated by situations in which the testing and evaluation of solutions is done off-line and is not included in the overall performance evaluation. In these situations the on-line system runs with the best solution generated to that point while off-line adaptation is continuing. Off-line performance is much closer to the standard measure of performance for function optimizers. The magnitude of trial errors is not included; only progress toward the minimum is

measured. As a consequence, off-line performance places heavier emphasis on convergence while on-line performance emphasizes initial performance.

In both cases, the weights c_t provide a means of shifting the emphasis. If they are increasing ($c_t < c_{t+1}$), more emphasis is placed on convergence. If they are decreasing ($c_t > c_{t+1}$), more emphasis is placed on initial performance. For our purposes $c_t=1$ for all t is sufficient.

Global robustness is now defined in terms of these local measures. On-line performance X: S $\rightarrow$R is given by:

$$X(s) = \frac{1}{\sum_{E} w_e} * \sum_{E} w_e \cdot x_e(s)$$

Off-line performance is similarly given by:

$$X^*(s) = \frac{1}{\sum_{E} w_e} * \sum_{E} w_e \cdot x_e^*(s)$$

In both cases, the weights w_e can be used to assign relative difficulties to the alternative environments.

1.6 Summary

In this chapter we have attempted to define formally what we mean by a problem in adaptation and have dis-

cussed some practical examples of such problems. We have noted the close relationship between the problems of adaptation and function optimization, and we have seen that the bulk of optimization techniques is not generally applicable to the design of adaptive systems. Finally, we have defined the specific class of adaptive systems which will be the subject of further study in the following chapters.

Chapter 2

GENETIC ADAPTIVE MODELS

2.1 Introduction

In the discussion of function optimization theory in chapter 1, we noted that, although the problems of adaptation and optimization are closely related, most of the standard optimization techniques are inadequate for adaptive problems of any complexity. This inadequacy can be viewed as an inability to process information relating to global aspects of the function to be optimized. Extremely efficient techniques have been developed for finding the nearest local maximum of a function; however, attempts to extend these techniques to find global maxima have met with little success. Some global search techniques have been proposed for low-dimensional problems (see, for example, Hill (1969) or Bremermann (1970)), their computation time growing rapidly with dimensionality. As a consequence, most global searching is accomplished with some form of random search. From an adaptive system point of view, random search is extremely inefficient because it makes no use of the available feedback information to reduce the initial uncertainty surrounding the problem for adaptation. These observations suggest a critical question for adaptive system design: are there efficient ways to exploit global information about a problem in order to generate better-performing solutions?

This thesis is part of a larger research project which is attempting to answer such questions under the direction of John Holland at the University of Michigan. The basic point of view of this research is that nature is an extremely rich source of examples of sophisticated information processing and adaptation. The goal of this project is to understand and abstract from natural systems the mechanisms of adaptation in order to design artificial systems of comparable sophistication. This research has centered around the design of artificial systems derived from standard models of heredity and evolution in the field of population genetics which we will briefly review.

2.2 Genetic Population Models

Population genetics is concerned with the characteristics of heredity and evolution at the population level. It assumes a Mendelian view of the mechanisms of heredity, i.e. genetic material is represented as strands of chromosomes consisting of genes which control observable properties in the individuals making up the population. A population is viewed as a dynamic pool of genetic information, the characteristics of which change from generation to generation in response to environmental factors. Numerous examples exist which demonstrate the ability of a population of organisms to adapt over a period of generations to complex changes in its environment. The goal is to explain these observable adaptations in terms of

the mechanisms of heredity and evolution.

In a genetic population model, individuals are represented purely in terms of their genetic makeup. Representations of genetic material vary from simple one-chromosome individuals (haploid models) to complex multi-chromosome individuals (polyploid models). Having specified a representation for genetic material, the observable characteristics of an individual are defined as functions of the chromosomal genes. Environmental pressures, specified in terms of these observable characteristics, assign a measure of "fitness" to an individual. Finally, the dynamics of population development are defined in terms of fitness, life-death cycles, mating rules, mobility, sex, species, and so on.

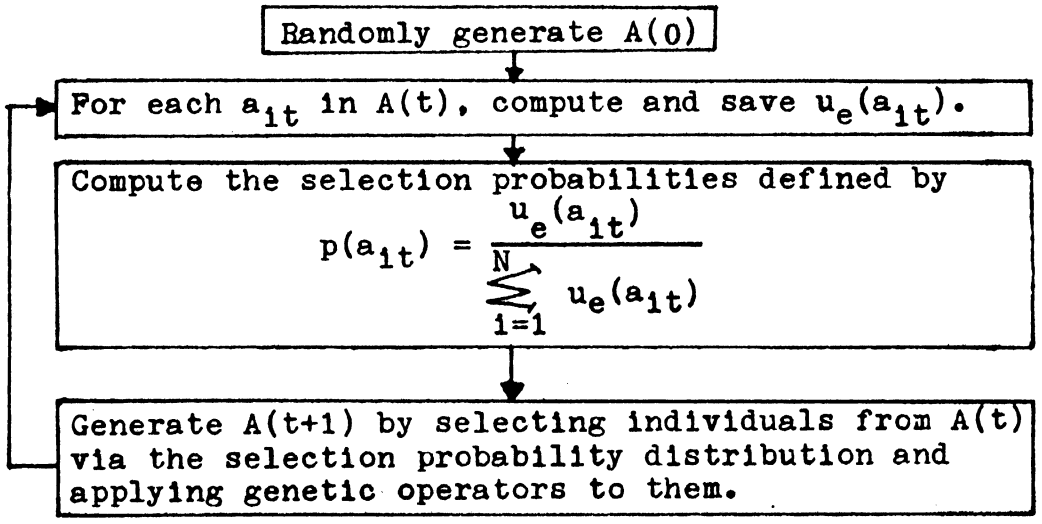
We, of course, are not concerned with modeling the development of biological populations per se; rather, we are concerned with understanding the mechanisms of adaptation which provide for such development. Unfettered by biological facts, we are free to construct artificial systems which capture the essence of these mechanisms. The exciting aspect of this approach, as we will see, is that even very simple artificial systems exhibit considerable adaptive capabilities.

2.3 Reproductive Plans

In this section we will describe the basic class of artificial adaptive systems which has arisen from the

genetic population models. This class of adaptive systems, called reproductive plans, was first proposed by Holland; subsequent variations have been studied by others (see, for example, Caviccio (1970), Hollstien (1971), Frantz (1972)).

Recall from the formalism introduced in chapter 1 that alternative solutions to the problem for adaptation are represented by the set A. In a reproductive plan, the memory of the system at time t consists of a population A(t) of N individuals a_{it} from A together with their associated performance ratings $u_e(a_{it})$. These representations a_{it} of solutions to the problem for adaptation are considered the genetic material to be processed by a reproductive plan. New individuals (and hence new alternative solutions) are produced by simulating genetic population dynamics. That is, individuals from A(t) are selected as parents and idealized genetic operators are applied to produce offspring. More specifically, a reproductive plan operates as follows:



To get a feeling for how reproductive plans work, note that the expected number of offspring produced by an individual is proportional to its performance. This can be seen by considering the process of selecting individuals for reproduction as N samples from A(t) with replacement using the selection probability distribution. Hence, the expected number of offspring from individual a_{it} is given by

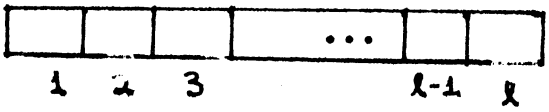
$$
\begin{aligned}
O(a_{it}) &= N \cdot P(a_{it}) \\
&= N \cdot \frac{u_e(a_{it})}{\sum_{i=1}^{N} u_e(a_{it})} \\
&= \frac{u_e(a_{it})}{\frac{1}{N}\sum u_e(a_{it})} \\
&= \frac{u_e(a_{it})}{\overline{u_e}(A(t))}
\end{aligned}
$$

So we see that individuals with average performance ratings produce on the average 1 offspring while better individuals produce more than 1 and poorer individuals produce less than 1. Hence, with no other mechanisms for adaptation, reproduction proportional to fitness produces a sequence of generations A(t) in which the best individual in A(o) takes over a larger and larger proportion of the population.

However, in nature and in these artificial systems, offspring are almost never exact duplicates of a parent. It is the role of genetic operators to exploit this selection process by producing new individuals which have high-performance expectations. The choice of operators is motivated by the mechanisms of nature: crossover, mutation, inversion, and so on. The exact form taken by such operators depends on the "genetic" representation chosen for individuals in A. In order to see more clearly the role of genetic operators, let us consider a very simple (from a biological viewpoint) reproductive plan which exhibits surprising adaptive capabilities.

2.4 The Basic Reproductive Plan: R1

The simplest reproductive plans use fixed-length haploid representations for elements of A. That is, an individual is represented by a single chromosome consisting of a fixed number (ℓ) of genes:



Each gene position is defined to take on one of a specified number of (allele) values. Hence, the set A of all possible individuals can be considered an ℓ-dimensional space in which an individual is represented by the value of its genes. To obtain a representation of this form for a specific problem for adaptation, alternative solutions to the problem are characterized uniquely by an ordered set

of ℓ parameters which in turn play the role of genes.

Having thus defined the representation space A, a reproductive plan is now free to explore A by submitting individuals for testing and evaluation as solutions to the problem. The basic reproductive plan accomplishes this via two genetic operators: crossover and mutation. To specify precisely how these operators work, we let an element a_i from A be represented as the string $v_{i1}v_{i2} \cdots v_{i\ell}$ in which the v_{ij} represent the gene values (alleles).

Crossover generates a new individual a_k from two existing individuals a_i and a_j by concatenating an initial gene segment from a_i with a final gene segment from a_j. The segments are defined by selecting a crossover point via a random sample from a uniform distribution over the ℓ-1 positions between the genes. So, for example, if crossover occurs between the second and third gene positions, individual a_k is generated from a_i and a_j as illustrated:

$$a_i = v_{i1}v_{i2}v_{i3} \cdots v_{i\ell}$$

$$a_j = v_{j1}v_{j2}v_{j3} \cdots v_{j\ell}$$

$$\rightarrow a_k = v_{i1}v_{i2}v_{j3} \cdots v_{j\ell}$$

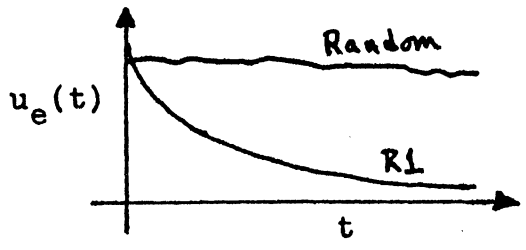
The crossover operation is embedded in plan R1 in the following way. Given an individual a_{it} selected from A(t) to produce an offspring, a mate a_{jt} is chosen from A(t) using the selection probabilities. An offspring is then produced by crossover.

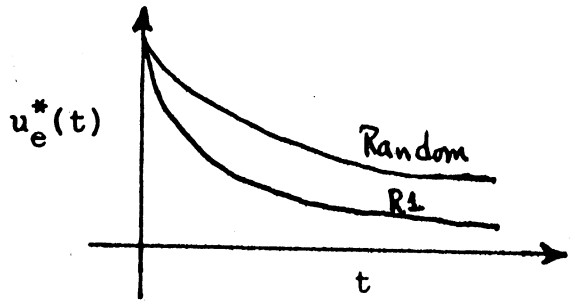
So we see that the strategy employed by crossover in searching A for better-performing solutions consists of constructing new sample points from existing ones selected on the basis of performance. Notice that if a particular allele (gene value) v_{ij} is not present in A(t), no offspring produced by crossover will contain v_{ij}. In other words, crossover is unable to generate points in the subspace $V_1 \times V_2 \times \ldots \times \{v_{ij}\} \times \ldots \times V_\ell$ of A. An allele can be missing from A(t) for several reasons. It may have been deleted by selection because of associated poor performance. It may also be missing simply because of the limited size of A(t). Obviously, if $|V_i| = 1000$, a minimum population of size 1000 is required for A(t) to contain an instance of each v_{ij}. In plan R1 new alleles are introduced into A(t) by the second genetic operator: mutation.

Mutation generates a new individual by independently modifying the value of one or more genes of an existing individual. A gene is selected for modification via a random sample from a uniform distribution over the ℓ gene positions. The new gene value is selected via a random sample from a uniform distribution over the associated set of alleles V_j. So, for example, if individual a_i is selected to undergo a mutation at position 2, an individual $a_j = v_{11}v_{j2}v_{13}\ldots v_{1\ell}$ is generated. The mutation operator is embedded in plan R1 as follows: a small

percentage of individuals generated by crossover for A(t+1) additionally undergo a mutation. In nature the probability of a gene undergoing mutation is generally less than .001 indicating that mutation (a form of random search) is not the primary genetic operator. Rather, it should be viewed as a background operator guaranteeing no allele will permanently disappear from A(t).

In order to evaluate the adaptive capabilities of plan R1, an environment E was defined consisting of a broad class of performance measures u_e defined on A (see appendix A). Included were instances of continuous, discontinuous, convex, non-convex, unimodal, multimodal, low-dimensional and high-dimensional functions as well as functions with Gaussian noise. The plan R1 was implemented in PL1 and its behavior observed over E in comparison to pure random search (see appendix C). While R1 did not always converge to a global maximum in the time allotted, it exhibited a considerable improvement over the performance generated by random search. Typical curves from these simulations are shown below:





Recall that the performance criteria X and X^* for adaptive systems were defined in terms of the average values of u_e and u_e^*, respectively, over time. With these encouraging results, we consider in more detail the properties of plan R1.

2.5 K-armed Bandits

Before we explore in more detail the way in which plan R1 searches the space A for better-performing elements, we will take a brief, but relevant, diversion to consider solutions to the generalization of the 2-armed bandit problem introduced in section 1.2, namely, the optimal allocation of trials to K machines. Holland (1975) has shown a mathematical solution exists if one is given a bit more a priori information about the K machines. Suppose we know that each machine pays stochastically according to a normal distribution $N(u_i, s_i^2)$, but we are not told which distribution is associated with which machine. In this case, an optimal strategy for allocating T trials to the K machines is roughly characterized as follows:

allocate exponentially more of the T trials to the observed best than to the remaining K-1 machines, where the exact form of the exponential depends on the K distributions $N(u_1,s_1^2)$. Notice that this strategy is non-realizeable in that no strategy can decide which machine will be the observed best after T trials without allocating the T trials, and then it is too late to distribute the trials optimally. However, such a solution gives us a characterization of the way in which trials should be allocated, and it yields a lower bound on the expected losses over T trials. The question, of course, is whether there are any realizeable strategies which are good approximations to the optimal one. To answer this question, we consider in more detail the optimal solution to the 2-armed bandit problem.

In this case we have two machines B1 and B2 which pay according to the distributions $N(u_1,s_1^2)$ and $N(u_2,s_2^2)$ respectively. For convenience, let B1 be the machine with the higher payoff and $\widetilde{B1}$ be the machine with the highest observed payoff after all T trials have been allocated with t_1 going to $\widetilde{B1}$ and t_2 to $\widetilde{B2}$. Holland (1975) has shown that the expected loss incurred over these T trials is given by:

$$L(t_1,t_2) = |u_1-u_2| * \left[t_1 * q(t_1,t_2) + t_2 * (1-q(t_1,t_2))\right]$$

where $q(t_1,t_2)$ is the probability that B2 will be the observed best and is well-approximated by

$$q(t_1,t_2) = \frac{1}{\sqrt{2\pi}} * \frac{\exp(-x^2/2}{x} \quad , \; x = \frac{u_1-u_2}{\sqrt{s_1^2/t_1 + s_2^2/t_2}}$$

To get a feeling for how $L(t_1,t_2)$ varies over the interval $0 < t_2 < T$, consider L rewritten as a function of T:

$$L(T,t_2) = \left|u_1-u_2\right| * \left[(T-t_2)*q(T,t_2) + t_2*(1-q(T,t_2))\right]$$

$$= \left|u_1-u_2\right| * \left[(T-2t_2)*q(T,t_2) + t_2\right]$$

As illustrated in figure 2.1, the term

$$(T-2t_2)*q(T,t_2) \tilde{=} (T-2t_2)* \frac{1}{\sqrt{2\pi}} * \frac{\exp(-x^2/2)}{x}$$

dominates L for small values of t_2, but drops off exponentially as a function of t_2 since

$$x = \frac{u_1-u_2}{\sqrt{s_1^2/t_1 + s_2^2/t_2}} \tilde{=} \frac{u_1-u_2}{\sqrt{s_2^2/t_2}} = (\frac{u_1-u_2}{s_2}) * \sqrt{t_2} = k_2\sqrt{t_2}$$

$$\therefore \quad (T-2t_2)* \frac{1}{\sqrt{2\pi}} * \frac{\exp(-x^2/2)}{x} \tilde{=} \frac{T}{\sqrt{2\pi}} * \frac{\exp(-k_2^2t_2/2)}{k_2\sqrt{t_2}}$$

$$= \frac{T}{k_2\sqrt{2\pi}} * \frac{\exp(-a_2t_2)}{\sqrt{t_2}}$$

FIG 2.1: BANDIT LOSS FUNCTION FOR T=50

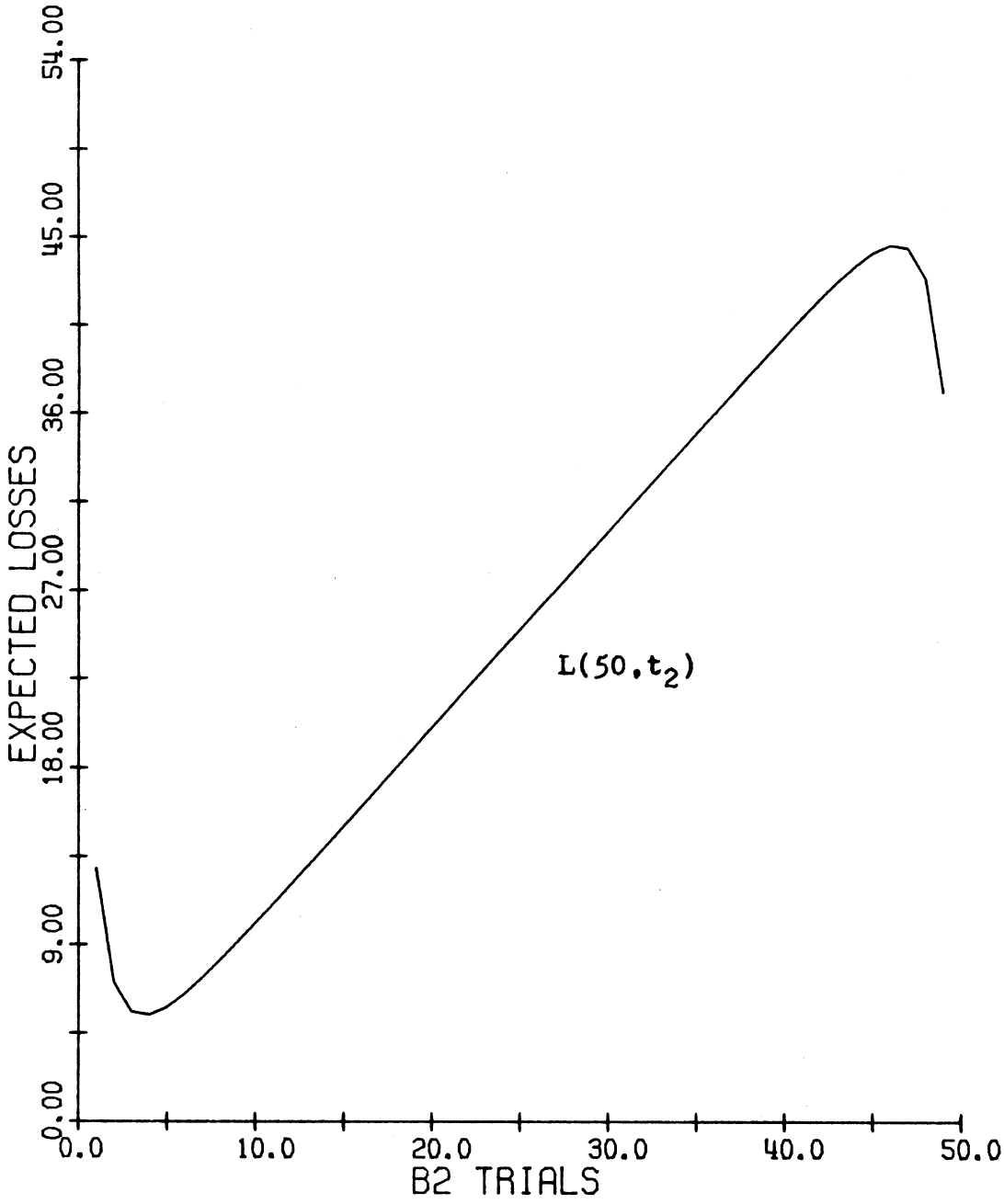


Figure 2.1: Expected losses over 50 trials on bandits B1(9,1) and B2(8,1).

As t_2 increases, the term t_2 dominates and L is essentially linear with respect to t_2. Finally, as $t_2 \rightarrow T$, the term

$$(T-2t_2)*q(T,t_2) \cong \frac{-T}{\sqrt{2\pi}} * \frac{\exp(-k_1^2 t_1/2)}{k_1\sqrt{t_1}} = \frac{-T}{k_1\sqrt{2\pi}} * \frac{\exp(-a_1 t_1)}{\sqrt{t_1}}$$

re-emerges as the dominant term with a negative sign.

In order to minimize our expected losses over T trials, we must find the value t_2^* such that

$$L(T,t_2^*) \leq L(T,t_2) \ , \ 0 < t_2 < T$$

Finding an analytic expression for t_2^* by considering those points at which $\frac{dL}{dt_2} = 0$ is fairly complex. Holland (1975), for example, has derived the approximation

$$t_2^* \sim b^{-2} * \ln\left[\frac{b^4 T^2}{8\pi \cdot \ln(T^2)}\right] \ , \ b = \frac{u_1 - u_2}{s_2}$$

For our purposes the optimum is found via a one-dimensional iterative search technique applied directly to L for various values of T. Figure 2.2 illustrates how the optimal loss function $L(T,t_2^*)$ varies with T. It is this kind of performance that a realizeable strategy must hope to approximate. Finally, figure 2.3 illustrates the previously mentioned relationship between $t_1^* = T - t_2^*$ and t_2^*, namely, that an optimal strategy allocates ex-

FIG 2.2: OPTIMAL LOSSES OVER T TRIALS

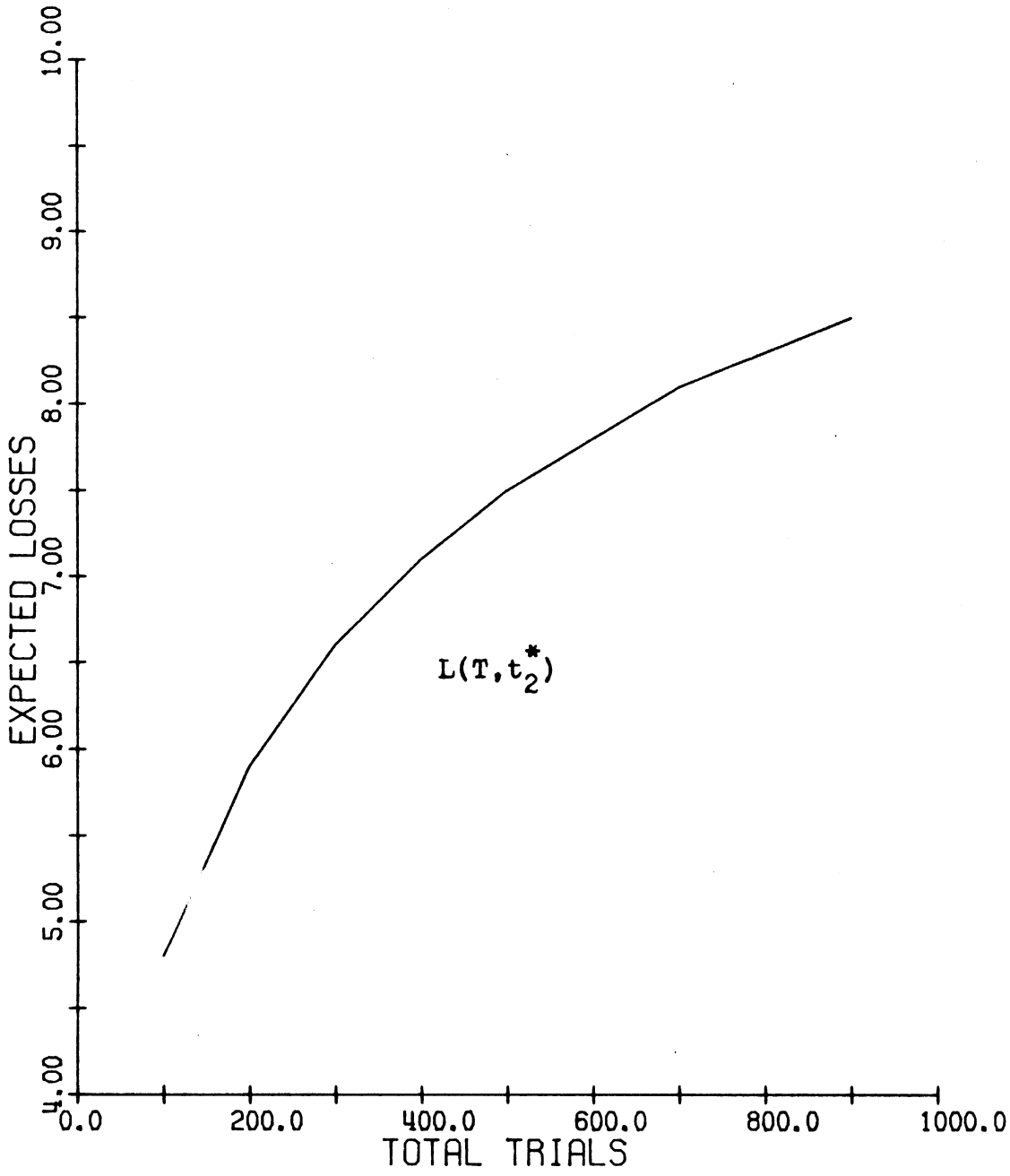


Figure 2.2: Optimal losses incurred over T trials on two bandits B1(9,1) and B2(8,1).

FIG 2.3: OPTIMAL DISTRIBUTION OF T TRIALS

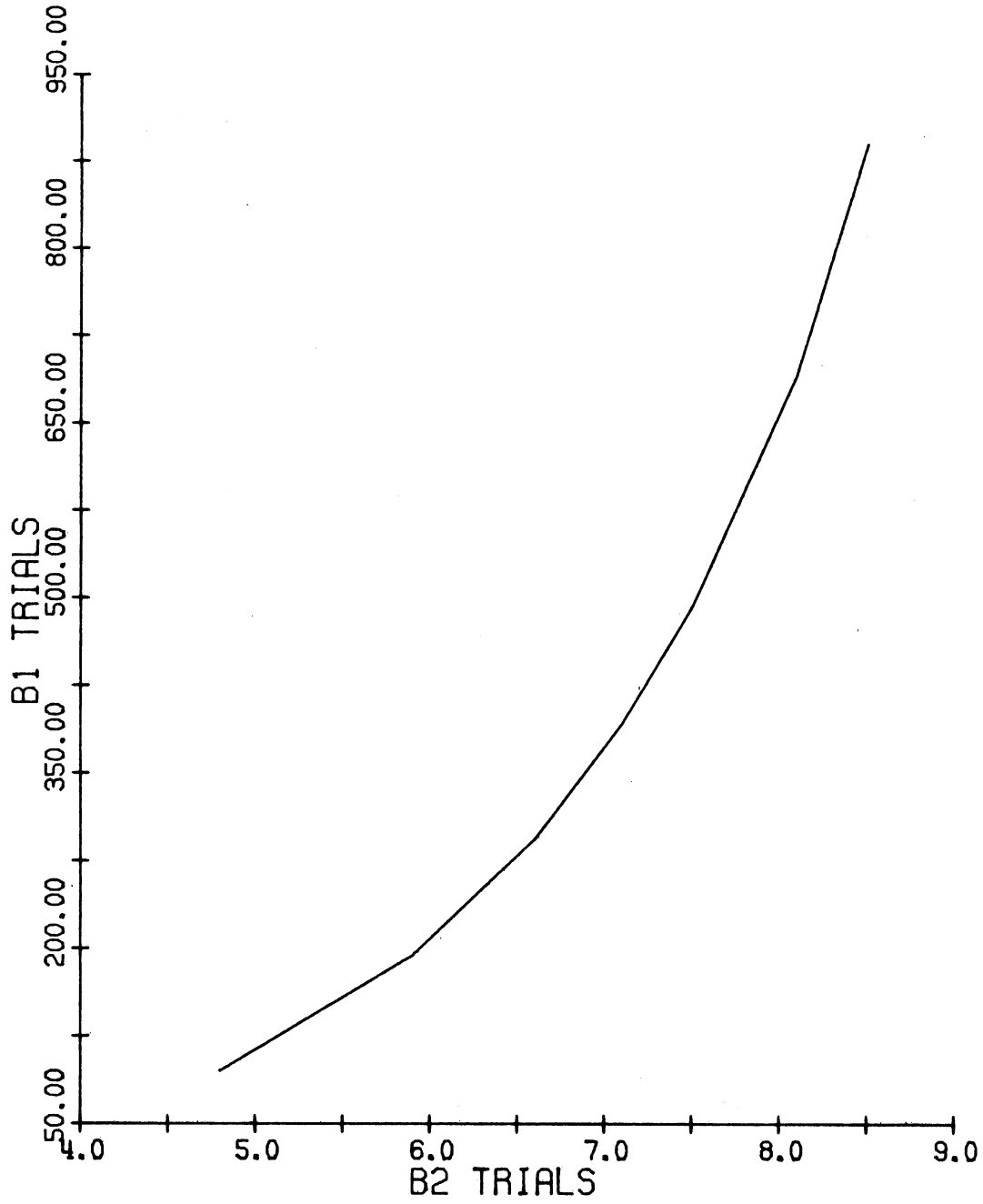


Figure 2.3: Optimal distribution of T trials between two bandits B1(9,1) and B2(8,1).

ponentially more trials to the observed best.

We are now in a position to evaluate the performance of some realizeable strategies. The first one which comes to mind is the standard decision theory approach (hereafter referred to as DTS) which goes as follows: allocate a small number t of trials to each machine; then allocate the remaining T-2t trials to the observed best. Paralleling the preceding analysis, we have an expected loss function:

$$L_1(T,t) = |u_1-u_2| * \left[(T-t)*q(t) + t*(1-q(t))\right]$$

where q(t) is the probability that B2 is the observed best after allocating t trials to each machine. In this case we have

$$q(t) \cong \frac{1}{\sqrt{2\pi}} * \frac{\exp(-x^2/2)}{x}, \quad x = \frac{u_1-u_2}{\sqrt{s_1^2/t + s_2^2/t}}$$

$$= \frac{u_1-u_2}{\sqrt{s_1^2 + s_2^2}} * \sqrt{t}$$

Again, we seek the value t^*, $0 < t^* < \frac{T}{2}$, which minimizes $L_1(T,t)$, as illustrated by figure 2.4. It should be clear that we can define a DTS which, when given T, u_1, u_2, s_1, and s_2, computes the optimal initial sample size t^* and allocates its trials accordingly. Intuitively one feels that this DTS will approximate the optimal strategy as T

FIG 2.4: DTS LOSS FUNCTION FOR T=50

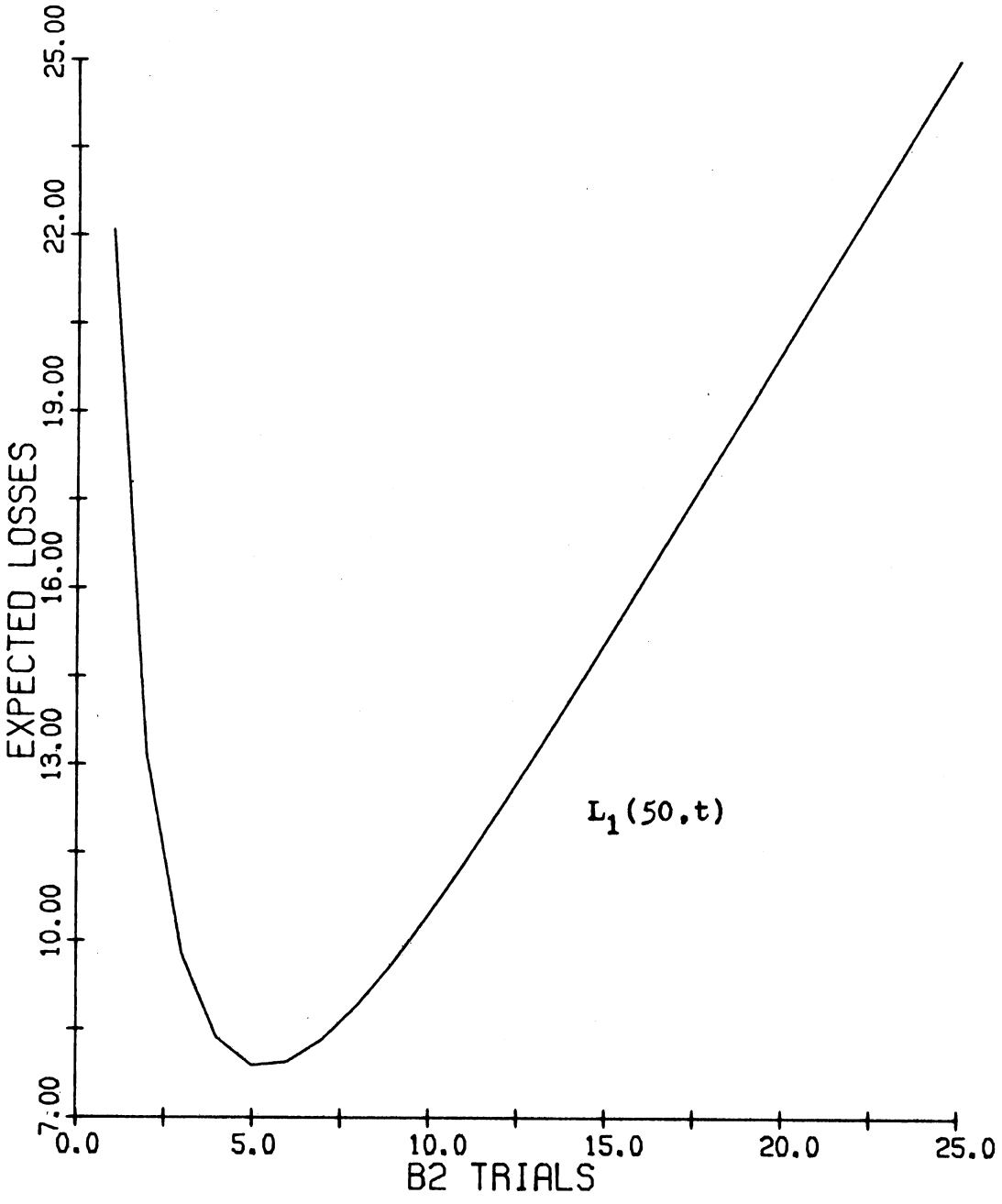


Figure 2.4: DTS expected losses over 50 trials on bandits B1(9,1) and B2(8,1).

increases. Figure 2.5, however, illustrates that it is a fairly crude approximation since the optimal number of trials allocated to B2 grows very slowly with T.

A second more interesting approach incorporates some of the ideas presented in the discussion of reproductive plans in section 2.3. The basic idea is to make a series of reversible decisions during the sequence of trials rather than one non-reversible decision. This is accomplished by defining a selection probability distribution over the machines. Initially, the distribution is uniform; however, it changes over time as follows:

$$P_i(t+1) = P_i(t) * \frac{\overline{f_i}(t)}{\overline{f}(t)} * K_{t+1}$$

That is, the probability of selecting machine i changes over time in proportion to its observed performance relative to the average, where K_{t+1} is the normalization factor required for $\sum_i P_i(t+1)=1$. If at each time step we select a machine for trial by sampling from this time-varying selection distribution, it should be clear that a machine which continues to show above-average performance will rapidly dominate the allocation of trials.

Initially t samples are allocated to each machine for estimates $\overline{f}_i(t)$ of u_i before the first decision is made. This, of course, incurs an initial loss $|u_1-u_2| * t$, but adds certainty to the subsequent decisions. As $t \to 1$, the initial overhead is reduced at the expense of making de-

FIG 2.5: EXPECTED DTS LOSSES OVER T TRIALS

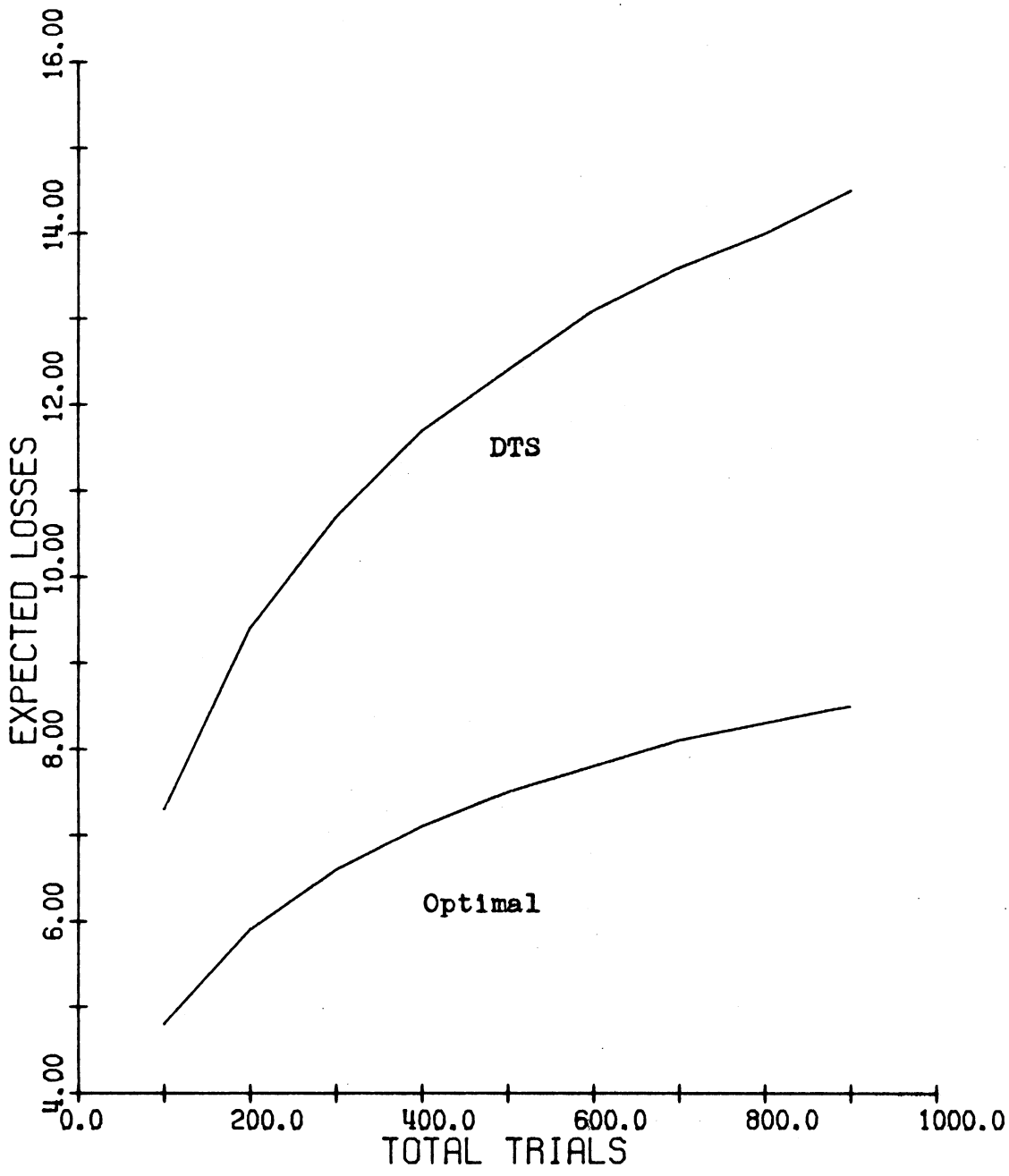


Figure 2.5: A comparison of the expected losses for DTS and the optimal on two bandits B1(9,1) and B2(8,1).

cisions with more uncertainty.

So we have expected losses over T trials given by:

$$L_2(T,t) = |u_1-u_2| * t + \tilde{L}_2(T,t)$$

where $\tilde{L}_2(T,t)$ specifies the expected losses during the time-varying decision processes from $2t+1$ to T. We can express $\tilde{L}_2$ as

$$\tilde{L}_2(T,t) = \sum_{j=2t+1}^{T} \ell(j)$$

where $\ell(j)$ is the expected loss on the j^{th} trial and is given by

$$\ell(j) = |u_1-u_2| * E\left[P_2(j)\right]$$

where $E\left[P_2(j)\right]$ is the expected value of the selection probability $P_2(j)$ at time j.

While it is relatively straightforward to calculate the expected initial value, $E\left[P_2(2t)\right]$, subsequent expected values are extremely difficult to analyze since the transition function

$$P_2(t+1) = P_2(t) * \frac{\bar{f}_2(t)}{\bar{f}(t)} * K_{t+1}$$

is non-Markovian and depends on the random variable $t_2(t)$, the number of trials allocated to B2 through time t.

Consequently, we are faced with optimizing $L_2(T,t)$ with respect to t by simulation as illustrated in figure 2.6. Two hundred samples were taken of $L_2(100,t)$ for

FIG 2.6: TVS LOSS FUNCTION FOR T=100

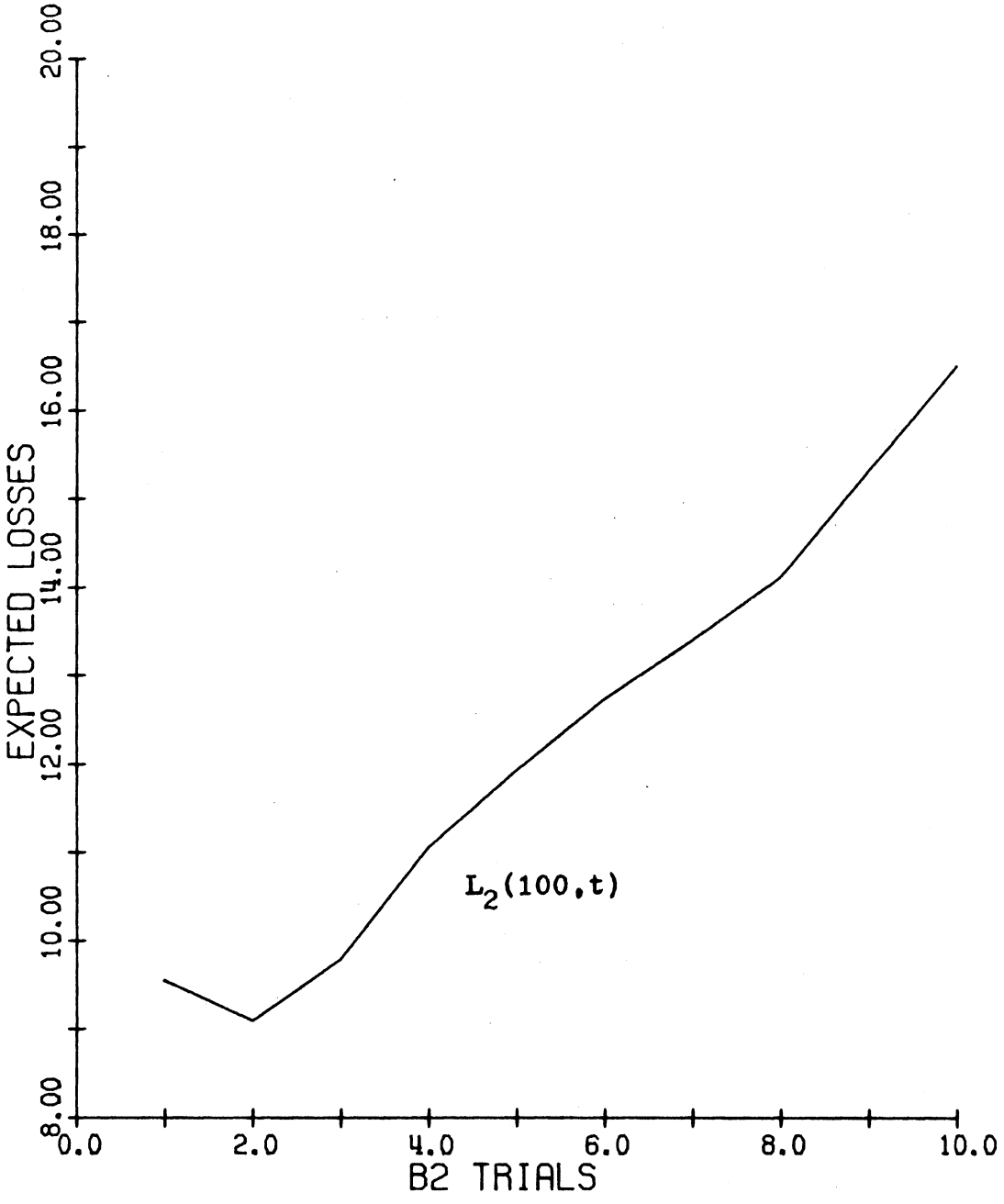


Figure 2.6: Simulated losses over 100 trials using TVS on two bandits B1(9,1) and B2(8,1).

t=1,2,3,....,10. These figures, and others not shown here, suggest that a good approximation for t^* is given by:

$$t^* \geq \frac{\sqrt{s_1^2 + s_2^2}}{u_1 - u_2}$$

which, for the illustrated case, yields $t^* \geq \sqrt{2}$ or $t^* \cong 2$. This formulation is motivated as follows: choose enough initial samples t so that, with a priori probability q, $\overline{f_1}(t) - \overline{f_2}(t)$ will have the same sign as $u_1 - u_2$. We know the a priori probabilities associated with $f_1(t) - f_2(t)$ falling in the interval $(u_1-u_2) \pm K^* \sqrt{\frac{s_1^2/t + s_2^2/t}{\sqrt{t}}}$ For the signs to be the same, we must have

$$|u_1 - u_2| \geq K^* \frac{\sqrt{s_1^2/t + s_2^2/t}}{\sqrt{t}}$$

or

$$t \geq K^* \frac{\sqrt{s_1^2 + s_2^2}}{|u_1 - u_2|}$$

The value K=1 or q=.68 seemed to fit the data best.

Using the above approximation for t^*, figure 2.7 compares the expected TVS losses with those of the two previous strategies and illustrates that it rapidly approaches the optimal one. Finally, figure 2.8 compares the way in which the three strategies divide the trials between the two machines.

With this analysis in mind, we now consider in more

FIG 2.7: EXPECTED TVS LOSSES OVER T TRIALS

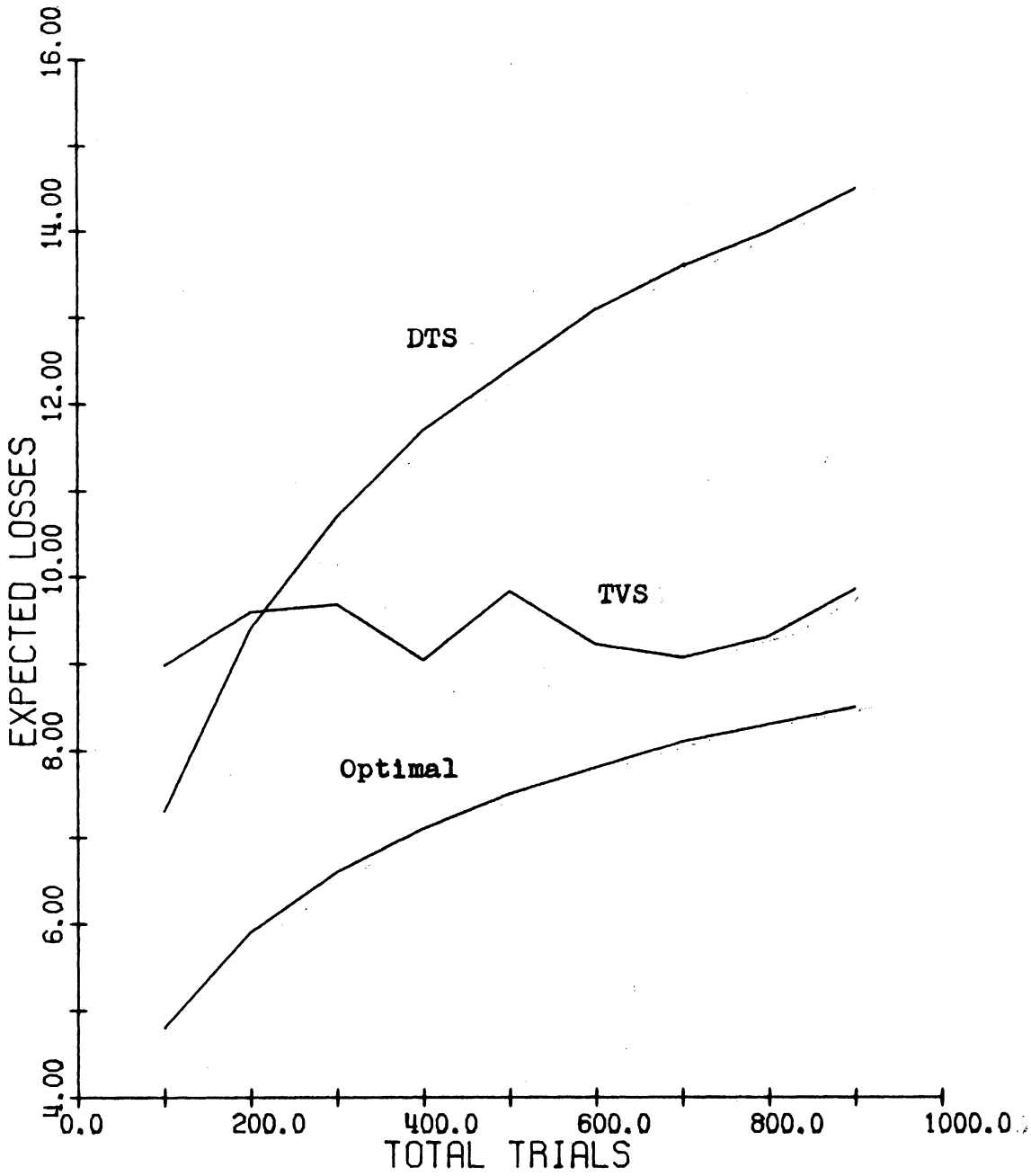


Figure 2.7: A comparison of expected losses over T trials on two bandits B1(9,1) and B2(8,1).

FIG 2.8: COMPARATIVE ALLOCATION OF T TRIALS

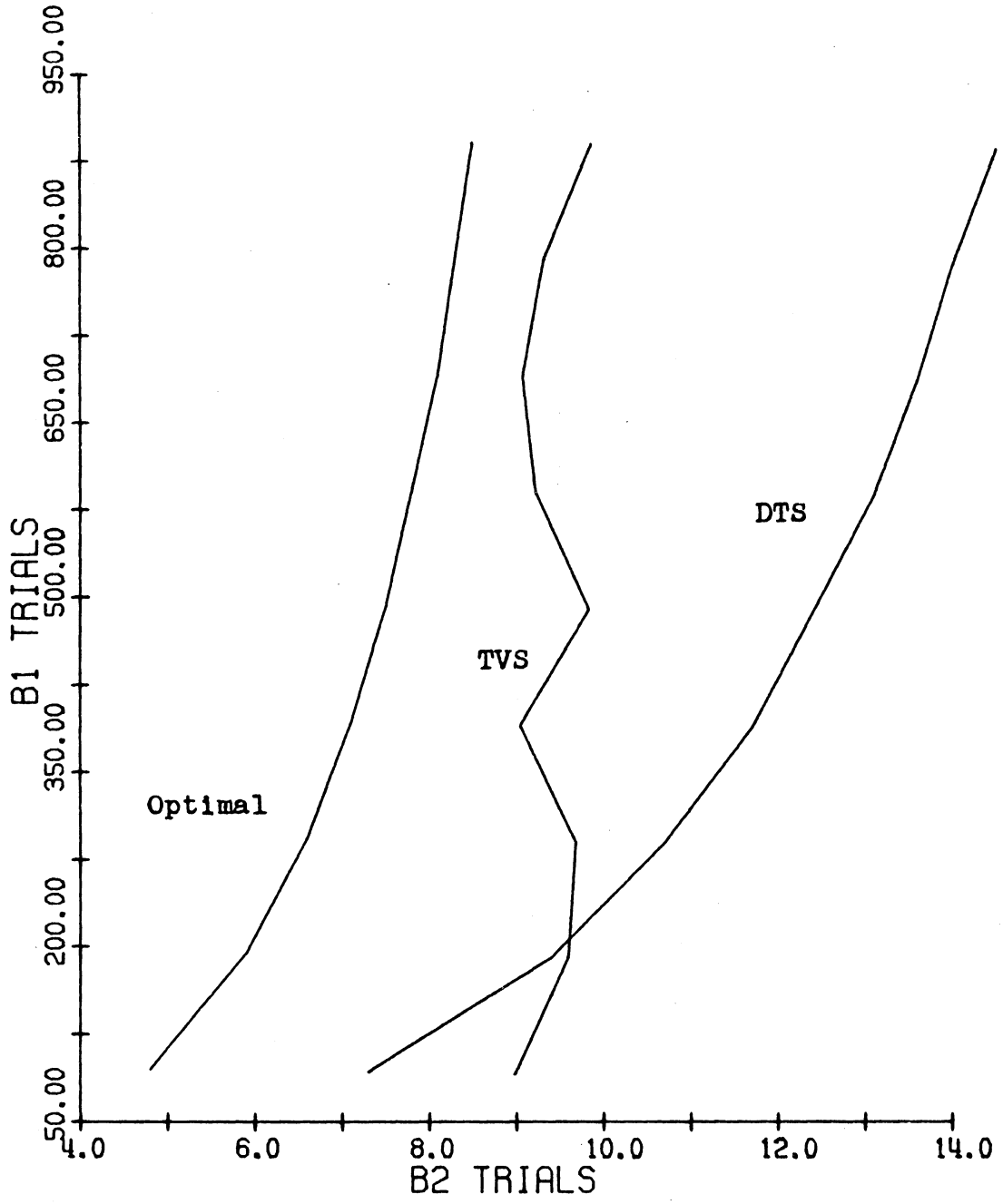


Figure 2.8: A comparison of the allocation of T trials to two bandits B1(9,1) and B2(8,1).

detail how adaptive plan R1 allocates its trials within the space A.

2.6 Hyperplane Analysis of R1

In this section we will attempt to understand more clearly how the genetic plan R1 searches the representation space A for better-performing elements by focusing our attention on hyperplane partitions of A as suggested by Holland (1975).

As suggested in section 2.4, we consider A as an ℓ-dimensional space in which a point $a_i \in A$ is specified by giving its ℓ gene values $\langle v_{i1}, \ldots, v_{i\ell} \rangle$. A k^{th}-order hyperplane is then defined to be the $(\ell-k)$-dimensional subspace of A specified by giving only k of the ℓ gene values. These hyperplanes can be represented visually as follows:

$$0\text{--}\ldots\text{-} \stackrel{d}{=} \left\{ a_i \in A : v_{i1} = 0 \right\}$$

$$\text{-}11\text{-}\ldots\text{-} \stackrel{d}{=} \left\{ a_i \in A : v_{i2} = 1 \ \& \ v_{i3} = 1 \right\}$$

If we consider all possible hyperplanes which can be defined by specifying the gene values of a fixed set of K positions, this set $\{H_i\}$ of hyperplanes forms a uniform partition of the space A. For example, if $V_1 = \{0,1\}$, the allowable values for the first position, then

$$H_1 = 0\text{--}\ldots\text{-}$$

$$H_2 = 1\text{--}\ldots\text{-}$$

form a first-order partition of A with exactly half of the points falling in each hyperplane. If we consider the performance measure $u_e : A \rightarrow R$ restricted to a particular hyperplane,

$$u_e\Big|_{H_1} : H_1 \rightarrow R$$

it has a well-defined mean and variance which are, of course, unknown to an adaptive strategy. Hence, associated with each hyperplane partition $\left\{H_i\right\}_{i=1}^{K}$ of the space A, is a K-armed bandit problem, namely, the optimal allocation of trials among the partition elements H_i. Since any sequence of trials in A simultaneously distributes trials among the elements of each of the $\sum_{j=0}^{\ell} \binom{\ell}{j} = 2^{\ell}$ distinct hyperplane partitions of A, we can view the problem of searching A as simultaneously solving 2^{ℓ} K_j-armed bandit problems. The question we are exploring in this section is how well plan R1 allocates its trials to these K_j-armed bandits.

In order to accomplish this we fix our attention on a particular hyperplane partition $\left\{H_i\right\}$ in relationship to the population A(t) of N individuals maintained by plan R1. Since $\left\{H_i\right\}$ is a partition of the space A, each a_{it} in A(t) lies in some H_i. Let $M_i(t)$ represent the number of individuals from A(t) which lie in H_i at time t. Because of the way in which the selection probabilities were defined for reproductive plans (section 2.3), we

know that the expected number of offspring $O(H_1)$ produced by individuals in H_1 at time t is given by:

$$O(H_1) = \sum_{j=1}^{M_1(t)} \frac{u_e(a_{jt})}{\bar{u}_e(t)}$$

$$= \frac{M_1(t)}{\bar{u}_e(t)} * \frac{\sum_{j=1}^{M_1(t)} u_e(a_{jt})}{M_1(t)}$$

$$= M_1(t) * \frac{\tilde{u}_e(H_1(t))}{\bar{u}_e(t)}$$

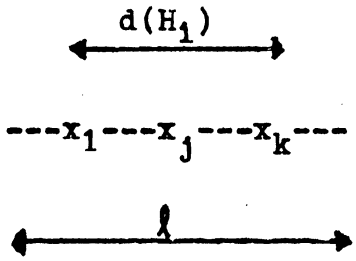
If in fact the offspring $O(H_1)$ themselves lie in H_1, then we have

$$M_1(t+1) = M_1(t) * \frac{\tilde{u}_e(H_1(t))}{\bar{u}_e(t)}$$

That is, the number of trials allocated to H_1 varies from one time step to the next in proportion to its performance relative to the average, which of course is the TVS solution to the K-armed bandit problem discussed in the preceding section.

Whether or not $O(H_1) \subseteq H_1$ depends on the genetic operators used to construct them. In plan R1 there are two such operators: crossover and mutation. An offspring will lie in H_1 only if the k positions which define H_1 re-

main unchanged between parent and offspring. Intuitively, if crossover occurs within these defining positions, one or more of them will likely be changed. Hence it is fairly easy to show that the probability of a parent in H_i producing an offspring outside H_i is no greater than $\frac{d(H_i)-1}{\ell-1}$, where $d(H_i)$ is the "definition length" of H_i, namely, the length of the smallest segment containing all the defining positions of H_i as illustrated below.



As a consequence we note that crossover has little effect on the allocation of trials to the bandits associated with short-definition hyperplanes (relative to ℓ), while the allocation of trials to long-definition hyperplanes is considerably disrupted.

The probability of a parent in H_i producing an offspring outside H_i via mutation is just $P_m * \frac{k}{\ell}$, where P_m is the probability of a gene undergoing a mutation and k is the order of H_i. In nature and generally in plan R1, $P_m \leq .001$. Hence, mutation has very little effect on the allocation of trials according to performance.

In summary, then, by looking at hyperplane partitions of A, we have gained considerable insight into the behav-

ior of reproductive plans. In the first place, this analysis yields a criterion for artificial genetic operators, namely, the ability to generate new individuals in A without disturbing too much the near-optimal TVS allocation of trials. Secondly, we can now describe the way plan R1 searches the space A. It generates near-optimal allocation of trials simultaneously to short-definition hyperplane partitions. As elements of high-performance hyperplanes begin to dominate A(t), we have a reduction in the dimension of A and a corresponding reduction in the definition lengths of hyperplanes, providing for another cycle of near-optimal sampling.

2.7 An Example of R1

As an illustration of the discussion in the previous sections, we consider a simple problem for adaptation. Suppose each alternative solution to a problem is represented by a single real number in the interval $[0,10]$ with a precision of 2 decimal places. Suppose further than the performance associated with each solution point is given by $f(x)=x^2$ with the higher valued solutions being the better ones. We choose the representation space A for R1 as follows: There are $(10-0)*10^2$ distinct solutions; hence, $\log_2(10^3)=10$ bits are required for a binary representation. The correspondence between $[0,10]$ and A is given by:

$$x = \frac{a_1}{100}$$

So, for example,

$$0.01 \longleftrightarrow 0000000001$$

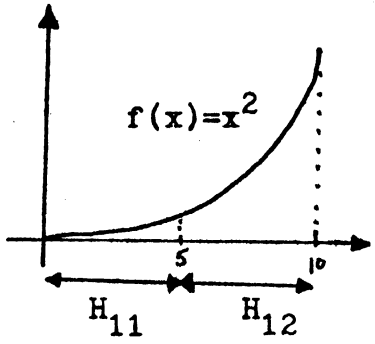
$$5.12 \longleftrightarrow 1000000000$$

In order to see how R1 searches A, we focus our attention initially on a first-order partition P_1 defined by:

H_{11}: 0---...---

H_{12}: 1---...---

P_1 simply divides the space in half:



Since R1 generates an initial population A(0) randomly from a uniform distribution over A, we expect half of A(0) to lie in H_{11} and half in H_{12}. Notice, however, that $\overline{f}(H_{11}) < \overline{f}(H_{12})$. Since P_1 is a short-definition partition relative to $\ell=10$, plan R1 will allocate trials to H_{11} and H_{12} according to the near optimal time-varying strategy (TVS) described earlier. In other words, R1 quickly generates a population A(t) consisting almost entirely of individuals from H_{12}.

We now consider a refinement P_2 of P_1 given by:

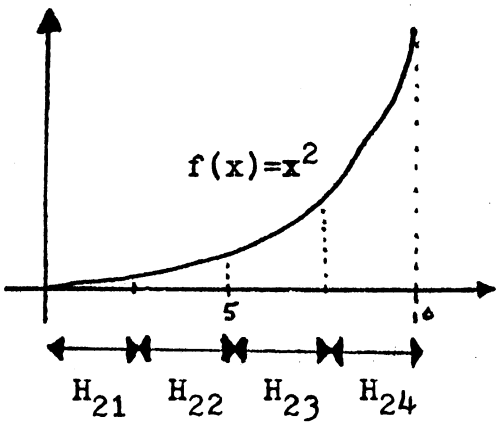
H_{21}: 00---...--

H_{22}: 01---...--

H_{23}: 10---...--

H_{24}: 11---...--

P_2 simply divides the space in quarters:



As we noted above, R1 rapidly generates a population $A(t_1)$ in which most individuals begin with a 1. This is in effect a reduction in the search space A to an ℓ-1 dimensional space. Hence, after a few generations, P_2 effectively becomes a first-order partition of $A_{\ell-1}$ to which R1 now allocates a near-optimal sequence of trials. Since $\overline{f}(H_{23}) < \overline{f}(H_{24})$, R1 rapidly generates a population $A(t_2)$ which lies almost entirely in H_{24}, effecting yet another reduction in the search space.

The important thing to note here is that the same remarks hold for any other short-definition partitions, for example:

---.....--0

---.....--1

While such partitions are harder to visualize, each is being sampled at a near-optimal rate simultaneously by R1. It is this parallelism which gives even simple reproductive plans like R1 their surprising adaptive capabilities.

2.8 Summary

In this chapter we have defined a class of genetic adaptive models called reproductive plans. These artificial systems are motivated by the kinds of models used in population genetics to explain the adaptive behavior of natural systems. The central feature of these reproductive plans is that new solutions to the problem for adaptation are generated by selecting individuals from the current population on the basis of their observed performance to produce offspring via genetic operators. By focusing our attention on hyperplanes on A rather than individual elements of A, we were able to characterize the way in which reproductive plans search A, and the characterization provided a criterion for genetic operators. Finally, we saw that even the simple reproductive plan R1, because of its ability to simultaneously allocate trials at a near-optimal rate to a large number of hyperplanes on A, exhibit considerable improvement over random search.

Chapter 3

STOCHASTIC EFFECTS IN FINITE GENETIC MODELS

3.1 Introduction

In this chapter we will explore the characteristics of finite genetic adaptive systems, that is, genetic plans which have limited memory and time to adapt to the problem at hand. As one might expect, the behavior of such systems can vary considerably from the norm predicted by mathematical analysis involving expected values, the law of large numbers, and limit theorems. The motivation for analyzing finite models, of course, is that they correspond to the observed behavior in any practical application of genetic adaptive systems. We will pursue this analysis by considering in more detail the characteristics of plan R1 introduced in the preceding chapter.

3.2 The Problem of Premature Convergence

We begin by analyzing the behavior of plan R1 on test function F1 (see appendix A). Here the problem consists of finding the minimum point on the three dimensional parabolic surface given by

$$F1(X) = \sum_{1}^{3} x_i^2, \quad x_i \leq 5.12, \quad \Delta x_i = .01$$

As illustrated in appendix C, plan R1 generates an

exponential decrease in both f(t) and $f^*(t)$ over the interval $1 \leq t \leq 10{,}000$. However, since R1 is a stochastic process, these curves represent the performance of R1 averaged over the number of independent runs. Table 3.1 depicts the behavior of R1 for a particular run on test function F1. Notice that there is little or no improvement in f(t) and $f^*(t)$ from t=3000 on, even though $f^*(t)$ is still greater than the minimum of zero at the origin. This behavior is typical of plan R1. After an initial reduction in f(t) and $f^*(t)$, a threshold seems to be crossed after which little or no improvement is generated. If we look more closely at the population A(t) maintained by R1, the reason for this lack of improvement becomes clear: each individual in A(3000) is very nearly alike. Recall that plan R1 uses a binary genetic representation for points in the solution space A. That is, each gene position can take on only the values 0 or 1, and for this problem, $|A| = (10^3)^3 = 10^9$ requiring $\ell = 30$ gene positions. If we consider A(t) a reservoir of gene values (alleles), the "lost" column in table 3.1 illustrates that in A(3000) 22 of the 30 gene positions have no instances of one of the two possible alleles. That is, plan R1 has converged to a particular allele in all but 8 positions and hence reduced the search space for crossover to $2^8 = 256$ points. Moreover, if we say that plan R1 has <u>effectively converged</u> to a particular allele whenever an allele is

trials t	generations	f(t)	$f^*(t)$	lost	converged
50	1	25.62	2.016	0	0
250	5	11.20	0.409	1	1
450	10	8.11	0.187	2	2
1000	25	5.48	.151	6	11
2000	60	2.19	.109	14	18
3000	100	1.42	.018	22	25
6000	335	1.05	.018	24	26
10000	560	.82	.018	25	27

Table 3.1: The data associated with a single run of R1 on F1.

found in more than 95% of the population, then the "converged" column in table 3.1 illustrates that by t = 3000, R1 has effectively converged in 25 of the 30 positions.

This reduction in the search space A is precisely the behavior discussed in chapter 2. Unfortunately, however, in this case the optimum for F1 is not contained in the reduced subspace. Nor is it very likely that plan R1 will find the optimum for $t > 3000$. To see this recall that crossover can generate a point in A for trial only if all the alleles for that point are present in the population. Hence, crossover effectively searches only the reduced subspace of 256 points. Moreover, because of the similarity of individuals in A(3000), the results of many crossovers will be to produce an offspring identical with one of the parents, providing no new points for trial. Comparing the "trials" column and the "generation" column in table 3.1 illustrates this reduced effectiveness of R1 as alleles are lost from the population. Initially, nearly 50 new trials are generated per generation by crossover and mutation. However, from generation 60 on ($t > 3000$), there are fewer than 15 new trials per generation.

Restating these observations in terms of the hyperplane analysis of chapter 2 yields further insight into the problem. For 25 of the 30 first-order hyperplane partitions of A, plan R1 has chosen to allocate

almost all of the trials from t = 3000 on to one of the two competing partition elements. However, if we consider the symmetry of test function F1 on A, it should be clear that every first-order partition of A presents plan R1 with a 2-armed bandit problem in which the machines have equal payoffs. In other words, no particular allele has an advantage over its competitor; yet in 25 of 30 positions, one allele seems to have almost completely dominated, effecting a dramatic reduction in the search space.

Immediately one thinks of increasing the mutation rate as a simple direct way of maintaining variability in the population. But we must be careful at this point of applying a cure to a symptom rather than the problem. Certainly increasing mutation will increase the variation in the population maintained by R1 on F1. But recall that on F1 no particular allele has an advantage over its competitor. For the other functions in E there clearly are alleles which yield much higher performance than their competitors. Increasing mutation in these cases will retard the dominance of the better performing alleles and slow the adaptive response. What we attempt to understand in this chapter is why R1 has such a high rate of allele loss on F1. The hope is that understanding this problem will provide insight into improved performance on E.

3.3 Genetic Drift

The phenomenon of genetic drift is a well-studied problem in population genetics. Since it is an artifact of the application of random selection processes to finite populations, it has considerable bearing on the finite genetic models under study in this chapter. Genetic drift can be illustrated by the following simple stochastic model. Suppose we have a population A(t) of N individuals and we generate A(t+1) by making N uniformly random selections from A(t) with replacement and apply no genetic operators. Again we focus our attention on the alleles of a particular gene and observe the number of instances of these alleles in the population. If we assume a binary genetic representation and a uniformly random initial population A(0), then the expected number of 0-alleles $R_i(t)$ for gene i is N/2. However, as t increases, the variance of $R_i(t)$ also increases to the extent that wide deviations from the norm are quite likely.

To see this more clearly, we can represent the above model as a Markov process in which the states are simply the N+1 possible values of $R_i(t)$. The transition probability P_{jk} is simply the probability of k successes in N Bernouli trials with a probability of success on each trial of j/N. That is,

$$P_{jk} = \binom{N}{k}\left(\frac{j}{N}\right)^k\left(1-\frac{j}{N}\right)^{N-k}$$

The initial state probabilities P_k are simply

$$P_k = P_{\frac{N}{2},k} = \binom{N}{k}\left(\frac{1}{2}\right)^k\left(\frac{1}{2}\right)^{N-k} = \binom{N}{k}\left(\frac{1}{2}\right)^N$$

With no genetic operators defined, it should be clear that states $R_1(t) = 0$ and $R_1(t) = N$ are absorbing states. Since the other N-1 states are all transient, the probability of being in either of the absorbing states increases over time and in fact approaches 1. Of even more interest is the expected number of generations to first entry into a particular state. We are interested in those states in which one of the alleles under observation has managed to dominate a certain percentage of the population. To illustrate the effects of genetic drift we will focus our attention on 4 states: 70, 80, 90, and 100% dominance.

The expected number of generations f_{jk} to first entry into state k from state j is given by:

$$f_{jk} = \sum_{n=0}^{\infty} n * f_{jk}^{n}$$

where f_{jk}^{n} is the probability of first entry to k from j in exactly n steps. Unfortunately, the computation of these expected values is difficult since the terms f_{jk}^{n} are computed recursively as

$$P_{jk}^{n} = \sum_{i=1}^{n} f_{jk}^{i} * P_{kk}^{n-i}$$

in terms of the extended transition probabilities P_{jk}^{n} which are themselves computed by raising the transition matrix

$$P = \left[P_{ij}\right]$$

to the n^{th} power.

However, for our purposes, we can estimate the expected values by simulation, the results of which are illustrated in figure 3.1. As might be expected, the number of generations to reach a particular state of dominance is a linear function of population size. The slopes associated with the (70, 80,90, and 100%) states are roughly 1/5, 2/5, 4/5, and 8/5 allowing for a predicted rate of dominance. Figure 3.2 illustrates more clearly the role of population size in increasing the expected number of generations to first entry into one of the four states. Moreover, it illustrates that the effects of genetic drift cannot be ignored even in a population of size 100 if the number of generations exceeds 50.

To reduce these stochastic effects over the interval of adaptation, we can of course increase the population size sufficiently to minimize genetic drift, but we do so at the expense of maintaining a larger population and in general a slower adaptive response. A second alternative which immediately comes to mind is to add a mutation operator which would counteract the allele loss

FIG 3.1: GENETIC DRIFT VARYING POPULATION SIZE

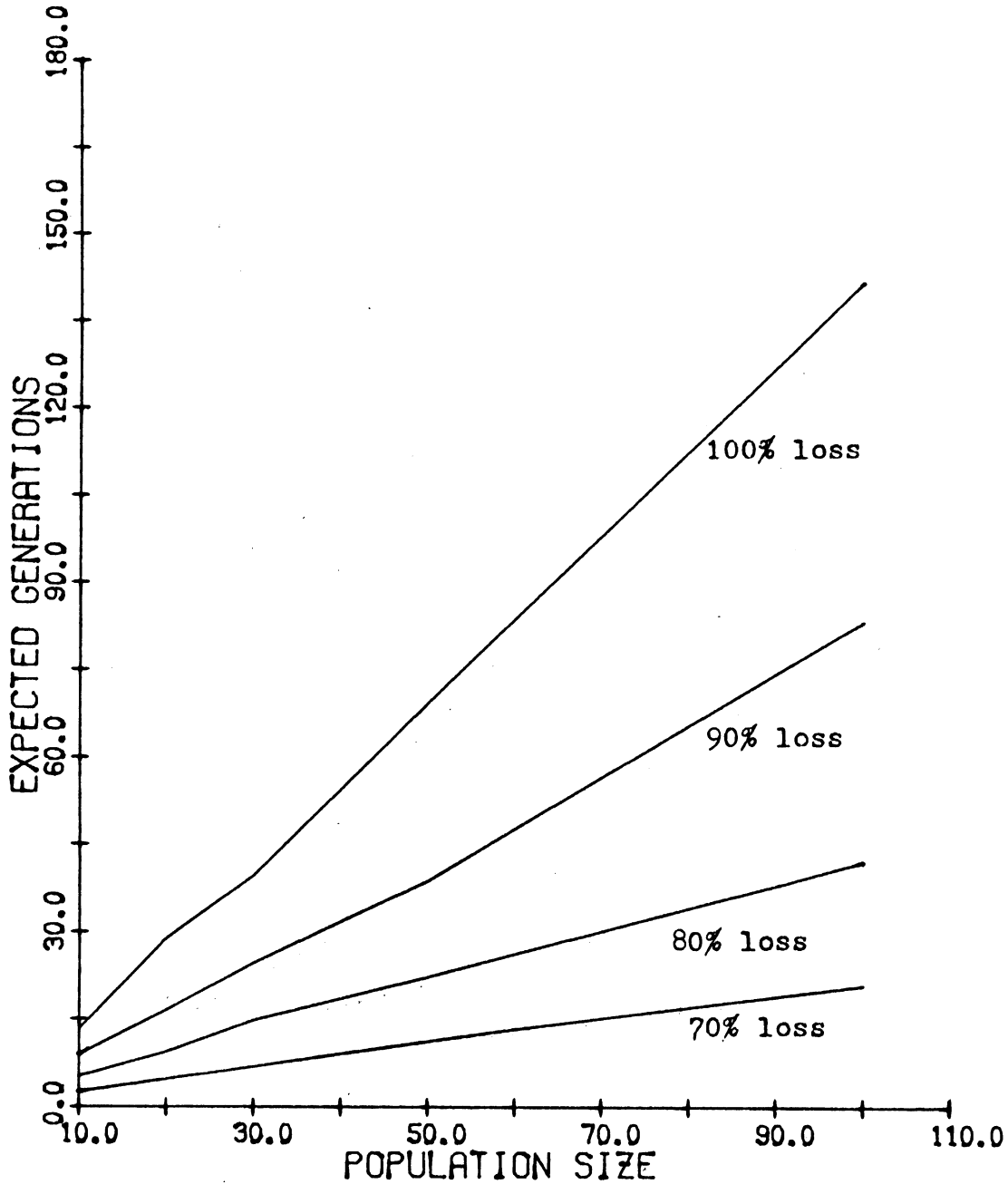


Figure 3.1: The rate of allele loss due to genetic drift as a function of population size.

FIG 3.2: GENETIC DRIFT VARYING POPULATION SIZE

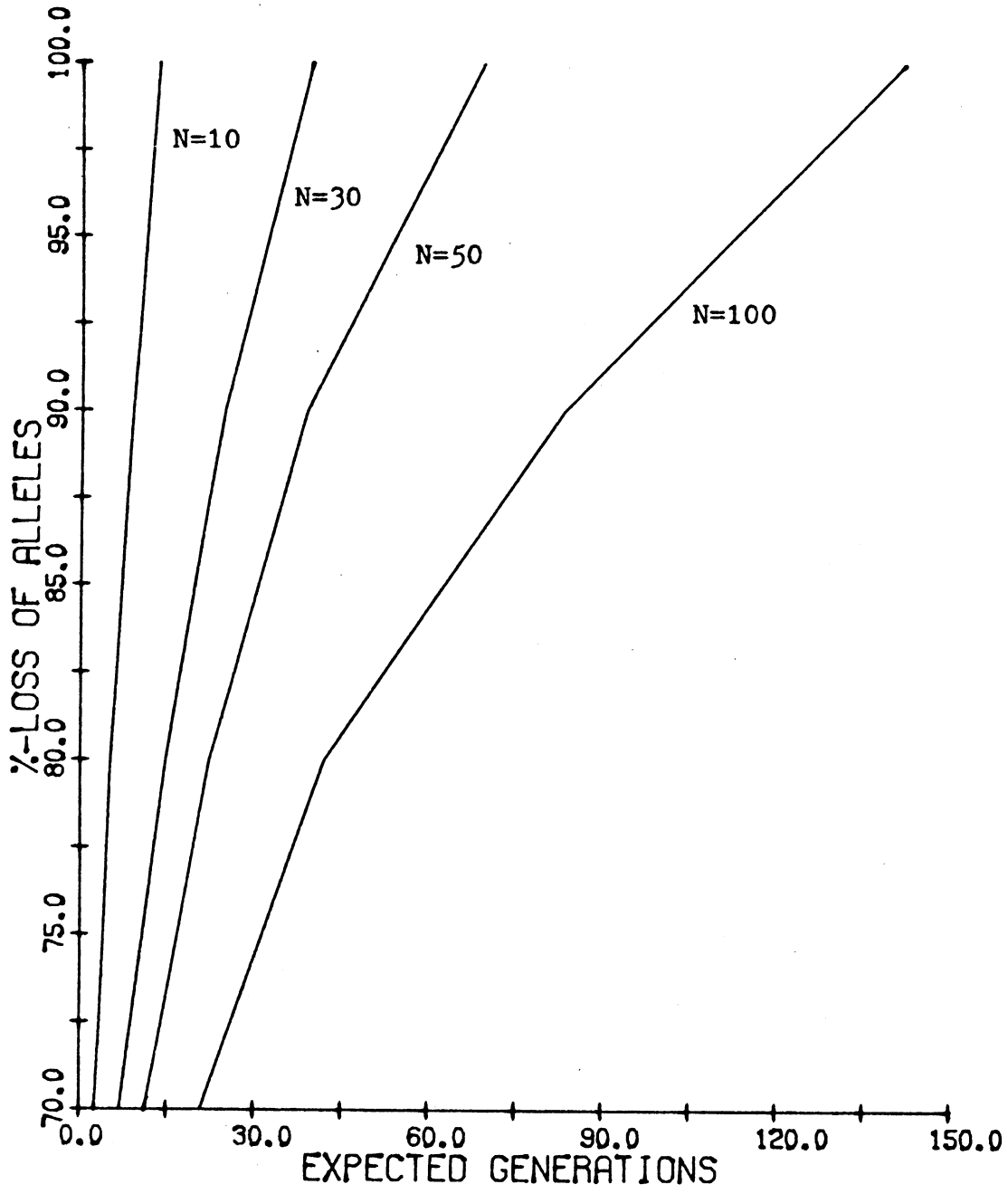


Figure 3.2: The rate of allele loss due to genetic drift as a function of population size.

due to genetic drift and allow smaller population sizes. To evaluate this alternative, a mutation operator can be easily added to the Markov process discussed above. While the addition of mutation complicates the expected value computations even more, it is intuitively clear that the states 0 and N are no longer absorbing states. Figures 3.3 and 3.4 illustrate the effects of several mutation rates on the simulated Markov process with populations of size 50 and 100. As might be expected, one mutation per generation is sufficient to increase the expected first-entry times to the 100%-loss state beyond the bounds of a practical adaptive interval. However, the effects on the first entry times to the other states are much less pronounced. The expected first entry to a 90%-loss state with a population of size 50 is still less than 60 generations.

3.4 The Effects of Population Size on R1

The analysis of the preceding sections has yielded considerable insight into the behavior of plan R1. As we have seen, the loss of alleles from A(t) corresponds to a dramatic decrease in the space being searched by R1. If this reduced space does not contain the optimum, we have seen that R1 will very likely remain on a non-optimal plateau with mutation providing only a low-probability chance of escape. Since this is the case, it is critical that alleles are lost only if their com-

FIG 3.3: GENETIC DRIFT VARYING MUTATION

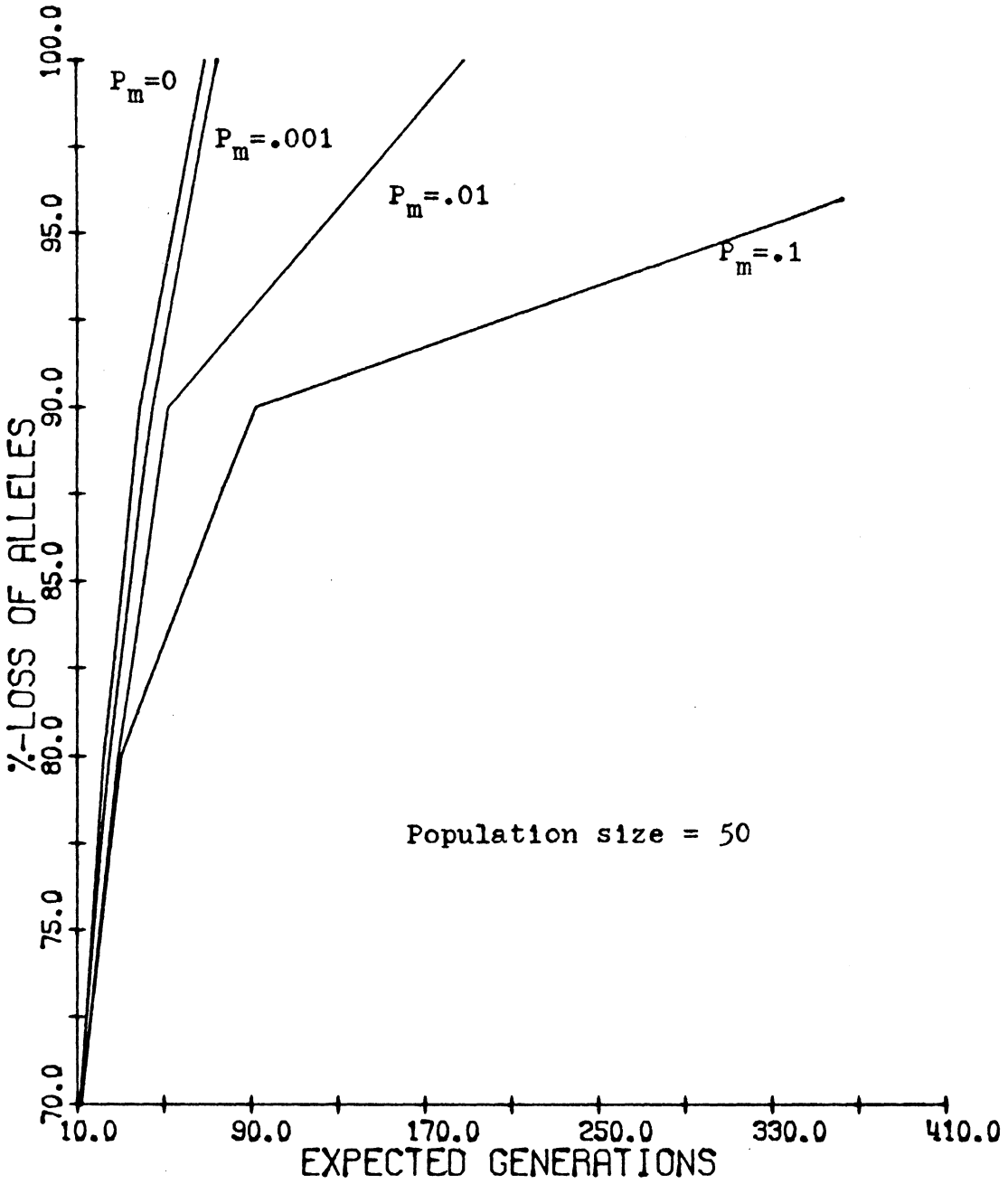


Figure 3.3: The rate of allele loss due to genetic drift as a function of the mutation rate.

FIG 3.4: GENETIC DRIFT VARYING MUTATION

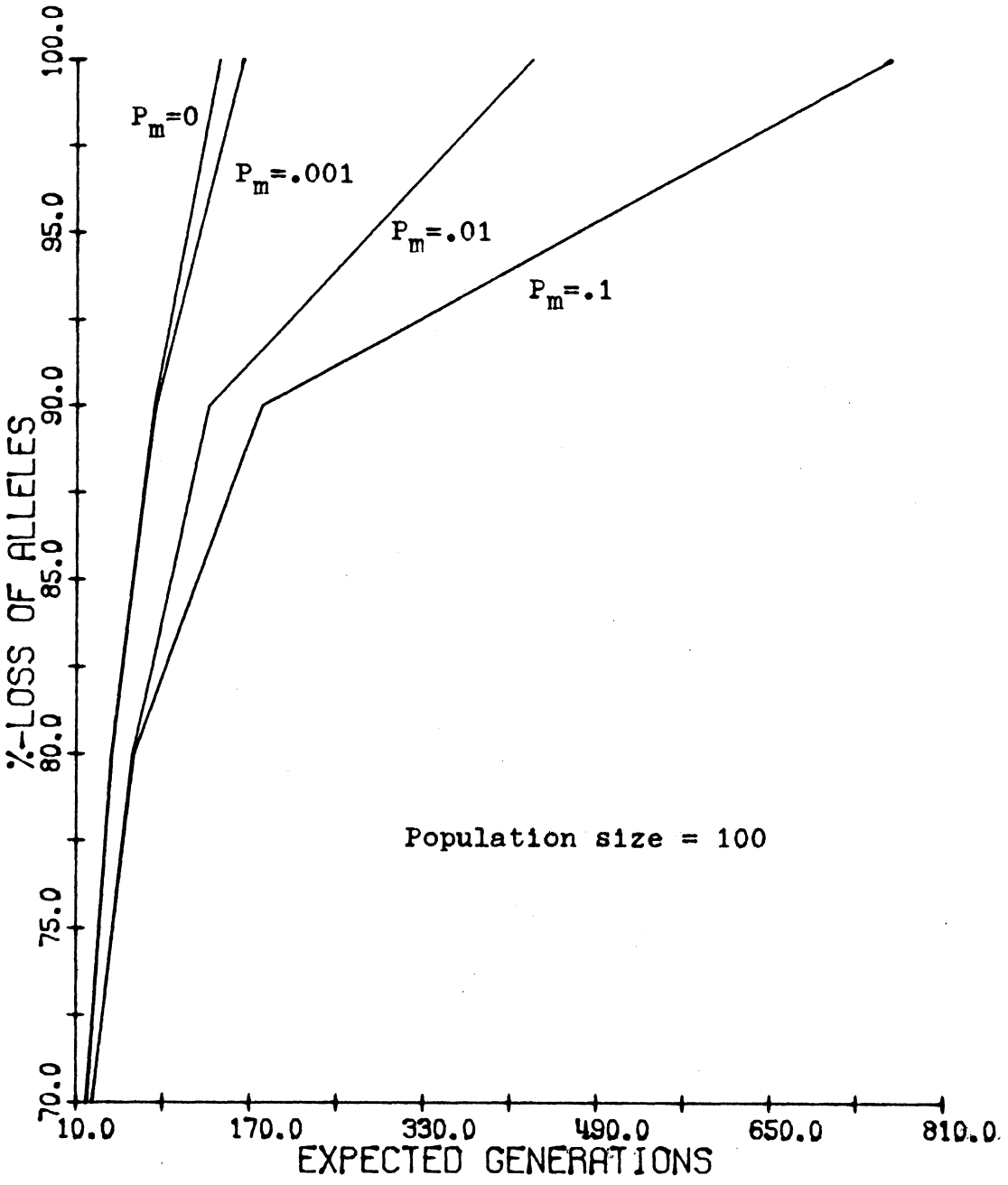


Figure 3.4: The rate of allele loss due to genetic drift as a function of the mutation rate.

petitors are in fact better. However, on test function F1, alleles are lost even when there is no selection differential and R1 converges to a non-optimal plateau. The preceding section suggests that this may be due in part to stochastic effects and suggests two approaches for alleviating the problem: changing the population size and the mutation rate of plan R1. In this section we explore the effects of population size on the behavior of R1.

Recall from appendix C that plan R1 maintained a population of 50 individuals and a mutation rate of .001. Note further from table 3.1 that 100 generations had elapsed by the time R1 generated A(3000). Referring back to figure 3.3, we see that for a population size of 50 and a mutation rate of .001, the expected number of generations for the simulated Markov process to enter the 100%-loss state was approximately 75. Hence, the allele loss observed in A(3000) could be due entirely to genetic drift. If this is the case, increasing the population size should reduce considerably the rate of allele loss on test function F1. Whether or not this will also improve the performance of R1 on F1 is not quite so obvious. Clearly, premature convergence is to be avoided. However, increasing the population size may also have the effect of slowing down the rate of convergence beyond acceptable bounds.

In order to evaluate these hypotheses, the behavior

of R1 on F1 was also observed with population sizes of 100 and 200, leaving the mutation rate unchanged at .001. Figure 3.5 contrasts the average rate of allele loss for the various population sizes. As expected, increasing the population size reduces the allele loss considerably over the interval of observation. The effect here is heightened by the fact that the time scale is in terms of the number of trials rather than the number of generations. That is, the allele loss was reduced in part because fewer generations (and hence fewer stochastic effects) were involved in generating the same number of sample points.

So we see that the problem of premature loss of alleles can be effectively removed by increasing the population size maintained by R1. However, it remains to be seen what effect this has on the performance of R1. Recall from chapter 1 that two local measures of adaptive performance were defined for functions in E:

$$\text{off-line} \quad x_e^*(s) = \frac{1}{T} \sum_{t=1}^{T} f_e^*(t)$$

$$\text{and on-line} \quad x_e(s) = \frac{1}{T} \sum_{t=1}^{T} f_e(t)$$

where $f_e(t)$ is the performance rating given to the sample solution generated by the adaptive plans for evaluation

FIG 3.5: R1 ALLELE LOSS VARYING POPULATION SIZE

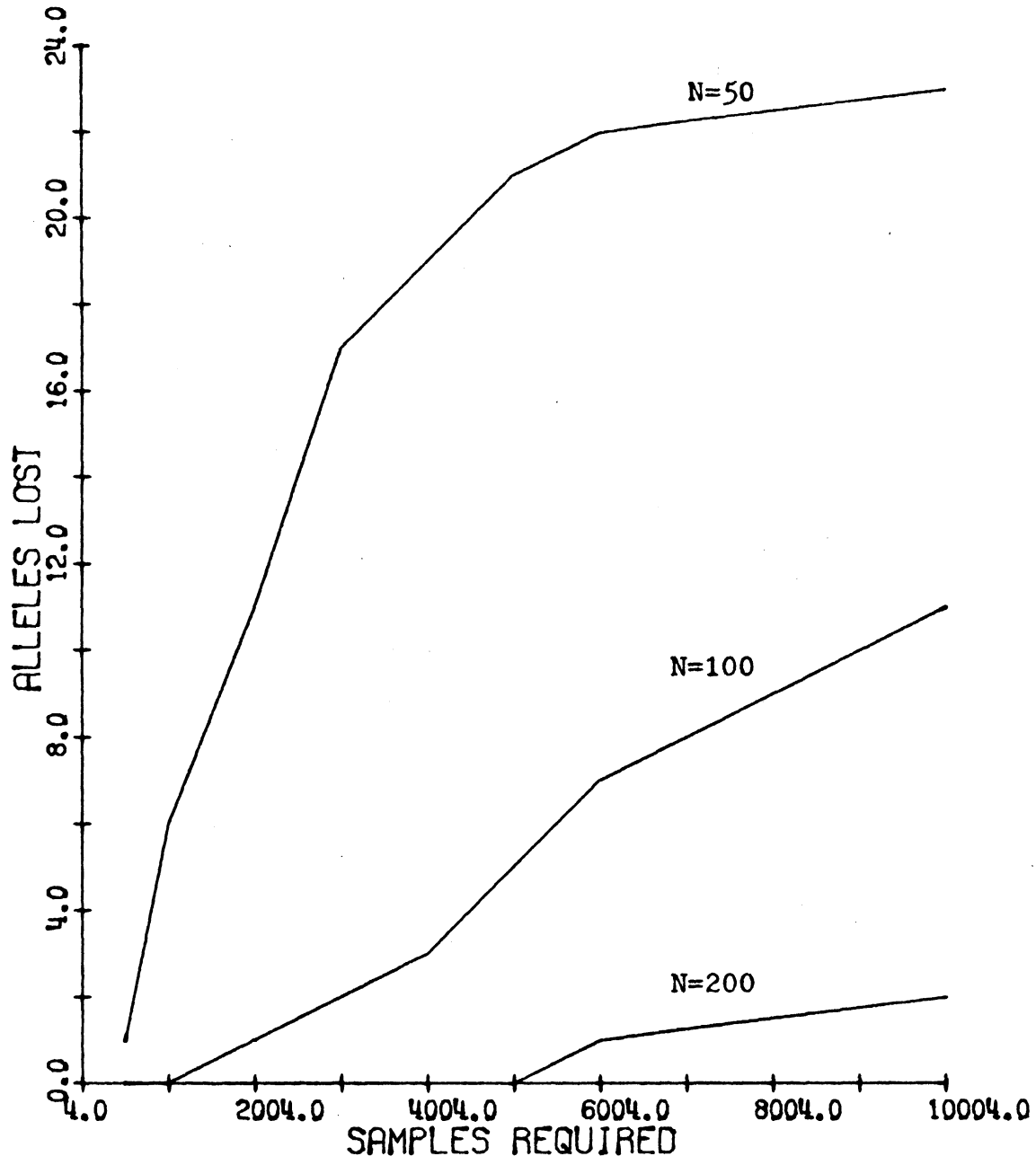


Figure 3.5: The effects of population size on allele loss for R1 on test function F1.

at time t, and where $f_e^*(t)$ is defined by:

$$f_e^*(t) = \min\left\{f_e(1), f_e(2), \ldots, f_e(t)\right\}$$

Figure 3.6 illustrates the effects of population size on $F1^*(t)$. The tradeoff here is clear. Initially R1(50) outperforms the larger populations, but converges prematurely to a non-optimal plateau. R1(100) and R1(200) respond more slowly but yield better long-term performance.

Figure 3.7 illustrates the effects of population size on F1(t). Here the interval required for the tradeoff to become apparent is considerably longer with R1(50) outperforming the others over the first 25,000 trials.

At this point a few words of explanation about the notation being developed in chapter 3 is in order. As we shall see, genetic plan R1 is really a family of plans defined by such parameters as the population size, the mutation rate, and so on. Specific members of this family will be designated by notation of the form R1(X,Y,Z) specifying the actual parameter values. For purposes of clarity, two notational conveniences will be used. First, parameters which have not yet been introduced into the discussion will be suppressed. So, for example, in the preceding paragraph we refer to R1(X) even though by the end of the chapter four parameters will have been defined. Secondly, in a particular context where it is clear that only one

FIG 3.6: R1 OFF-LINE VARYING POPULATION SIZE

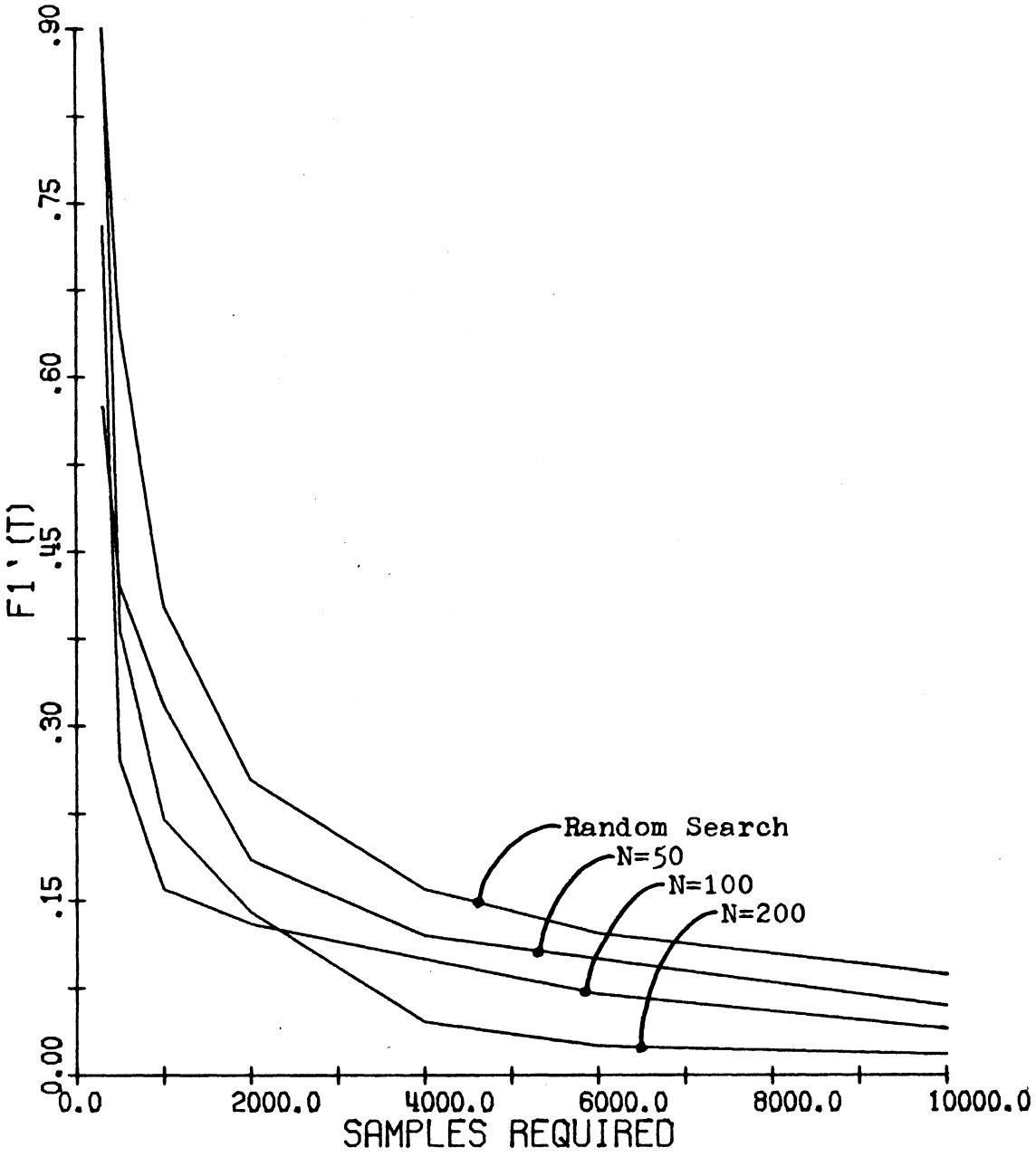


Figure 3.6: The effects of population size on off-line performance of R1 on test function F1.

FIG 3.7: R1 ON-LINE VARYING POPULATION SIZE

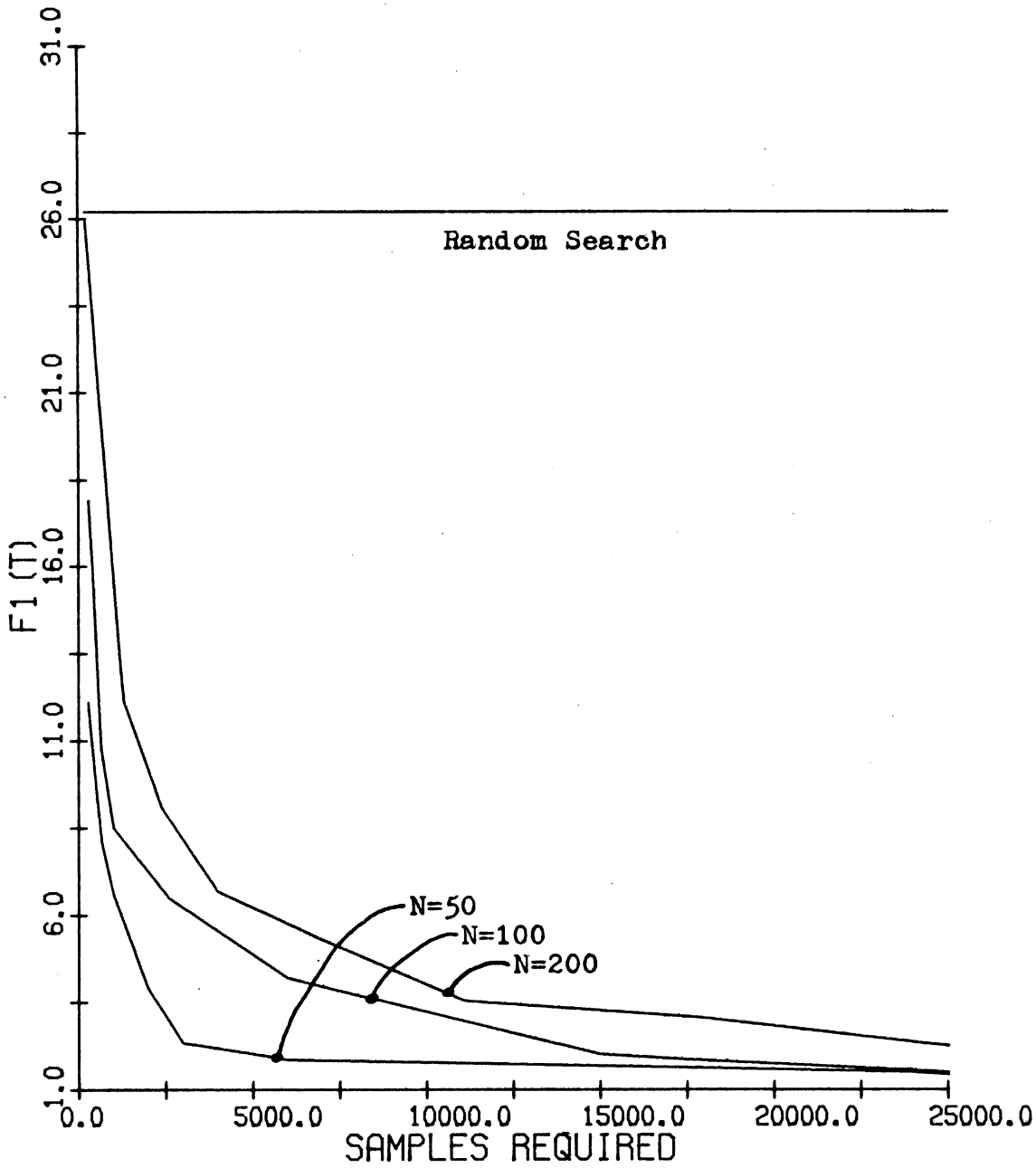


Figure 3.7: The effects of population size on on-line performance of R1 on test function F1.

parameter is under study, the values of the other parameters will be suppressed. So, for example, we may refer to R1(Z) in situations in which X and Y are clearly fixed.

3.5 The Effects of Mutation Rate on R1

In this section we explore the second alternative approach to the problem of premature allele loss on F1, namely, changing the mutation rate for R1. Recall from appendix C that R1 maintained a population of 50 individuals and a mutation rate of .001. Referring back to figure 3.3 we note that a considerable reduction in allele loss was achieved in the simulated Markov process with a population size of 50 by increasing the mutation rate. This suggests that the allele loss in R1 might also be reduced by increasing the mutation rate. How an increase in the mutation rate will affect the performance of R1 is not so obvious. Clearly, reducing the premature allele loss will increase the potential for improving long-term performance as we saw in the previous section. However, recall that in chapter 2 we were able to neglect the effects of mutation (at .001) on the near-optimal sampling rate of R1. As we increase the rate of mutation, we increase its effects on sampling which, in turn, may negatively affect the performance of R1.

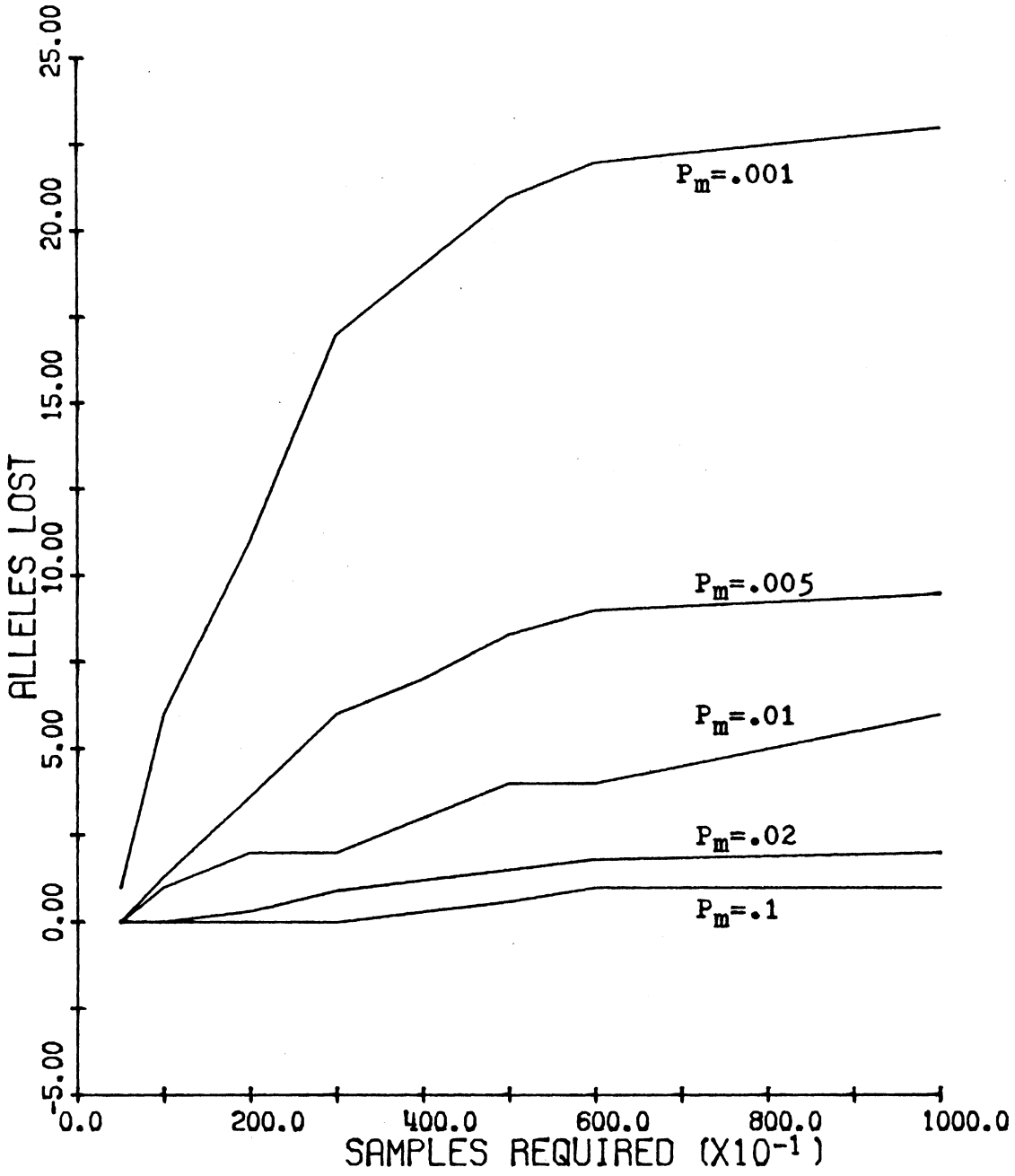
In order to evaluate these hypotheses, the behavior of R1 on F1 was observed with mutation rates of .005, .01,

.02, and .1, leaving the population size unchanged at 50. Figure 3.8 contrasts the average rate of allele loss for the various mutation rates. As expected, increasing the mutation rate reduces considerably the allele loss over the interval of observation. Clearly, the problem of premature allele loss can be solved by raising the mutation rate. However, its effect on the performance of R1 must also be considered.

Figure 3.9 illustrates the effects of increasing the mutation rate on the off-line performance of R1 on F1. Increasing the mutation rate has the effect of improving initial performance. As noted earlier, a mutation rate of the same order of magnitude as 1/POP_SIZE seems to be about the best setting. Increasing the rate more definitely degrades off-line performance. These observations tend to confirm our intuition about R1. With too low a mutation rate, the performance of R1 is degraded by the premature loss of alleles. With too high a mutation rate, the performance is degraded by the sub-optimal allocation of trials to competing hyperplanes.

Figure 3.10 illustrates the effects of increasing the mutation rate on the on-line performance of R1 on F1. Here the effects of mutation are clear. When every trial counts in the performance rating, any increase in the application of a random search operator like mutation has a degrading effect on the performance of R1.

FIG 3.8: R1 ALLELE LOSS VARYING MUTATION



3.8: The effects of mutation rate on allele loss for R1 on test function F1.

FIG 3.9: R1 OFF-LINE VARYING MUTATION

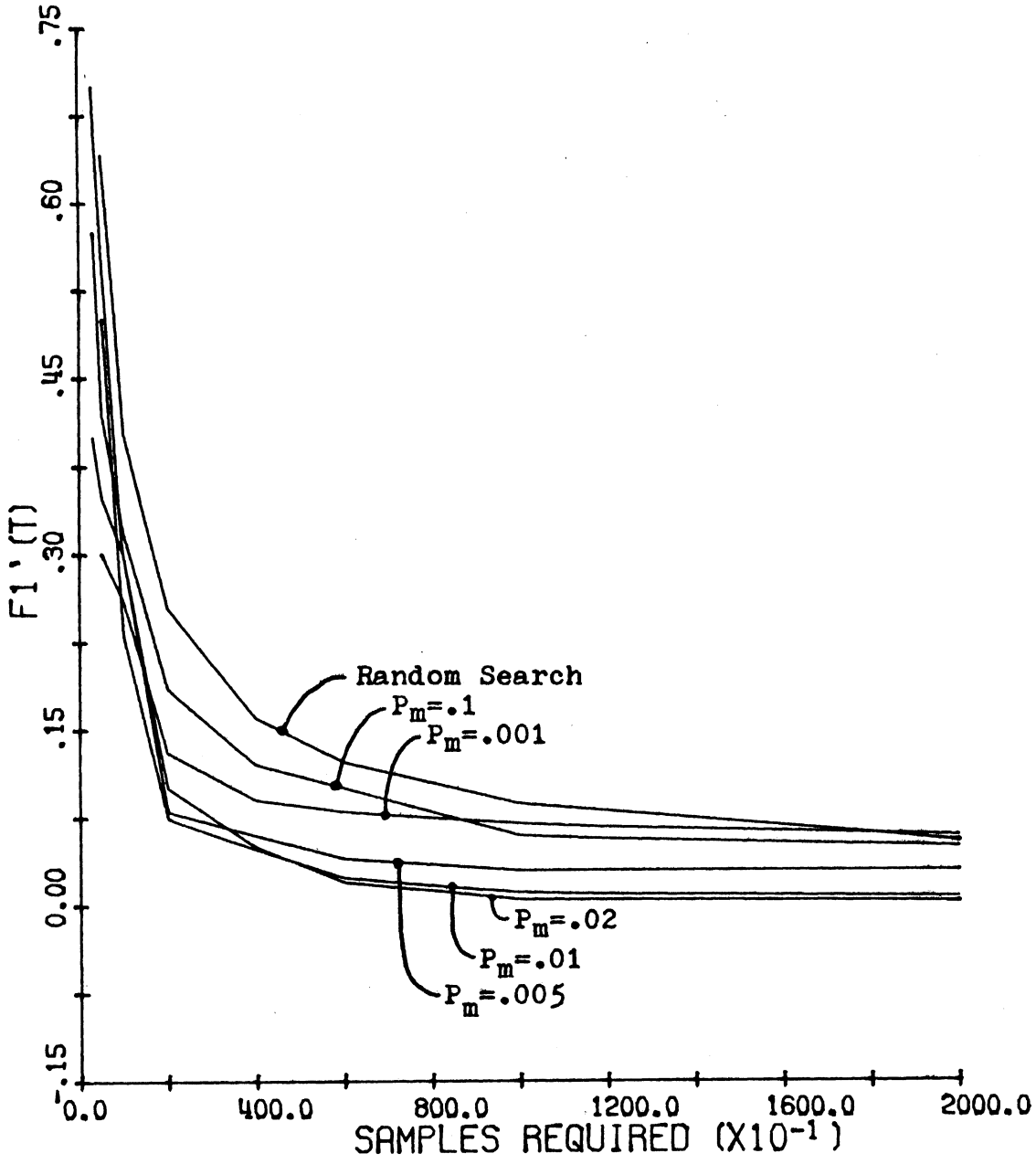


Figure 3.9: The effects of mutation rate on off-line performance of R1 on test function F1.

FIG 3.10: R1 ON-LINE VARYING MUTATION

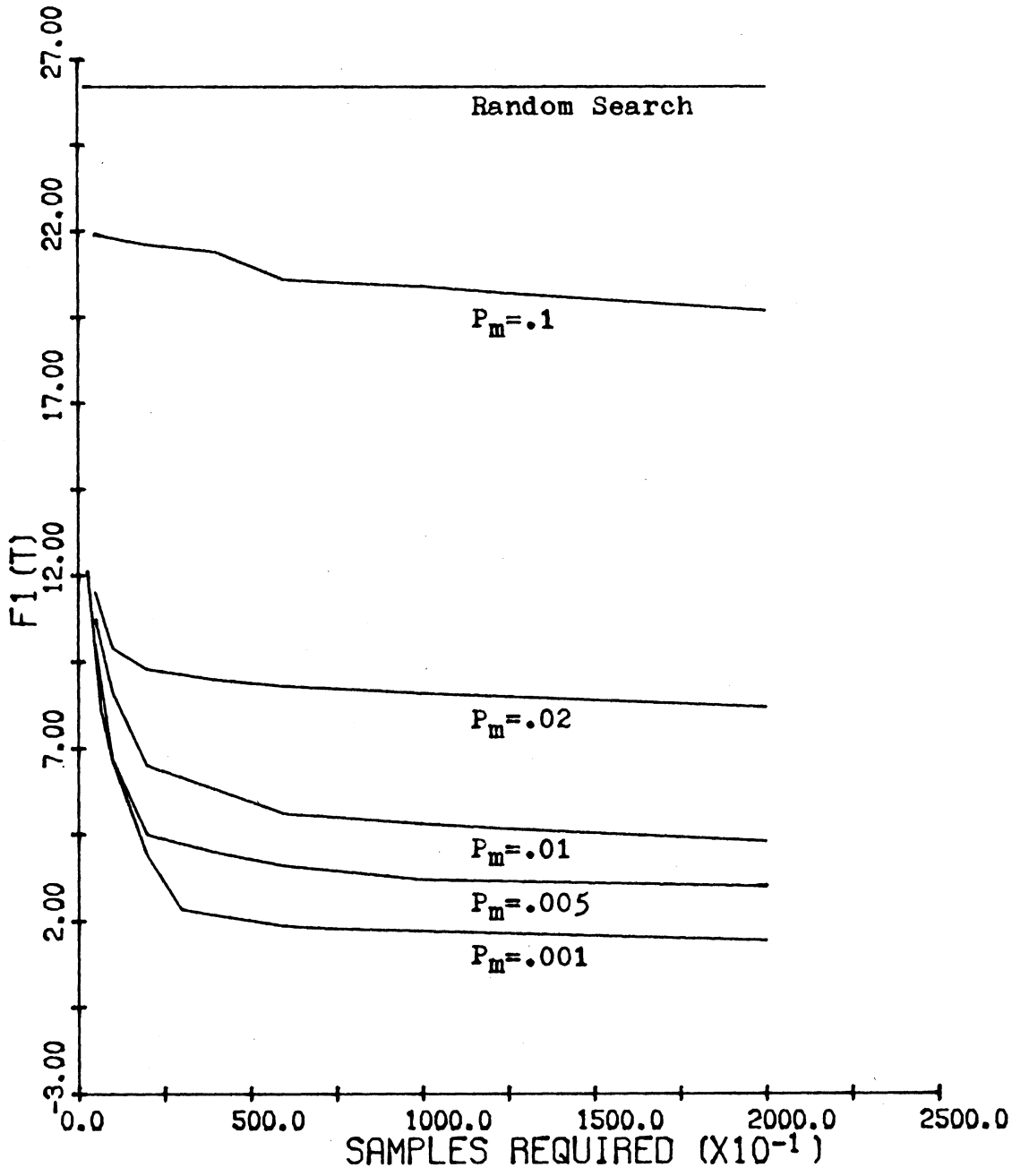


Figure 3.10: The effects of mutation rate on on-line performance of R1 on test function F1.

3.6 The Effects of Crossover Rate on R1

As described in appendix C, plan R1 produces an individual for the next generation A(t+1) by selecting two parents, applying crossover to produce an offspring, and then applying mutation to each gene position with probability P_m. In this section we explore the effects of reducing the number of individuals in A(t+1) produced by crossover. This variation is easily accomplished within the framework of R1 as follows:

Do I=1 to POP_SIZE:

- select an individual a_{it} from A(t) using the selection probabilities.
- with probability P_c apply crossover to a_{it} by selecting a mate from A(t) using the selection probabilities and choosing a crossover point.
- apply mutation at each gene position with probability P_m.

Since crossover is the principle search operator in R1, the effect of lowering the crossover rate is to reduce the number of new trials per generation. This reduction should in turn heighten the stochastic effects noted in the previous sections and increase the rate of allele loss generated by R1 on F1. As a consequence, we would expect the performance of R1 to be adversely affected, since fewer trials will have been allocated before the allele loss has reduced A(t) to a nearly uniform population.

In order to evaluate these hypotheses, the behavior

of R1 on test function F1 was observed with crossover rates of P_c = .8, .6, and .4, leaving the population size and mutation rate unchanged at N = 50 and P_m = .001.

Figure 3.11 compares the rate of allele loss for R1 on F1 as a function of the crossover rate. As expected, the rate of allele loss increases as the crossover rate decreases. Figures 3.12 and 3.13 compare the off-line and on-line performance curves for R1 on F1 as a function of the crossover rate. Here the results were unexpected. In spite of the fact that the rate of allele loss is increased, lowering the crossover rate initially improved performance. Only when the crossover rate was lowered to .4 was any negative effect on performance observed.

In an attempt to understand this phenomenon, consider for a moment the effects of the two genetic operators: crossover and mutation. Until the allele loss in A(t) is extensive, applying crossover to two individuals generally produces an offspring quite distinct from either parent. On the other hand, applying mutation to an individual at the rate of .001 changes on the average $\ell*(.001)$ alleles. In the case of test function F1, the number of genes per individual is ℓ=30, so that crossover affects on the average .03 gene positions. So we see that with only these two genetic operators, lowering the crossover rate in R1 has the effect of increasing the likelihood that members of A(t) will produce an offspring nearly identical to themselves, if not identical. Since parents are

FIG 3.11: R1 ALLELE LOSS VARYING CROSSOVER

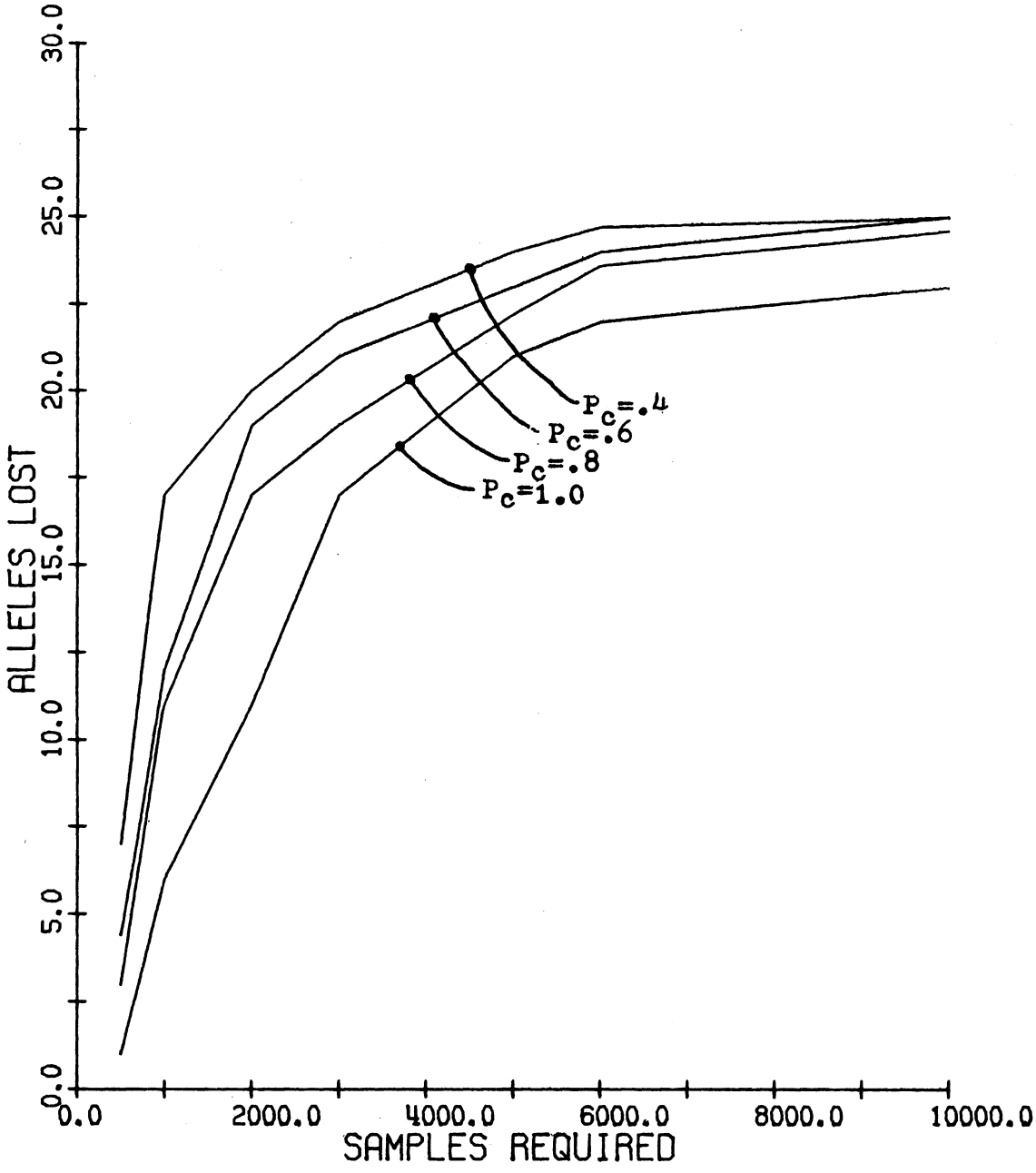


Figure 3.11: The effects of crossover rate on allele loss for R1 on test function F1.

FIG 3.12: R1 OFF-LINE VARYING CROSSOVER

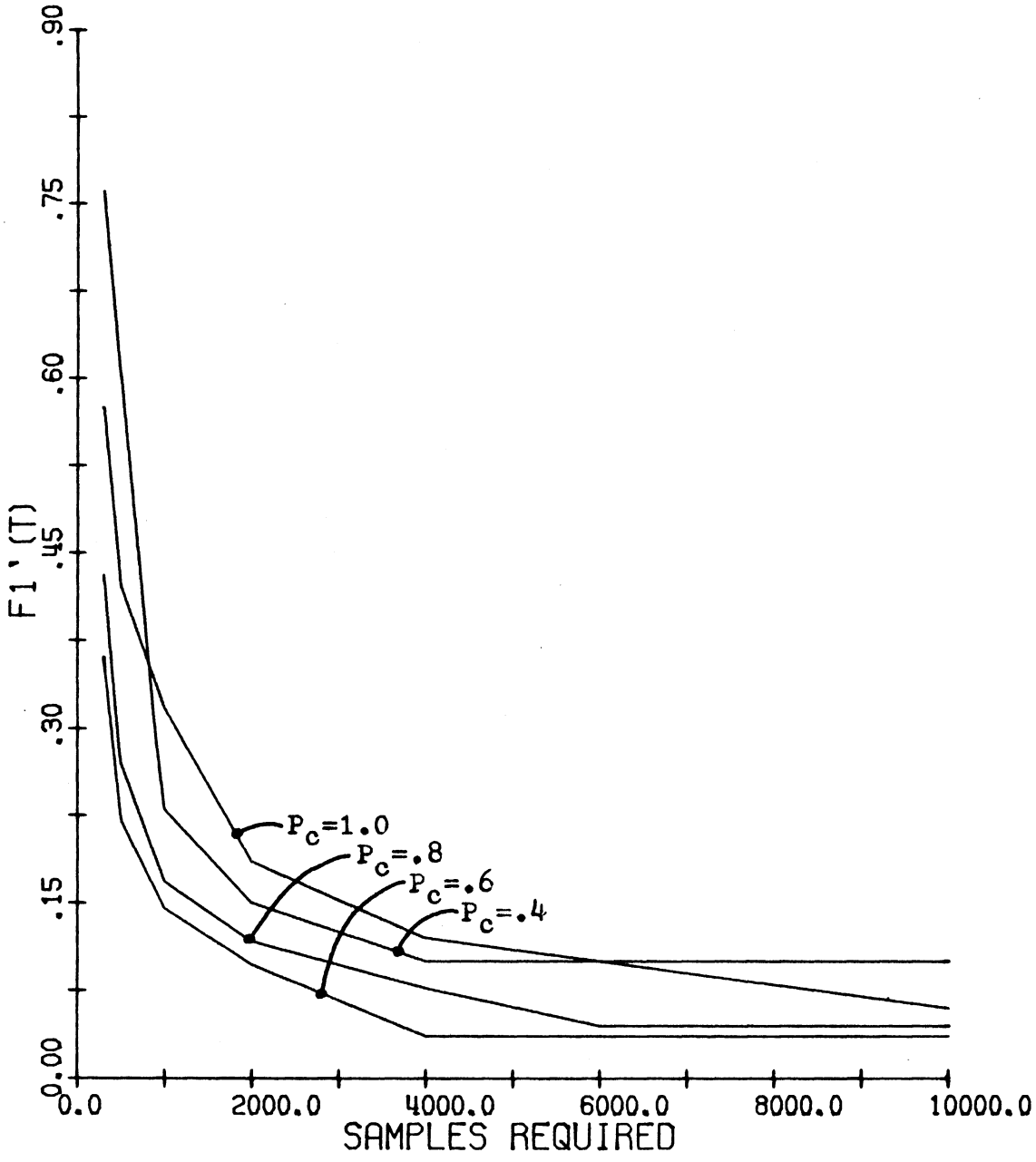


Figure 3.12: The effects of crossover rate on off-line performance of R1 on test function F1.

FIG 3.13: R1 ON-LINE VARYING CROSSOVER

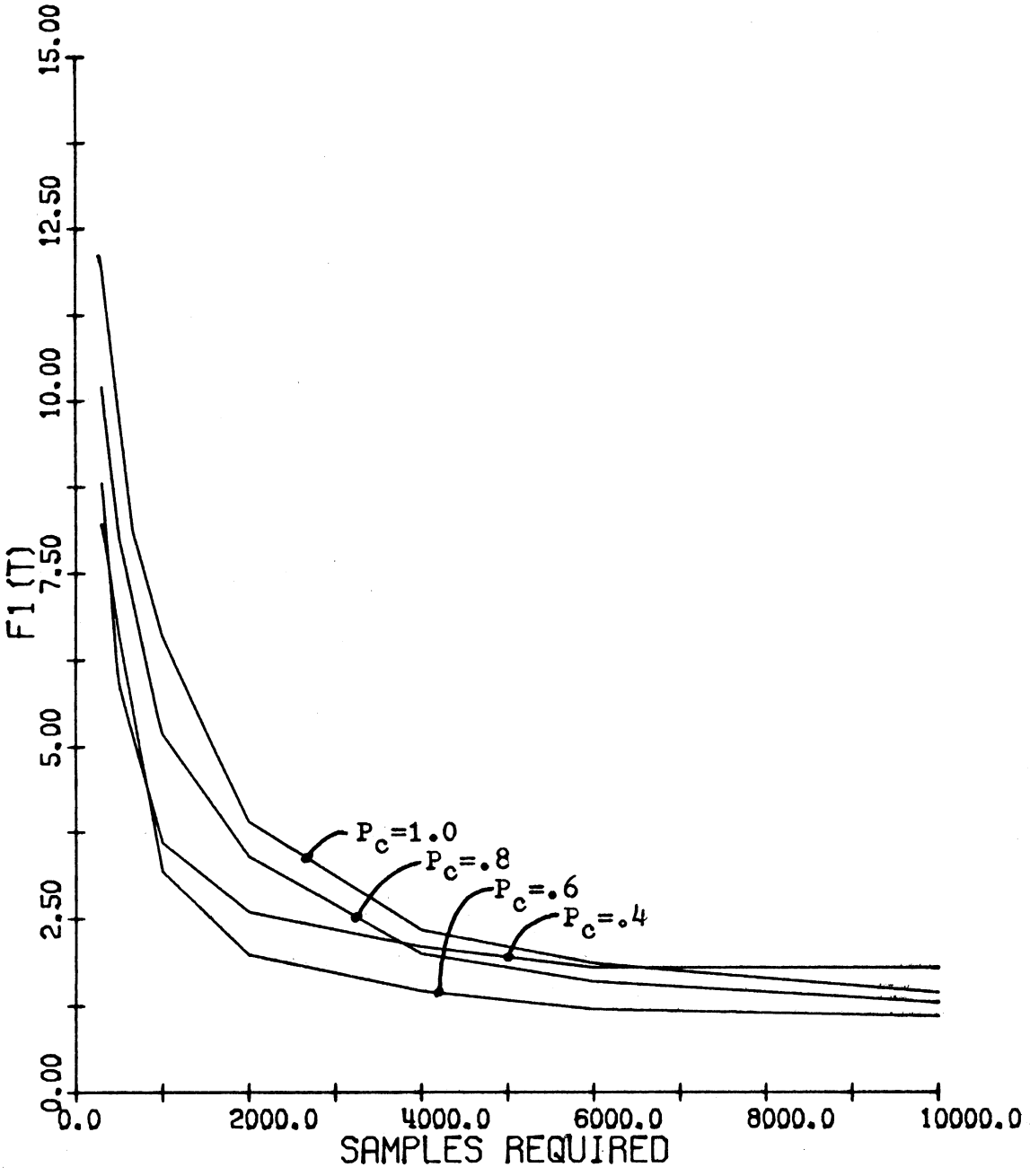


Figure 3.13: The effects of crossover rate on on-line performance of R1 on test function F1.

selected on the basis of performance, the result is to increase the probability of high-performance individuals surviving into the next generation. Here again we encounter the delicate tradeoff between further exploration and preserving the status quo. Applying crossover at the rate of 1.0 seems to be too high a sampling rate for R1(50,.001). High-performance individuals are discarded faster than crossover can produce improvements, terminating with the usual premature convergence due to allele loss. On the other had, a crossover rate of .4 seems to be too low a sampling rate for R1(50,.001). Too little exploration combined with the increased rate of allele loss causes rapid convergenge to a non-optimal plateau.

3.7 The Effects of Generation Gap on R1

Recall that plan R1 is designed to produce the next generation A(t+1) by replacing all N individuals from A(t). A genetic model of this type is described as having non-overlapping generations; that is, parents do not exist simultaneously with their offspring. It is not immediately clear whether non-overlapping generations are good or bad in an artificial genetic adaptive model. From an implementation point of view, the distinction poses the classic tradeoff between storage and cpu time. Non-overlapping models require storage for two populations: A(t) and A(t+1). If generations overlap, less storage is required, but

more generations are required (recomputing selection probabilities) to produce the same number of trials. In this section we ignore the time-space tradeoff and explore the effect of overlapping generations on the performance of R1.

Overlapping generations can be incorporated into R1 by adding a new parameter called the generation gap G which specifies the fraction of A(t+1) to be generated via the genetic operators. Obviously, G must lie in the range $0 \leq G \leq 1$ with G = 1 the default value used in the previous simulations. If $G < 1$, the remaining positions in A(t+1) are filled by selecting individuals from A(t) <u>without</u> replacement using a uniform distribution. As before, we inquire as to the expected number of offspring produced by an individual a_{it} in A(t). If we assume that the selection probabilities do not change much over the life-time of an individual, then on any particular generation the expected number of offspring from a_{it} is given by:

$$(N*G) * p(a_{it})$$

where N is the population size and $p(a_{it})$ is the probability of selecting a_{it}. The number of generations a_{it} is expected to survive is simply the waiting time to extinction. Each generation a_{it} has a probability G of disappearing; hence, the waiting time is $\frac{1}{G}$ and the total number of offspring produced by a_{it} is given by:

$$(\frac{N*G}{G})*p(a_{it}) = N*p(a_{it})$$

which is the same as the non-overlapping model.

On the basis of our experiences with the crossover rate, we would expect that reducing the generation gap should increase the rate of allele loss since fewer trials are made per generation. Its effect on performance is not quite so obvious. Clearly, the reduced sampling rate should improve the performance of R1(50,.001) on F1 as it did in the case of crossover. However, note that the individuals which are likely to survive into the next generation are selected at random, rather than on the basis of performance. This should reduce the extent of the improvement observed when the crossover rate was reduced.

In order to evaluate these hypotheses, the behavior of R1 was observed on F1 with generation gaps of .8, .6, and .4 leaving the population size, mutation rate, and crossover rate unchanged at N = 50, P_m = .001 and P_c = 1.0.

Figure 3.14 compares the rate of allele loss for R1 on F1 as a function of the generation gap. As expected, the rate of allele loss increases as the generation gap decreases. Figures 3.15 and 3.16 compare the off-line and on-line performance curves for R1 on F1 as a function of the generation gap. As expected, lowering the generation gap provides an initial improvement in performance

FIG 3.14: R1 ALLELE LOSS VARYING GENERATION GAP

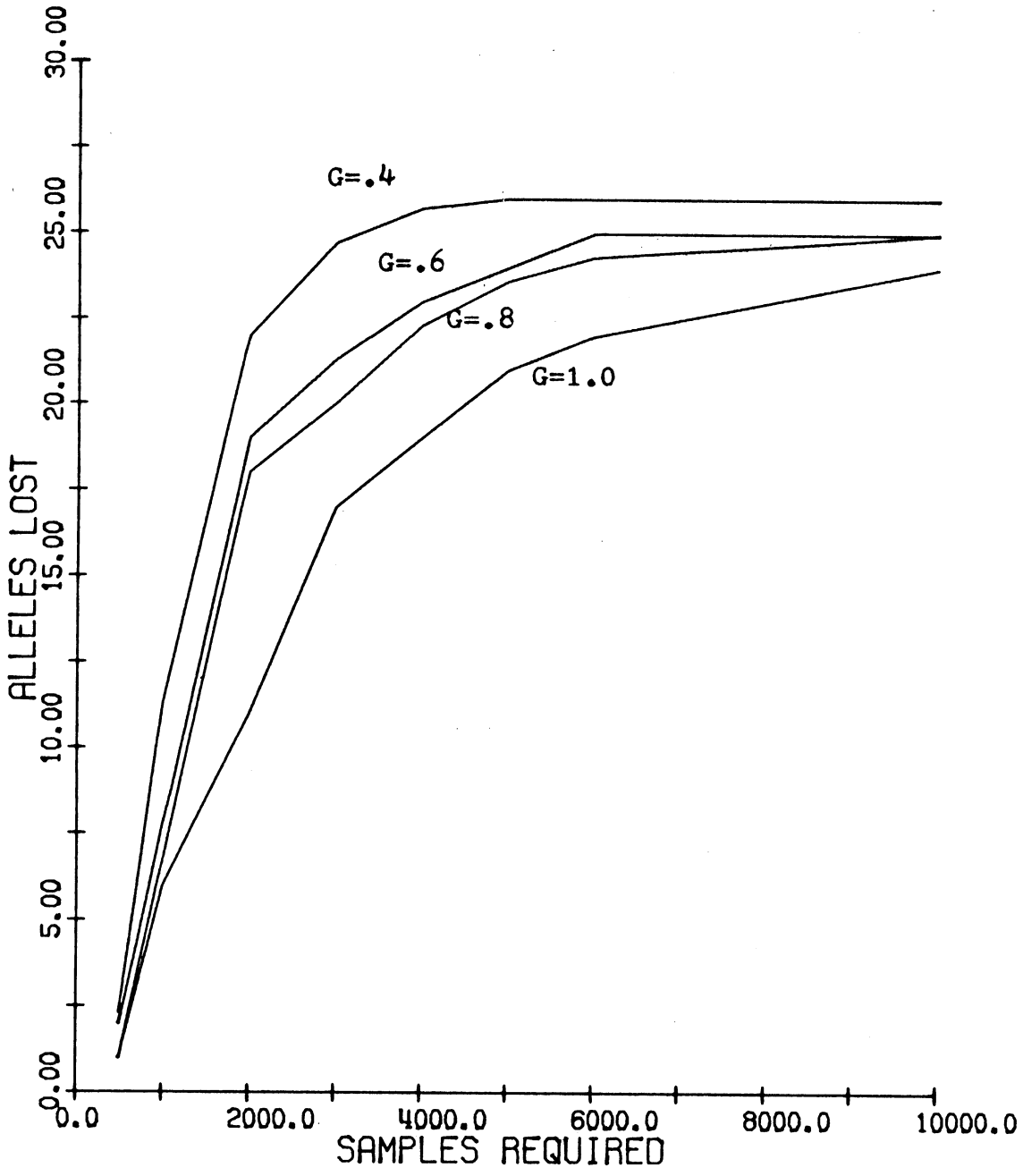


Figure 3.14: The effects of generation gap on allele loss of R1 on test function F1.

FIG 3.15: R1 OFF-LINE VARYING GENERATION GAP

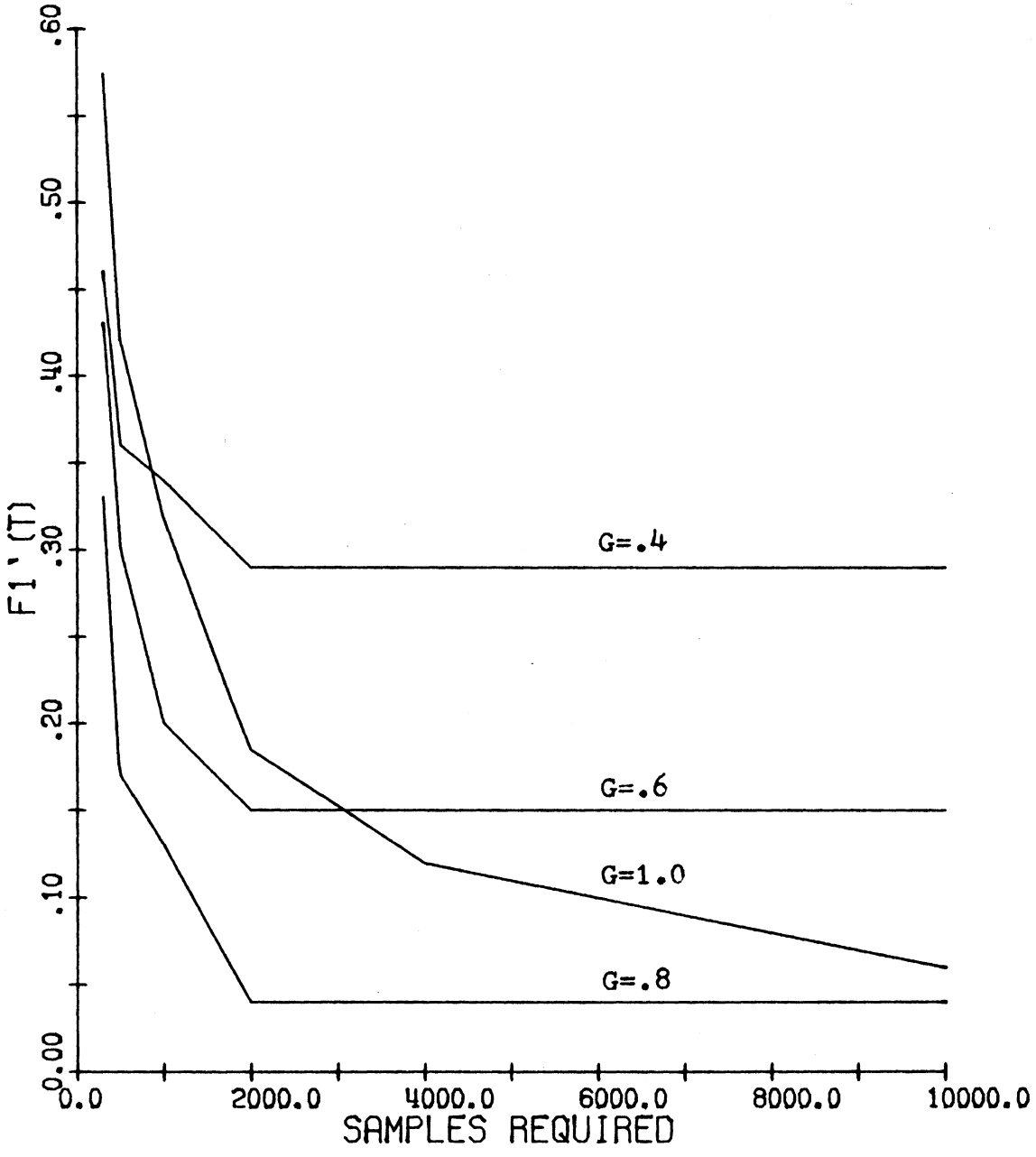


Figure 3.15: The effects of generation gap on off-line performance of R1 on test function F1.

FIG 3.16: R1 ON-LINE VARYING GENERATION GAP

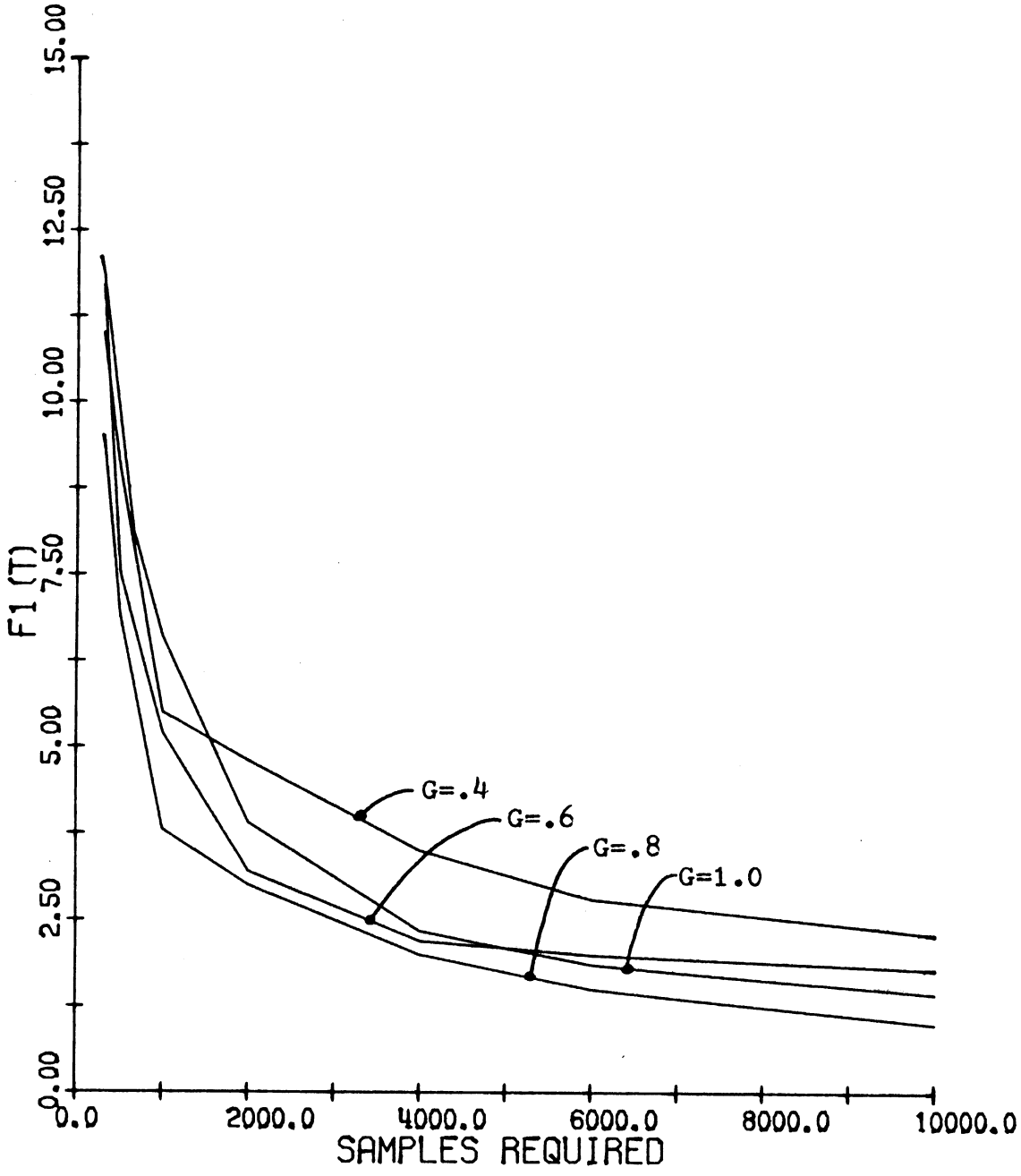


Figure 3.16: The effects of generation gap on on-line performance of R1 on test function F1.

but also produces an earlier convergence to a non-optimal plateau, the improvement being considerably less dramatic than that generated by a corresponding reduction in the crossover rate.

3.8 Improving the Performance of R1 on F1

In the preceding sections we have isolated several parameters in the definition of plan R1 and have explored the effects of independently changing these parameters on the behavior of R1 on test function F1. The motivation for these studies was to gain further insight into how R1 operates and, in particular, to analyze the problem of premature convergence to a non-optimal plateau. As we have seen, no one of the parameters studied both satisfactorily resolves the problem of premature convergence and substantially improves the performance of R1 on F1. In this section we explore the possibility of resolving these problems by changing various combinations of parameter settings for R1.

In this chapter R1 has evolved into a family of genetic plans, a member of which is selected by specifying the values of four parameters: the population size N, the mutation rate P_m, the crossover rate P_c, and the generation gap G. Ideally, we would like to apply optimization techniques to the space of algorithms defined by these parameters and optimize with respect to premature allele loss, off-line, and on-line performance.

In reality, however, this approach is prohibited by the cost involved in analyzing the behavior of a single member of this family. Because each plan is a stochastic process, at least 5 (and often more) simulations are required to produce analysis measurements within reasonable standard error limits. In terms of present university rates, this can mean a cost of as much as $50 to evaluate a single plan on F1 alone. We will avoid this problem by applying the insight gained from the previous sections to the selection of a few well-chosen combinations of parameters to confirm and extend our understanding of the basic genetic plan R1.

We begin by noting that of the four parameters analyzed, reducing the crossover rate produced the single best improvement in the performance of R1 on F1, even though the allele loss rate actually increased in the process. This, we felt, was due to the reduced sampling rate effected by reducing the number of new individuals produced by crossover. As we observed, reducing the generation gap also lowered the sampling rate, but the improvement in performance is not as substantial as the corresponding reduction in crossover because of the difference in the kind of individual most likely to survive into the next generation. If these observations are correct, we would expect that sampling rates produced by a combination of reduced crossover rates and generation gaps should not be as effective in improving

the performance of R1 on F1 as the equivalent sampling rate produced by crossover alone.

In order to evaluate this hypothesis, the behavior of R1 on F1 was observed for 4 different combinations of crossover rates and generation gaps (P_c=.8,G=1.0), (P_c=.8,G=.8), (P_c=.6,G=1.0), and (P_c=.6,G=.8), holding the population size and mutation rate fixed at N=50 and P_m=.001. The performance curves generated by these combinations on test function F1 are illustrated in Figures 3.17 and 3.18, and they confirm our intuition about the behavior of plan R1. R1(.8,.8) performed better on F1 than R1(.8,1.0), but not as well as R1(.6,1.0) which has an equivalent sampling rate. As we saw previously, a combined sampling rate of less than .6 (in this case R1(.6,.8)) adversely affects the performance of R1 on F1. These observations suggest that reasonable settings for the crossover rate and generation gap of R1 are approximately P_c=.6 and G=1.0.

Alternatively, we saw that increasing the mutation rate improved considerably the allele loss rate, but the effects on performance were mixed. The best on-line performance was generated by a mutation rate of approximately P_m=1/N while any increase in mutation adversely affected on-line performance. This, we felt, was due to the fact that mutation is in fact an effective method for combatting premature allele loss and, hence, improving off-line performance. But because it accomplishes

FIG 3.17: R1(50,.001,X,Y) OFF-LINE PERFORMANCE

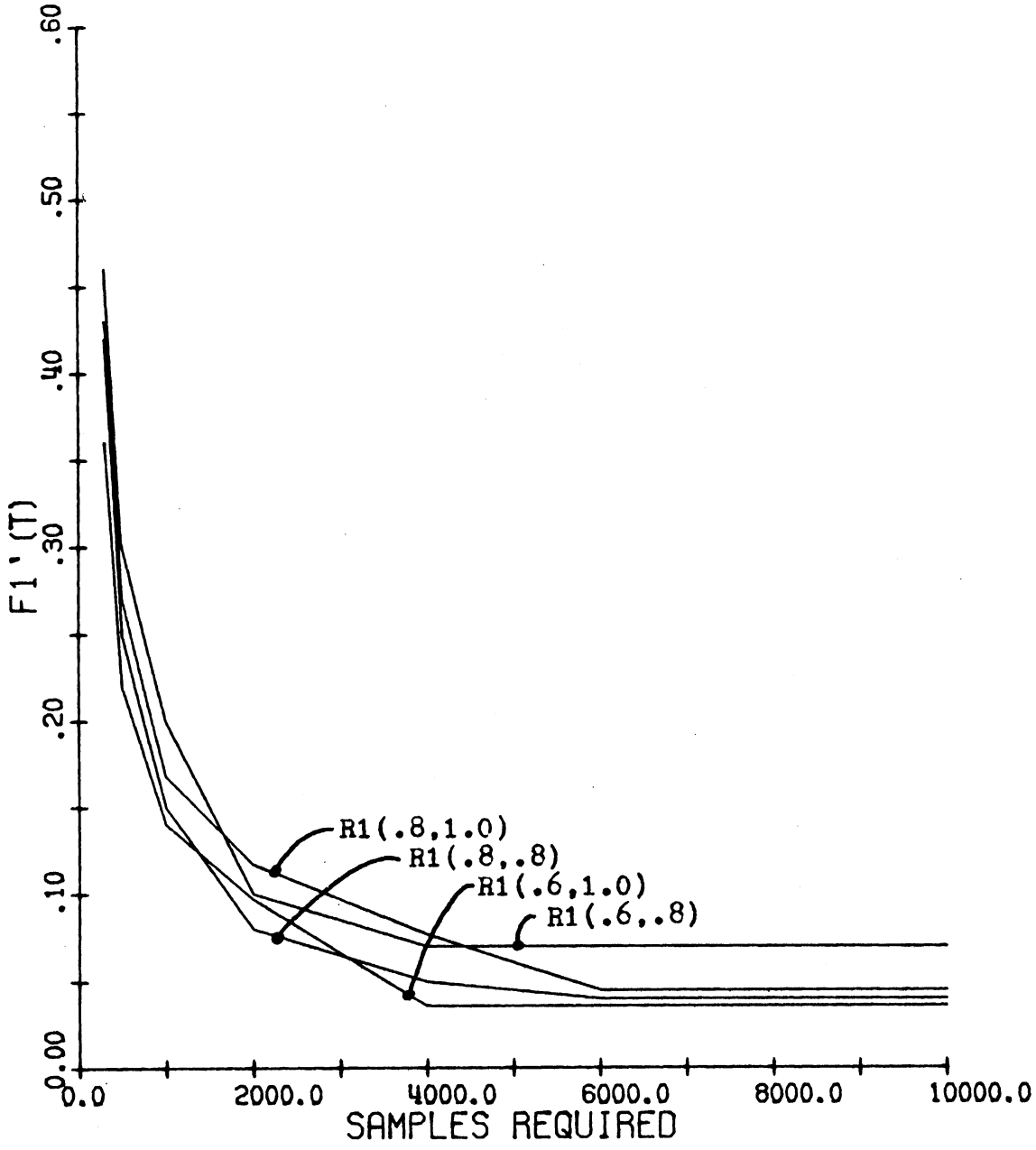


Figure 3.17: Off-line performance of R1 on F1 as a function of crossover rate and generation gap.

FIG 3.18: R1(50,.001,X,Y) ON-LINE PERFORMANCE

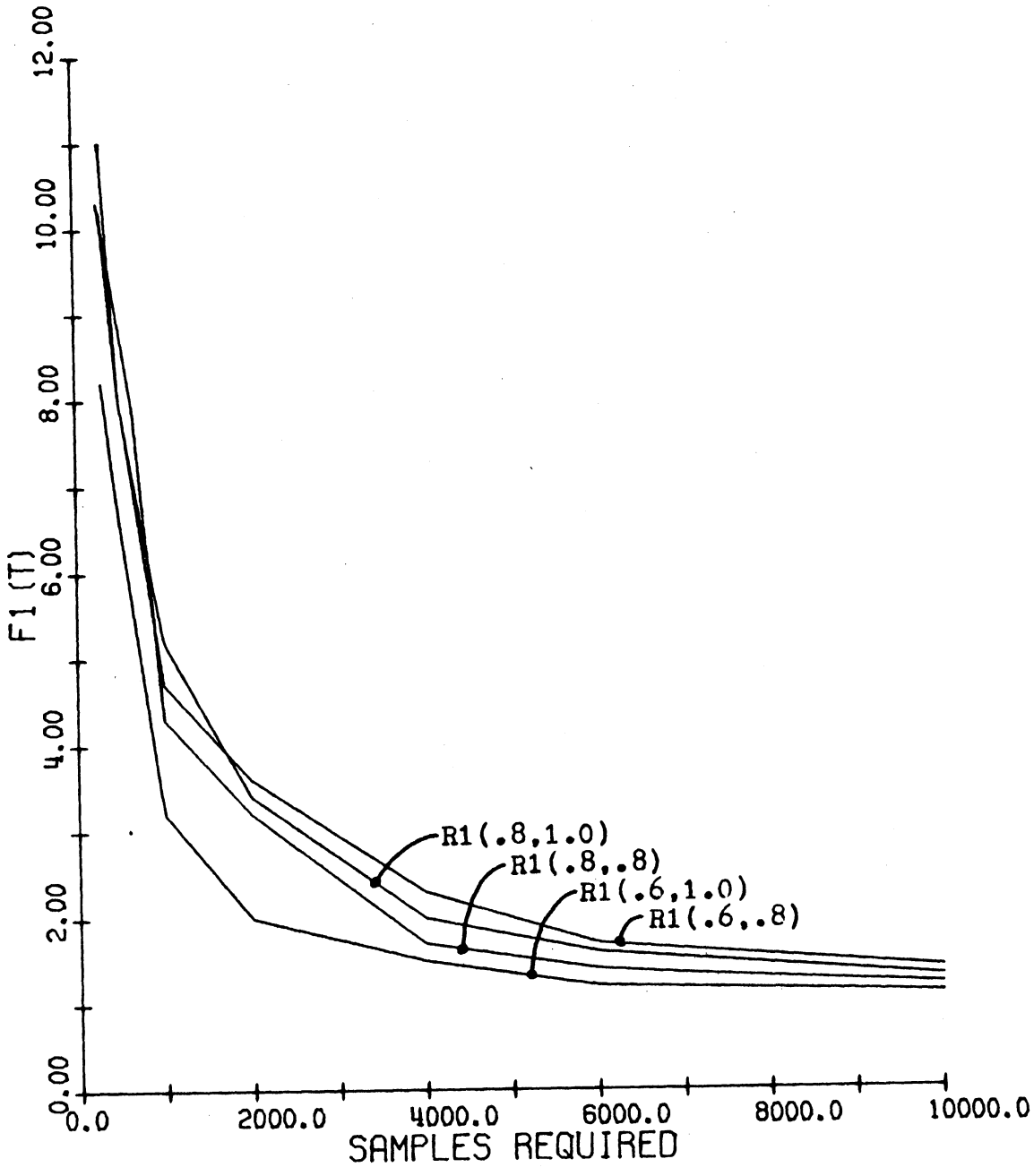


Figure 3.18: On-line performance of R1 on F1 as a function of crossover rate and generation gap.

this in its random sampling style, the price is paid in its adverse effect on on-line performance. If these observations are correct, we should expect to see the same kind of behavior changes produced by varying the mutation of R1(50,x,.6,1.0) as we saw with R1(50,x,1.0,1.0), but perhaps less dramatic changes since a crossover rate of .6 has already improved the performance curves.

To evaluate these hypotheses, the behavior of R1 on F1 was observed with mutation rates of P_m=.001, .01, and .1, leaving the population size, the crossover rate, and the generation gap fixed at N=50, P_c=.6, and G=1.0. Figures 3.19 and 3.20 compare the performance curves generated by the various mutation rates. These observations confirm our intuition about the effects of mutation on the performance of R1 and emphasize again the tradeoff between on-line and off-line performance.

Finally, we observed that increasing the population size reduced the rate of premature allele loss, but its effects on the performance of R1 were mixed. Larger populations responded more slowly but generated better long-term off-line performance, while increasing the population size adversely affected on-line performance over the interval of observation. This, we felt, was due to the fact that increasing the population size reduces considerably the allele loss and hence improves long-term performance, but at the cost of taking more samples before a decision (a generation) is made con-

FIG 3.19: R1(50,X,.6,1.0) OFF-LINE PERFORMANCE

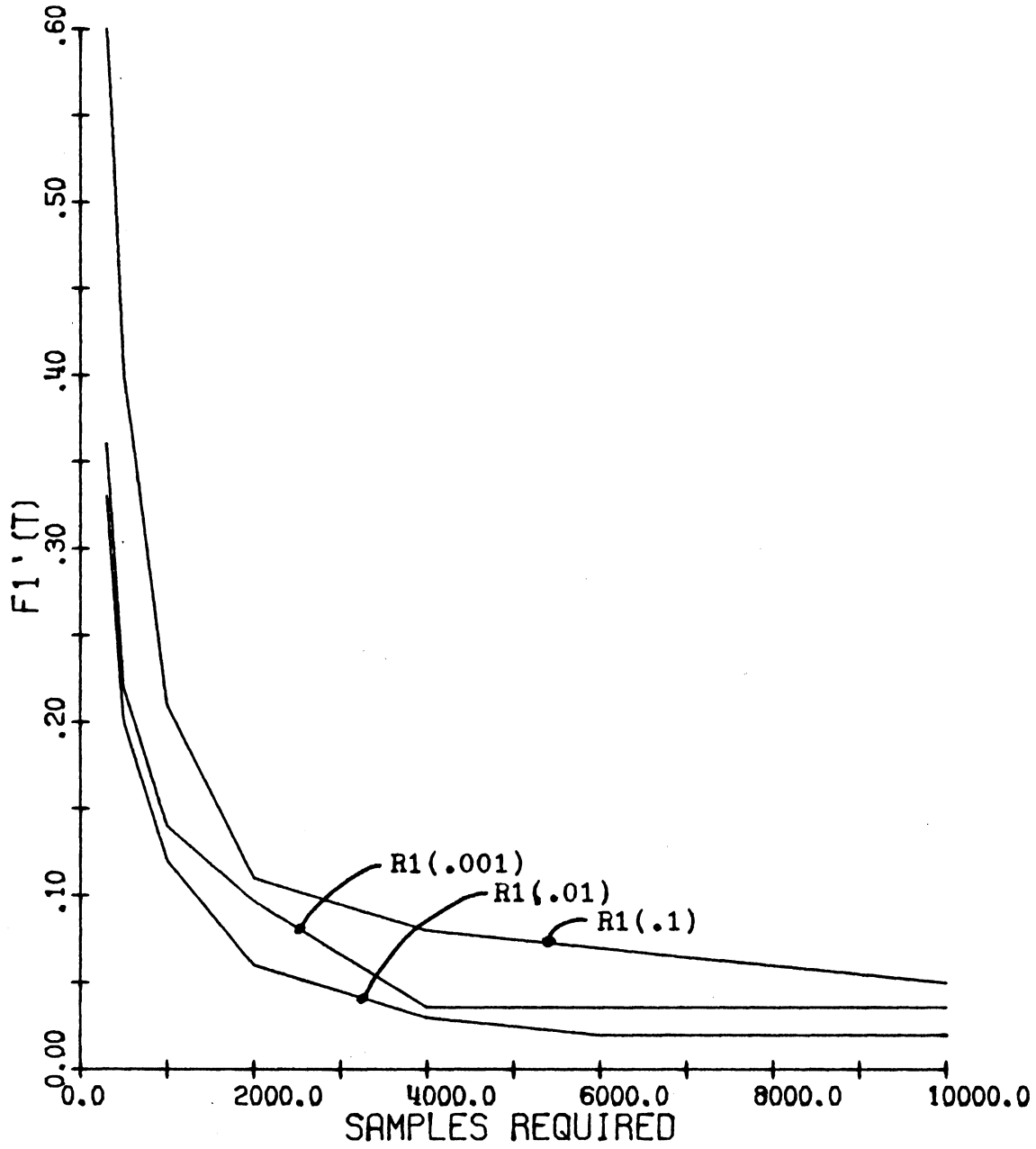


Figure 3.19: Off-line performance of R1 on F1 as a function of mutation rate.

FIG 3.20: R1(50,X,.6,1.0) ON-LINE PERFORMANCE

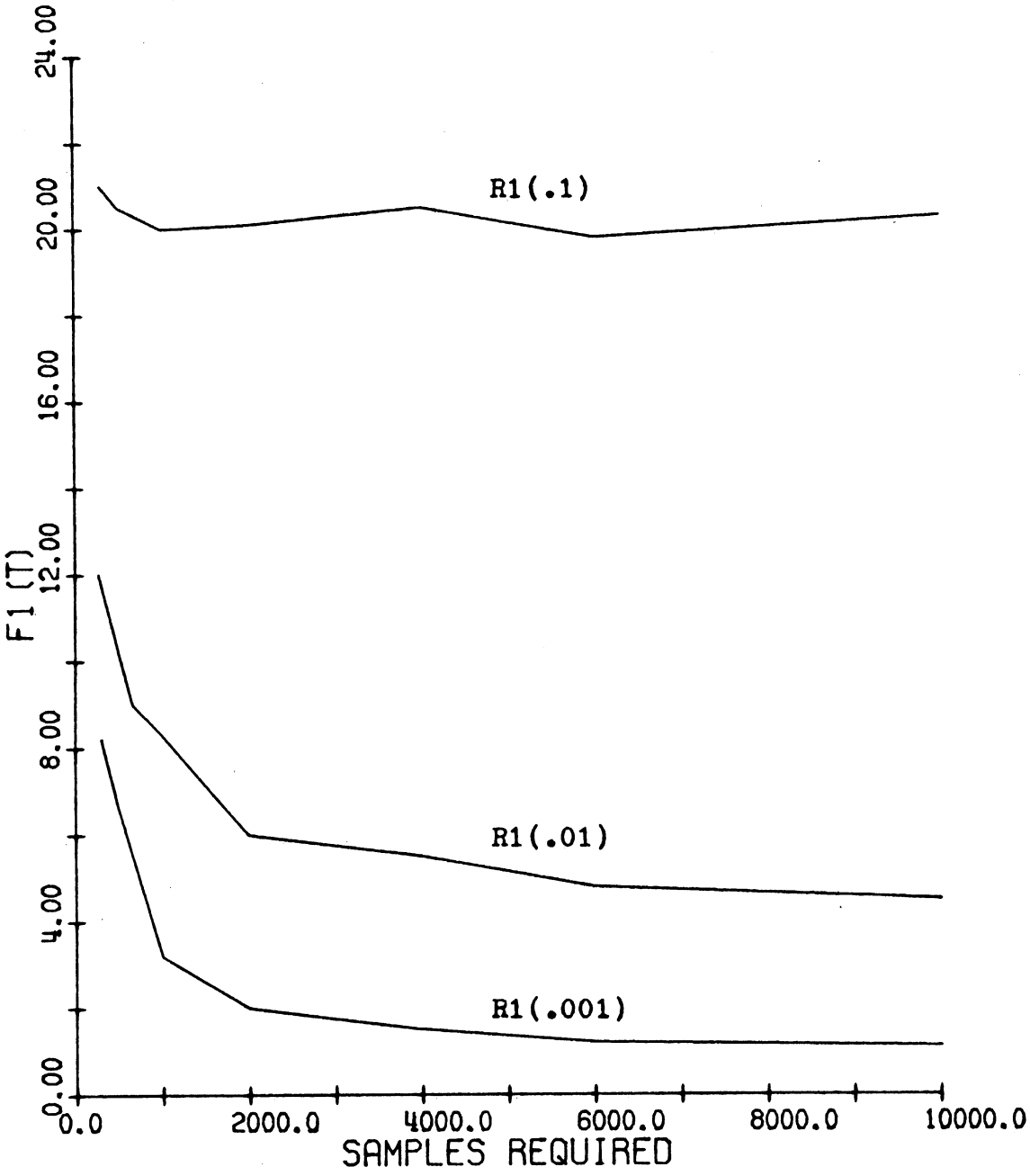


Figure 3.20: On-line performance of R1 on F1 as a function of mutation rate.

cerning the re-distribution of trials. If these observations are correct, we should expect to see the same kind of changes in the behavior produced by increasing the population size of R1(x,.001,.6,1.0) as we saw with R1(x,.001,1.0,1.0), but perhaps less dramatic changes since a crossover rate of .6 has already improved the performance curves.

To evaluate these hypotheses, the behavior of R1 on F1 was analyzed for population sizes of N=50, 100, and 200, leaving the mutation rate, the crossover rate, and the generation gap unchanged at P_m=.001, P_c=.6, and G=1.0. Figures 3.21 and 3.22 compare the performance curves produced by the various population sizes. These observations confirm our intuition about the effects of population size and emphasize again the tradeoff between on-line and off-line performance.

These observations also suggest that no particular combination of the four parameter settings is going to dramatically improve the performance of R1 on F1, and that perhaps the off-line performance generated by R1(50,.01,.6,1.0) and the on-line performance generated by R1(50,.001,.6,1.0) are about the best that can be expected from the basic genetic plan R1.

3.9 Summary

We began this chapter by noting that, although plan R1 outperforms random search on test function F1, it

FIG 3.21: R1(X,.001,.6,1.0) OFF-LINE PERFORMANCE

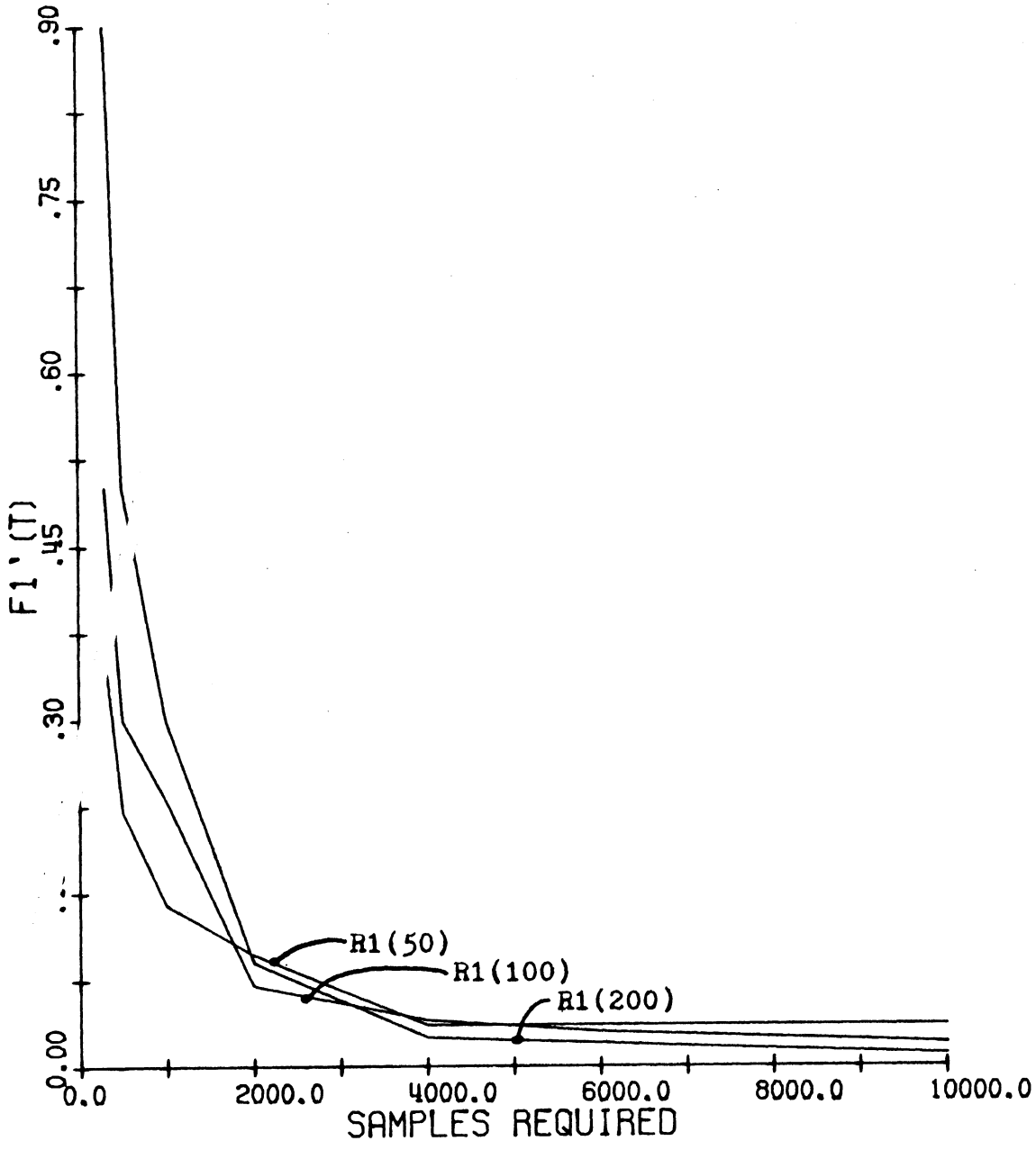


Figure 3.21: Off-line performance of R1 on F1 as a function of population size.

FIG 3.22: R1(X,.001,.6,1.0) ON-LINE PERFORMANCE

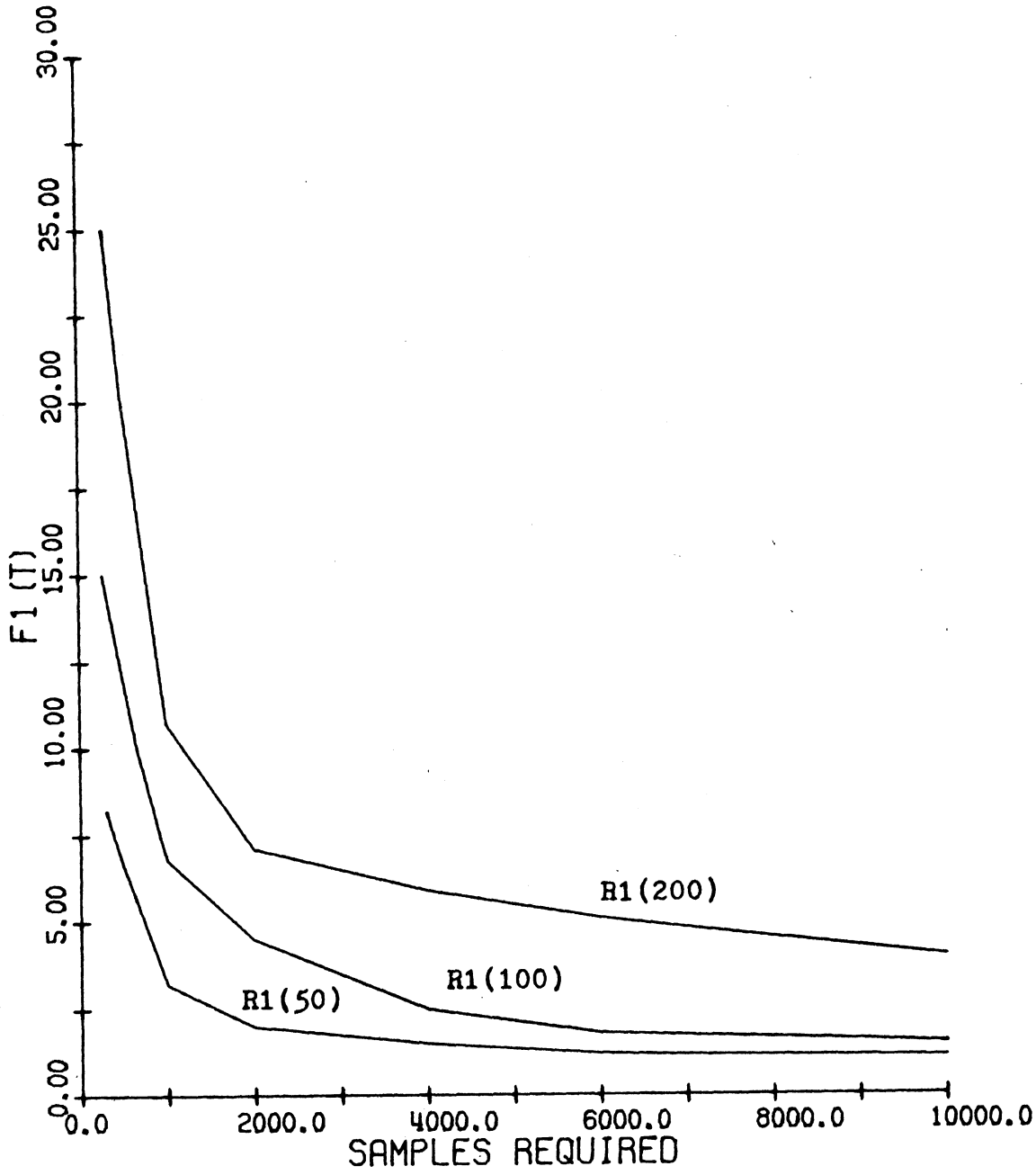


Figure 3.22: On-line performance of R1 on F1 as a function of population size.

suffers from the problem of premature convergence to a non-optimal performance plateau caused by a loss of alleles in A(t), even though on F1 no allele has any selective advantage over its competitor. We saw via Markov process simulation that such allele loss rates can in fact be caused by the stochastic side-effects of generating new populations from old ones using only a finite number of random samples. In order to understand and, perhaps, alleviate the problem, the effects of changing various parameters of genetic plan R1 were analyzed. As we observed, increasing the population size maintained by R1 reduces considerably the rate of allele loss, but also poses a tradeoff in performance. Larger populations respond more slowly, but yield better long-term performance. Alternatively, the allele loss can be counteracted by increasing the mutation rate. However, the effects on performance are mixed. A mutation rate of about 1/POP_SIZE seems to generate the best off-line performance for R1. But any increase in the mutation rate adversely affects on-line performance. Reducing the crossover rate did nothing to alleviate the premature convergence problem; rather, it increased the rate of allele loss. Surprisingly, however, it did effect an improvement in the initial performance of R1, suggesting that generating A(t+1) by replacing every individual in A(t) was, perhaps, too high a sampling rate. Reducing the generation gap of R1 was also ob-

served to increase the rate of allele loss rather than alleviate it. As with crossover, even with the increased rate of allele loss, an improvement in initial performance was observed. Finally, several combinations of parameter values were analyzed in an attempt to improve the performance of R1 on F1. As we observed, no particular settings significantly improved performance suggesting that this is about the best we can expect from R1 on F1.

Chapter 4

PERFORMANCE EVALUATION OF GENETIC ADAPTIVE PLANS

4.1 Introduction

In the preceding chapters we have introduced a class of genetic adaptive algorithms for study, and we have focused our attention in particular on the behavioral characteristics of the basic family of plans R1 on test function F1. The emphasis has been on understanding how these adaptive models operate in finite time and space. In this chapter we apply the insight gained by these studies to the problem of improving the performance of genetic adaptive plans on E.

4.2 The Performance of R1 on E

In the last chapter we studied the effects of changing the various parameters of R1 on its performance on test function F1. In this section we will extend these observations to the performance of R1 on E. As noted earlier, optimizing the performance of R1 over its parameter space is prohibited by the cost of simulation analysis on existing facilities. As before, however, we extend our insight by analyzing a few well-chosen members of the family of plans defined by R1. Recall that in choosing a particular member in R1, four parameters must be specified: the population size N, the mutation rate P_m, the crossover rate P_c, and the

generation gap G. Based on the results of the previous chapter, the following members were chosen for analysis on E:

	N	P_m	P_c	G
R1(	50,	.001,	1.0,	1.0)
R1(	50,	.001,	.8,	1.0)
R1(	50,	.001,	.6,	1.0)
R1(	50,	.001,	.8,	.8)
R1(	50,	.01 ,	.8,	1.0)
R1(	50,	.01 ,	.6,	1.0)
R1(	100,	.001,	.8,	1.0)
R1(	100,	.001,	.6,	1.0)

Recall from chapter 1 that, for each f_e in the environment E, local robustness was defined by

$$x_e(T) = \frac{1}{T} \sum_{i=1}^{T} f_e(t)$$

for on-line performance and

$$x_e^*(T) = \frac{1}{T} \sum_{i=1}^{T} f_e^*(t)$$

for off-line performance with the associated <u>global</u> measures of robustness defined by

$$X_E(T) = \frac{1}{|E|} \sum_{E} x_e(T)$$

and

$$X_E^*(T) = \frac{1}{|E|} \sum_{E} x_e^*(T)$$

respectively. Based on previous experience and with an eye for practical applications, T=6000 was chosen as a reasonable bound on the interval of observation. Tables 4.1a and 4.1b summarize the performance measures obtained from this evaluation and there were very few surprises.

The first three members analyzed differed only in their crossover rates of 1.0, .8, and .6 respectively. As we observed before on test function F1, reducing the crossover rate improves both off-line and on-line performance. The fourth member analyzed illustrates that on E as well as F1, reducing the generation gap is not as effective as reducing the crossover rate. The fifth and sixth members analyzed confirm on E the observation regarding the tradeoff between off-line and on-line performance presented by changing the mutation rate. The only mild surprise came with the evaluation of the last two members supporting a population of size 100. Contrary to our earlier observations, increasing the population size degraded both off-line and on-line performance indices. Upon reflection, however, the reason for this change seems clear. Recall that our earlier observations over 10,000 trials suggested that increasing the population size improved long-term performance at the expense of short-term performance. By shortening the evaluation period to 6000 trials, we have put more emphasis on the short-term behavior and

T=6000	N =50 P_m=.001 P_c=1.0 G = 1.0	N = 50 P_m=.001 P_c=.8 G = 1.0	N = 50 P_m=.001 P_c=.6 G = 1.0	N = 50 P_m=.001 P_c=.8 G =.8	N = 50 P_m=.01 P_c=.8 G =1.0	N =50 P_m=.01 P_c=.6 G =1.0	N =100 P_m=.001 P_c=.8 G =1.0	N =100 P_m=.001 P_c=.8 G =1.0
$x^*_{F1}(T)$	.22	.199	.146	.185	.136	.169	.214	.157
$x^*_{F2}(T)$	.34	.23	.31	.36	.199	.147	.288	.248
$x^*_{F3}(T)$	-26.2	-26.5	-27.2	-26.4	-26.6	-26.4	-26.1	-26.8
$x^*_{F4}(T)$	35.9	34.67	34.5	37.5	34.1	34.2	36.6	37.8
$x^*_{F5}(T)$	4.13	3.75	2.56	3.58	3.31	2.46	4.92	4.75
$X^*_E(T)$	2.88	2.47	2.06	2.64	2.23	2.11	3.18	3.23

Table 4.1a: Off-line performance of R1 on E

T=6000	N =50 P_m=.001 P_c=1.0 G =1.0	N =50 P_m=.001 P_c=.8 G =1.0	N =50 P_m=.001 P_c=.6 G =1.0	N =50 P_m=.001 P_c=.8 G =.8	N =50 P_m=.01 P_c=.8 G =1.0	N =50 P_m=.01 P_c=.6 G =1.0	N =100 P_m=.001 P_c=.8 G =1.0	N =100 P_m=.001 P_c=.8 G =1.0
$x_{F1}(T)$	4.6	4.27	3.65	4.02	7.35	7.6	6.94	5.69
$x_{F2}(T)$	78.4	76.8	76.7	78.4	161.6	160.5	134.8	114.37
$x_{F3}(T)$	-22.1	-23.17	-23.3	-22.1	-18.4	-19.01	-21.4	-21.3
$x_{F4}(T)$	97.3	93.2	89.9	96.5	127.8	128.4	102.7	98.14
$x_{F5}(T)$	42.2	34.1	36.5	41.2	84.1	92.2	45.4	38.1
$X_E(T)$	40.08	37.04	36.69	39.60	72.49	73.94	53.61	47.01

Table 4.1b: On-line performance of R1 on E

hence should expect to see a performance degradation.

These results confirm our earlier observations of the behavior of R4, and they suggest that the off-line performance of R1(50,.01,.6,1.0) and the on-line performance by R1(50,.001,.6,1.0) are about the best that can be expected from these simple genetic plans.

4.3 Elitist Model R2

Earlier observations of the behavior of R1 suggested that generating N new individuals for each new population A(t+1) was in fact too high a sampling rate. High-performance individuals were lost before the genetic operators were able to produce improvements. An improvement in performance was obtained by reducing the crossover rate and/or the generation gap which, in turn, reduced the number of new individuals produced for A(t+1). Moreover, we observed that reducing the crossover rate produced better performance improvements than a corresponding reduction in the generation gap. This, we felt, was due to the fact that, because of the selection processes, high-performance individuals were more likely to survive into the next generation via a reduction in the crossover rate than with a reduction in the generation gap. In this section we consider the implications of giving high-performance individuals special treatment by modifying the basic plan R1 to include the following elitist policy:

Let $a^*(t)$ be the best individual generated up to time t. If, after generating A(t+1) in the usual fashion, $a^*(t)$ is not in A(t+1), then include $a^*(t)$ to A(t+1) as the $(N+1)^{th}$ member.

Such a policy guarantees that the best individual generated will not be lost from one generation to the next as a consequence of sampling effects or the application of genetic operators. From the hyperplane analysis point of view, this policy will bias the distribution of trials in favor of those hyperplane partition elements which have produced the best-performing individual. This suggests that the effect of such a policy on performance may be to improve local search at the expense of global search.

In order to evaluate the effects of such a policy, two members of this family were evaluated on E:

R2(50,.001,.8,1.0)
and R2(50,.001,.6,1.0)

Figures 4.1 - 4.3 compare the behavior of these plans with their R1 counterparts on test function F1. Figure 4.1 illustrates that the allele loss rate is slightly better with R2. This is probably due to the fact that appending the best individual to A(t+1) prevents one or two alleles from being counted as lost. Figures 4.2 and 4.3 illustrate that R2 produces both off-line and on-line curves for F1 which are significantly better than those produced by R1.

FIG 4.1: R2 ALLELE LOSS ON F1

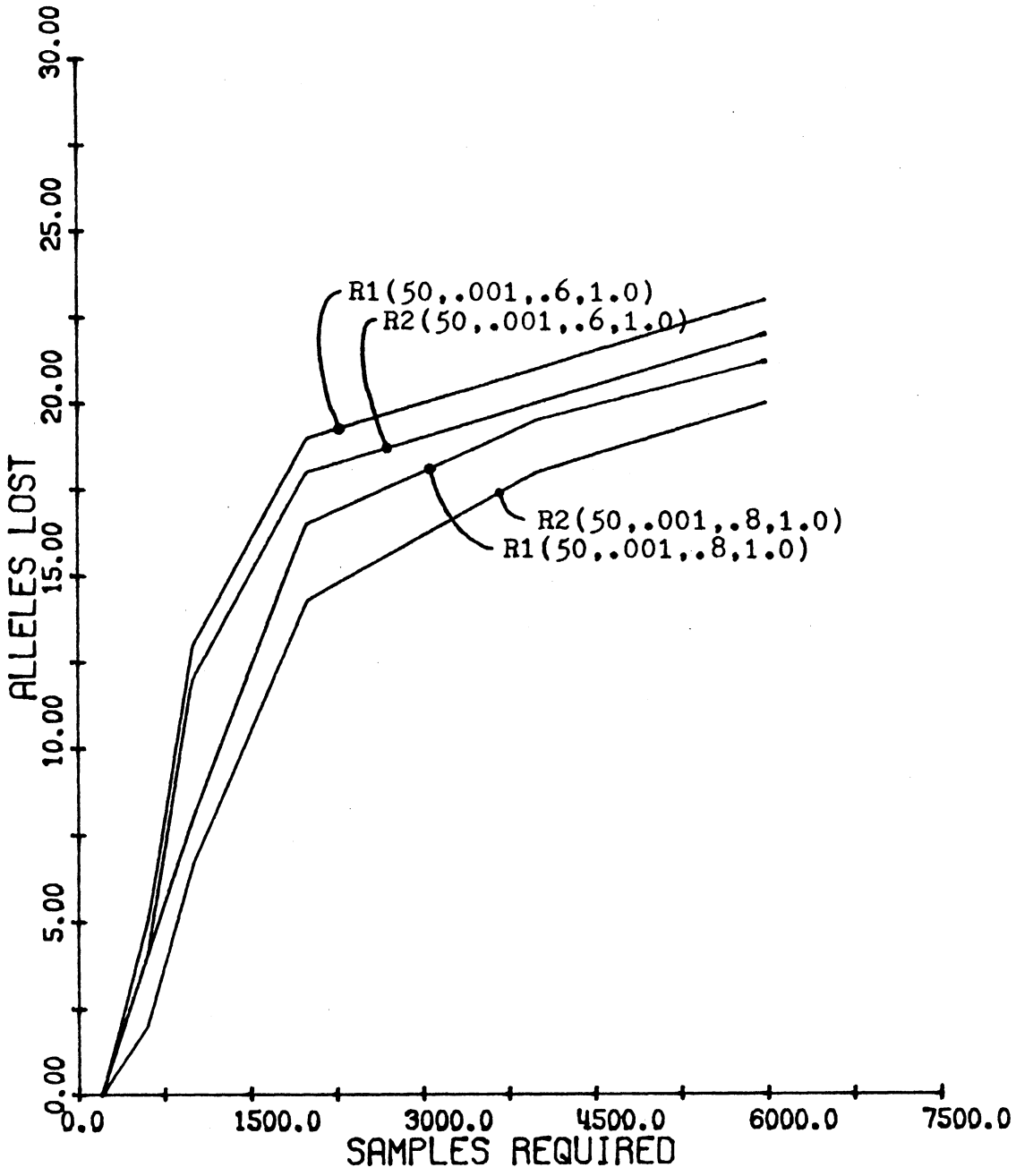


Figure 4.1: Allele loss for R2 on F1.

FIG 4.2: OFF-LINE PERFORMANCE OF R2 ON F1

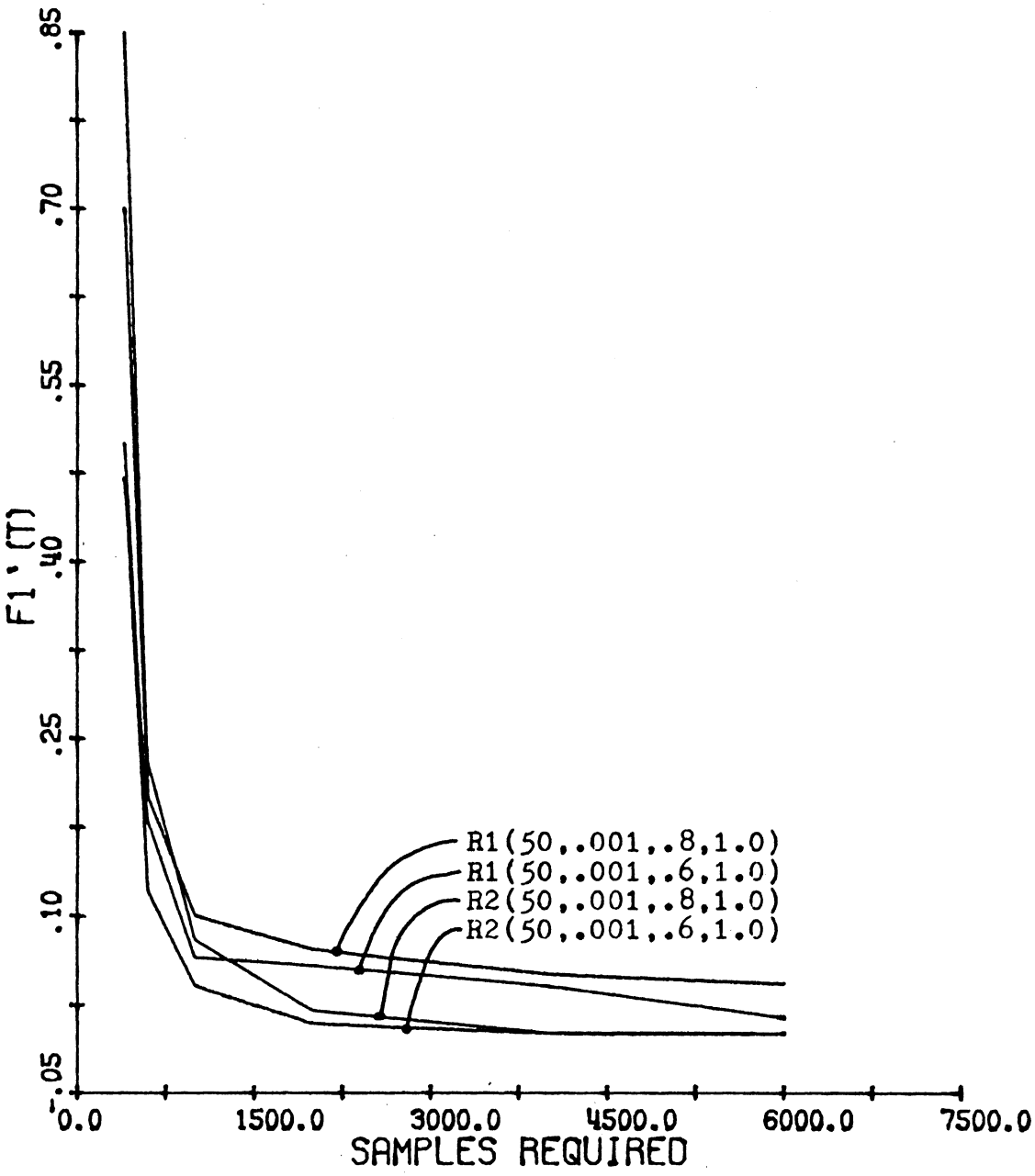


Figure 4.2: Off-line performance curves for R2 on F1.

FIG. 4.3: ON-LINE PERFORMANCE OF R2 ON F1

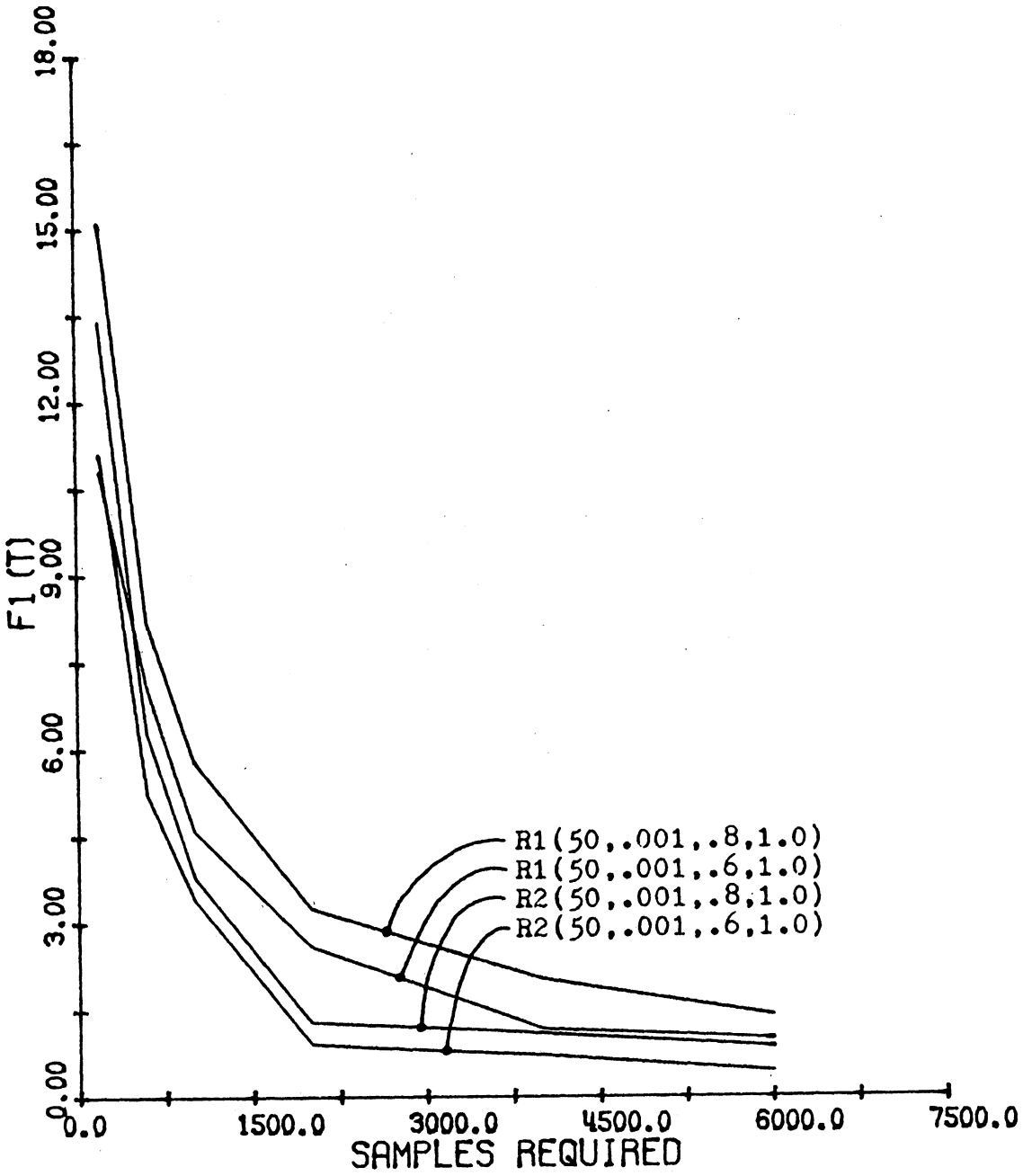


Figure 4.3: On-line performance curves for R2 on F1.

Finally, the associated performance indices for both off-line and on-line performance are tabulated below in comparison with their R1 counterparts:

T=6000	R1(.8)	R2(.8)	R1(.6)	R2(.6)
$x^*_{F1}(T)$	.199	.178	.146	.093
$x^*_{F2}(T)$	.230	.076	.310	.182
$x^*_{F3}(T)$	-26.5	-26.3	-27.2	-27.1
$x^*_{F4}(T)$	34.67	29.61	34.5	26.76
$x^*_{F5}(T)$	3.75	5.86	2.56	3.95
$X^*_E(T)$	2.47	1.88	2.06	.778

T=6000	R1(.8)	R2(.8)	R1(.6)	R2(.6)
$x_{F1}(T)$	4.27	2.88	3.65	2.71
$x_{F2}(T)$	76.8	51,73	76.7	50.46
$x_{F3}(T)$	-23.17	-22.9	-23.3	-24.02
$x_{F4}(T)$	93.2	65.8	89.9	59.92
$x_{F5}(T)$	34.1	36.2	36.2	37.49
$X_E(T)$	37.04	26.74	36.69	25.31

These results confirm our intuition about the behavior of elitist plan R2. Because of its more conservative sampling policy, on-line performance is consistently improved. Off-line performance is improved as well, but notice that the improvements come on the unimodal surfaces, particularly F4, while the performance degraded significantly on F5.

4.4 Expected Value Model R3

The problem of premature allele loss and the subsequent convergence to a non-optimal plateau which was analyzed closely in the previous chapter has still not been resolved. As we have seen, changing the various parameters of the genetic models affects both the allele loss rate and performance curves on test function F1, but gives no satisfactory solution to either. In this section we explore the possibility of resolving these problems by modifying the sampling techniques used in R1 and R2.

We begin by focusing our attention again on the two competing hyperplanes associated with a particular gene position. As we have seen, test function F1 has the characteristic that it has the same average value on both partition elements, so that neither hyperplane theoretically has any selective advantage over the other. However, the next generation A(t+1) is produced by taking a finite number of samples from A(t) using a selection distribution computed from sample means. This opens the door for stochastic side effects from two sources: the error involved in the sample means (and hence the selection probabilities), and the error involved in only taking a finite sample from A(t) using the selection distribution. The error in the sample means is, of course, a function of the population size and the variance of F1 on the associated hyperplanes, and can be resolved, as we have

seen, by increasing the population size at the expense of initial performance. The Markov process simulations in the section on genetic drift modelled the second source of error which, we have seen, cannot be ignored. In R1 and R2 this sampling process is used to produce the offspring of individuals in A(t). As a consequence, the actual number of offspring produced by an individual can differ markedly from the expected number of offspring. As we saw in chapter 2, the offspring determine the number of trials allocated to a particular hyperplane in the next generation. Hence, the sampling side-effects can lead to considerable disparities between the expected and actual number of trials allocated to competing pairs of hyperplanes. This suggests that we consider redefining the sampling process used in R1 and R2 in such a way as to force the actual number of offspring to more closely approximate the expected number.

Genetic adaptive plan R3 attempts to accomplish this in the following way. The expected number of offspring, $u(a_{it})/\overline{u(t)}$, is computed and associated with each individual a_{it} in A(t) before selection begins. Each time an individual is selected as a partner for crossover, its associated offspring count is decremented by .5. Each time an individual is selected to produce an offspring without applying crossover, its associated offspring count is decremented by 1. When the offspring count falls below zero, an individual is no longer available

for selection.

This modification to the selection process forces the actual number of offspring to always be less than $u(a_{1t})/\overline{u(t)} + 1$ and generally less than $u(a_{1t})/\overline{u(t)} + .5$, resulting in a leveling effect on the sampling error. If the high rate of allele loss exhibited by R1 and R2 on F1 is due in part to this sampling error, R3 should exhibit a reduced rate of allele loss and a corresponding improvement in performance.

In order to evaluate these hypotheses, the following three members from the family of plans defined by R3 were chosen for evaluation on E:

R3(50,.001,1.0,1.0)
R3(50,.001, .8,1.0)
R3(50,.001, .6,1.0)

Figures 4.4 - 4.6 compare the behavior of R3(50,.001,.6,1.0) on F1 with its corresponding R1 and R2 counterparts. Figure 4.4 illustrates that, as we had hoped, the allele loss rate is considerably reduced with the modified sampling technique. Figures 4.5 and 4.6 compare the performance curves of R1, R2, and R3 for test function F1. Notice that R3 performed significantly better than R1 on F1, but not quite as well as R2.

Finally, the associated performance indices for both off-line and on-line performance of R3 on E are tabulated below in comparison with R1 and R2:

FIG 4.4: R3 ALLELE LOSS ON F1

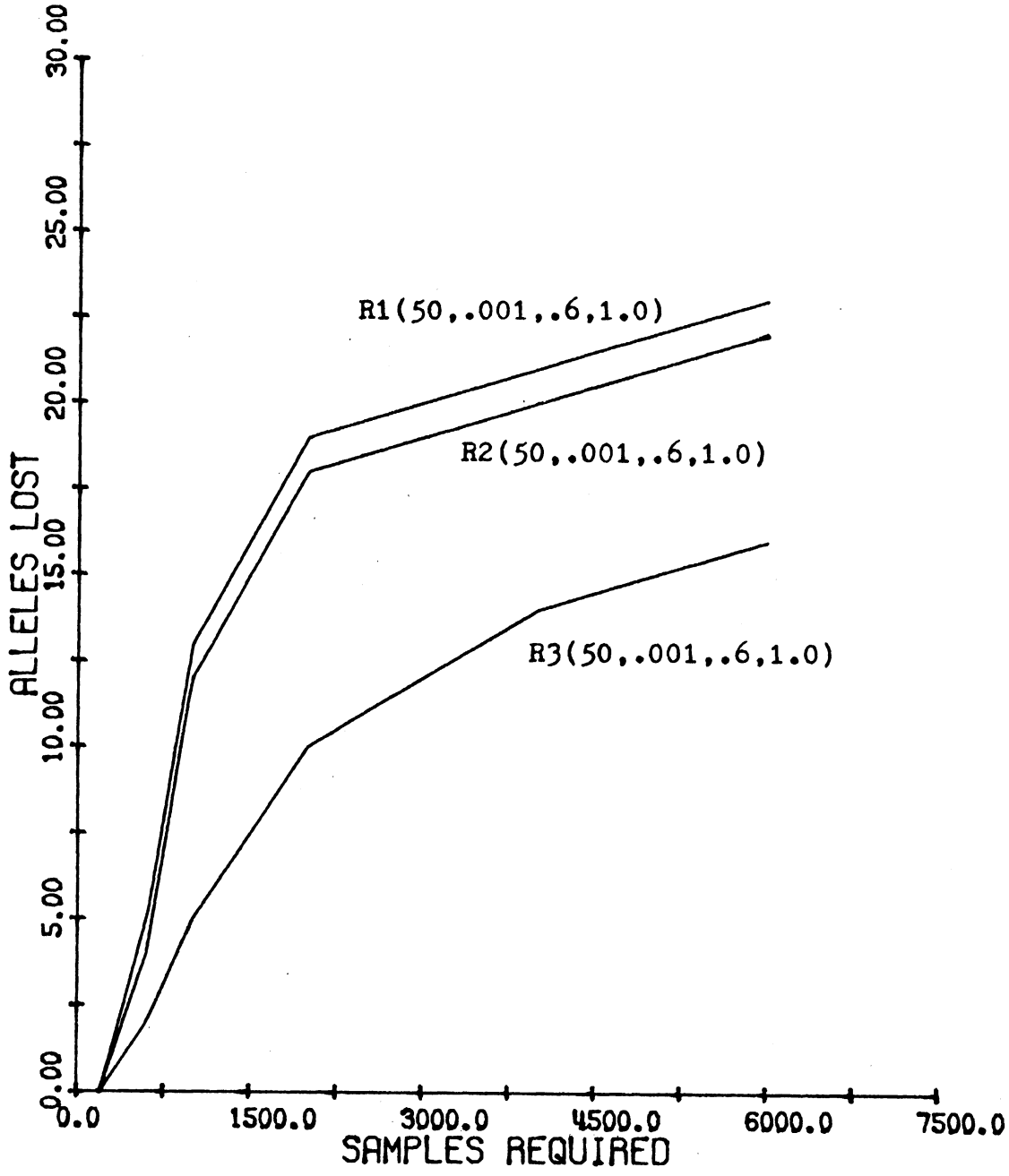


Figure 4.4: Allele loss for R3 on F1.

FIG 4.5: OFF-LINE PERFORMANCE OF R3 ON F1

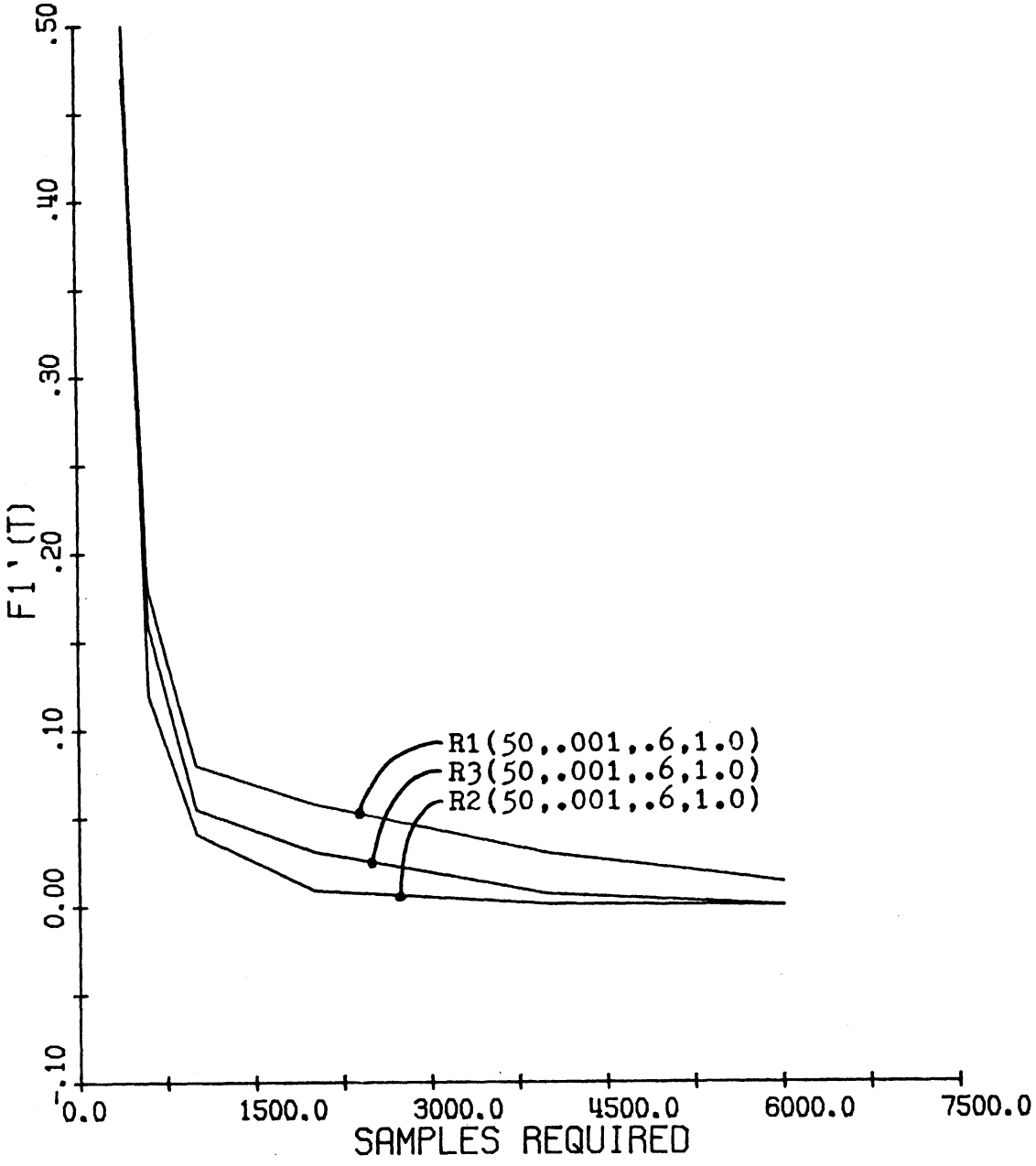


Figure 4.5: Off-line performance curve for R3 on F1.

FIG. 4.6: ON-LINE PERFORMANCE OF R3 ON F1

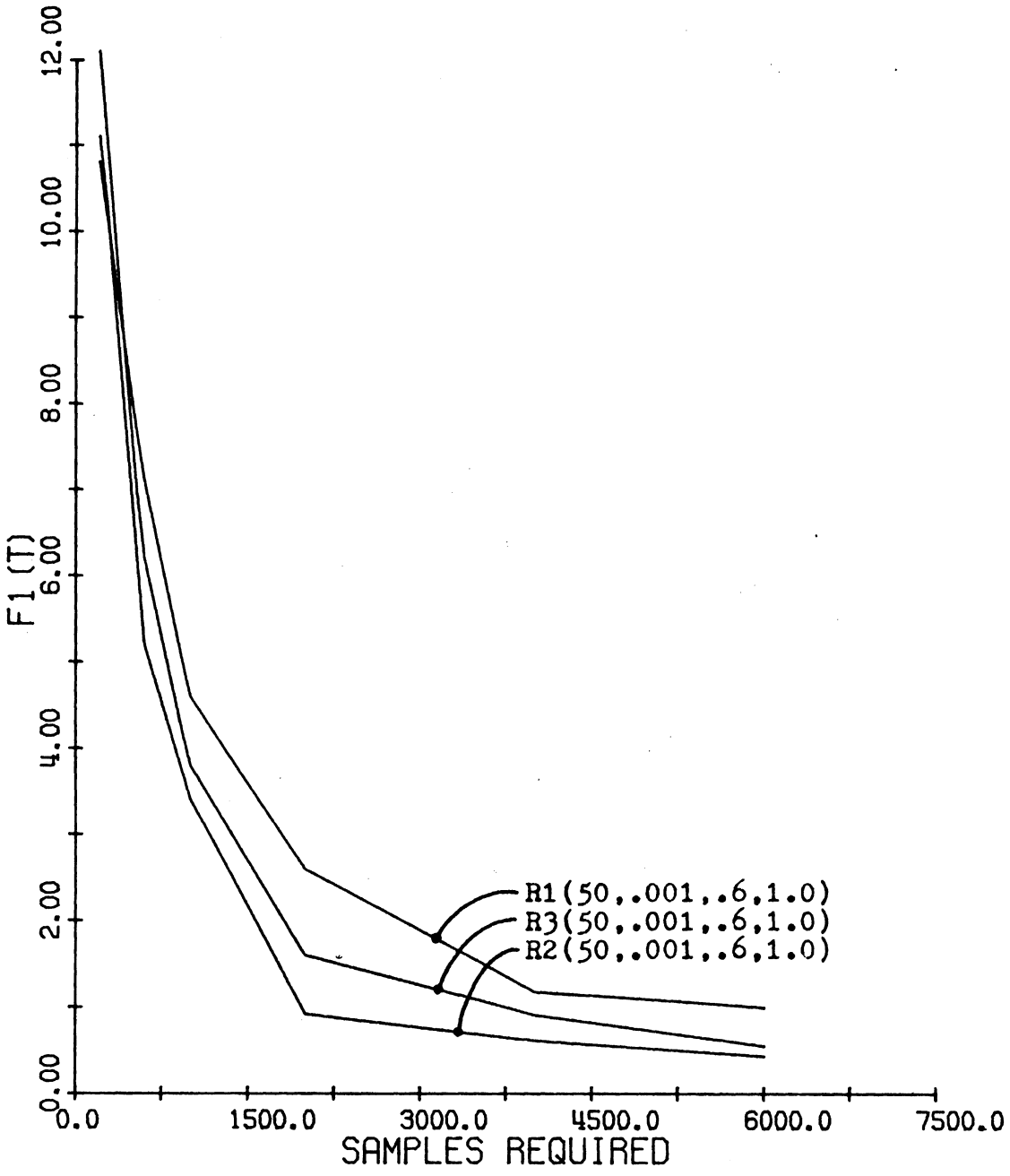


Figure 4.6: On-line performance curve for R3 on F1.

T=6000	R3(1.0)	R3(.8)	R3(.6)	R2(.6)	R1(.6)
$x^*_{F1}(T)$	.410	.130	.166	.093	.146
$x^*_{F2}(T)$	.428	.213	.364	.182	.310
$x^*_{F3}(T)$	-27.3	-27.1	-27.2	-27.1	-27.2
$x^*_{F4}(T)$	21.17	20.22	21.07	26.76	34.5
$x^*_{F5}(T)$	4.31	2.86	3.20	3.95	2.56
$X^*_E(T)$	.196	-.735	-.483	.778	2.06

T=6000	R3(1.0)	R3(.8)	R3(.6)	R2(.6)	R1(.6)
$x_{F1}(T)$	3.53	2.41	3.42	2.71	3.65
$x_{F2}(T)$	65.65	46.64	55.15	50.46	76.7
$x_{F3}(T)$	-24.6	-24.87	-24.94	-24.02	-23.3
$x_{F4}(T)$	48.28	49.08	44.18	59.92	89.9
$x_{F5}(T)$	36.46	32.29	37.31	37.49	36.2
$X_E(T)$	25.46	21.68	22.65	25.31	36.69

These results yield two interesting observations. First, note that R3(.8) performed slightly better on E than R3(.6), suggesting that, by reducing the sampling error, a higher sampling rate can be supported. Secondly, note that, although R2 outperformed R3 on F1, R3 showed an overall improvement in robustness on E. Both of these observations confirm our intuition about the behavior of the elitist and expected value models.

4.5 Elitist Expected Value Model R4

At this point in the analysis of genetic adaptive plans, it is difficult to resist combining the two previous models to produce an expected value model with an elitist policy. The motivation here is to increase our confidence in the observations and inferences made so far about the behavior of genetic plans, rather than providing new insights. If our analysis of the preceding sections is correct, we should expect that adding an elitist policy to R3 should improve its performance on the unimodal surfaces at the expense of multimodal performance.

In order to evaluate this hypothesis, two members of R4 were chosen for analysis on E:

R4(50,.001,.8,1.0)
R4(50,.001,.6,1.0)

Figures 4.7 - 4.9 compare the behavior of R4(.6) on test function F1 with its R2 and R3 counterparts. Figure 4.7 illustrates again that adding an elitist policy reduces slightly the allele loss rate on F1. Figures 4.8 and 4.9 illustrate the improved off-line and on-line performance curves generated by R4.

Finally, the associated performance indices for both off-line and on-line performance of R4 on E are tabulated below in comparison with R3 and R2:

FIG 4.7: R4 ALLELE LOSS ON F1

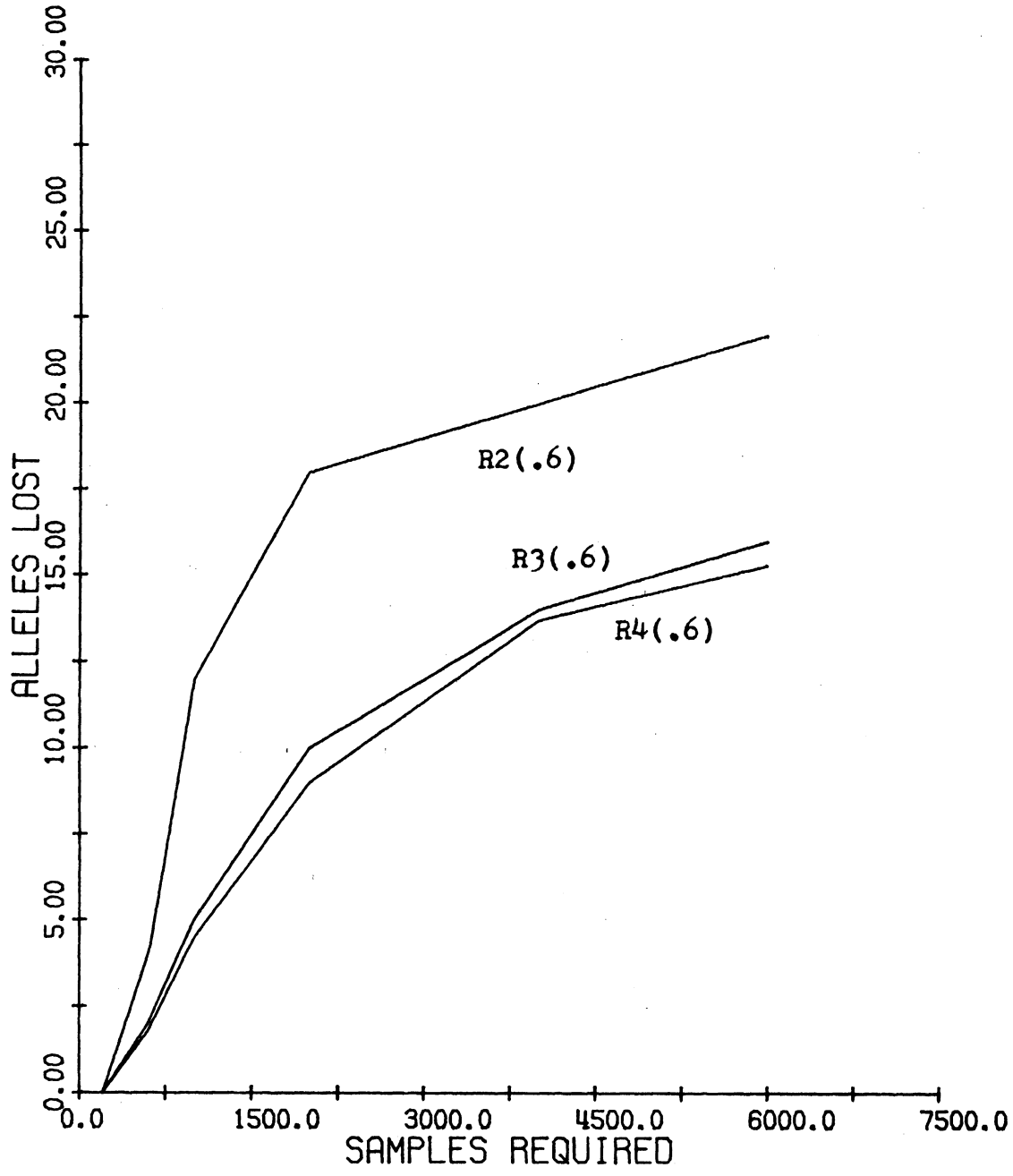


Figure 4.7: Allele loss for R4 on F1.

FIG 4.8: OFF-LINE PERFORMANCE OF R4 ON F1

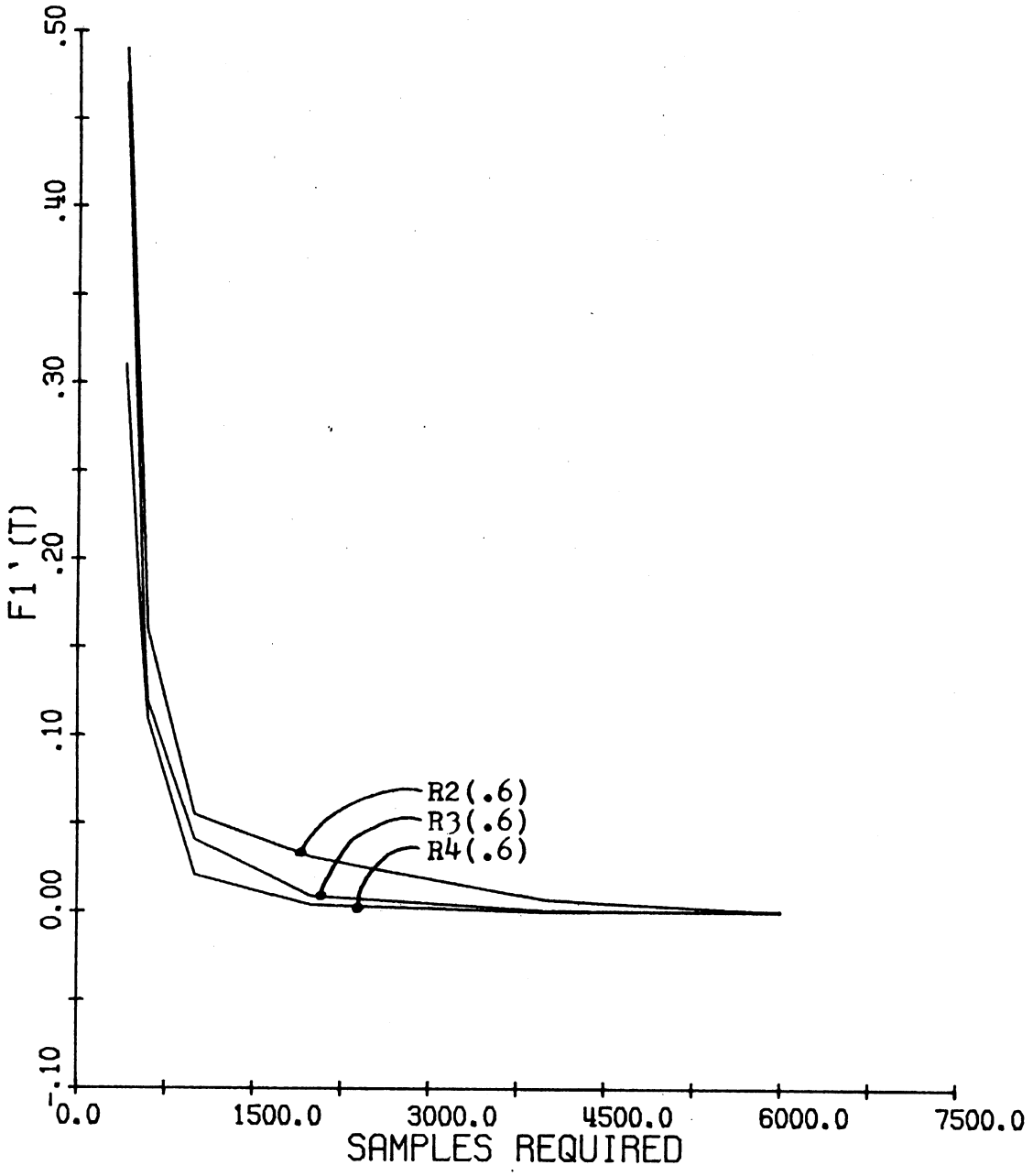


Figure 4.8: Off-line performance curve for R4 on F1.

FIG. 4.9: ON-LINE PERFORMANCE OF R4 ON F1

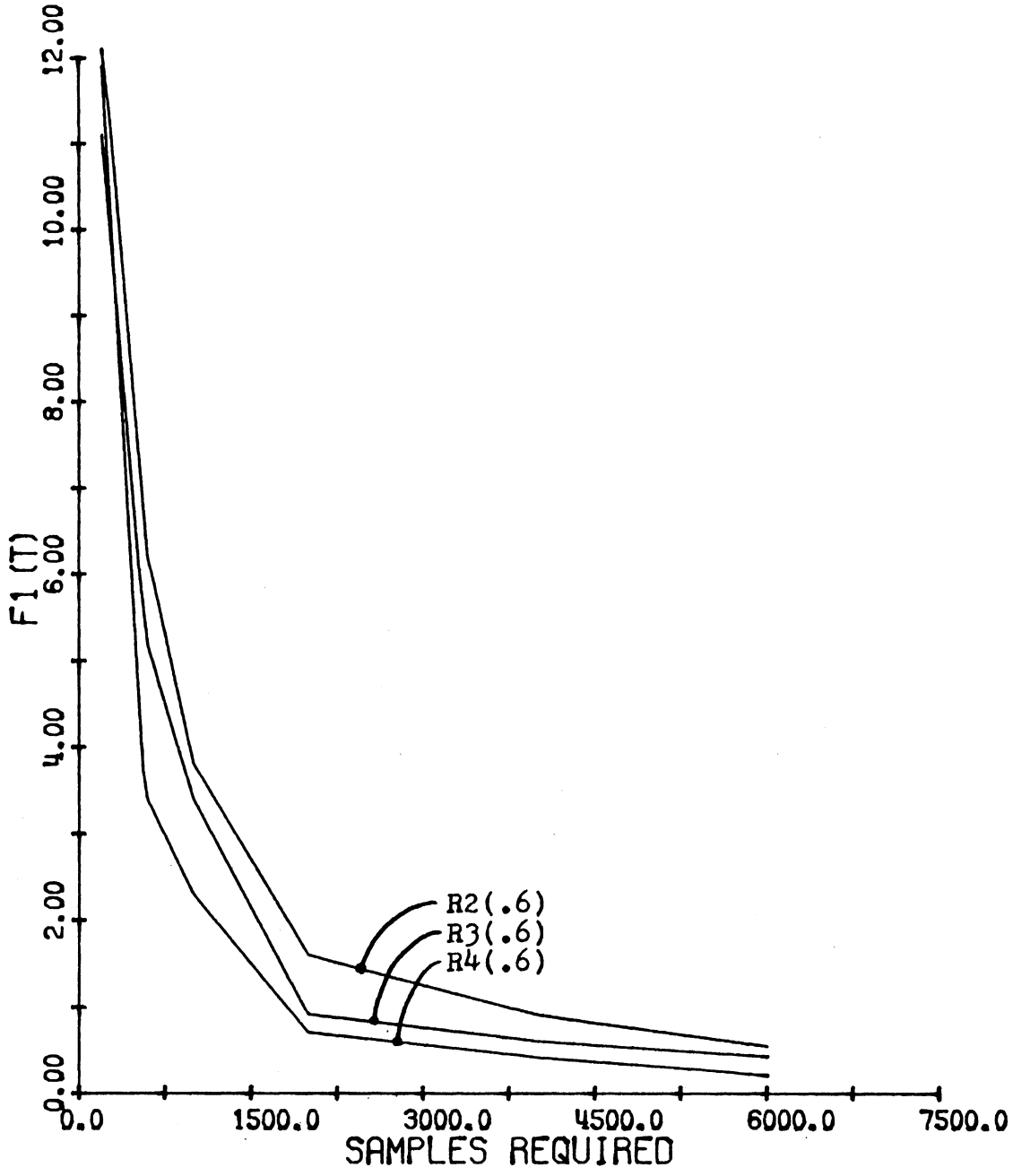


Figure 4.9: On-line performance curve for R4 on F1.

T=6000	R4(.8)	R4(.6)	R3(.8)	R3(.6)	R2(.6)
$x^*_{F1}(T)$	.097	.113	.130	.166	.093
$x^*_{F2}(T)$	.201	.221	.213	.364	.182
$x^*_{F3}(T)$	-27.5	-28.2	-27.1	-27.2	-27.1
$x^*_{F4}(T)$	18.21	17.62	20.22	21.07	26.76
$x^*_{F5}(T)$	2.98	3.34	2.86	3.20	3.95
$X_E(T)$	-1.20	-1.38	-.735	-.483	.778

T=6000	R4(.8)	R4(.6)	R3(.8)	R3(.6)	R2(.6)
$x_{F1}(T)$	2.41	2.32	2.41	3.42	2.71
$x_{F2}(T)$	35.46	34.76	46.64	55.15	50.46
$x_{F3}(T)$	-25.35	-26.49	-24.87	-24.94	-24.02
$x_{F4}(T)$	42.53	40.73	49.08	44.18	59.92
$x_{F5}(T)$	33.59	34.34	32.29	35.31	37.49
$X_E(T)$	17.73	17.12	21.68	22.65	25.31

These results suggest that our intuition about the effects of an elitist policy on the behavior of these genetic plans is correct. The performance on the unimodal surfaces (F1-F4) is considerably improved at the expense of performance on the difficult multimodal surface F5. Notice however that performance on F5 was only slightly affected (an observation which will be explored more fully later) and, as a consequence, R4 generated the best overall performance we have seen on E.

It is worth considering at this point whether the performance generated by R4(50,.001,.6,1.0) is about the best R4 can do on E. We have been evaluating these particular parameter settings to provide straightforward comparisons with the preceding models. However, the changes we have made to the genetic algorithm may also affect the choice of the parameter settings. To answer this question, the performance of the following members of R4 was evaluated on E:

R4(50,.001,.8,1.0)
R4(50,.001,.6,1.0)
R4(50,.001,.4,1.0)
R4(50,.01 ,.6,1.0)
R4(50,.05 ,.6,1.0)
R4(50,.001,.6, .8)
R4(50,.001,.6, .6)
R4(100,.001,.6,1.0)

Tables 4.2a and 4.2b compare the off-line and on-line performance indices for each of the parameter settings. The first three members evaluated differ only in their crossover rates and they suggest that a crossover rate of .6 is still a reasonable choice. Notice, however, that because of the overall improvement in performance, R4 is considerably less sensitive than R1 to changes in the crossover rate. The fourth and fifth members illustrate the effects of an increased mutation rate. Here the results differed from before. Increasing the mutation rate degraded both the off-line and on-line performance of R1. This observation will be explored in more detail later. The sixth and seventh members illus-

T=6000	N =50 P_m=.001 P_c=.8 G =1.0	N =50 P_m=.001 P_c=.6 G =1.0	N =50 P_m=.01 P_c=.4 G =1.0	N =50 P_m=.01 P_c=.6 G =1.0	N =50 P_m=.05 P_c=.6 G =1.0	N =50 P_m=.001 P_c=.6 G =.8	N =50 P_m=.001 P_c=.6 G =.6	N =100 P_m=.001 P_c=.6 G =1.0
$x^*_{F1}(T)$	.097	.113	.134	.068	.128	.213	.112	.103
$x^*_{F2}(T)$	.201	.221	.205	.259	.240	.193	.410	.148
$x^*_{F3}(T)$	-27.5	-28.2	-28.5	-27.6	-26.1	-27.9	-28.2	-27.5
$x^*_{F4}(T)$	18.21	17.62	16.24	21.81	35.98	21.66	17.1	28.5
$x^*_{F5}(T)$	2.98	3.34	8.53	3.98	3.51	2.81	11.5	5.39
$X^*_E(T)$	-1.20	-1.38	-.678	-.297	2.75	-.605	.184	1.33

Table 4.2a: Off-line performance of R4 on E

T=6000	N =50 P_m=.001 P_c=.8 G =1.0	N =50 P_m=.001 P_c=.6 G =1.0	N =50 P_m=.01 P_c=.4 G =1.0	N =50 P_m=.01 P_c=.6 G =1.0	N =50 P_m=.05 P_c=.6 G =1.0	N =50 P_m=.001 P_c=.6 G =.8	N =50 P_m=.001 P_c=.6 G =.6	N =100 P_m=.001 P_c=.6 G =1.0
$x_{F1}(T)$	2.41	2.32	2.30	5.88	15.83	2.35	2.42	4.63
$x_{F2}(T)$	35.46	34.76	40.02	105.0	221.72	42.35	41.44	62.9
$x_{F3}(T)$	-25.35	-26.49	-27.01	-21.5	-14.04	-26.1	-26.2	-24.05
$x_{F4}(T)$	42.53	40.73	39.03	96.38	182.0	47.14	49.2	68.38
$x_{F5}(T)$	33.59	34.34	35.06	91.46	207.1	36.76	41.28	43.82
$X_E(T)$	17.73	17.12	17.88	55.45	122.5	20.50	21.63	31.14

Table 4.2b: On-line performance of R4 on E

trate again the negative effect on performance generated by reducing the generation gap. And the last member evaluated illustrates again that increasing the population size degrades performance over the interval of observation.

In summary, then, these results suggest that the performance generated by R4(50,.001,.6,1.0) is about the best R4 can do on E. It is important, however, to emphasize the extent of the improvements in performance we have achieved by moving to type R4 genetic plans. Recall from the evaluation in appendix C that, if on-line performance is desired, one simply cannot afford to use random search. Even the simplest genetic plan generates significantly better on-line performance. However, when measuring off-line performance, we saw that random search gave R1 considerably stiffer competition and produced in several cases better performing individuals over the interval of observation. To illustrate the improved performance of R4, figures 4.10 - 4.14 compare the off-line performance curves of random search on each of the test functions in E with the best curves generated by R1 and R4 as well as the (unattainable) optimal off-line curve given by:

$$f^*(t) = MIN(f) \ , \ t=1,\ldots,T$$

On test functions F1 and F2 plan R4 located the minimum without difficulty within the interval of observation. On the larger search spaces associated with F3 and F4, the minimum was not found within 6000 trials. But notice

FIG. 4.10: OFF-LINE PERFORMANCE ON F1

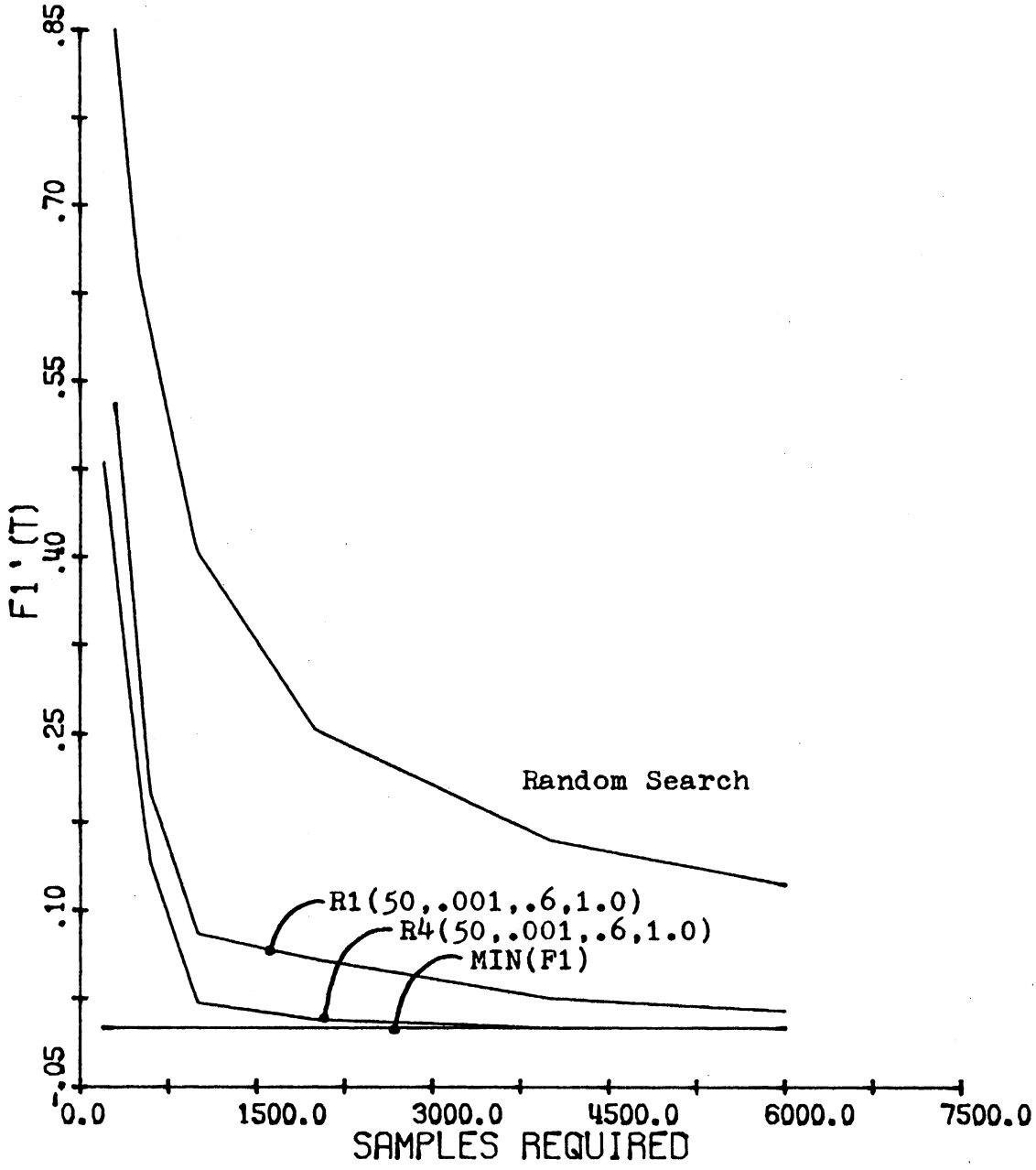


Figure 4.10: A comparison of off-line performance curves generated on F1.

FIG. 4.11: OFF-LINE PERFORMANCE ON F2

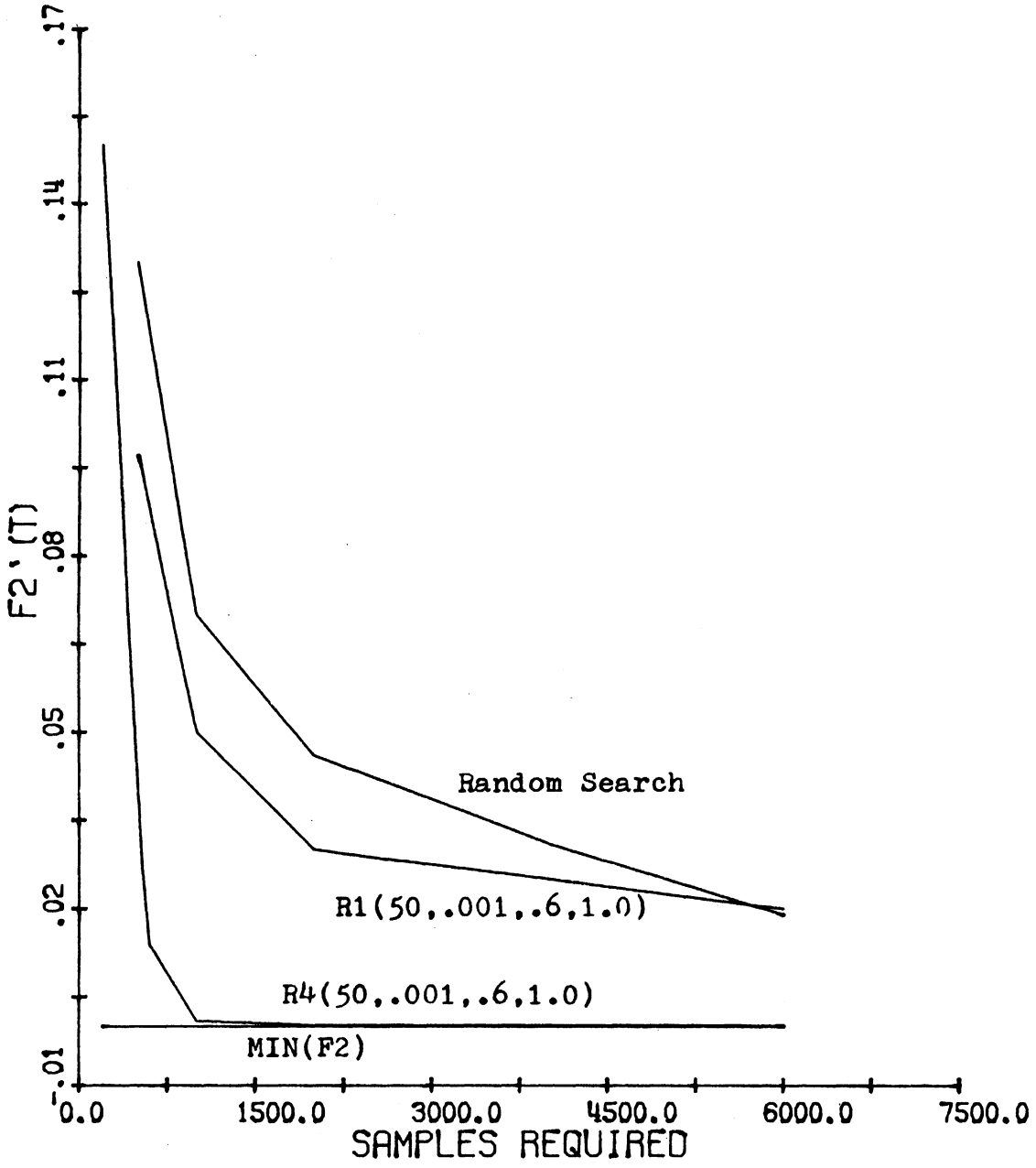


Figure 4.11: A comparison of off-line performance curves generated on F2.

FIG 4.12: OFF-LINE PERFORMANCE ON F3

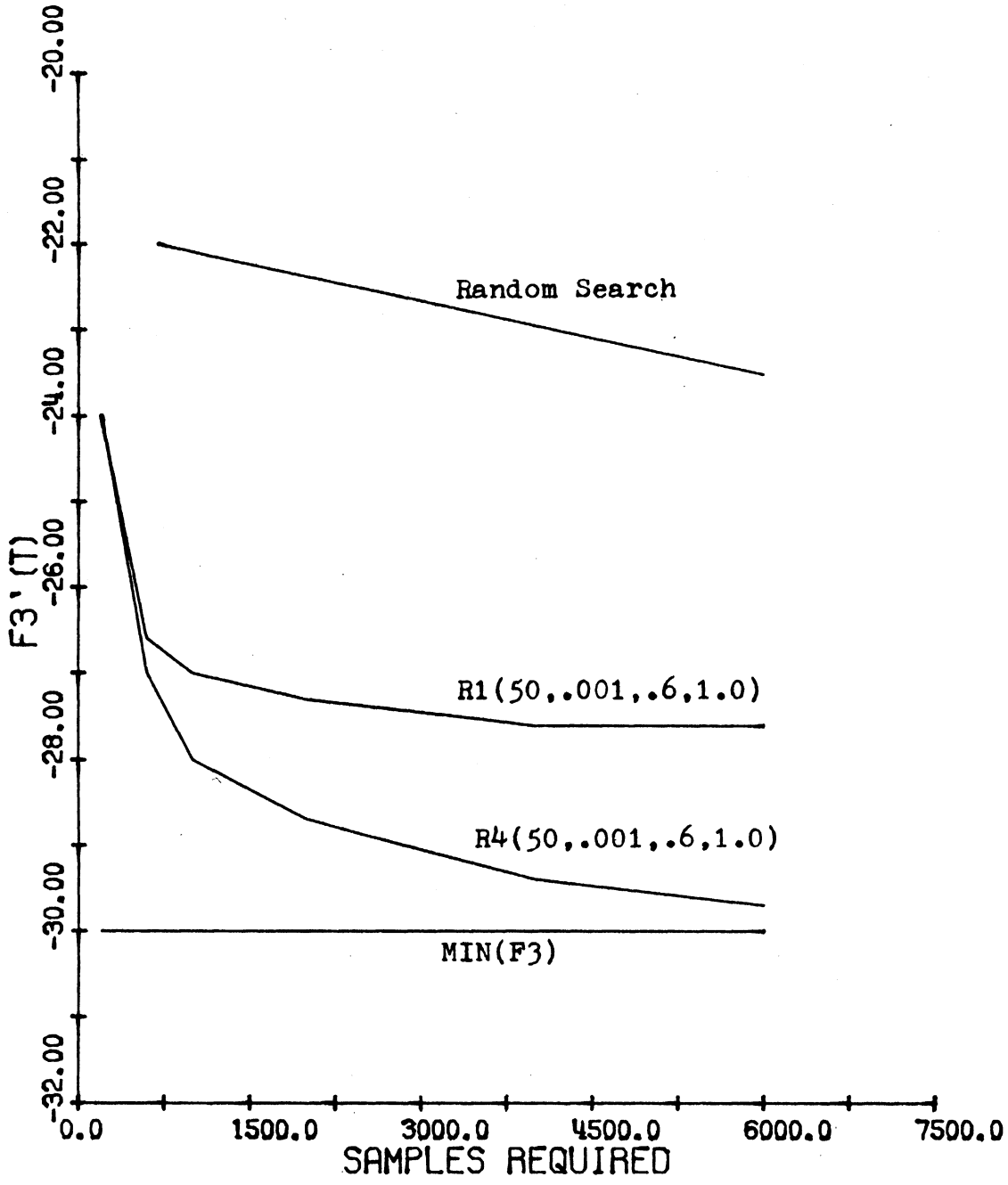


Figure 4.12: A comparison of off-line performance curves generated on F3.

FIG. 4.13: OFF-LINE PERFORMANCE ON F4

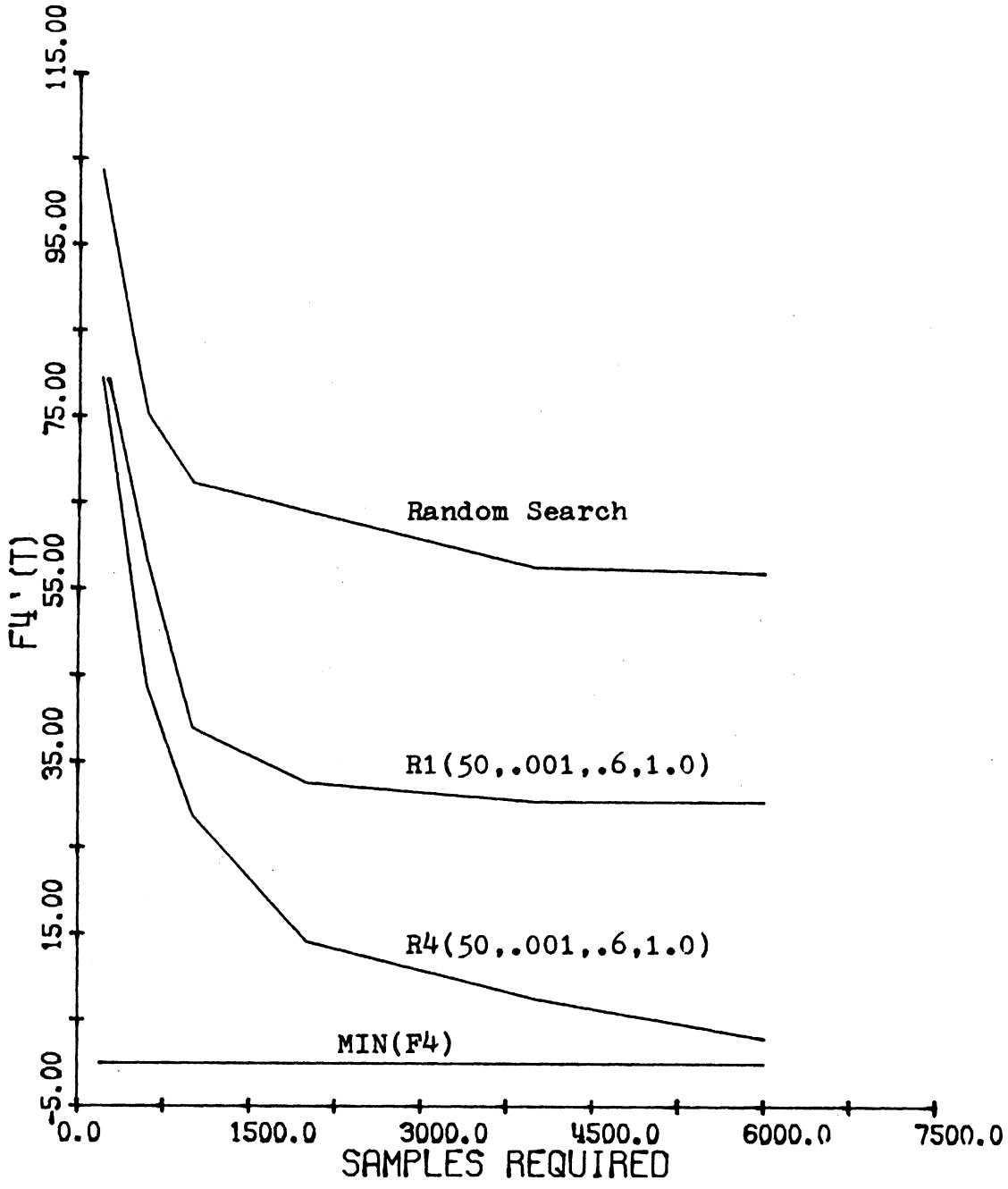


Figure 4.13: A comparison of off-line performance curves generated on F4.

FIG. 4.14: OFF-LINE PERFORMANCE ON F5

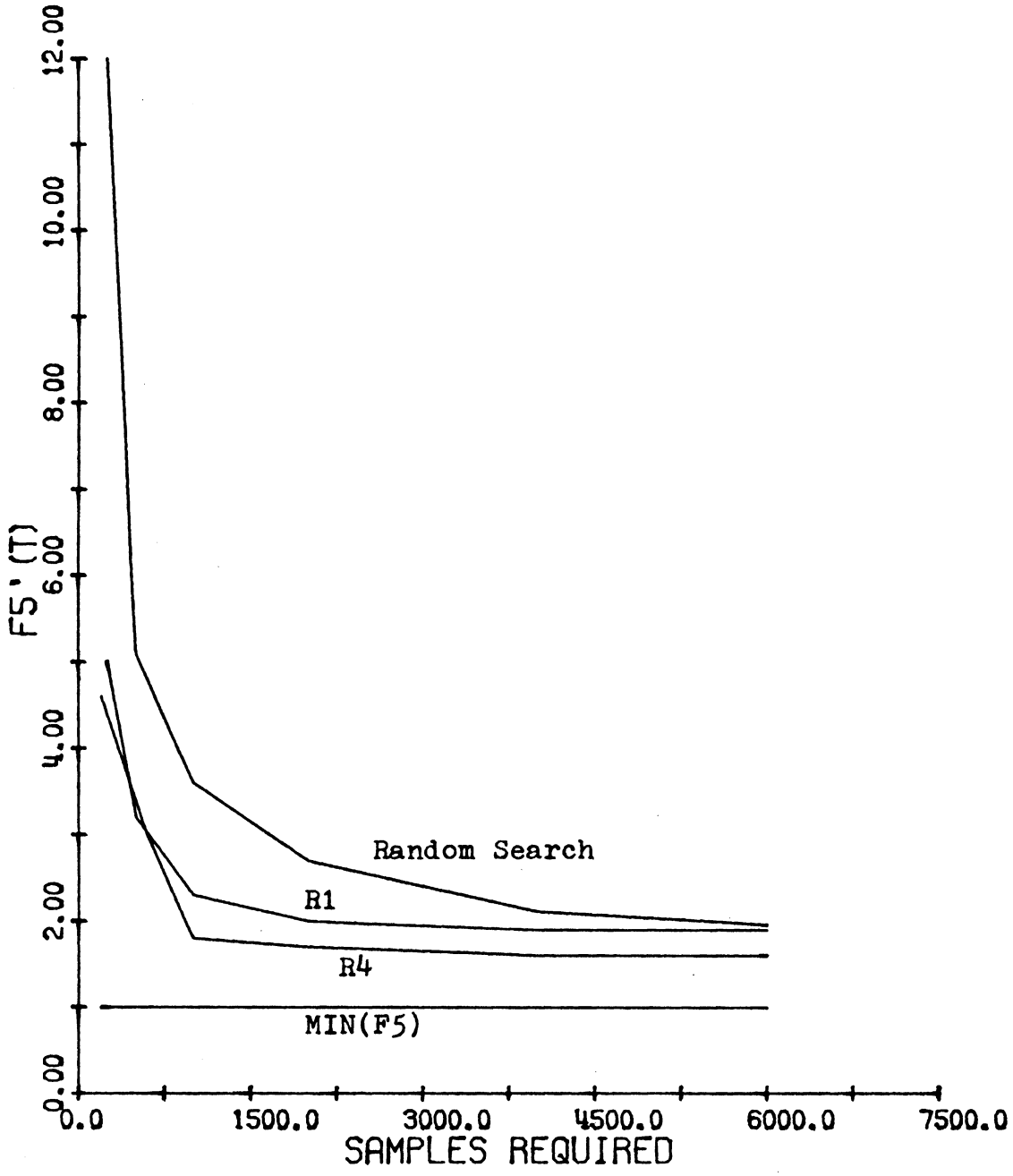


Figure 4.14: A comparison of off-line performance curves generated on F5.

that the problem of pre-mature convergence to a non-optimal plateau is no longer evident. Consistent progress is made over the entire interval of observation. Only on the difficult multimodal surface defined by F5 do we still see convergence to a non-optimal plateau. It is this problem which will be addressed in the next sections.

4.6 Improving the Performance of R4 on F5

As noted in the previous section, considerable progress has been made in improving the performance of finite genetic models on E. By modifying the sampling technique used in R1 so that the actual number of offspring more closely approximate the expected number, the allele loss rate due to stochastic side-effects has been reduced considerably with a corresponding improvement in performance. In addition, by adding an elitist policy to R3, significant improvement on the unimodal surfaces was observed. Figure 4.15 summarizes the remaining fly in the ointment: the performance of genetic plans on F5. By going to plan R3, the premature convergence generated by R1 was replaced by an off-line performance curve which made slow but steady progress over the interval of observation. Adding the elitist policy to R3 improved initial off-line performance, but once again we observe convergence to a non-optimal plateau. This also suggests an explanation to the

FIG. 4.15: OFF-LINE PERFORMANCE ON F5

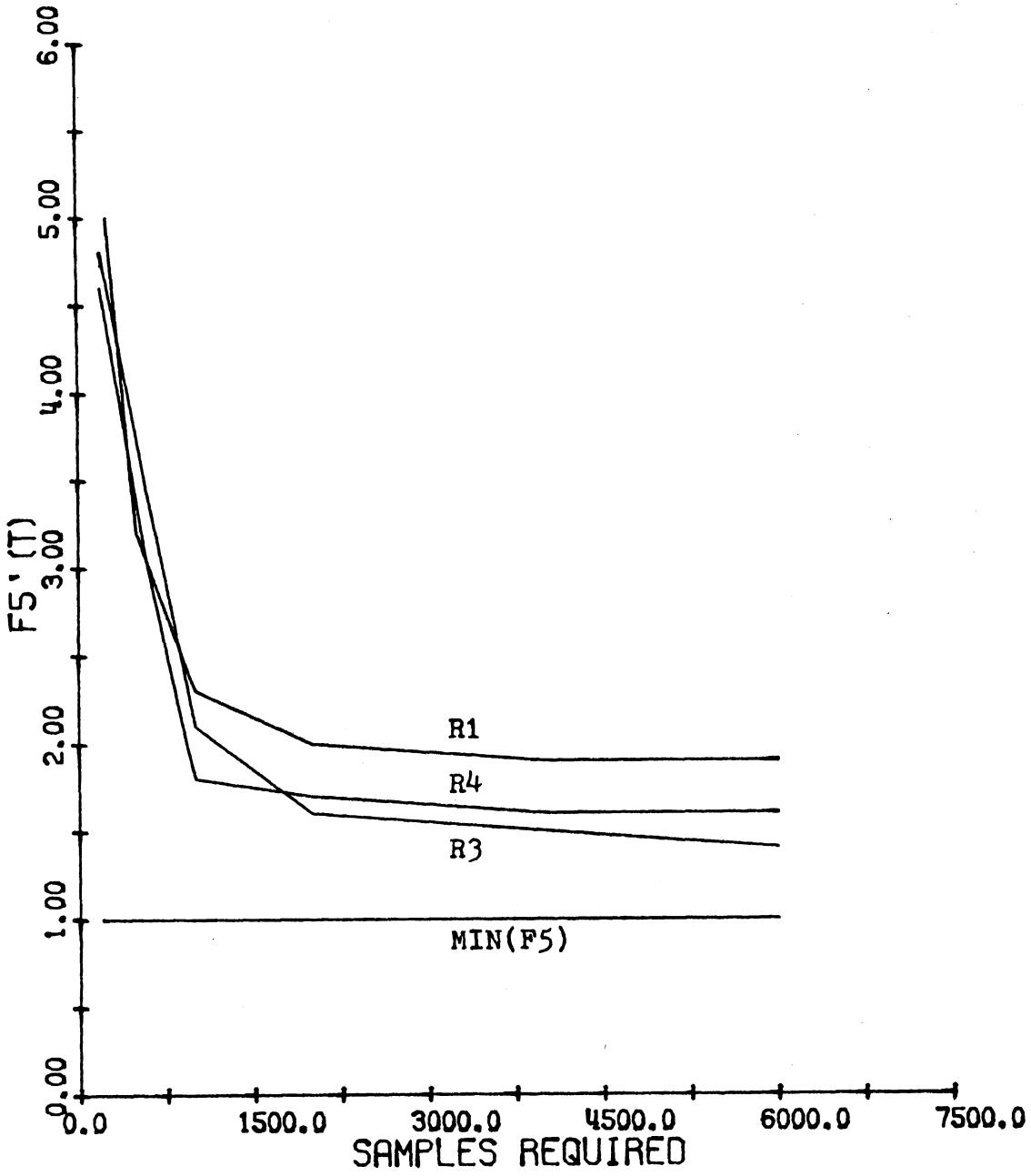
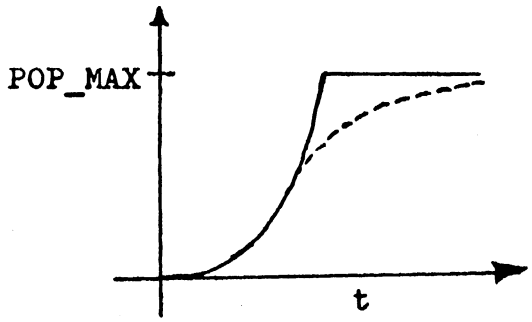


Figure 4.15: Off-line performance curves for genetic plans on F5.

observation noted in the previous section that adding an elitist policy to R3 did not degrade the performance indices for R4 on F5 as much as had been expected. As figure 4.15 illustrates, the average values (performance indices) of the off-line curves generated by R3 and R4 are very nearly the same. Over a longer time interval, the negative effects of the premature convergence with R4 would have been more clearly seen. In this section we address the problem of improving the global search properties of R4 without, hopefully, having to give up the improved local performance.

We begin by considering this problem in terms of the hyperplane analysis introduced in chapter 2. Recall that genetic plans have the property that the best of competing hyperplanes were allocated an exponentially increasing number of trials relative to their competitors. This property was shown to be a consequence of the fact that the number of instances of a particular hyperplane in A(t) changed over time in proportion to the hyperplane's performance relative to its competitors. If a particular hyperplane outperforms its competitors for a relatively small number of generations, we saw that the number of instances of that hyperplane in A(t) increased exponentially to a point of complete dominance. When such a hyperplane is in fact the best, this is precisely what we want to happen with the corresponding allele loss effecting a reduction in the search space.

However, we are working with finite genetic plans which maintain reasonably small populations which evaluate hyperplane performance via sample means based on a relatively small number of samples. As a consequence, it is not difficult to imagine that some of the premature allele loss observed may be the result of non-optimal hyperplanes appearing to be the best for a sufficiently long time to effect a reduction in the search space. More to the point, it is easy to imagine that on the difficult surface defined by F5 a hyperplane associated with a relatively good local optimum could quickly dominate A(t) and cause the observed premature convergence. These observations suggest that the premature allele loss rate and the performance of genetic plans on multimodal functions could be improved by making it more difficult for hyperplanes to dominate A(t). It should be clear, however, that overall performance on E may be seriously degraded unless a solution is chosen carefully, since it is precisely this exponential increase in trials which generates the kind of performance exhibited by R4 on the unimodal surfaces. What we seek is a solution which permits exponential exploitation of the observed best without allowing them to readily dominate a finite population. This suggests that, rather than allow exponential growth until total dominance occurs, genetic plans should admit "controlled" growth in the form of an "S" curve as illustrated below:



Such an approach permits initial exponential exploitation of hyperplanes for rapid performance improvements, while at the same time making it considerably more difficult for a hyperplane to completely dominate A(t).

The difficulty, of course, is in finding a reasonable implementation for this conceptually simple solution. Consider for a moment the alternatives within the R4 framework. Increasing the population size serves both to improve the sample means and increase the time required for a hyperplane to dominate A(t). As we have seen, however, this results in a significant degradation in performance over the interval of observation. Increasing the generation gap reduces the rate at which decisions are made and hence the rate of dominance. However, within the genetic context, a value larger than 1.0 makes no sense. Increasing the crossover rate has the opposite effect from that desired. As we saw in chapter 2, crossover becomes increasingly less likely to interfere with hyperplane growth as it begins to dominate A(t). Increasing the mutation rate seems to be the only possible solution within the R4

framework. As the number of instances of a particular hyperplane begin to dominate A(t), the number of their offspring expected to undergo mutation increases and effects a reduction in the hyperplane's growth rate.

In order to explore the aspects of such a solution, the performance of R4 was evaluated on F5 with mutation rates of .001, .005, .01, and .05, respectively. Figure 4.16 illustrates clearly the effect that increasing the mutation rate has on the off-line performance of R4 on F5. As the mutation rate increases, the shape of the off-line performance curve changes to reflect less dramatic initial performance and more uniform progress over the entire interval of observation. Note that R4(.01) very nearly converges to the minimum within 6000 trials. However, its initial performance is less impressive than R4(.001). This suggests an explanation to the observation noted in the previous section that, unlike R1, the performance of R4 on E was actually degrading slightly by increasing the mutation rate from .001 to .01. With R1, increasing the mutation rate served to reduce its high rate of allele loss and improve performance. However, with R4's reduced allele loss rate and improved local performance, increasing the mutation rate generated an improvement in longer-term performance at the expense of initial performance. Had we evaluated R4 over a longer time interval, the long-term improvements would have been more clearly visible.

Finally, figure 4.17 illustrates clearly what we have

FIG. 4.16: OFF-LINE PERFORMANCE OF R4 ON F5

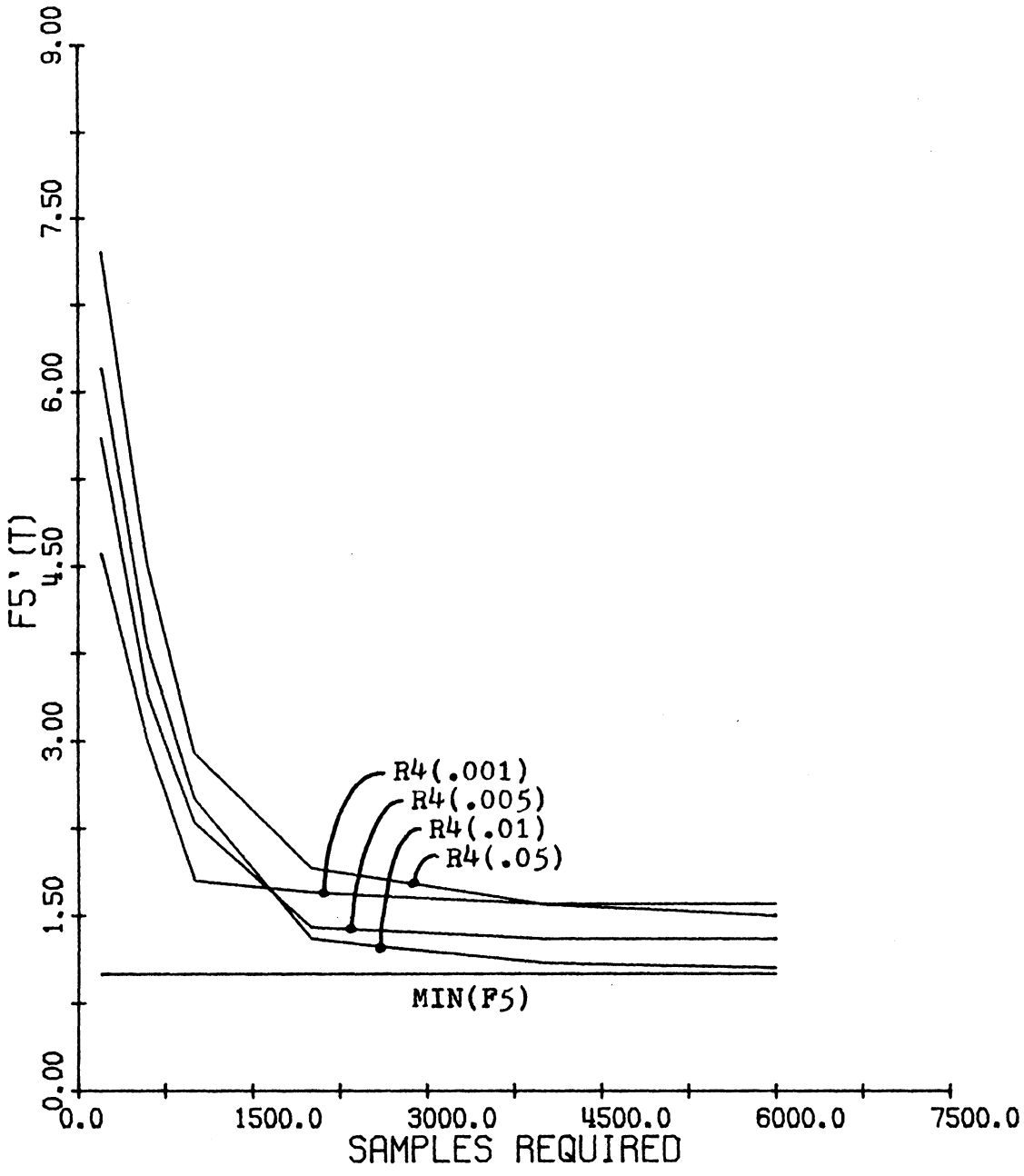


Figure 4.16: Off-line performance for R4 on F5 as a function of mutation rate.

FIG. 4.17: ON-LINE PERFORMANCE OF R4 ON F5

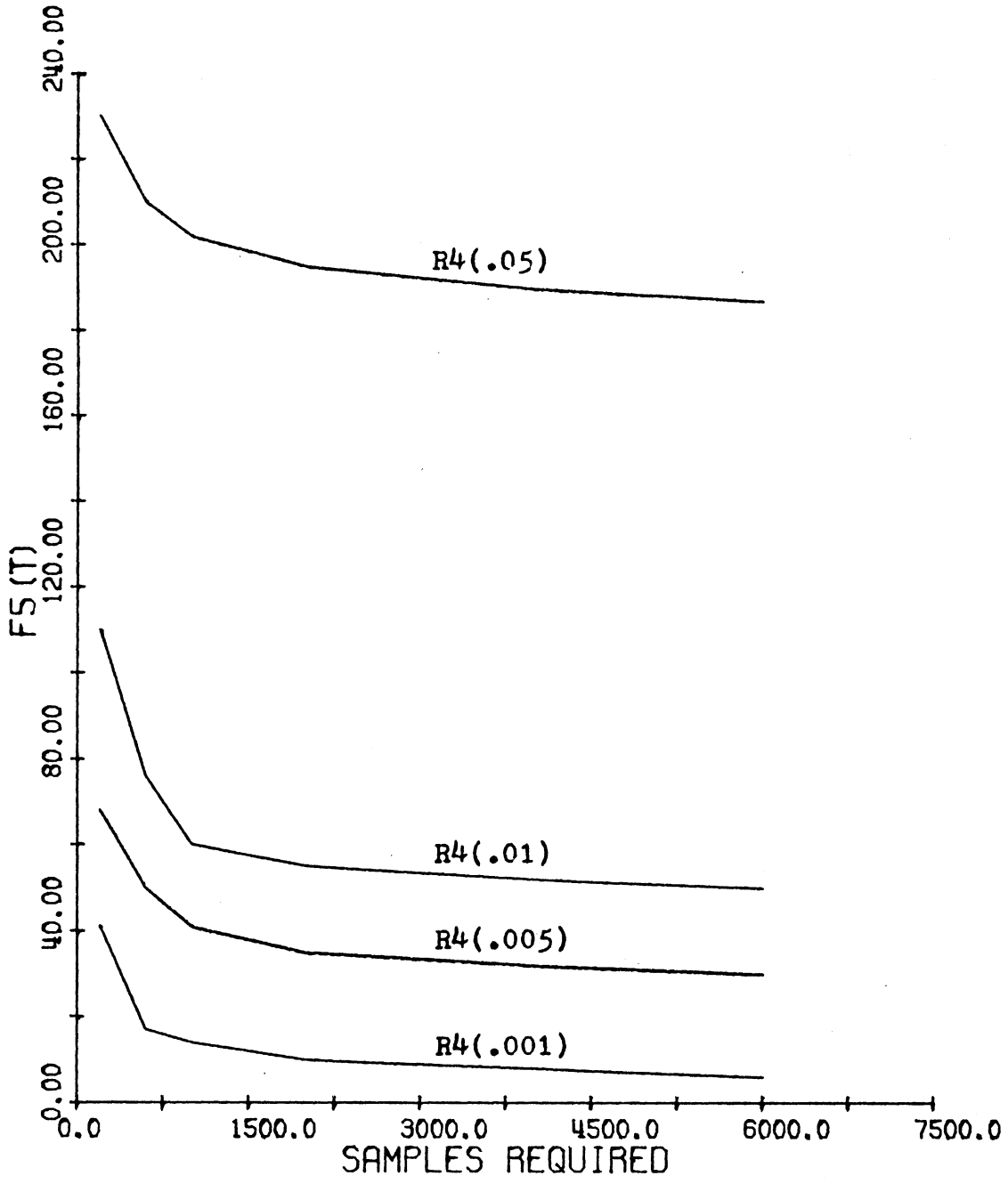


Figure 4.17: On-line performance for R4 on F5 as a function of mutation rate.

seen before. If on-line performance is required, any increase in the mutation rate seriously degrades performance.

4.7 Crowding Factor Model R5

Holland has suggested that the kind of controlled growth we seek in finite genetic models occurs in nature as a consequence of crowding. That is, as more and more like individuals dominate an environmental niche, the competition for limited resources increases rapidly resulting in lower life expectancies and birth rate. In this section we consider the effects of including such a feature in genetic adaptive plans as an alternate to increasing the mutation rate in order to improve performance on multimodal surfaces.

If we think of the genetic plans in terms of the overlapping generation models, connections between the natural and artificial systems are more intuitive. In particular, consider a model in which only a few offspring are produced each generation (e.g. G=.1). Plans of this sort produce A(t+1) as follows:

- produce G*N offspring using selection and the genetic operators
- using a uniform distribution on A(t), insert the G*N offspring into A(t) by selecting G*N individuals to "die".

Stated in this form, the concept of life expectancy is more clearly defined for these artificial systems. And, as we noted in chapter 3, the expected number of offspring of an

individual is directly related to the number of generations it survives. What we seek is a method for reducing the life expectancy of individuals which are instances of a hyperplane rapidly dominating A(t).

One interesting approach to this problem is as follows. When selecting an individual in A(t) to die, pick several candidates initially and choose that one which is most similar to the new individual being inserted into the population. For the genetic models under study here, similarity is defined in terms of the number of matching binary alleles. Intuitively, this approach has the right characteristics. Until a hyperplane begins to dominate, the modified replacement policy has little effect, allowing initial exponential growth. However, as a hyperplane begins to dominate A(t), instances of that hyperplane become increasingly more likely to be replaced by other instances, resulting in reduction in the hyperplane growth rate.

This approach will clearly increase the amount of processing required to produce the next generation. Additional information (which, incidentally, has a derivative-like flavor) is being computed at each time step to control the allocation of trials. In this section, however, we will ignore the processing time tradeoffs and concentrate on the effects of this approach on the performance of genetic plans on E.

In order to gain further insight into this approach to controlled growth, a fifth parameter was defined for the

genetic plans under study: a crowding factor parameter CF which specifies the number of individuals initially selected as candidates to be replaced by a particular offspring. CF=1 is equivalent to no crowding factor, and as CF increases, the more likely it becomes that similar individuals replace one another. As an initial study of the effects of the crowding factor, the behavior of the following four members of this new class of genetic plans was analyzed: R5(50,.001,.6,.1,CF) where CF = 1, 2, 3, and 4. Figure 4.18 illustrates the allele loss rates of each of the plans on test function F1. Recall that, because of the symmetry of F1, theoretically there should be no allele loss. As one can see, increasing the crowding factor results in a dramatic decrease in the allele loss rate. Figures 4.19 and 4.20 give the off-line and on-line performance curves generated by these plans on test function F5. Recall that these studies were motivated by the observation that the off-line performance curves of R4 on F5 suggested that premature convergence was still a problem on multimodal surfaces. Figure 4.19 illustrates that the crowding factor has in fact the right effect on F5 with R5(2) very nearly converging to the global minimum within the interval of observation. Figure 4.20 illustrates that, like mutation, increasing the crowding factor adversely affects on-line performance. This, of course, is due to the constraints placed on the number of samples allocated to the observed best.

FIG 4.18: R5 ALLELE LOSS ON F1

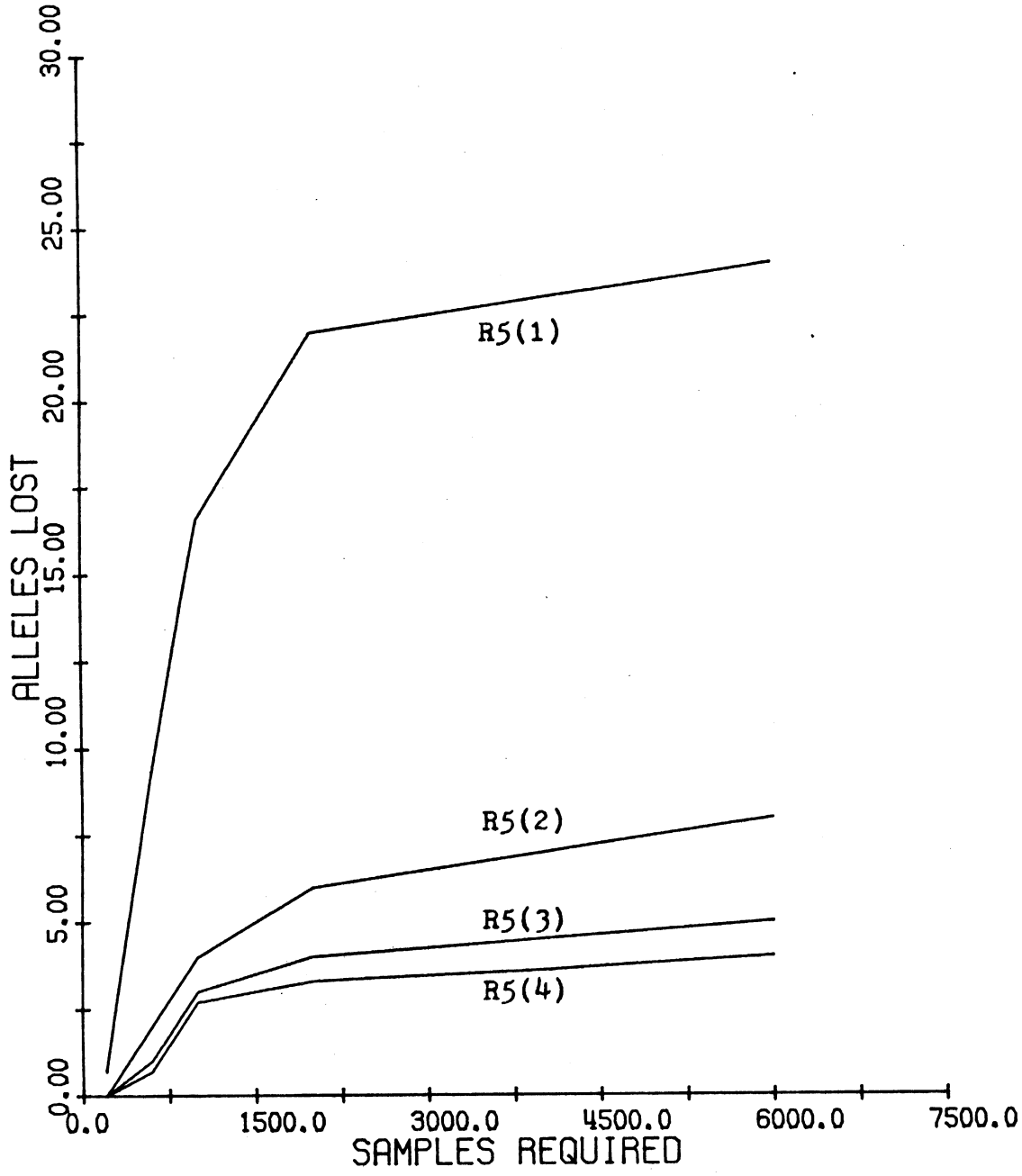


Figure 4.18: Allele loss for R5 on F1.

FIG. 4.19: OFF-LINE PERFORMANCE OF R5 ON F5

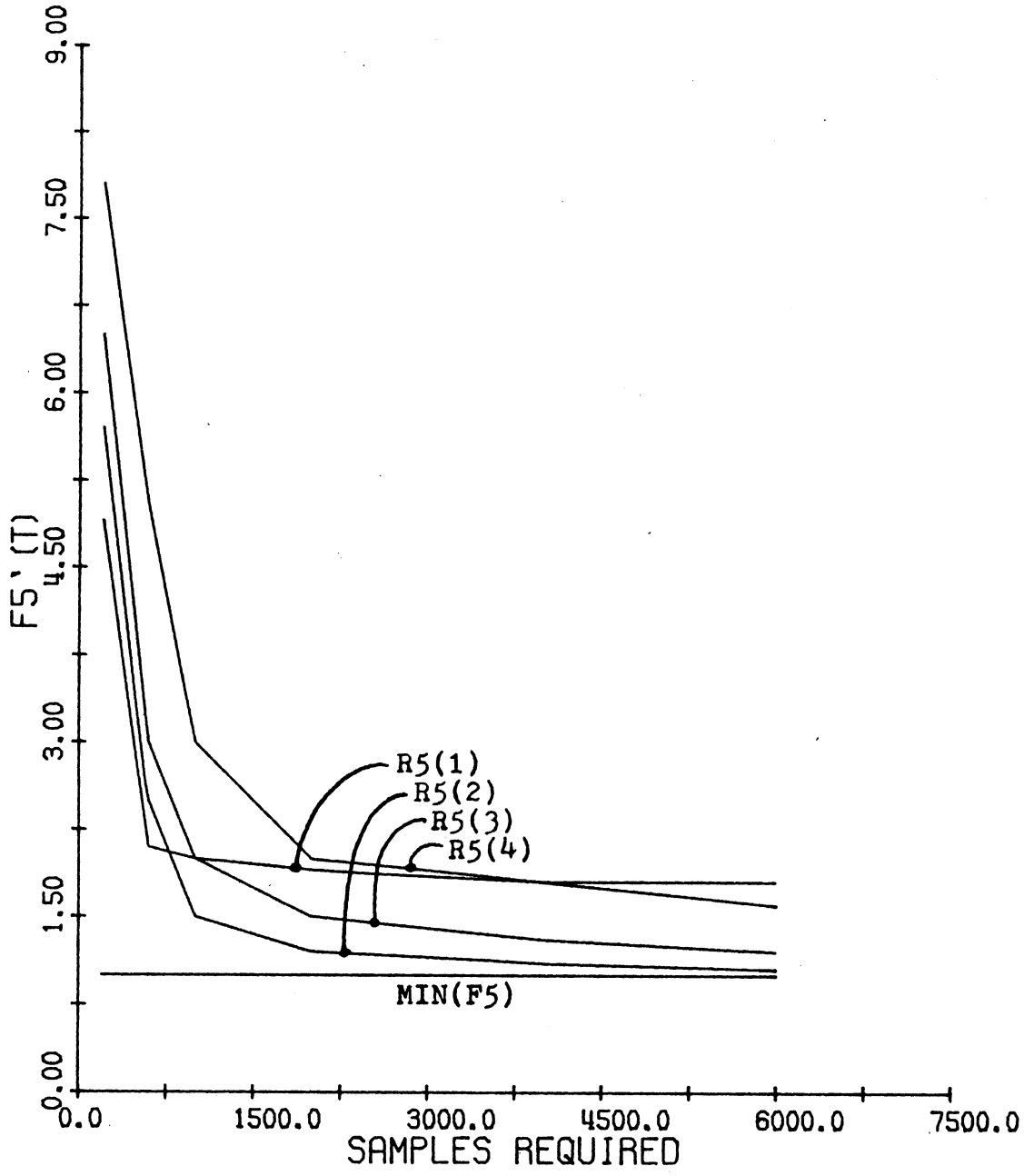


Figure 4.19: Off-line performance for R5 on F5 as a function of the crowding factor.

FIG. 4.20: ON-LINE PERFORMANCE OF R5 ON F5

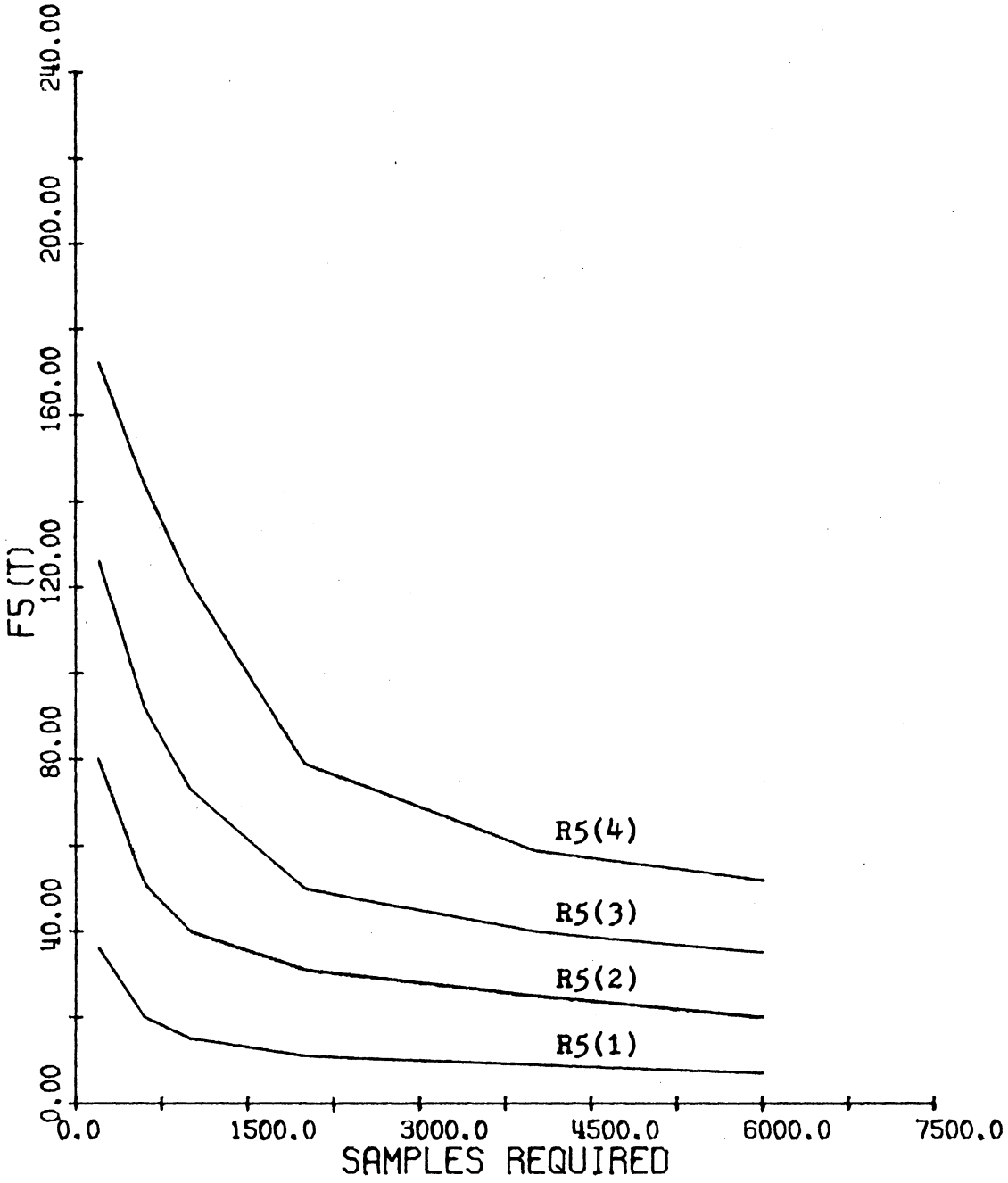


Figure 4.20: On-line performance for R5 on F5 as a function of the crowding factor.

Any reduction slows the overall improvement rate. Notice, however, that these negative effects are not nearly so severe with increases in the crowding factor as they are with increases in mutation. The reason seems obvious. Mutation controls the growth rate by randomly changing allele values, an approach which becomes more likely to produce performance degradation as adaptation progresses. On the other hand, the crowding factor provides the same kind of controlled growth rate by reducing the number of offspring produced by instances of dominating hyperplanes, rather than modifying offspring allele values.

We began this analysis of the crowding factor by arbitrarily choosing an overlapping generation model for which G=.1. Since we saw in previous studies that increasing the generation gap led to improved performance, it is of interest to explore that possibility here. In this situation we do not have quite the freedom in choosing G that we had before since as G increases beyond .5, the concept of the crowding factor becomes less meaningful, and makes little sense at all if nearly all the population is being replaced. As a consequence, the effects of crowding were analyzed for two other values of G, .2 and .4. Figures 4.21 and 4.22 give the off-line performance curves for each of these settings and illustrate two points of interest. The first observation is that, for the same crowding factor, increasing the

FIG. 4.21: OFF-LINE PERFORMANCE OF R5 ON F5

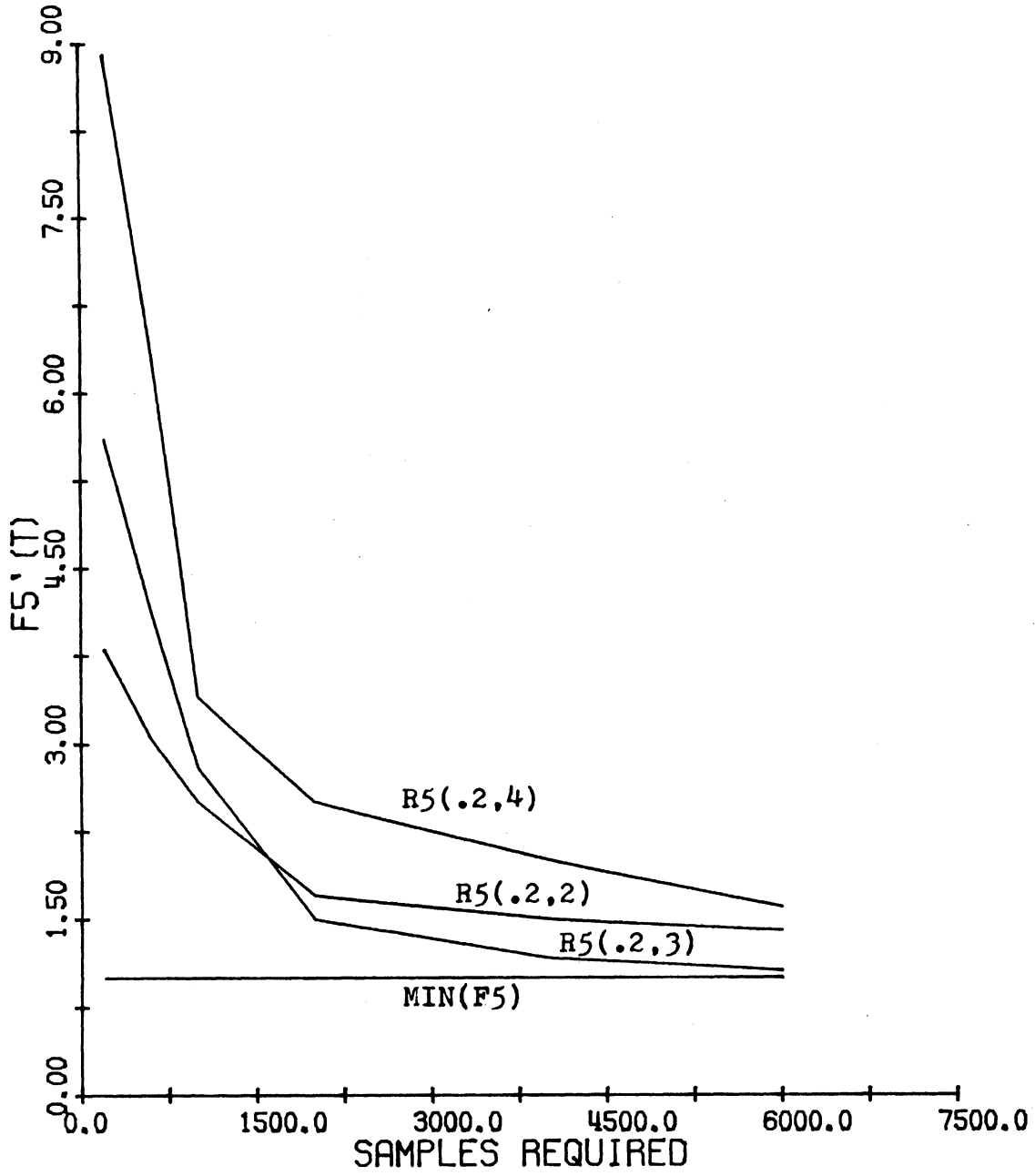


Figure 4.21: Off-line performance for R5 on F5 as a function of the crowding factor.

FIG. 4.22: OFF-LINE PERFORMANCE OF R5 ON F5

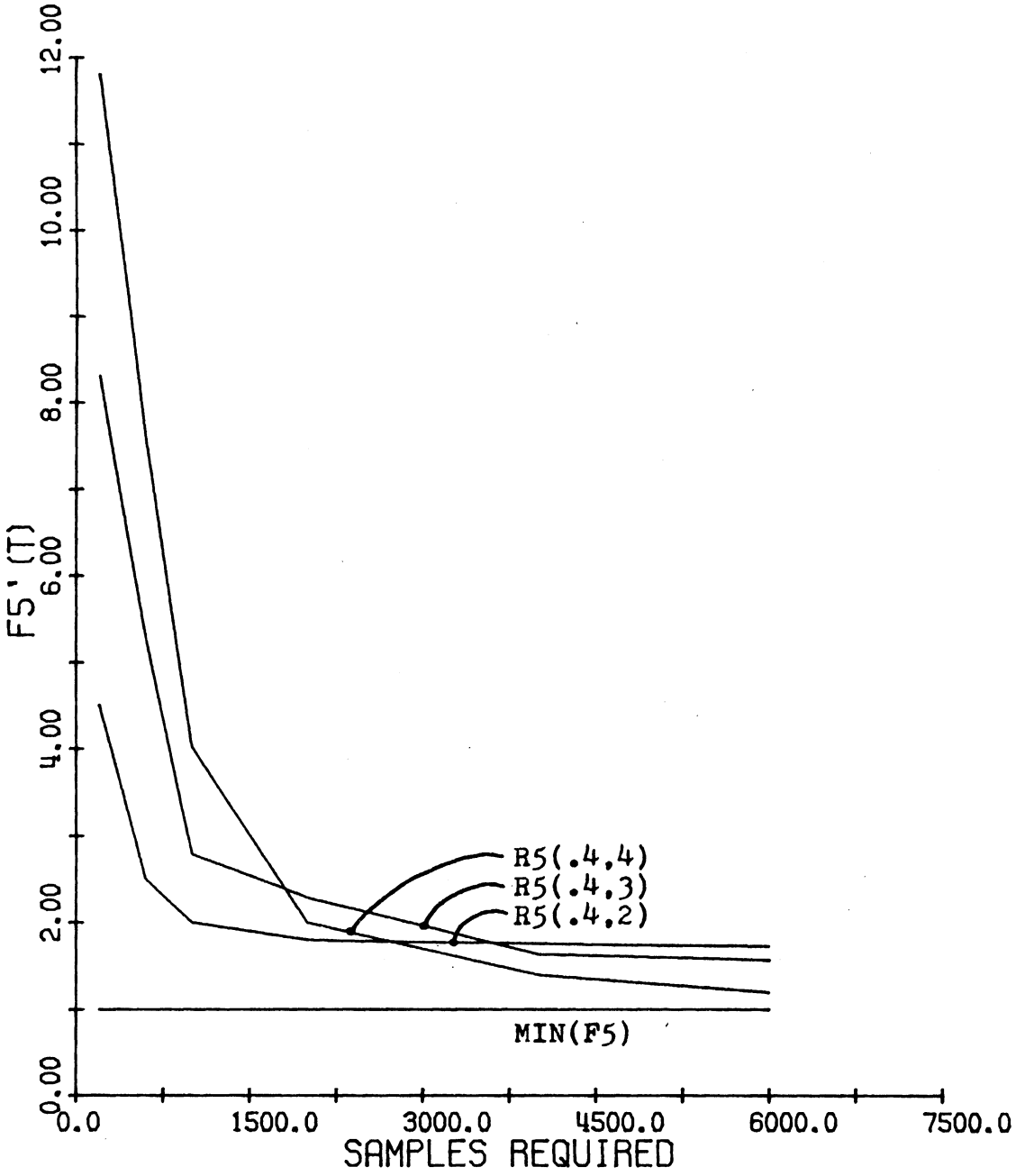


Figure 4.22: Off-line performance for R5 on F5 as a function of the crowding factor.

generation gap actually degraded the off-line performance curves of R5 on F5. In each case an increase in the crowding factor was required to preserve the same kind of performance curve. These observations suggest an interesting interaction between crowding and generation gaps: the larger the generation gap, the less effective crowding becomes. One possible explanation for this interaction is as follows. As the generation gap increases, the lifespan (in terms of generations) of an individual is reduced with a compensating increase in the number of offspring per generation. The crowding factor operates by reducing the lifespan of individuals and, hence, with shorter average lifespans its effectiveness is reduced.

Finally, it remains to be seen what effect crowding has on the overall performance of genetic plans on E. To analyze these effects, the following members of R5 were evaluated on E:

R5(50,.001,.6,.1,2)
R5(50,.001,.6,.1,3)
R5(50,.001,.6,.1,4)
R5(50,.001,.6,.2,2)
R5(50,.001,.6,.2,3)
R5(50,.001,.6,.2,4)
R5(50,.001,.6,.4,3)
R5(50,.001,.6,.4,4)

Tables 4.3a and 4.3b give the corresponding off-line and on-line performance indices computed over 6000 trials. As in the previous section these results indicate quite clearly the tradeoffs in performance we face. Including a crowding factor in genetic plans improves significantly

T=6000	R5(.1,2)	R5(.1,3)	R5(.1,4)	R5(.2,2)	R5(.2,3)	R5(.2,4)	R5(.4,3)	R5(.4,4)
$x^*_{F1}(T)$	.186	.185	.093	.092	.103	.161	.170	.129
$x^*_{F2}(T)$	.284	.111	.103	.226	.125	.097	.476	.411
$x^*_{F3}(T)$	-26.8	-26.38	-25.9	-27.5	-26.73	-27.08	-27.18	-26.5
$x^*_{F4}(T)$	21.79	28.55	28.85	19.36	25.13	23.85	22.73	23.36
$x^*_{F5}(T)$	3.07	4.48	5.07	5.28	4.79	4.37	3.88	3.21
$X^*_E(T)$	-.299	1.39	1.64	-.508	.684	.279	.015	.122

Table 4.3a: Off-line performance of R5 on E

T=6000	R5(.1,2)	R5(.1,3)	R5(.1,4)	R5(.2,2)	R5(.2,3)	R5(.2,4)	R5(.4,3)	R5(.4,4)
$x_{F1}(T)$	6.49	7.47	7.02	5.03	6.94	6.79	6.42	6.90
$x_{F2}(T)$	130.05	145.85	152.24	100.84	134.26	128.11	112.85	127.7
$x_{F3}(T)$	-20.95	-18.64	-17.7	-21.46	-20.4	-18.8	-21.1	-19.0
$x_{F4}(T)$	76.2	95.85	101.87	71.37	87.12	91.83	74.99	87.11
$x_{F5}(T)$	45.3	58.5	65.14	46.7	56.39	66.28	67.12	67.89
$X_E(T)$	47.36	57.81	61.71	40.49	52.86	54.84	48.06	53.12

Table 4.3b: On-line performance of R5 on E

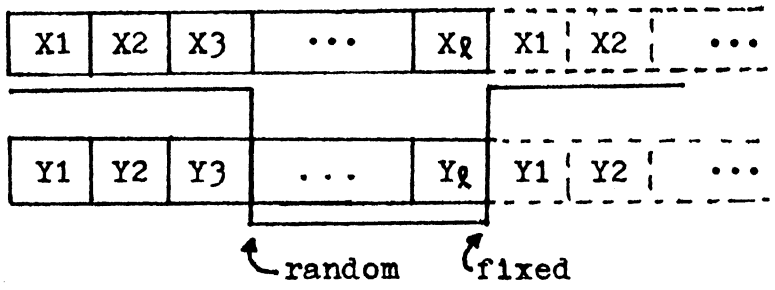
their performance on multimodal surfaces at the expense of rapid convergence on the unimodal surfaces. On the other hand, the distinction between these tradeoffs becomes less evident as the interval of observation increases and more emphasis is placed on convergence.

4.8 Generalized Crossover Model R6

Recall from the hyperplane analysis introduced in chapter 2 that genetic plans generate near-optimal allocation of trials to competing hyperplanes whose definition length (the shortest gene segment containing all the fixed positions) was short relative to the chromosome length ℓ. This was due to the fact that crossover disrupted the allocation of trials according to performance with a probability directly proportional to the definition length of a hyperplane. This means that the performance of the genetic plans under study may in fact be representation dependent. That is, one binary representation of the space to be searched may be less affected by crossover than another representation because high-performance groups of alleles are physically closer together. One solution to this problem is to allow the representation itself undergo adaptation be introducing a genetic inversion operator which physically permutes genes on a chromosome without loss of functional position. Studies by Franz (1972) suggest that changes effected by inversion are difficult to

detect except perhaps in long-term behavior. Since we are concerned with practical applications, we explore an alternate approach in this section: the possibility of modifying the crossover operator itself to reduce representation dependencies.

Recall again how crossover has been defined to this point. After selecting two individuals, a crossover point is selected uniformly from the ℓ-1 positions between the ℓ genes. The offspring consists of the first segment of the first parent up to the crossover point and the remaining segment of the second parent. If we think of a chromosome as a circle with the first gene immediately following the last
then it becomes immediately clear that there are in fact 2 crossover points: one fixed at position zero and the other randomly selected.



An immediate generalization to the present crossover operator is to allow both crossover points to be randomly selected. Further generalization can be made by allowing an arbitrary number of crossover points. But notice that the actual number of crossover points is always

an even number since (from the circular viewpoint) you always end up back where you started.

In order to understand what effects these changes have on the allocation of trials to competing hyperplanes, we generalize the discussions of chapter 2. We need to compute the probability that an offspring after crossover lies in a different hyperplane partition element than its parent. If we let $x_1, x_2, \cdots, x_k$ be the k positions defining a hyperplane, then the offspring will certainly lie in the same hyperplane if there were an even number of crossover points between each consecutive pair of fixed points (x_i, x_j). Hence, if we think of the hyperplane's k fixed points as dividing a chromosome into k segments (in the circular viewpoint), the probability of staying in the same hyperplane is at least as great as the probability that each segment contains an even number of crossover points. Restating this as the probability of the loss of an offspring to another hyperplane, we have:

$$\Pr\left[\text{crossover loss}\right] \leq 1 - P_{nk}$$

where P_{nk} is the probability that each of the k segments received an even number (including zero) of the n=2m crossover points.

In order to get a feeling for how these probabilities change by increasing the number of crossover points, consider the effects on second-order hyperplanes. They divide the chromosome up into two segments of length

$\ell_1=x_2-x_1$ and $\ell_2=\ell-(x_2-x_1)$ with the probability of a randomly selected crossover point falling in one of them given by ℓ_1/ℓ. If we assume the convention that whenever an odd number of crossover points are randomly selected, the final even crossover point is defined to occur at position zero, then we have:

$$P_{n2} = \sum_{i=0}^{m} \binom{n}{2i} \left(\frac{\ell_1}{\ell}\right)^{2i} \left(\frac{\ell_2}{\ell}\right)^{n-2i}$$

where n is the number of randomly selected crossover points with m=n/2 (integer division).

Figure 4.23 illustrates how the loss probability $1-P_{n2}$ changes both as a function of the definition length ℓ_1 and the number of randomly selected crossover points n for second order hyperplanes with a chromosome length of $\ell=30$. Notice that there are two distinct families of curves: one for n even and one for n odd. When an odd number of crossover points are randomly selected, the probability of loss for widely spaced fixed points remains high since it remains likely that all the crossover points will fall in the long segment defined by the two fixed points. On the other hand, randomly selecting an even number of crossover points immediately drops the loss probabilities to .5 or less with $\ell/2$ spacings becoming the most likely victims.

How these generalizations of the crossover operator

FIG 4.23: PROBABILITY OF LOSS DUE TO CROSSOVER

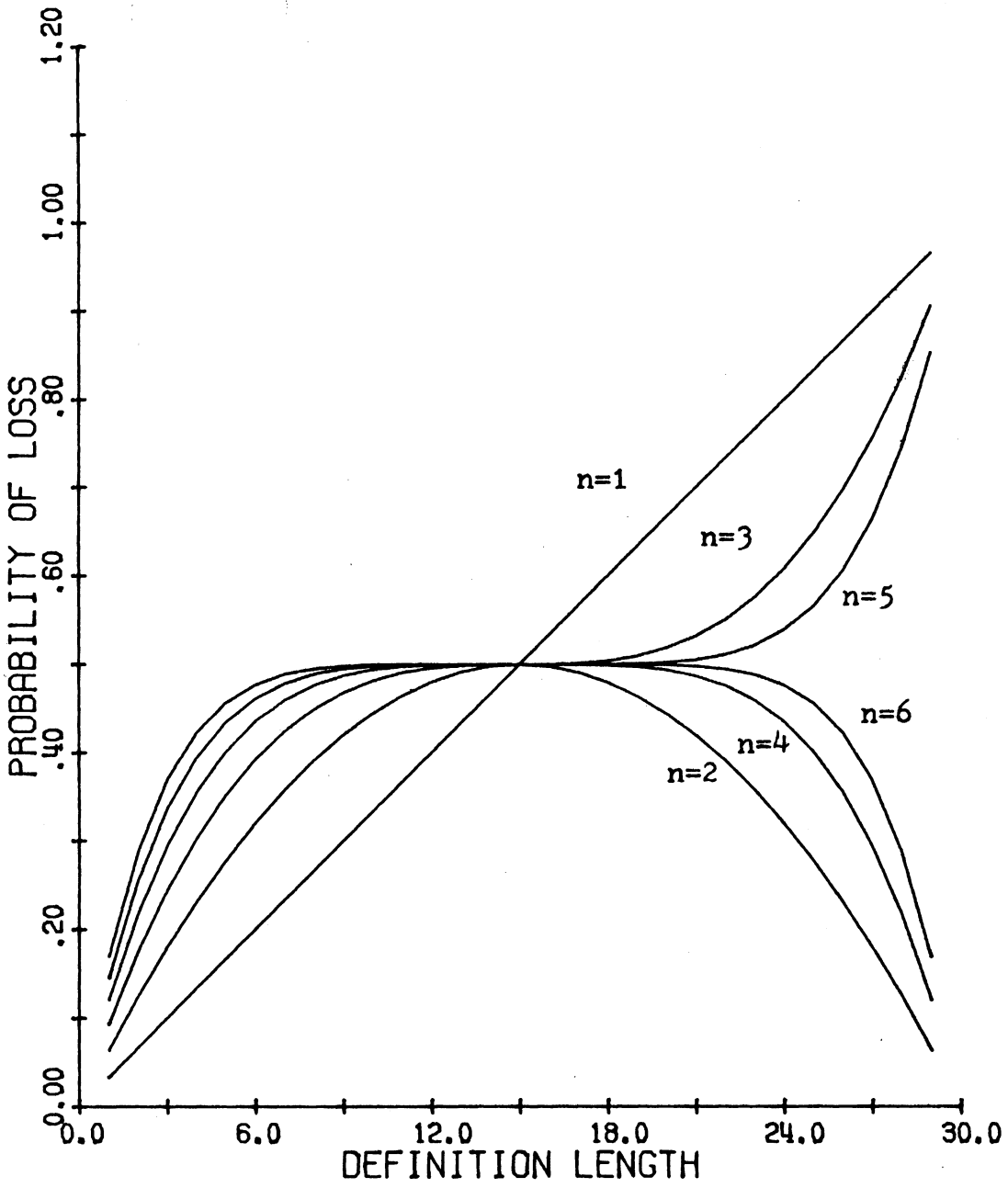


Figure 4.23: Loss probability curves for second order hyperplanes on chromosomes of length 30 as a function of the number of crossover points.

will affect the performance of genetic plans is not clear. We may find that they reduce representation dependencies at the cost of lower overall performance on E. On the other hand one can imagine that perhaps a modest increase in the number of randomly selected crossover points may generate performance improvements while larger numbers of crossover points may be too disruptive to the allocation of trials to higher order hyperplanes.

To explore these possibilities, plan R5 was modified to accept a sixth parameter, CP, which specifies the number of crossover points to be randomly selected. Up to this point we have been implicitly using a value of CP=1. If CP is odd, the final crossover point is assumed to occur at position zero.

As an initial attempt to understand the implications of generalized crossover, five members of R6 were evaluated on test function F1:

R6(50,.001,.6,1.0,1,1)
R6(50,.001,.6,1.0,1,2)
R6(50,.001,.6,1.0,1,3)
R6(50,.001,.6,1.0,1,4)
R6(50,.001,.6,1.0,1,8)

Figure 4.24 depicts the allele loss generated by these members of R6 on test function F1, and illustrates that the allele loss rate actually increases as CP does. This is a surprising observation for which an explanation is not immediately clear. It may very well be the case that the previous disruption of the allocation of trials to the longer hyperplanes may have counteracted some

FIG 4.24: R6 ALLELE LOSS ON F1

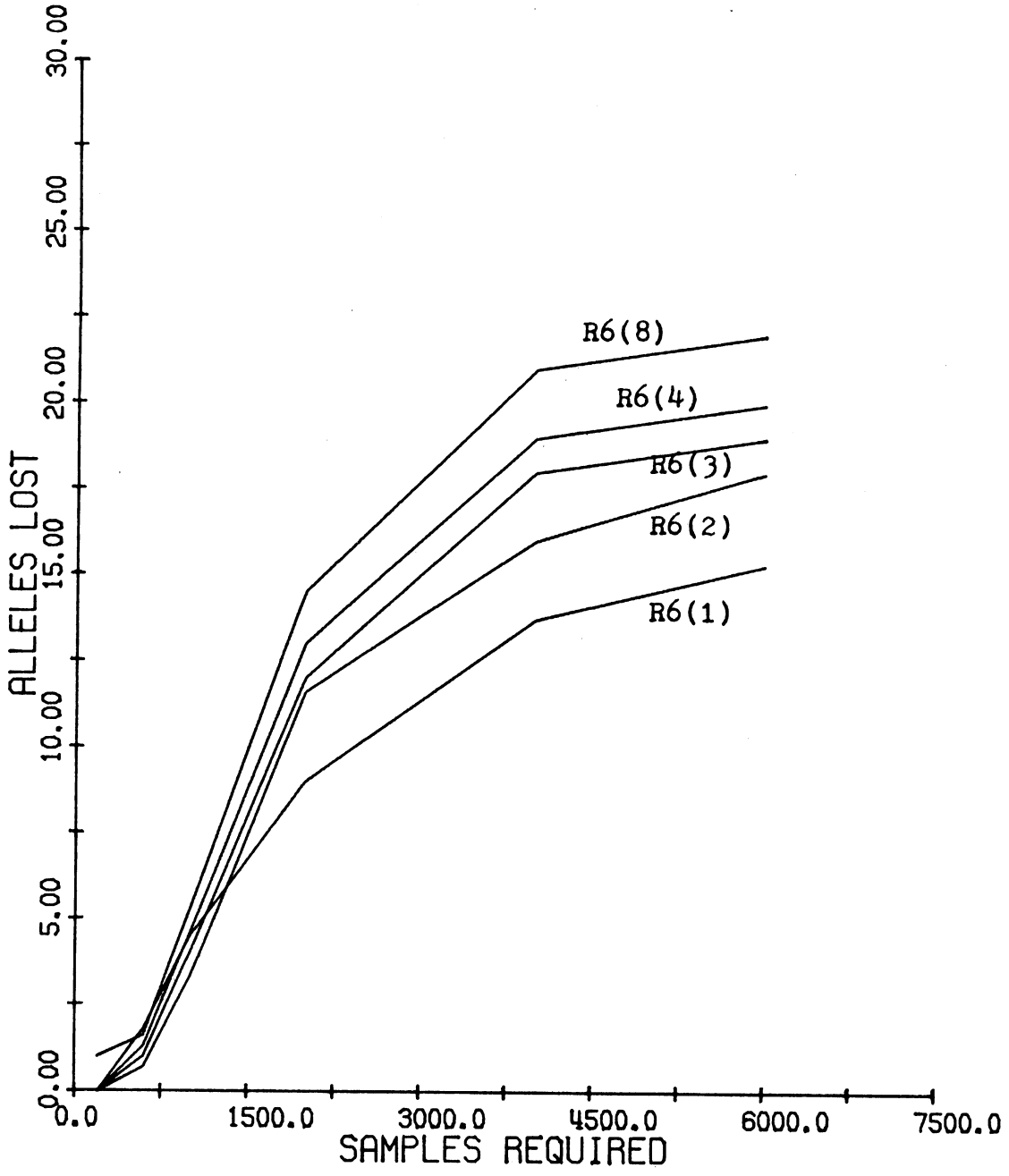


Figure 4.24: Allele loss for R6 on F1 as a function of the number of crossover points.

of the stochastic side-effects of genetic drift and, hence, the removal of some of this disruption opened the door for increased allele loss.

Figures 4.25 and 4.26 give the off-line and on-line performance curves generated by these members of R6 on F1. In general, increasing the number of crossover points seems to degrade slightly the off-line performance of R6 with R6(8) exhibiting the old problem of premature convergence. This is probably due to the previously noted increased allele loss rate. It is also interesting to note that initial on-line performance also degrades somewhat as CP increases, suggesting that increasing the number of crossover points leads to a less conservative sampling policy in the initial stages of adaptation.

Finally, to evaluate the effects of generalized crossover on the performance of R6 on E, the behavior of the following members was analyzed on E:

R6(50,.001,.6,1.0,1,1)
R6(50,.001,.6,1.0,1,2)
R6(50,.001,.6,1.0,1,3)
R6(50,.001,.6,1.0,1,4)
R6(50,.001,.6,1.0,1,8)
R6(50,.01, .6,1.0,1,2)
R6(50,.001,.6, .1,2,2)

Tables 4.4a and 4.4b summarize the off-line and on-line performance indices for these evaluations over 6000 trials. The first five members analyzed differed only in the number of crossover points selected. The best overall off-line performance was achieved with CP=2, chiefly on the basis of its performance on F4. Note that

FIG. 4.25: OFF-LINE PERFORMANCE OF R6 ON F1

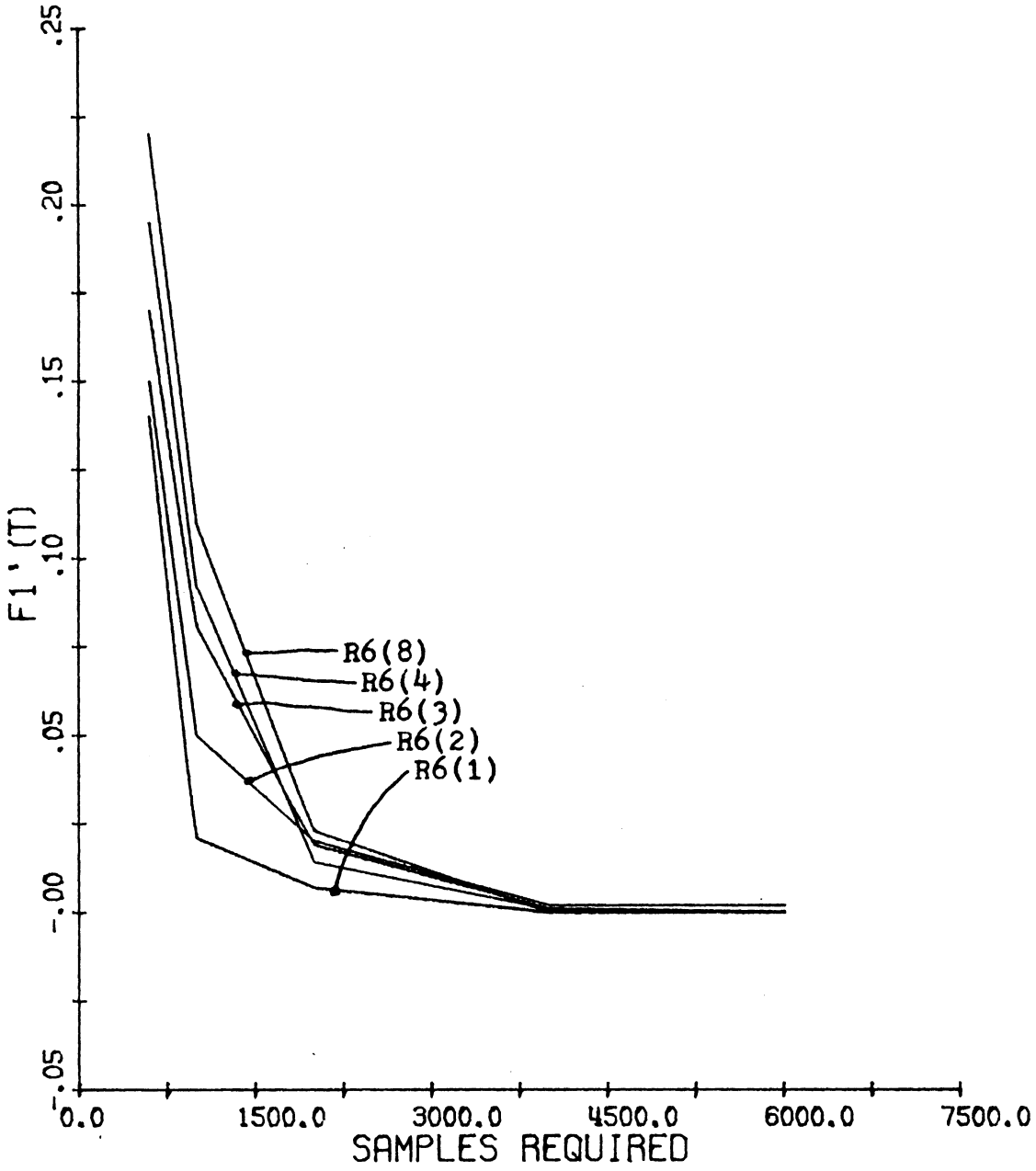


Figure 4.25: Off-line performance curves for R6 on F1 as a function of the number of crossover points.

FIG. 4.26: ON-LINE PERFORMANCE OF R6 ON F1;

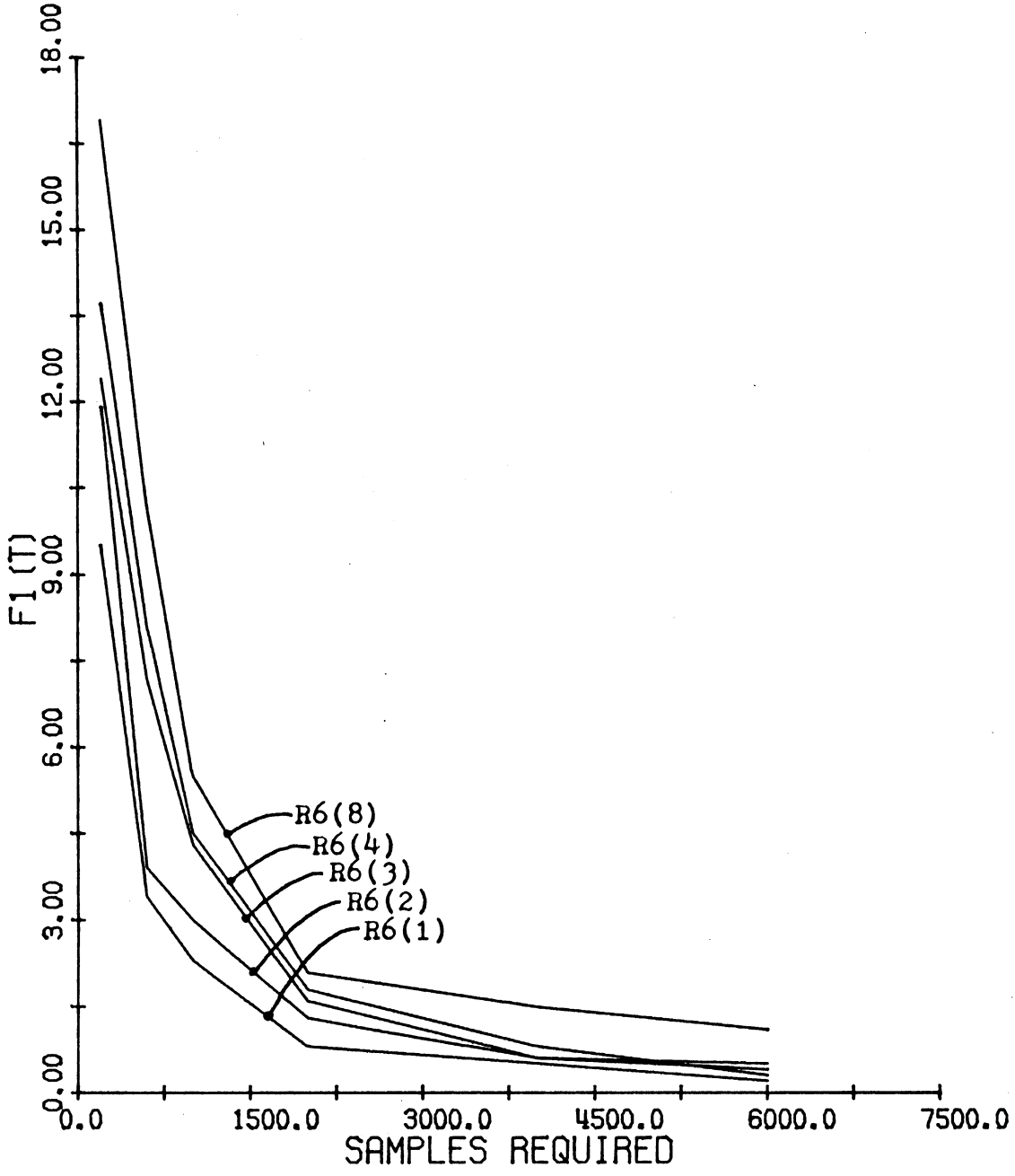


Figure 4.26: On-line performance curves for R6 on F1 as a function of the number of crossover points.

T=6000	CP=1	CP=2	CP=3	CP=4	CP=8	P_c=.01 CP=2	G =.1 CP=2 CP=2
$x^*_{F1}(T)$	.113	.117	.122	.127	.138	.129	.116
$x^*_{F2}(T)$	.221	.197	.237	.244	.289	.133	.215
$x^*_{F3}(T)$	-28.2	-27.99	-27.86	-27.53	-26.91	-28.3	-27.1
$x^*_{F4}(T)$	17.62	14.01	18.19	21.06	25.23	19.59	23.9
$x^*_{F5}(T)$	3.34	4.74	5.27	5.61	5.85	3.95	3.87
$X^*_E(T)$	-1.38	-1.79	-.808	-.098	.919	-.899	.200

Table 4.4a: Off-line performance of R6 on E

T=6000	CP=1	CP=2	CP=3	CP=4	CP=8	P_c=.01 CP=2	G =.1 CF=2 CP=2
$x_{F1}(T)$	2.32	2.67	2.88	3.23	3.51	6.47	5.07
$x_{F2}(T)$	34.76	36.99	38.97	41.25	46.85	112.6	96.7
$x_{F3}(T)$	-26.49	-26.17	-25.83	-25.14	-24.61	-22.16	-22.5
$x_{F4}(T)$	40.73	37.54	42.64	47.97	49.8	95.9	83.8
$x_{F5}(T)$	34.34	36.91	38.27	42.38	45.63	82.65	74.5
$X_E(T)$	17.12	17.59	19.39	21.94	24.24	55.09	47.51

Table 4.4b: On-line performance of R6 on E

performance of F5 degrades significantly as CP increases, indicating again how important a low allele loss rate is. As we saw on F1, on-line performance degrades significantly as CP increases.

The last two members were chosen as likely candidates to improve the off-line performance of R6 on E. Time and resources prohibited a more detailed analysis of the six parameters defining R6. However, since we had earlier noted the increased allele loss rate of R6 on F1, the two most likely candidates for improvement were increased mutation rates and the crowding factor. One instance of each was chosen; neither improved off-line performance over 6000 trials and, as we have seen before, both degrade on-line performance.

4.9 Summary

We began this chapter by analyzing the performance of the basic family R1 of genetic plans on E. While this family outperforms random search on E, we noted that there was considerable room for improvement. To this basic plan we added an elitist policy which biased the allocation of trials slightly toward the hyperplanes which produced the currently best individual. This resulted in improved performance on E, particularly on the unimodal surfaces. In fact, performance on F5 was degraded suggesting that an elitist policy improves the local search properties of genetic plans. As an

alternative, expected value model R3 was introduced which attempted to improve performance by minimizing the difference between the actual and expected number of offspring produced by individuals in A(t). This produced a significant increase in performance on E. R3 was then modified to include the previously mentioned elitist policy. Again the performance on E was improved to the extent that on test functions F1-F4, there were no signs of premature convergence over the interval of observation. Rather, R4 generated steady progress toward the minimum with convergence within 6000 trials on F1 and F2. F5, however, remained a difficult challenge. To that end, several members of the R4 family were analyzed on F5 to see whether a change in parameters would improve the off-line performance curve. Increasing the mutation rate to .01 seemed about the most effective change for F5 but resulted in an overall decrease in off-line performance on E. As an alternative approach to improving the global search properties of R4, a crowding factor model was introduced which attempted to slow the growth rate of hyperplanes beginning to dominate the finite population A(t). Like increasing the mutation rate, the crowding factor improved off-line performance on F5 at the expense of on-line performance, but the tradeoff was less pronounced. Finally, generalized crossover model R6 was introduced in an attempt to alleviate possible representation problems caused by

the disrupting effect crossover has on long-definition hyperplanes. By allowing several crossover points to occur when generating an offspring, the disruptiveness on long-definition hyperplanes can be considerably reduced. Although time did not permit a complete analysis of the effects, no significant improvement in performance on E was noted. In fact overall performance was seen to degrade as the number of crossover points increased.

Chapter 5

PERFORMANCE ANALYSIS OF FUNCTION OPTIMIZERS

5.1 Introduction

In the last two chapters we have developed a class of genetic plans which performs well on the environment E in comparison with random search. In this chapter we provide an alternate point of comparison by evaluating the performance of several function optimization techniques on E.

In our discussion of function optimization in chapter 1, we noted that the problem of finding function extrema has generally been divided into two subproblems: finding the nearest local extremum (local or unimodal search) and finding the global extremum (global or multimodal search). Many sophisticated techniques have been developed which solve the local search problem (see, for example, Jacoby and Kowalik(1972) or Huang(1970)). However, much less success is evident for the global search problem. Several approaches have been proposed if appropriate bounds can be assumed on the derivatives of the functions to be optimized (see, for example, Bremermann (1970) or Brent (1971)). Alternatively, one can perform some sort of patterned search in an attempt to locate the global optimum (see, for example, Hill (1969)). Unfortunately, for most of these techniques the computation time grows rapidly with the dimensionality

of the problem, and they are used in practice only for low-dimensional problems. As a consequence, one is left with two alternatives for a more general global function optimizer. Either one runs a good local optimizer a sufficiently large number of times to assure all local optima have been found or revert to some form of random search. Since we know we can do better than random search with the genetic algorithms, we consider in this chapter the alternative of restarting a local optimizer.

5.2 Local Optimization Techniques

The most successful local minimization techniques have come from the area of iterative descent methods. The idea here is to reduce the problem of finding the minimum to a sequence of one-dimensional searches along a direction vector p_k. That is, at each step a point x_{k+1} is generated where $x_{k+1} = x_k + \lambda_k p_k$ and $f(k_{k+1}) < f(k_k)$. The techniques differ in how the direction vector p_k and the step size λ_k are chosen. We will consider two different techniques, one which requires that derivative information be available for the function being optimized and one which does not.

A well-studied approach to the choice of direction vectors is to perform a sequence of one-dimensional minimizations along a sequence of conjugate directions. If the function to be minimized is quadratic of dimension n, then in theory only n such one-dimensional

minimizations are required for convergence. In practice, however, conjugate direction techniques are applied to arbitrary functions with heuristic modifications to prevent the conjugate directions from becoming linearly dependent and reducing the search space. Powell (1964) proposed a technique for calculating conjugate directions without derivative information by means of a series of one-dimensional minimizations. Subsequently, Brent (1971) and others have made modifications to improve the performance of this approach. Since Brent's algorithm is available in software form (PRAXIS), it seemed a reasonable choice as a representative of conjugate direction methods which do not require derivatives.

A somewhat more sophisticated approach to the problem of local minimization, using variable metric methods, was introduced by Fletcher and Powell (1963) with subsequent variations proposed by Broyden (1970) and Huang (1970). In this case a sequence of nxn "metric" matrices (where n is the dimensionality of the search space) are constructed using gradient information to simultaneously provide a linear transformation of the search space into one less badly scaled and provide a sequence of conjugate directions for one-dimensional searches. As a consequence, each step is computationally more expensive than, for example, the previous approach (particularly when n is large), but generally requires no more than n steps on quadratic functions even when

the function to be minimized is badly scaled. In practice, variable metric methods are applied to arbitrary functions with heuristics for preventing the metric matrices H_k from becoming singular. Since the Fletcher-Powell algorithm is available in software form (DFP), it seemed a reasonable choice as a representative of the variable metric methods.

5.3 Performance Evaluation Conventions

One of the most difficult things to find in the function optimization literature is a comprehensive comparative analysis of the performance of various optimization techniques. Invariably, a paper will cite as performance evidence the fact that one technique required fewer function evaluations to minimize a particular function from a particular starting point. In reality one finds that the comparison depends not only on the starting point but also on a number of "hidden" parameters such as one-dimensional search accuracies, initial step sizes, estimates of scaling, and so on. Applying the algorithm to a different function or even a different starting point requires more parameter "tuning" for the published results. In this thesis we have been concerned with the quality of robustness, that is, the ability of an algorithm to perform well in a wide variety of situations. For function optimization analysis, this suggests that we evaluate algorithms

over a variety of starting points and require that any "hidden" parameters be fixed over the duration of the evaluation.

In order to provide direct comparisons with the performance of the genetic algorithms on E, several conventions were adopted. On-line and off-line performance measurements were made for PRAXIS and DFP in each of two modes: local mode and global mode. In local mode, a random starting point is chosen and the algorithm is run until it converges. If convergence occurs within 6000 trials (the interval of observation), the performance measures are extrapolated from the point of convergence out to 6000 trials by assuming $f_e(t) = f_e^*(t) = \widetilde{FMIN}$ where $\widetilde{FMIN}$ is the minimum at convergence. By averaging this performance over a number of random starting points, we have a direct comparison between local optimizers and genetic plans. In global mode, the algorithm is restarted with a new random starting point each time it converges until it has allocated 6000 trials. By averaging global performance over a number of random initial starting points, we have a direct comparison between proposed global optimizers and genetic plans.

Recall from appendix A that each test function in E was in fact restricted to a bounded subspace of R^n of the form $|x_i| \leq b$. PRAXIS and DFP are unconstrained minimization techniques. No attempt was made to prevent excursions outside of the bounded search space. How-

ever, all random starting points were chosen from a uniform distribution over the bounded search space. Recall also that the bounded spaces were discretized by specifying a resolution factor Δx_i. For fairness in comparison, no finer resolution of the minimum was required of PRAXIS and DFP. For PRAXIS convergence is assumed when successive extimates of the minimum essentially satisfy

$$|x_k - x_{k-1}| \leq T$$

For each test function in E, T was set to the discretization factor Δx_i. For DFP, convergence is assumed whenever the gradient at a particular estimate of the minimum x_k satisfies

$$G(x_k) \leq \epsilon$$

For each test function in E, ϵ was set to $G(x_{min} + \Delta x_i)$, the gradient one resolution step away from the minimum.

Finally, an attempt was made to equalize the fact that DFP required derivative information about the functions being minimized. As we noted in chapter 1, in many adaptive system applications, the performance functions to be optimized are not available in mathematical closed forms. Rather, they are usually "black box" problems for which only function values are readily available. This means that derivative information must be estimated by function evaluations taken small distances away. This suggests that gradient information

is equivalent at the very least to n function evaluations (where n is the dimension of the space) and perhaps 2n or more depending on the accuracy required. Since the DFP algorithm we used expected exact derivative information, it seemed unfair to use derivative estimates. Rather, exact derivatives were computed upon request for each of the test functions in E, but at the same time n function evaluations were computed as a conservative way of equalizing for this additional information.

5.4 Performance Evaluation of PRAXIS and DFP

Figures 5.1 - 5.10 give the off-line and on-line performance curves produced by PRAXIS and DFP in local mode. Each of these curves represents the average of 20 independent trials using random starting points within the bounded subspaces defined in appendix A.

Figures 5.1 and 5.2 illustrate the performance of PRAXIS and DFP on test function F1. Notice the time scale here in relationship to those of the genetic algorithms in the previous chapter. Both PRAXIS and DFP converge within 60 trials while the genetic algorithms required several thousand. These curves indicate the kind of performance which is possible with local optimizers when the assumptions about the function being minimized actually hold. With a low-dimensional, nicely scaled quadratic function like F1, one can hardly do better. The on-line performance curves on F1 bring

FIG 5.1: OFF-LINE PERFORMANCE ON F1

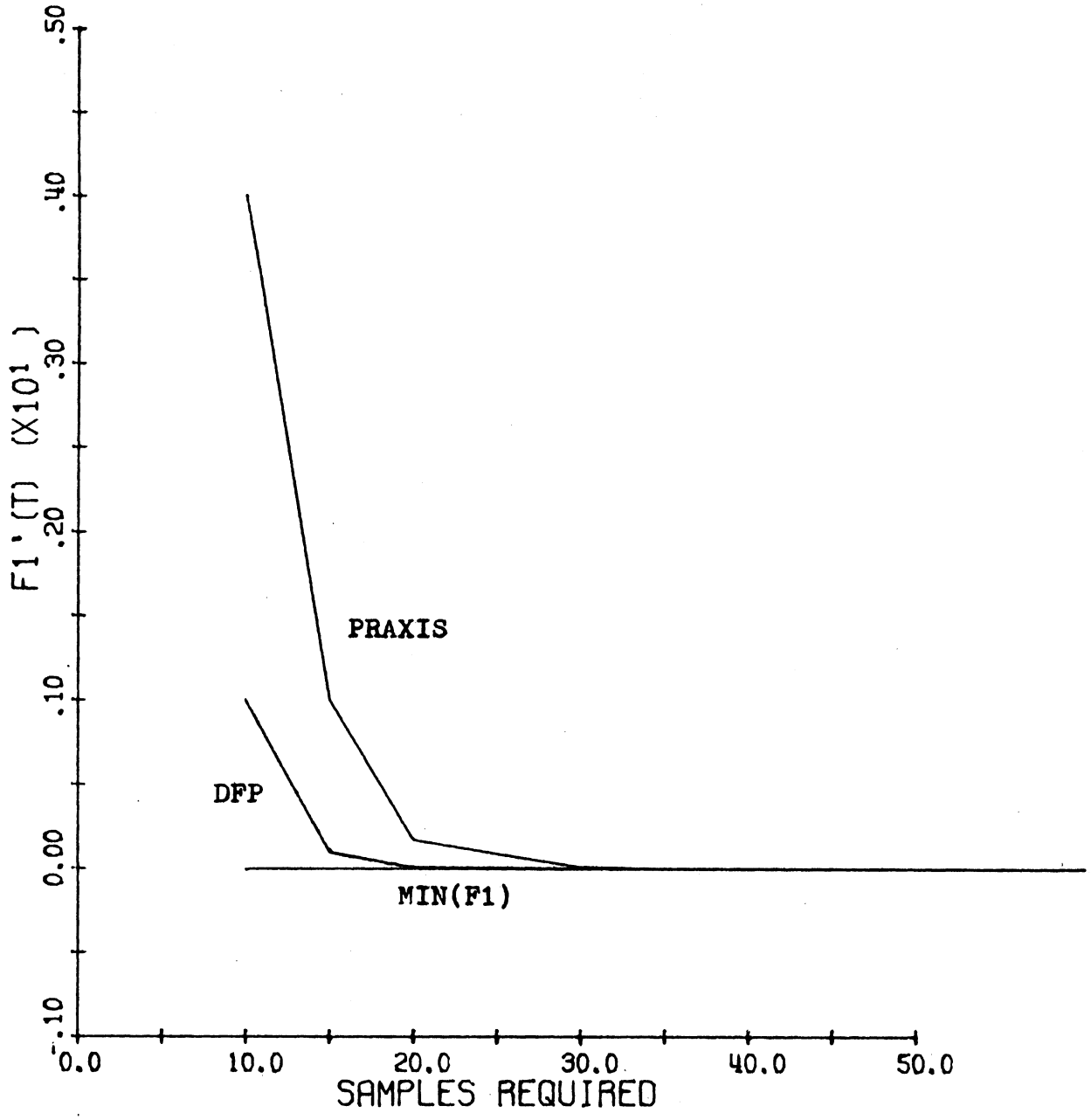


Figure 5.1: Off-line performance curves for PRAXIS and DFP in local mode on F1.

FIG. 5.2: ON-LINE PERFORMANCE ON F1

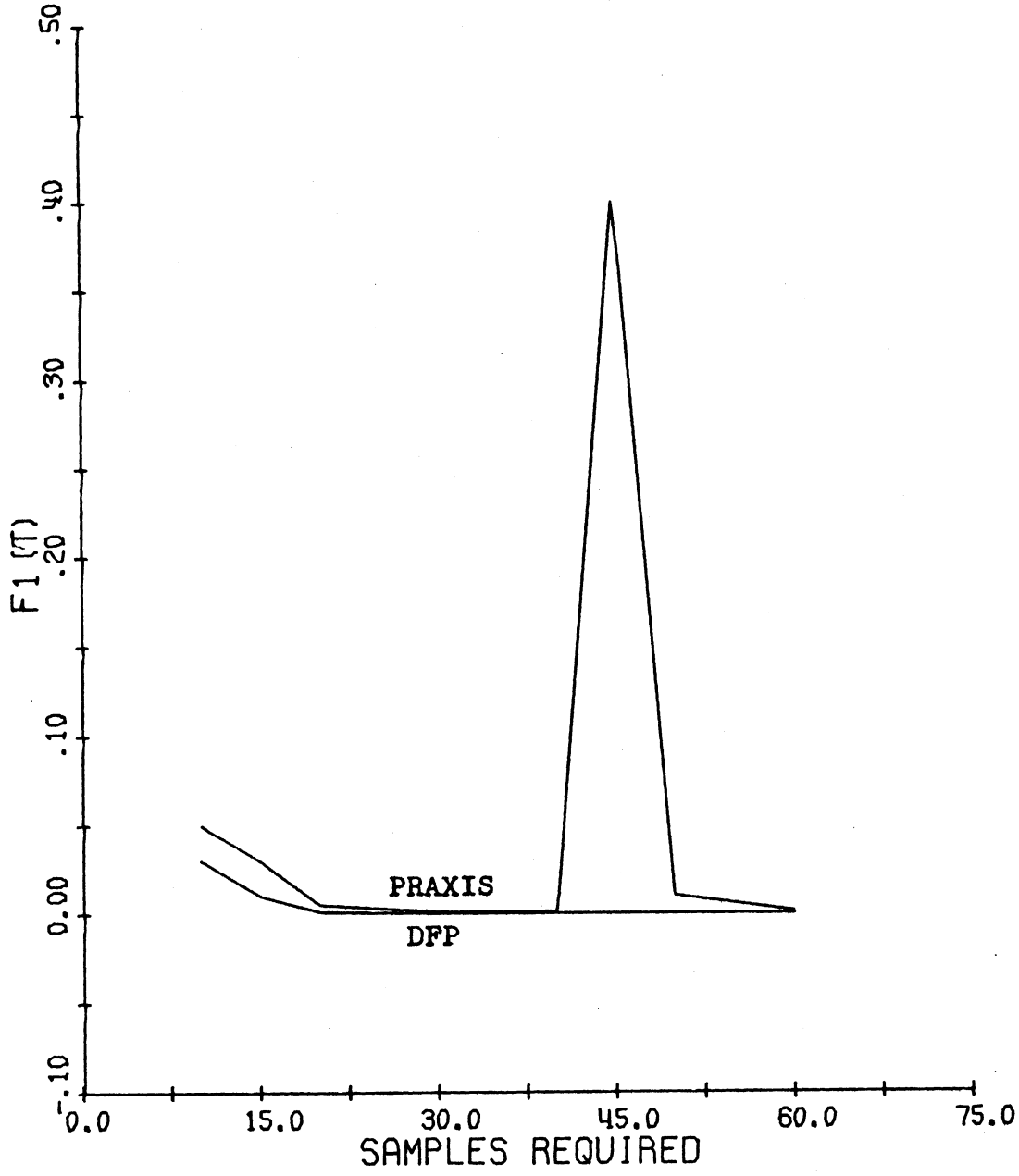


Figure 5.2: On-line performance curves for PRAXIS and DFP in local mode on F1.

out an interesting characteristic of PRAXIS. To avoid the problem of being caught in a narrow valley, when convergence is imminent, PRAXIS tries several steps in random directions. This shows up immediately in on-line performance since the probability of improvement is small.

Figures 5.3 and 5.4 illustrate the local performance curves generated on F2. F2 violates several of the assumptions made concerning the function to be minimized. It is non-convex and non-quadratic, and is also badly scaled. Again, both DFP and PRAXIS converged in far less time than the genetic algorithms, although they both required considerably more trials than on F1. Notice again how the random strategy of PRAXIS shows up in the final stages of on-line performance.

Figures 5.5 and 5.6 illustrate the local performance curves generated on F3. As they indicate, F3 gave both local optimizers considerably more difficulty than F1 and F2. The stopping criterion used by PRAXIS was never satisfied within 6000 trials, requiring manual termination. On the other hand, because DFP used a gradient stopping criterion, it stopped almost immediately on whatever plateau was selected by the random starting point. As a consequence, in local mode it converged on the average to AVE(F3) = -2.5, which was then extrapolated as discussed earlier over the remainder of the 6000 trials.

Figures 5.7 and 5.8 illustrate the local performance

FIG. 5.3: OFF-LINE PERFORMANCE ON F2

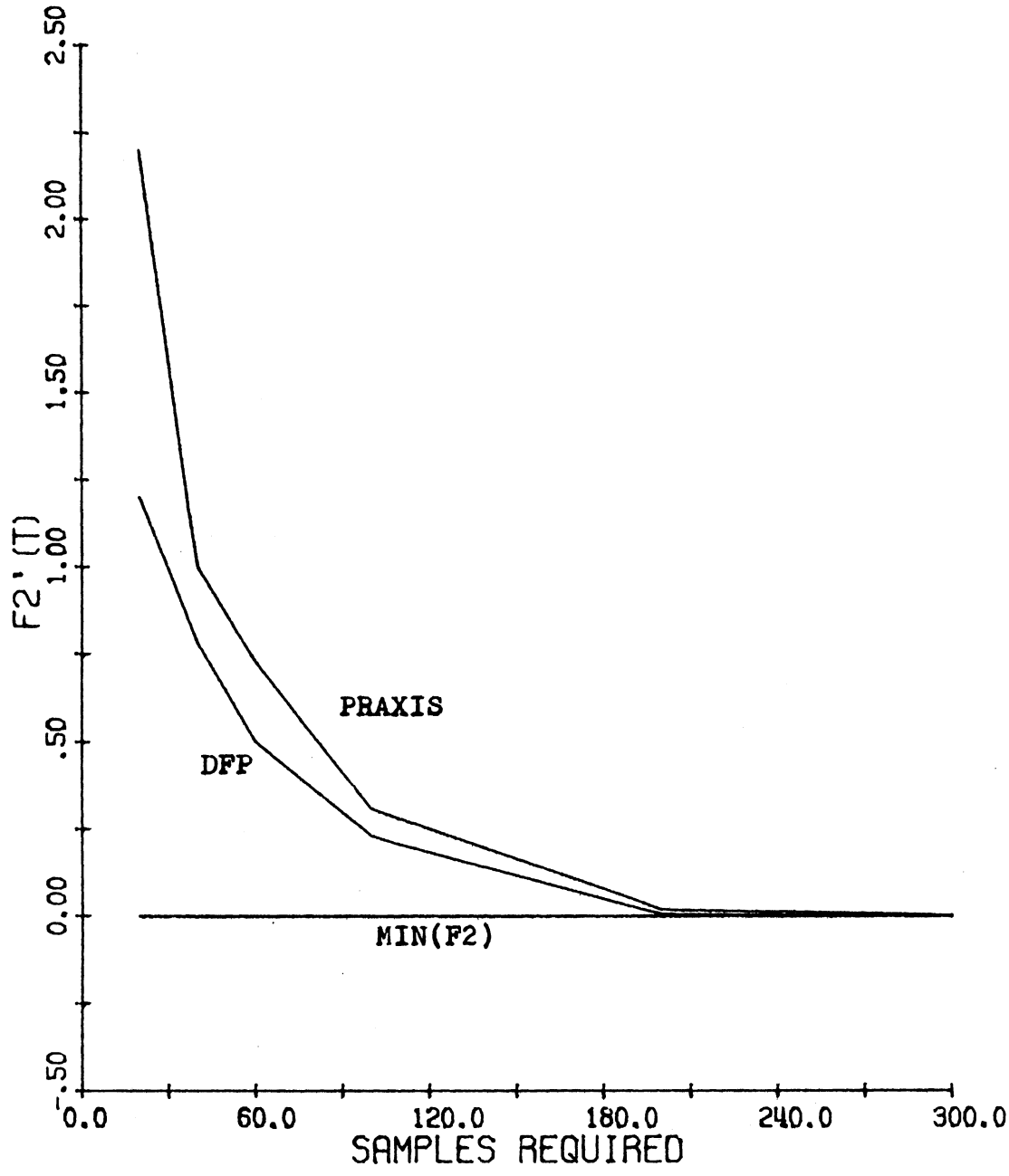


Figure 5.3: Off-line performance curves for PRAXIS and DFP in local mode on F2.

FIG. 5.4: ON-LINE PERFORMANCE ON F2

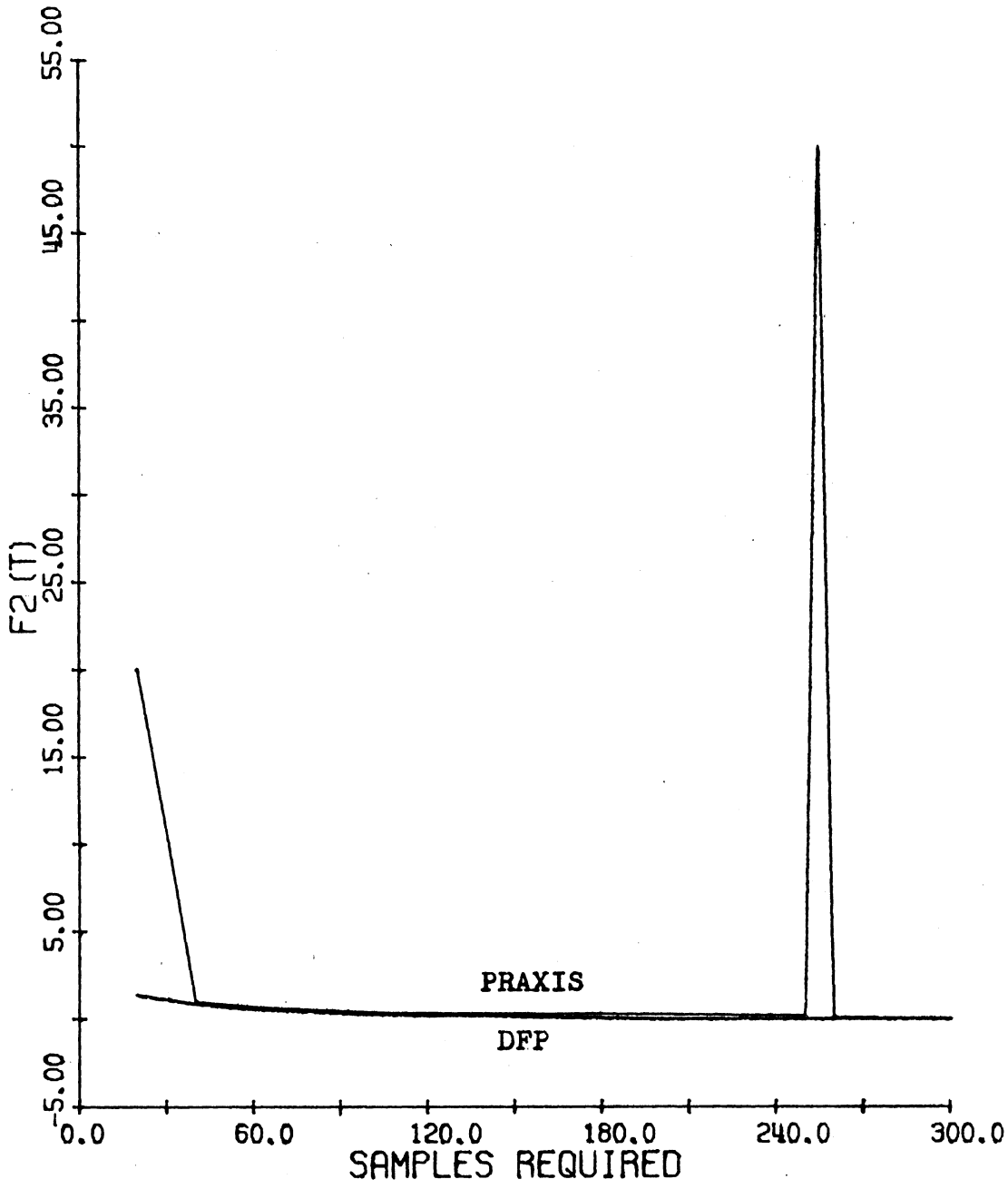


Figure 5.4: On-line performance curves for PRAXIS and DFP in local mode on F2.

FIG 5.5: OFF-LINE PERFORMANCE ON F3

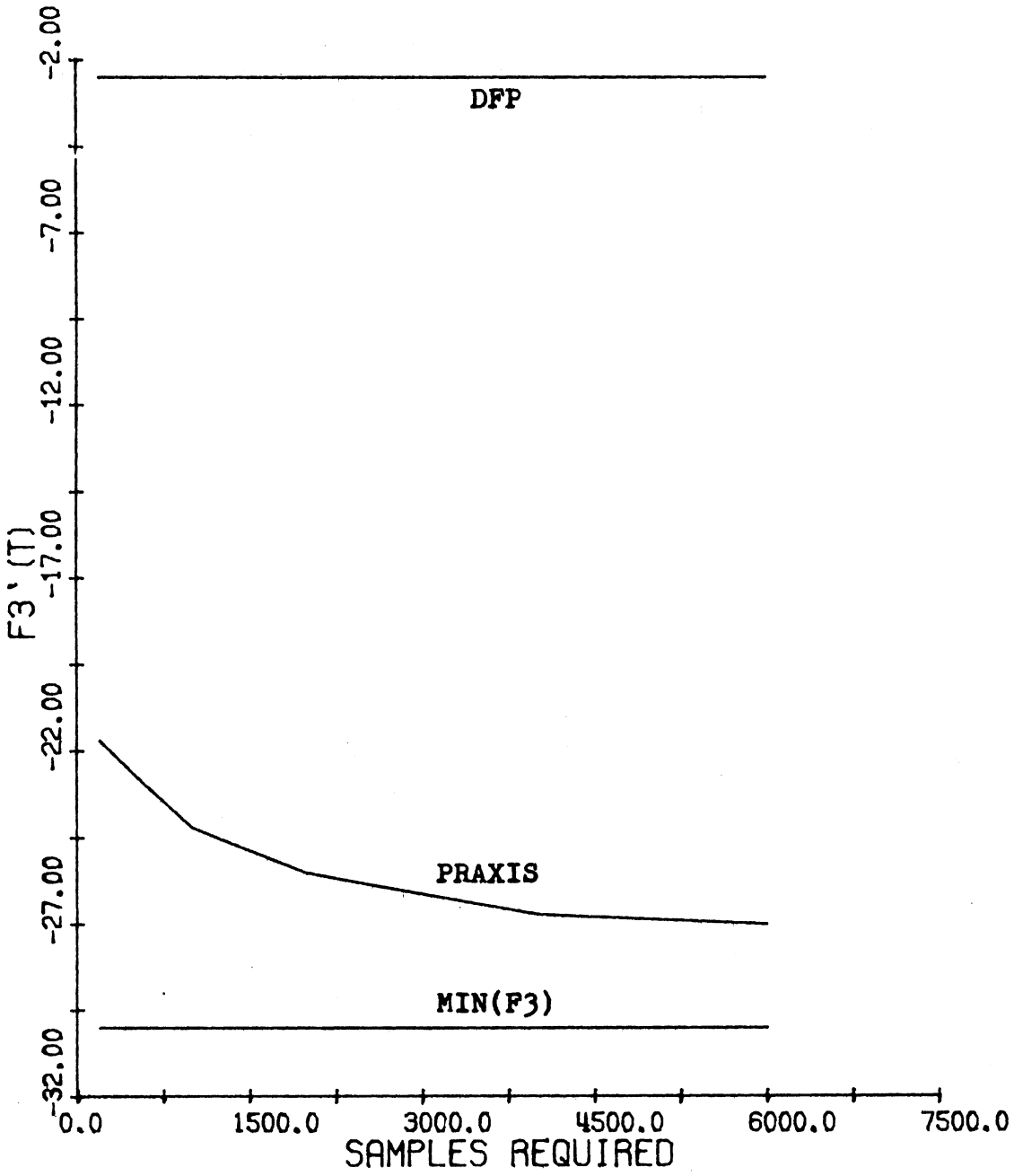


Figure 5.5: Off-line performance curves for PRAXIS and DFP in local mode on F3.

FIG. 5.6: ON-LINE PERFORMANCE ON F3

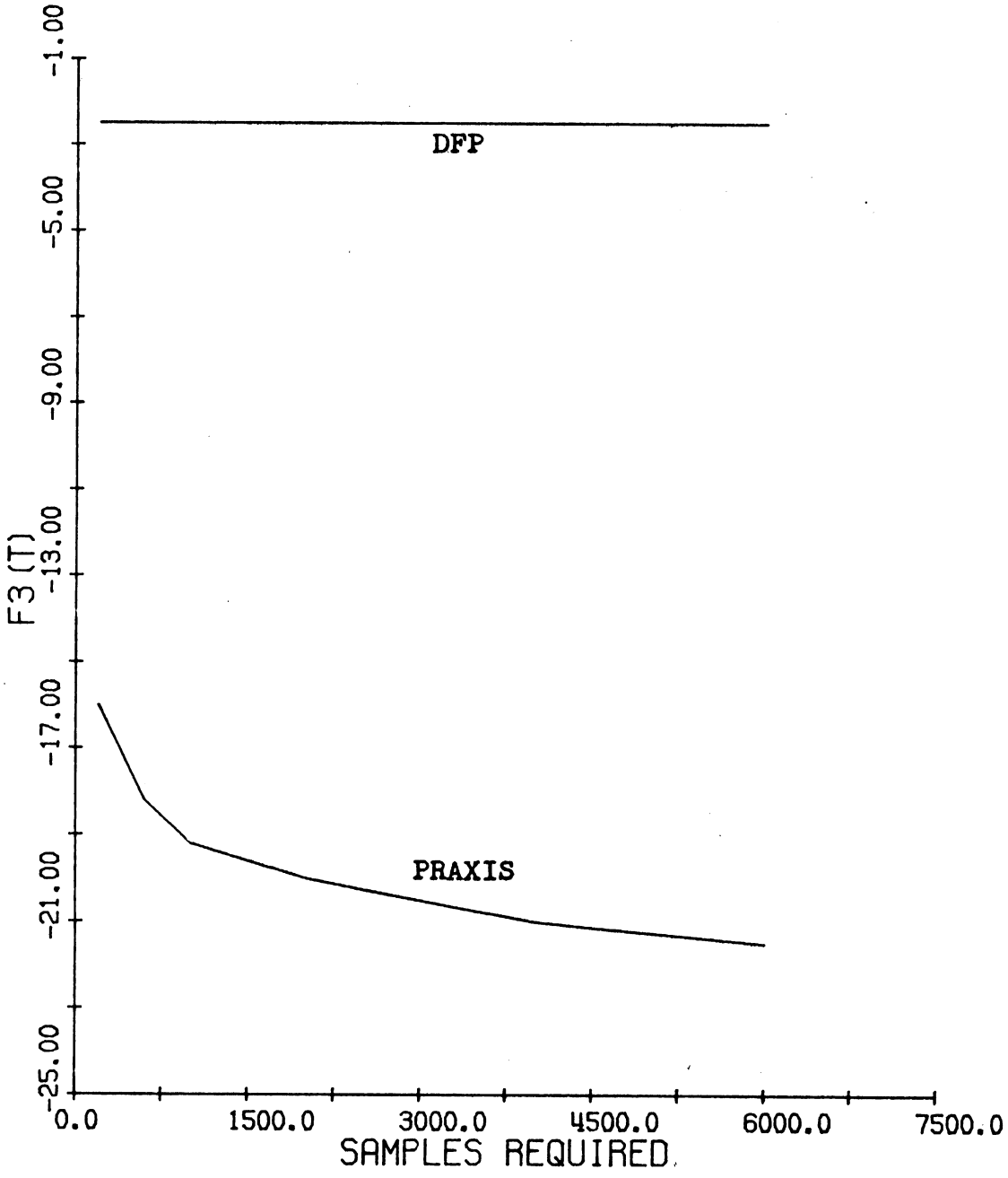


Figure 5.6: On-line performance curves for PRAXIS and DFP in local mode on F3.

curves generated on F4. Recall that F4 was a high dimensional quartic with Gaussian noise. Here we see a considerable difference in the performance of the algorithms. PRAXIS seemed to make no better progress than random search on F4, while DFP easily outperformed the genetic algorithms. This suggests that PRAXIS is considerably more sensitive to noise than DFP. To verify this, PRAXIS was evaluated on F4 with the Gaussian noise reduced from N(0,1) to N(0,.01). As illustrated, this resulted in considerable improvement in the performance of PRAXIS. It is interesting to speculate why PRAXIS is so much more sensitive to noise. Recall that PRAXIS constructs a conjugate direction without derivatives via a sequence of n one-dimensional minimizations. It is quite easy to imagine that this process is sensitive to noise, particularly with high-dimensional problems.

Finally, figures 5.9 and 5.10 illustrate the local performance curves generated on F5. The results are pretty much as expected. In local mode, both PRAXIS and DFP converge rapidly to the nearest local minimum, which when evaluated over a number of independent trials with random starting points yields convergence to the average value of F5 on its local minima.

At this point it is fairly easy to predict what will happen when we switch PRAXIS and DFP to global mode. On F1 and F2 there will be essentially no change in off-line performance since convergence is already

FIG. 5.7: OFF-LINE PERFORMANCE ON F4

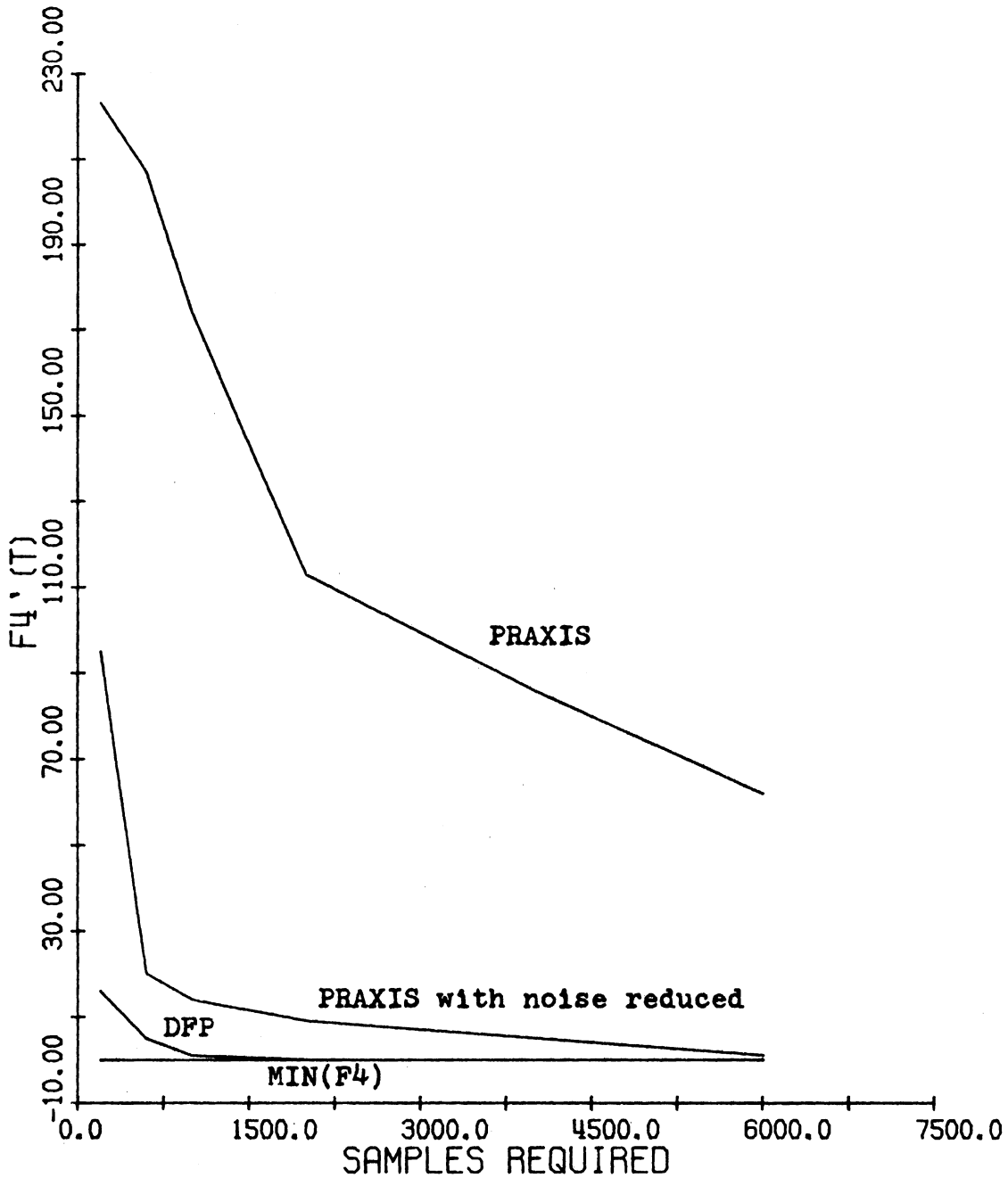


Figure 5.7: Off-line performance curves for PRAXIS and DFP in local mode on F4.

FIG. 5.8: ON-LINE PERFORMANCE ON F4

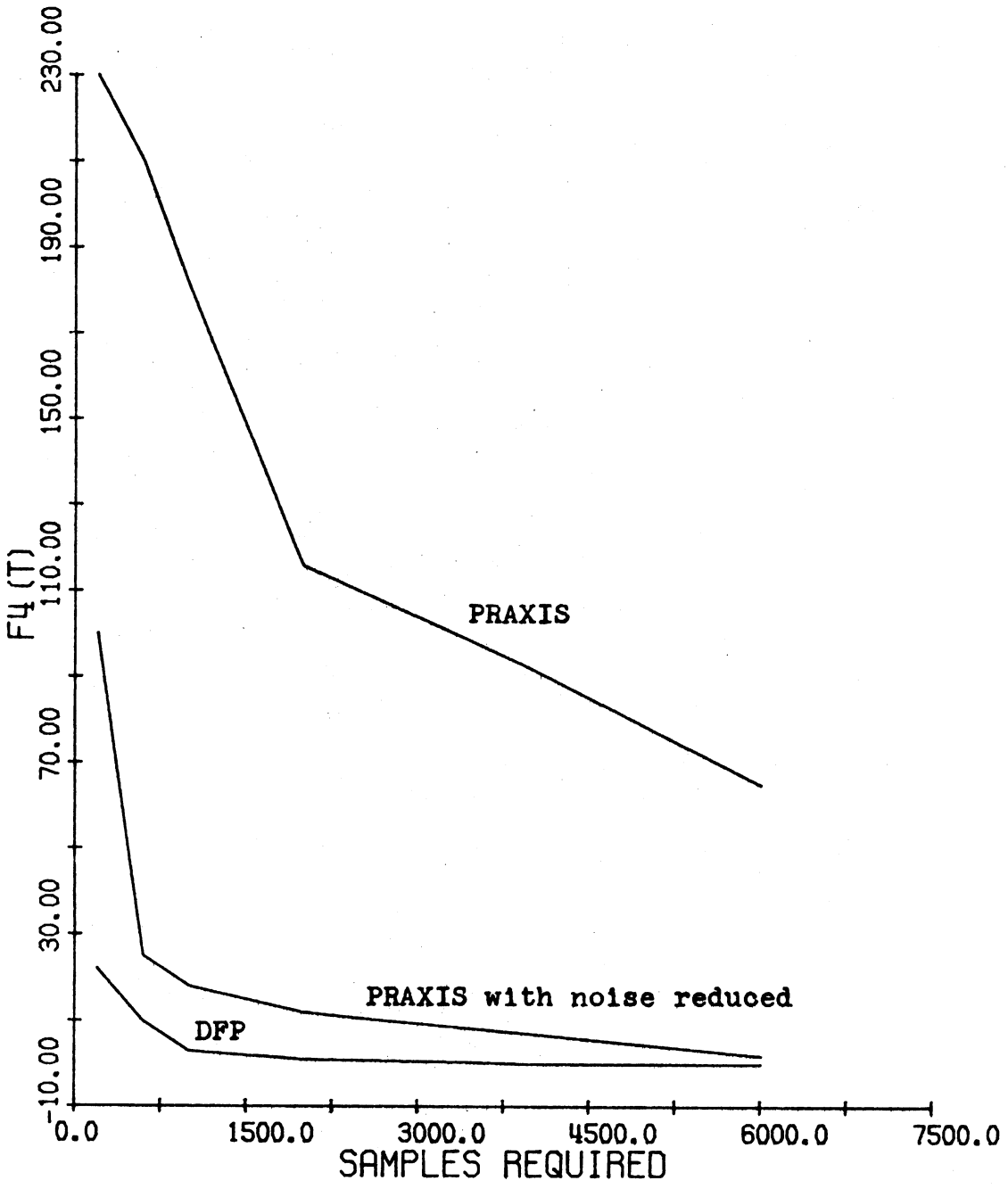


Figure 5.8: On-line performance curves for PRAXIS and DFP in local mode on F4.

FIG. 5.9: OFF-LINE PERFORMANCE ON F5

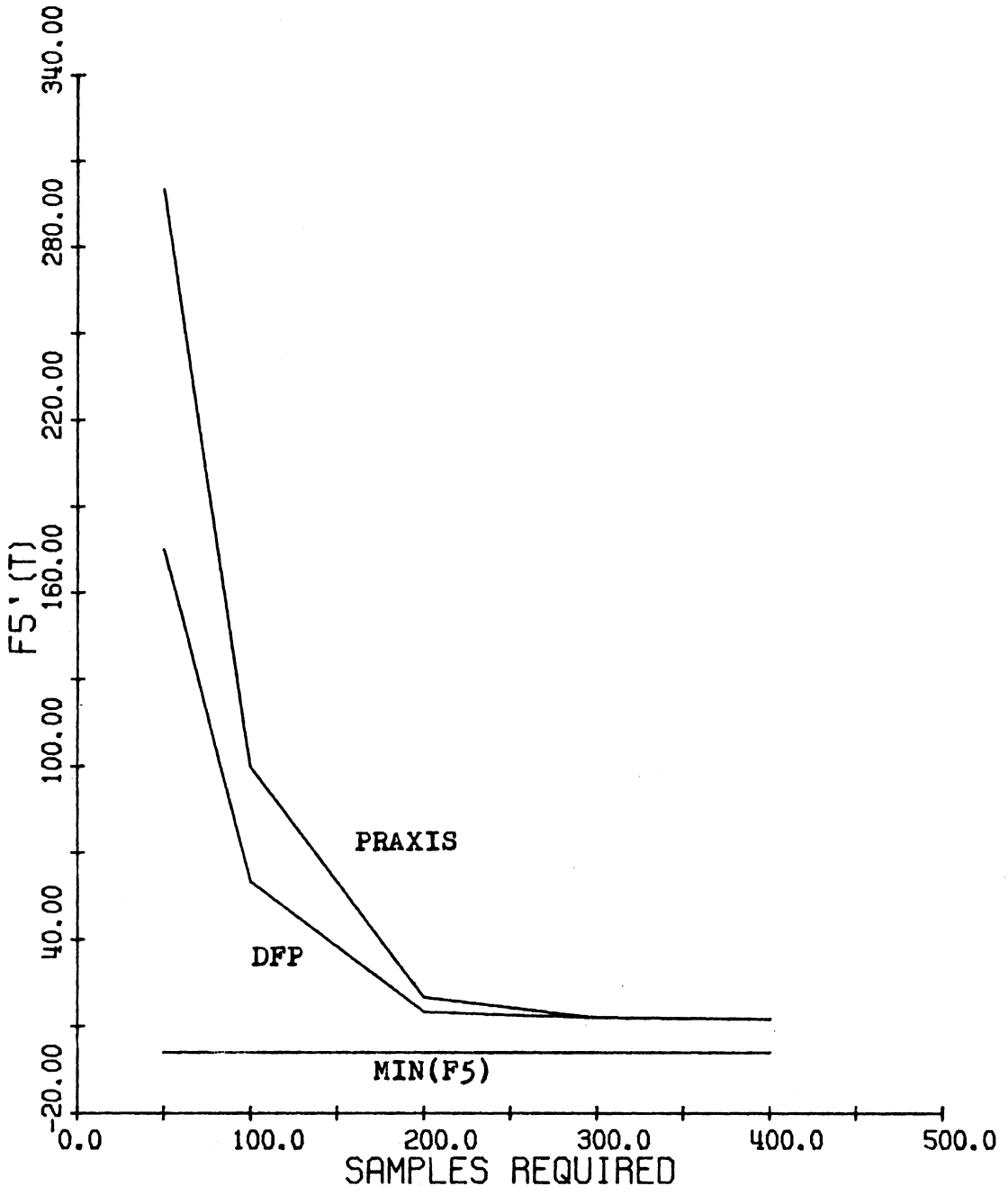


Figure 5.9: Off-line performance curves for PRAXIS and DFP in local mode on F5.

FIG. 5.10: ON-LINE PERFORMANCE ON F5

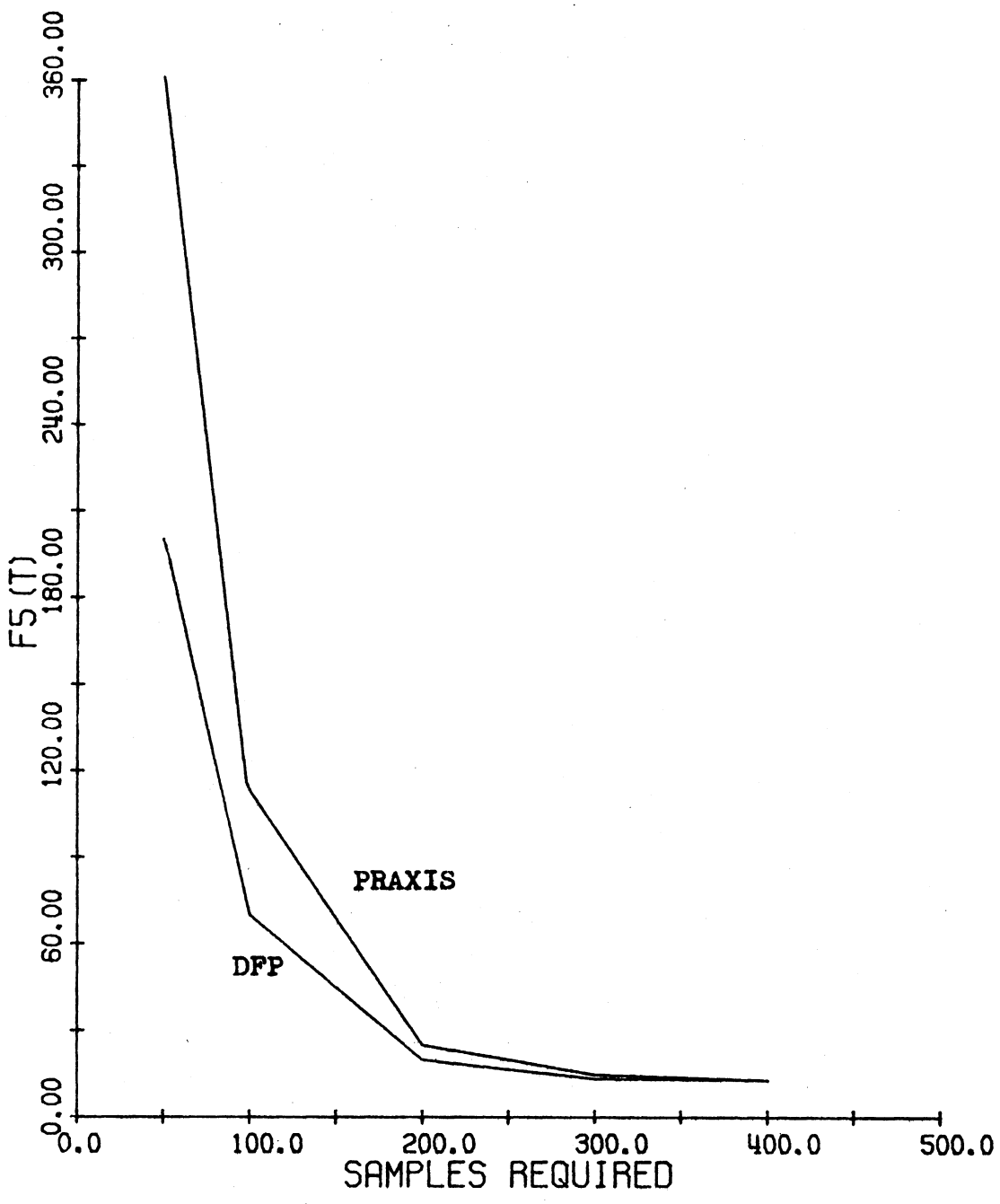


Figure 5.10: On-line performance curves for PRAXIS and DFP in local mode on F5.

achieved within 6000 trials. Notice, however, that restarting a local optimizer will have a definite effect on on-line performance, degrading it considerably. This is the same kind of tradoff between local and global search we observed with the genetic algorithms.

Switching to global mode on F3 will not affect the performance of PRAXIS at all since it did not converge in local mode within 6000 trials. Global mode will, however, improve the performance of DFP on F3, but, as illustrated by 5.11, it can do no better than random search since each restart is followed almost immediately by convergence. As a consequence both are outperformed, for example, by R4 on F3.

On F4, PRAXIS in global mode is unchanged since it did not converge in 6000 trials. Since DFP did converge, switching to global mode will leave off-line performance unchanged, and degrade on-line performance.

The interesting case is, of course, the effect of switching PRAXIS and DFP to global mode on the performance curves for F5. Since in local mode both PRAXIS and DFP converged to the nearest local minimum in about 300-400 trials, each gets restarted about 15-20 times in 6000 trials. Since each of the 25 local optima is about the same size, we would expect that on the average the global minimum will not be found in 6000 trials. Figure 5.12 illustrates that this is, in fact, true for both optimizers, and indicates that R4, for

FIG 5.11: OFF-LINE PERFORMANCE ON F3

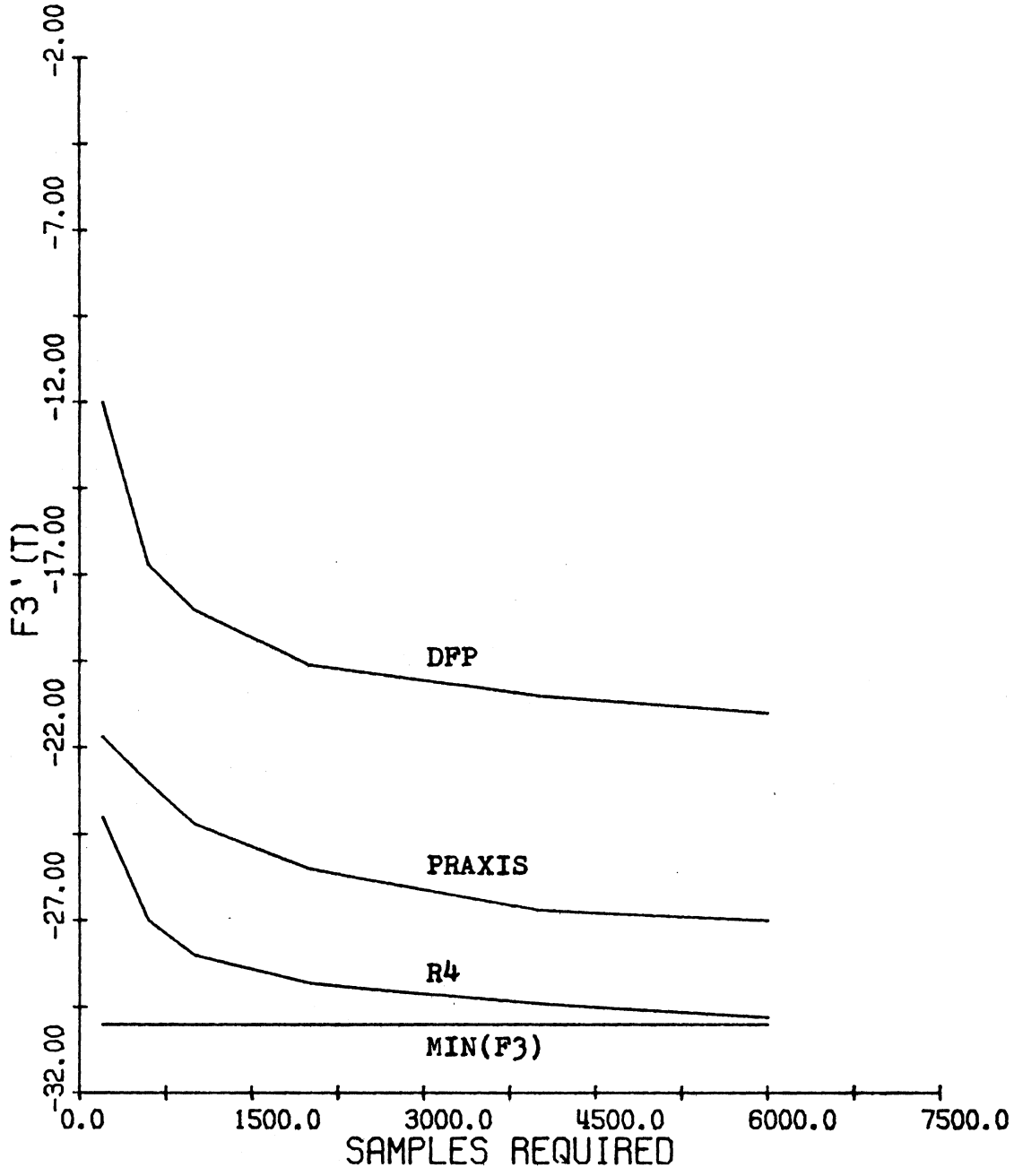


Figure 5.11: Off-line performance curves for PRAXIS and DFP in global mode on F3.

FIG. 5.12: OFF-LINE PERFORMANCE ON F5

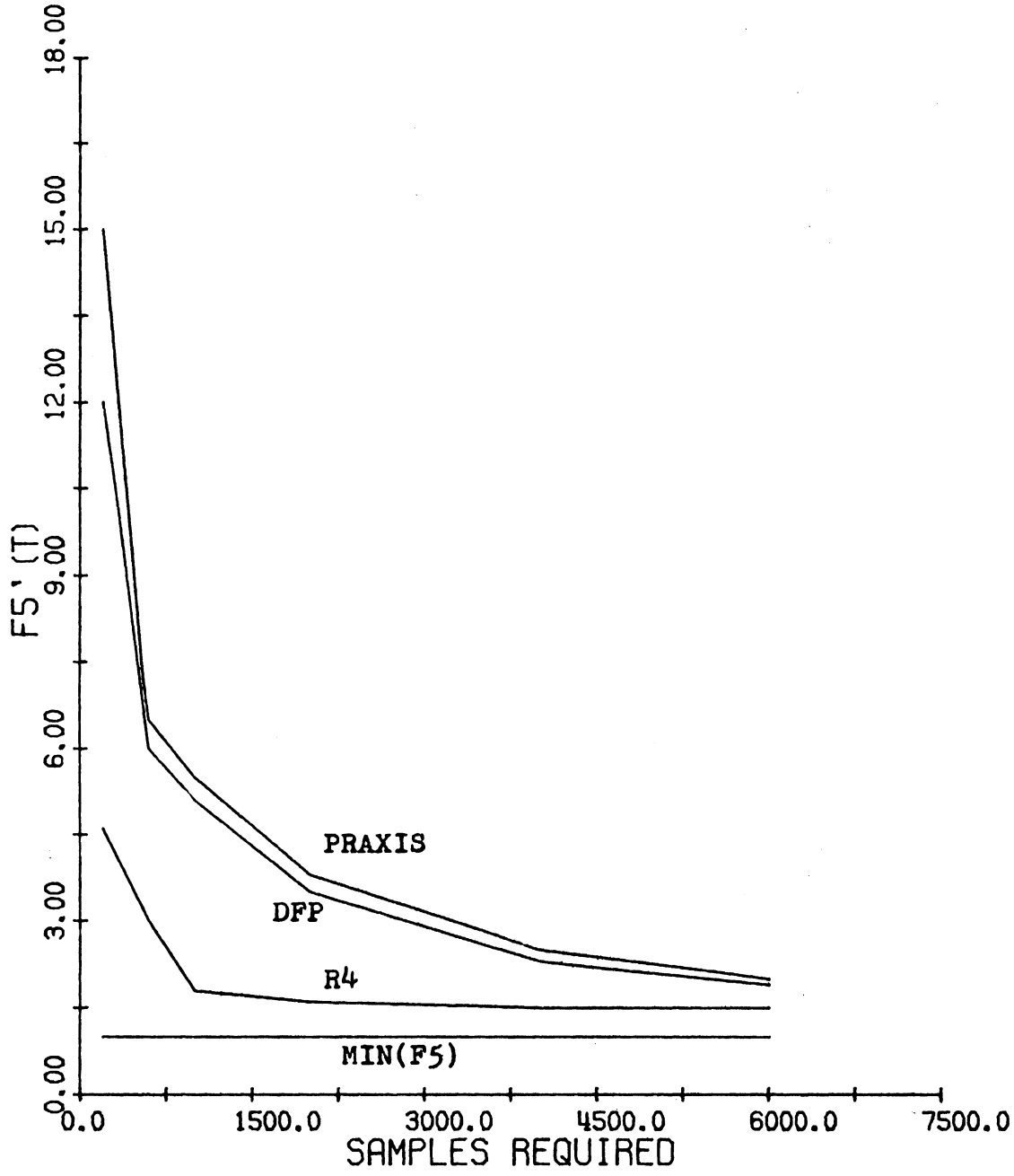


Figure 5.12: Off-line performance curves for PRAXIS and DFP in global mode on F5.

example, outperforms both on F5.

Finally, tables 5.1a and 5.1b give the off-line and on-line performance indices for PRAXIS and DFP evaluated over 6000 trials in both local and global mode. In general DFP outperformed PRAXIS on E, both in local and global mode. This is particularly true for on-line performance where the random search element in PRAXIS degraded performance considerably. Notice that both present a tradeoff between local and global mode. In local mode, rapid convergence to a possibly non-optimal minimum yields better on-line performance. On the other hand, restarting the optimizers increases the chances of finding the global optimum at the expense of on-line performance. In global mode, DFP came in a close second to R4 on off-line performance. DFP did better on F1, F2, and F4 while R4 was clearly the winner on F3 and F5. With respect to on-line performance, DFP outperformed R4 in local mode but not in global mode.

5.5 Summary

In this chapter we have attempted to provide a point of comparison for the behavior of genetic plans on E by evaluating the performance of two well-known function optimization techniques: a conjugate direction method and a variable metric method. Each were evaluated on E over 6000 trials in both local (normal) mode and a global mode which continued to restart the algorithms

T=6000	Local PRAXIS	Global PRAXIS	Local DFP	Global DFP	R4(50, .001,.6, 1.0)	Random Search
$x^*_{F1}(T)$	.003	.003	.002	.002	.114	.36
$x^*_{F2}(T)$	.051	.051	.003	.003	.221	.35
$x^*_{F3}(T)$	-25.47	-25.47	-2.5	-18.27	-28.2	-22.7
$x^*_{F4}(T)$	135.39	135.39	3.95	3.95	17.62	66.3
$x^*_{F5}(T)$	18.65	10.52	17.14	9.63	3.34	4.82
$X^*_E(T)$	25.72	24.1	3.72	-.937	-1.38	9.83

Table 5.1a: Off-line performance indices for PRAXIS and DFP on E.

T=6000	Local PRAXIS	Global PRAXIS	Local DFP	GLOBAL DFP	R4(50, .001,.6, 1.0)	Random Search
$x_{F1}(T)$	.005	6.55	.004	4.71	2.32	26.2
$x_{F2}(T)$	1348.6	2967.6	.042	11.57	34.76	494.05
$x_{F3}(T)$	-16.84	-16.84	-2.5	-2.5	-26.49	-2.5
$x_{F4}(T)$	149.57	149.57	5.62	54.62	40.73	249.6
$x_{F5}(T)$	21.98	203.17	18.57	187.56	34.34	473.3
$X_E(T)$	244.2	662.01	4.35	51.19	17.12	147.61

Table 5.1b: On-line performance indices for PRAXIS and DFP on E.

after convergence if 6000 trials had not been allocated. In general, we found that the variable metric method, DFP, outperformed the conjugate direction method. In fairness to PRAXIS, however, we must remember that DFP was given exact derivative information about the test functions in E upon request. If the derivatives had been estimated or if a cost of more than n function evaluations had been exacted for each derivative computation, the differences in performance would have been less. During the evaluation, two interesting facts about PRAXIS were uncovered. In the first place, its heuristic strategies for avoiding stagnation on resolution ridges exacted a heavy toll when measuring on-line performance. This, of course, reflects the emphasis of the function optimization literature on convergence. Secondly, PRAXIS was seen to be quite sensitive to noise, performing no better on F4 than random search. This suggests that the matrix updating techniques used by DFP to generate the conjugate directions of search are considerably less sensitive to noise than the constructive techniques used by PRAXIS, particularly in high-dimensional spaces.

Finally, we saw that DFP performed about as well as the genetic plans on E, but the tradeoffs were sharply drawn. On functions F1, F2, and F4 which approximate the assumptions made by DFP about the function to be minimized, DFP was clearly the better choice. The

situation is, however, exactly reversed on F3 and F5 with R4 the obvious choice. These observations suggest that the genetic plans hold a valid position in both the fields of adaptation and function optimization, filling the gap between the efficient local search techniques and inefficient random search.

Chapter 6

SUMMARY AND CONCLUSIONS

We began this thesis by introducing a formalism for the study of adaptive systems and, within this framework, we defined a means of evaluating the performance of adaptive systems. The central feature of the evaluation process was the concept of robustness: the ability of an adaptive system to rapidly respond to its environment over a broad range of situations. To provide a concrete measure of robustness, a family E of environment response surfaces was carefully chosen to include representatives of a wide variety of response surfaces. When evaluating an adaptive system on a member of E, two distinct performance curves were monitored during adaptation: on-line performance and off-line performance. On-line performance evaluated every trial produced during adaptation, reflecting those situations in which an adaptive system is used to dynamically alter the performance of a system. Off-line performance evaluated only trials produced during adaptation which resulted in improved performance, reflecting situations in which testing can be done independently of the system being controlled.

Within this evaluation framework, a class of genetic adaptive systems was introduced for analysis and evaluation. These artificial genetic systems, called reproductive plans, generate adaptive responses by simulating

the information processing achieved in natural systems via the mechanisms of heredity and evolution. By introducing the concept of hyperplane partitions of the representation space, it was shown that reproductive plans have good theoretical properties with respect to the optimal allocation of trials to competing hyperplane partition elements. The performance of an elementary member, R1, of this class of genetic plans was evaluated on E and was shown to be superior to pure random search both in on-line and off-line performance. However, as defined, R1 was seen to converge quite regularly to a non-optimal plateau. Analysis suggested that this was due in part to the stochastic side-effects of random samples on a finite population. The effects of varying four parameters in the definition of R1 were analyzed in the hope of removing the problem of premature convergence and improving the performance of R1 on E. Increasing the population size was shown to reduce the stochastic effects and improve long-term performance at the expense of slower initial response. Increasing the mutation rate was seen to improve off-line performance at the expense of on-line performance. Reducing the crossover rate resulted in an overall improvement in performance, suggesting that producing a generation of completely new individuals was too high a sampling rate. Finally, reducing the generation gap was shown to yield the same kind of overall improvement in performance as crossover, but not as

dramatic.

As an alternative to modifying the parameters of R1, several modifications to the basic algorithm were analyzed for their effects on premature convergence and performance. An elitist policy favoring hyperplanes which produced the best individuals was shown to improve local search performance at the expense of global search. Modifying the sampling technique so that the actual number of offspring of an individual more closely approximated the expected value produced the best overall improvement in performance. Combining the expected value model with the elitist policy generated the best performance of any of the genetic plans analyzed on E, although the problem of premature convergence on F5 still remained. Both increasing the mutation rate and including a crowding factor were shown to resolve the problem of local convergence on R5, but at the expense of performance on the unimodal surfaces. Finally, a brief analysis of a generalized crossover model produced no significant improvement in the performance of genetic plans on E.

To provide an alternate point of comparison for performance on E (in addition to random search), two standard function optimization techniques were also evaluated on E. Each was run in their normal (local search) mode as well as a global mode in which, after local convergence, they were restarted at random starting points in an attempt to find the global minimum. Both

techniques outperformed the genetic algorithms on those surfaces in E for which they were designed. However, the genetic algorithms were seen to be superior on the discontinuous and multimodal surfaces in E.

As a consequence of these studies, several points of interest have been brought out. In the first place, it seems fairly clear that it is difficult, if not impossible, to simultaneously provide high-level off-line and on-line performance. Fortunately, most applications demand one, but not both. But it would be nice to provide a single solution to both. Off-line performance emphasizes convergence while on-line performance stresses short-term performance. As we have seen, better convergence properties are obtained by bold exploration in the early stages of adaptation, while short term performance is improved by the more conservative sampling policies. These distinctions are sharpened by the fact that in any practical situation evaluation is performed over a relatively short time interval. The longer the evaluation period, the less important short-term performance becomes.

A second point of interest brought out is the disparity between the mathematical characteristics of genetic plans and their implementation behavior. As we have seen, the fact that genetic plans support a finite population over a finite period of time can lead to considerably lower performance levels than predicted math-

ematically. Because the genetic plans operate in terms of a sequence of trial allocation decisions based on finite sampling distributions and sample means, they were seen to be quite sensitive to the associated stochastic errors. By minimizing as much as possible these stochastic side-effects, the implementation performance of genetic plans was improved considerably.

A third point of interest was brought out in the comparative analysis of function optimization techniques. As with many other difficult problems in computer science, adaptation poses the tradeoff between a general solution to the problem and performance. By making assumptions about the kind of response surfaces to be faced, extremely efficient performance can be generated on a relatively small class of functions. It is clear that no genetic algorithm is going to minimize a convex quadratic function in 50 trials. On the other hand, it is clear that if such assumptions are not met, then genetic algorithms provide a considerably better alternative than random search without making any assumptions about the form of the response surface.

These studies also suggest several interesting areas for future research. Neither time nor resources permitted extensive evaluation of the generalized crossover operator. Further study may support the intuition that performance improvements could be achieved for small increases in the number of crossover points.

Another area worth exploring is to consider the effects of a diploid representation. That is, each gene position has two alleles with a "dominance" map specifying its functional value. Dominance has a very direct bearing on the problem of allele loss in finite populations.

Finally, it would be of interest to explore the possibility of introducing "species" into the genetic algorithm. This also has direct bearing on the problem of allele loss allowing, for example, exponential exploitation of a local minimum without the problem of complete population dominance.

Appendix A

THE ENVIRONMENT E

A.1 Introduction

In order to evaluate the capabilities of different adaptive strategies, an environment E was chosen to include instances of performance measures representing a broad class of functions. For convenience, each function in E was defined to be a performance index to be minimized. Care was taken to include instances of continuous, discontinuous, convex, non-convex, unimodal, multimodal, quadratic, non-quadratic, low-dimensional and high-dimensional functions as well as functions with Gaussian noise.

For testing purposes, each function was restricted to a bounded subspace of R^n of the form $a_i \leq x_i \leq b_i$, $i=1,\ldots,n$. Within this subspace each function was discretized by specifying a resolution factor Δx_i for each axis. Since the genetic algorithms use a binary representation of the search space, the number of discrete points on each axis, $(b_i-a_i)/\Delta x_i + 1$, was chosen to be a power of 2 so that a direct comparison with alternative adaptive plans could be made.

A.2 Test Function F1

Test function F1 is given by:

$$F1(X) = \sum_{i=1}^{3} x_i^2$$

F1 is a simple 3-dimensional parabola with spherical constant-cost contours. It is a continuous, convex, unimodal, low-dimensional quadratic function with a minimum of zero at the origin. Because of its simplicity and symmetry, F1 provides an easily analyzable first test for an adaptive plan. For testing purposes F1 was restricted to the space A defined by $-5.12 \leq x_i \leq 5.12$, $i=1,2,3$ with a resolution factor $\Delta x_i = .01$ on each axis. So the space A to be searched consisted of $(1024)^3 \cong 10^9$ alternative solutions on which:

$$\text{MAX(F1)} = F(\pm 5.12, \pm 5.12, \pm 5.12) = 78.6$$

$$\text{MIN(F1)} = F(0,0,0) = 0$$

$$\text{AVE(F1)} = \frac{1}{(10.24)^3} \int_A F1(X)\, dX = 26.2$$

Figures A.1a and A.1b illustrate the surface defined by F1 in its two-dimensional form.

A.3 Test Function F2

Test function F2 is given by:

$$F2(X) = 100*(x_1^2 - x_2)^2 + (1-x_1)^2$$

F2 is a standard test function in the optimization literature, first proposed by Rosenbrock(1960). It is a continuous, non-convex, unimodal, low-dimensional quartic function with a minimum of zero at (1,1). It is a difficult minimization problem because it has a deep

FIG. A.1A: 2-DIMENSIONAL VERSION OF F1

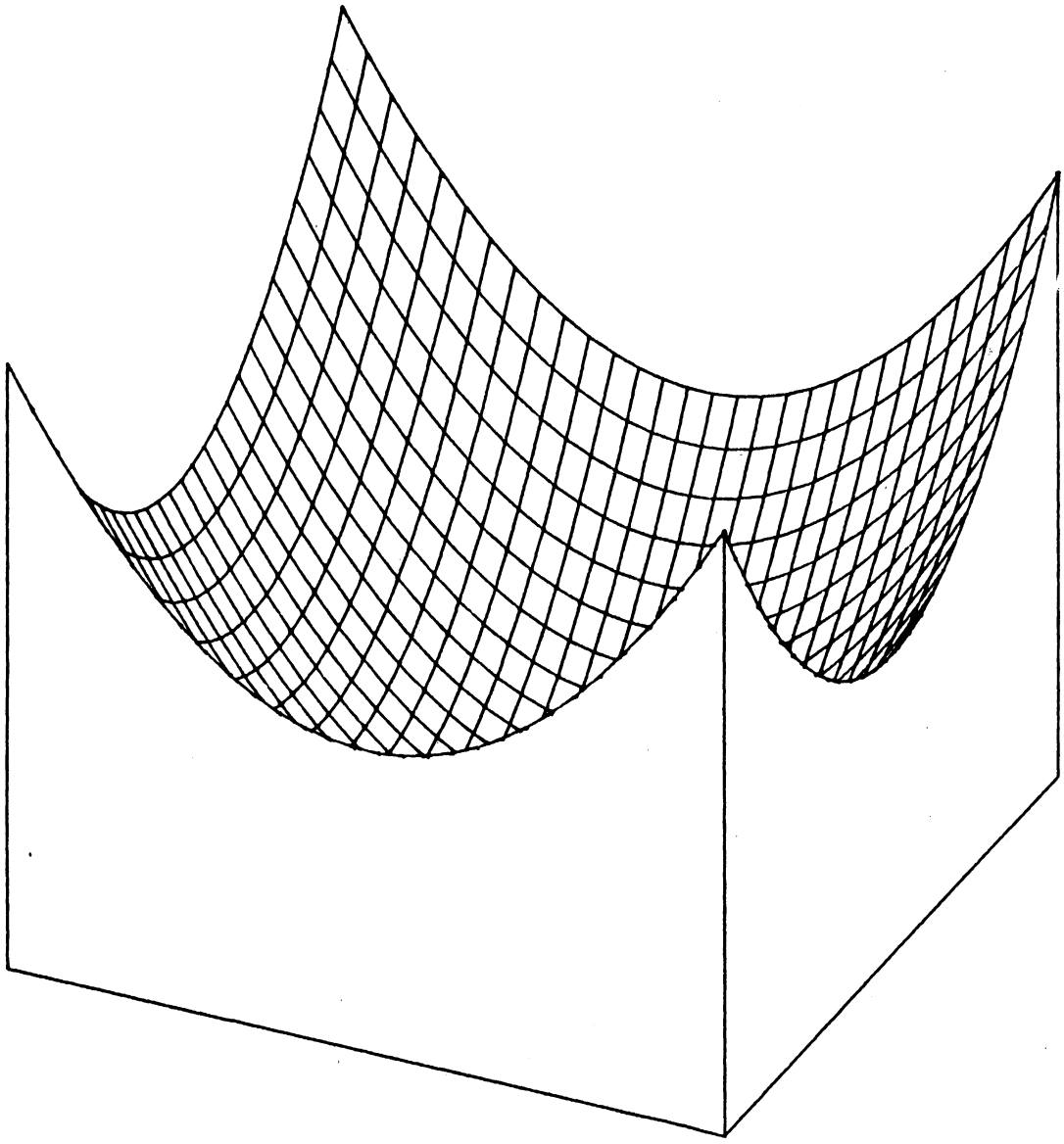


Figure A.1a: Top surface defined by the 2-dimensional version of F1.

FIG. A.1B: INVERTED 2-DIMENSIONAL VERSION OF F1

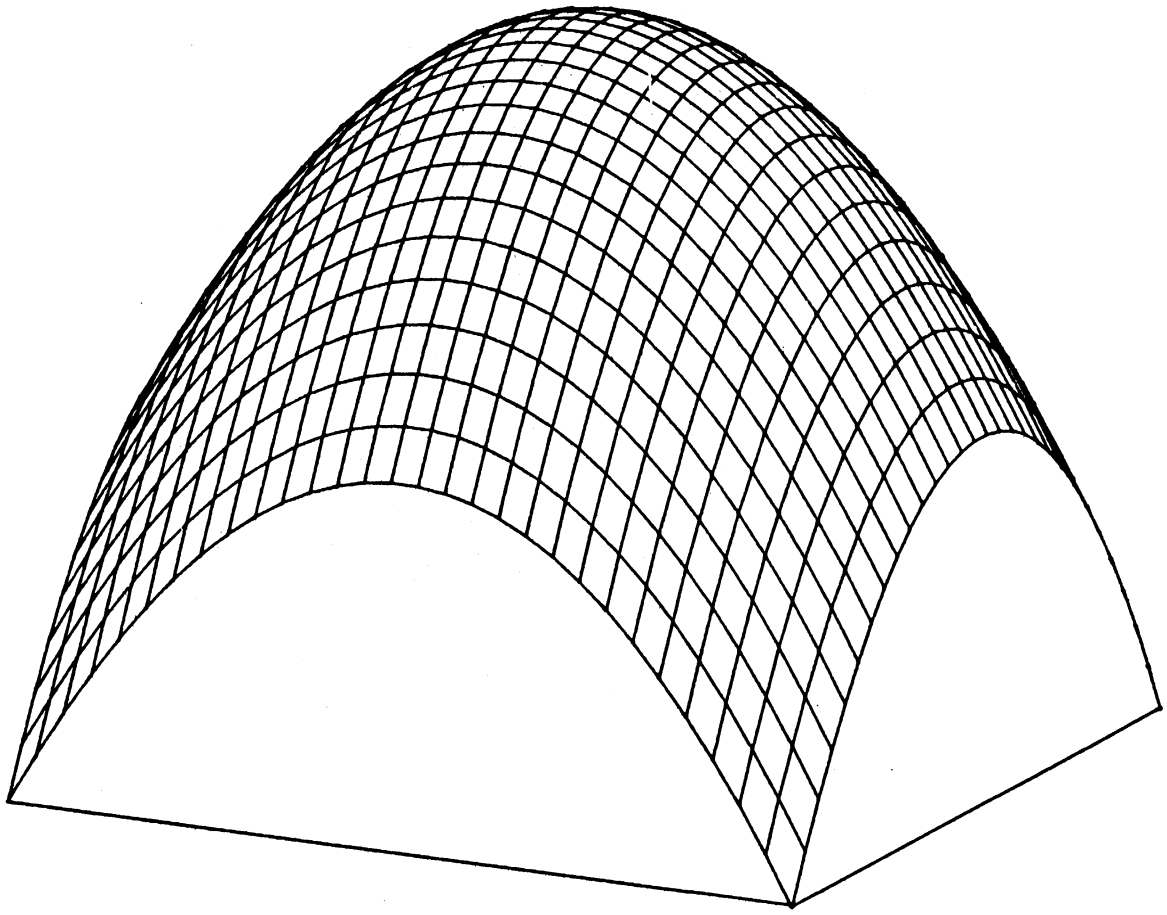


Figure A.1b: Bottom surface defined by the 2-dimensional version of F1.

parabolic valley along the curve $x_2 = x_1^2$. For testing purposes, F2 was restricted to the space A defined by $-2.048 \leq x_i \leq 2.048$, i=1,2 with a resolution factor of $\Delta x_i = .001$ along each axis. So the space A to be searched consisted of $(4096)^2 \cong 1.7*10^6$ alternative solutions on which:

$$\text{MAX(F2)} = F(-2.048,-2.048) = 3905.93$$

$$\text{MIN(F2)} = F(1,1) = 0$$

$$\text{AVE(F2)} = \frac{1}{(4.096)^2} \int_A F2(X)\, dX = 494.05$$

Figures A.2a and A.2b illustrate the surface defined by F2.

A.4 Test Function F3

Test function F3 is given by:

$$F3(X) = \sum_{i=1}^{5} [x_i]$$

where $[x_i]$ represents the greatest integer less than or equal to x_i. Hence, F3 is a 5-dimensional step function. It is a discontinuous, non-convex, unimodal function of moderate dimension which is piece-wise constant. F3 was chosen as a test for handling discontinuities. For testing purposes, F3 was restricted to the space A defined by $-5.12 \leq x_i \leq 5.12$ with a resolution factor of $\Delta x_i = .01$ on each axis. So the space A to be searched consisted of $(1024)^5 \cong 10^{15}$ alternative solutions on which:

FIG. A.2A: F2 (ROSENBROCK'S FUNCTION)

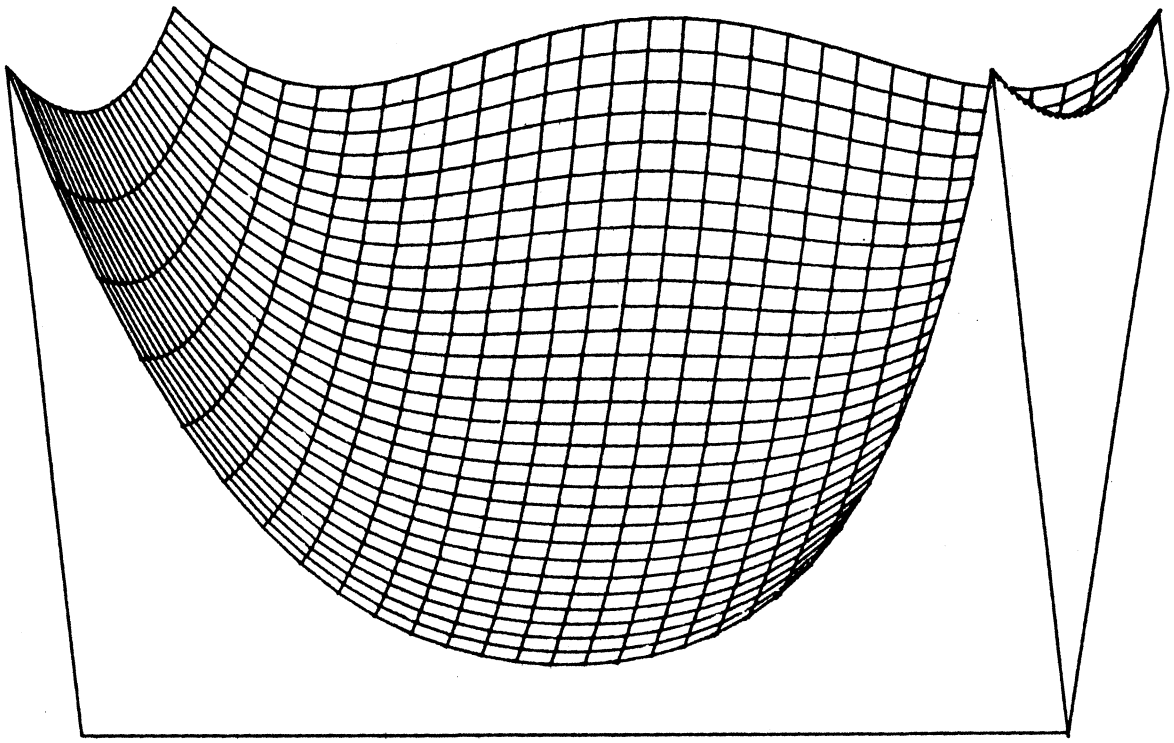


Figure A.2a: Top surface defined by test function F2.

FIG. A.2B: INVERTED F2 (ROSENBROCK'S FUNCTION)

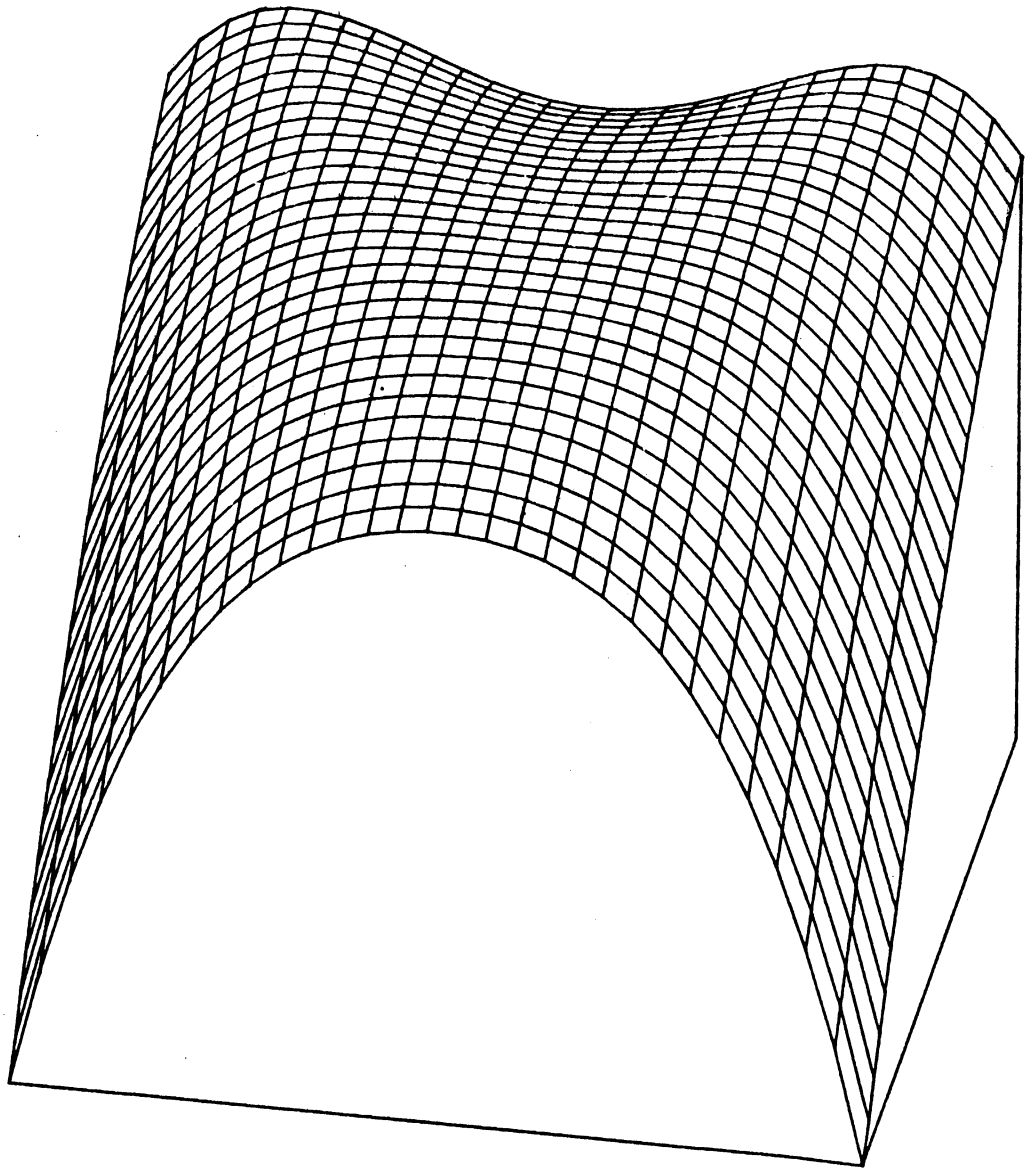


Figure A.2b: Bottom surface defined by test function F2.

$$\text{MAX(F3)} = \text{F3}(5.12,5.12,5.12,5.12,5.12) = 25$$

$$\text{MIN(F3)} = \text{F3}(-5.12,-5.12,-5.12,-5.12,-5.12) = -30$$

$$\text{AVE(F3)} = \sum_{i=1}^{5} \text{AVE}\left[x_i\right] = -2.5$$

Figure A.3 illustrates the surface defined by F3 in its two-dimensional form.

A.5 Test Function F4

Test function F4 is given by:

$$\text{F4(X)} = \sum_{i=1}^{30} i x_i^4 + \text{GAUSS}(0,1)$$

F4 is a continuous, convex, unimodal, high-dimensional quartic function with Gaussian noise. For testing purposes, F4 was restricted to the space A defined by $-1.28 \leq x_i \leq 1.28$, $i=1,\ldots,30$ with a resolution factor of $\Delta x_i = .01$ on each axis. So the space A to be searched consisted of $(256)^{30} \cong 10^{72}$ alternative solutions on which:

$$\text{MAX(F4)} = \text{F4}(\pm 1.28, \pm 1.28, \ldots, \pm 1.28) = 1248.2$$

$$\text{MIN(F4)} = \text{F4}(0,0,\ldots,0) = 0$$

$$\text{AVE(F4)} = 249.6$$

Figures A.4a and A.4b illustrate the surface defined by F4 in its two-dimensional form without Gaussian noise.

A.6 Test Function F5

Test function F5 is given by:

FIG. A.3: 2-DIMENSIONAL VERSION OF F3

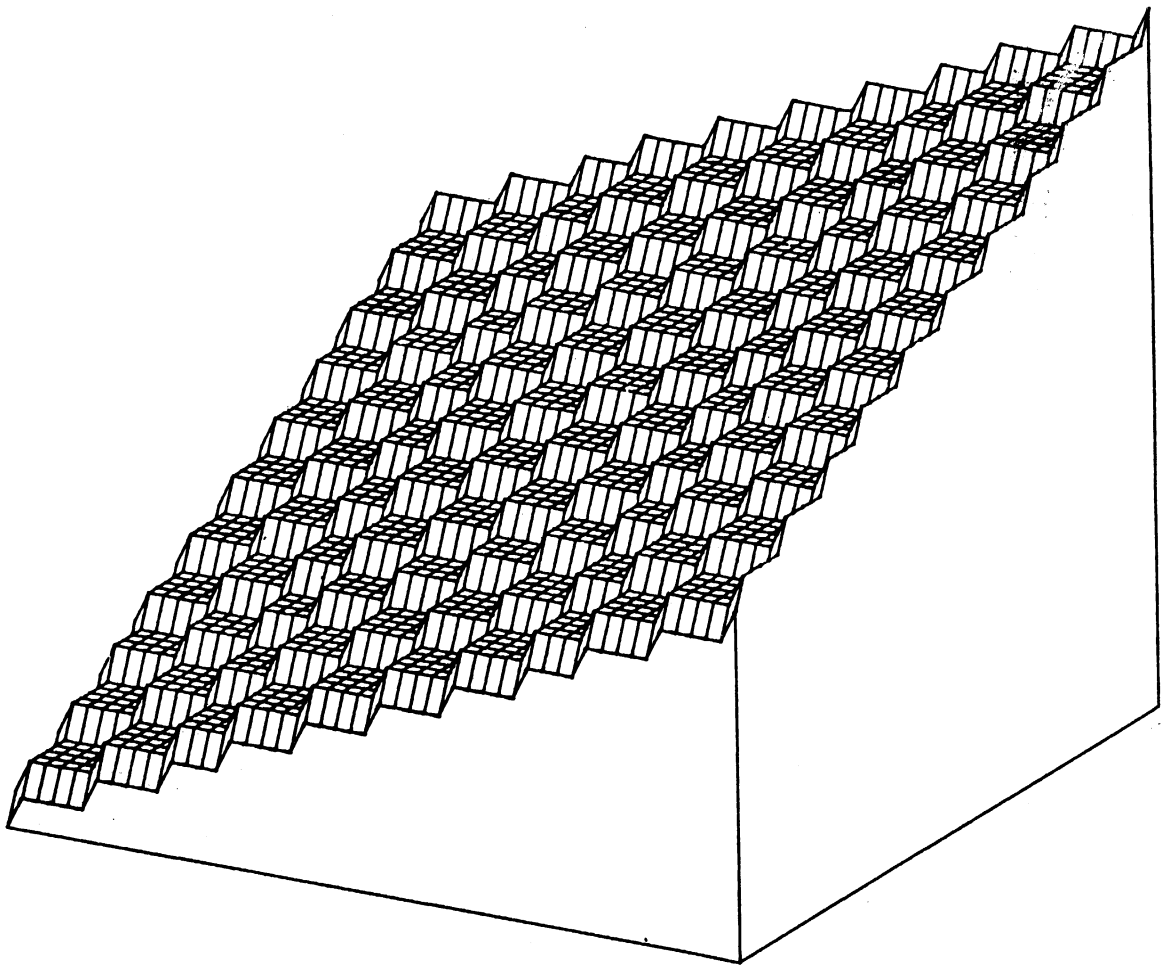


Figure A.3: Surface defined by the 2-dimensional version of test function F3.

FIG. A.4A: 2-DIMENSIONAL VERSION OF F4

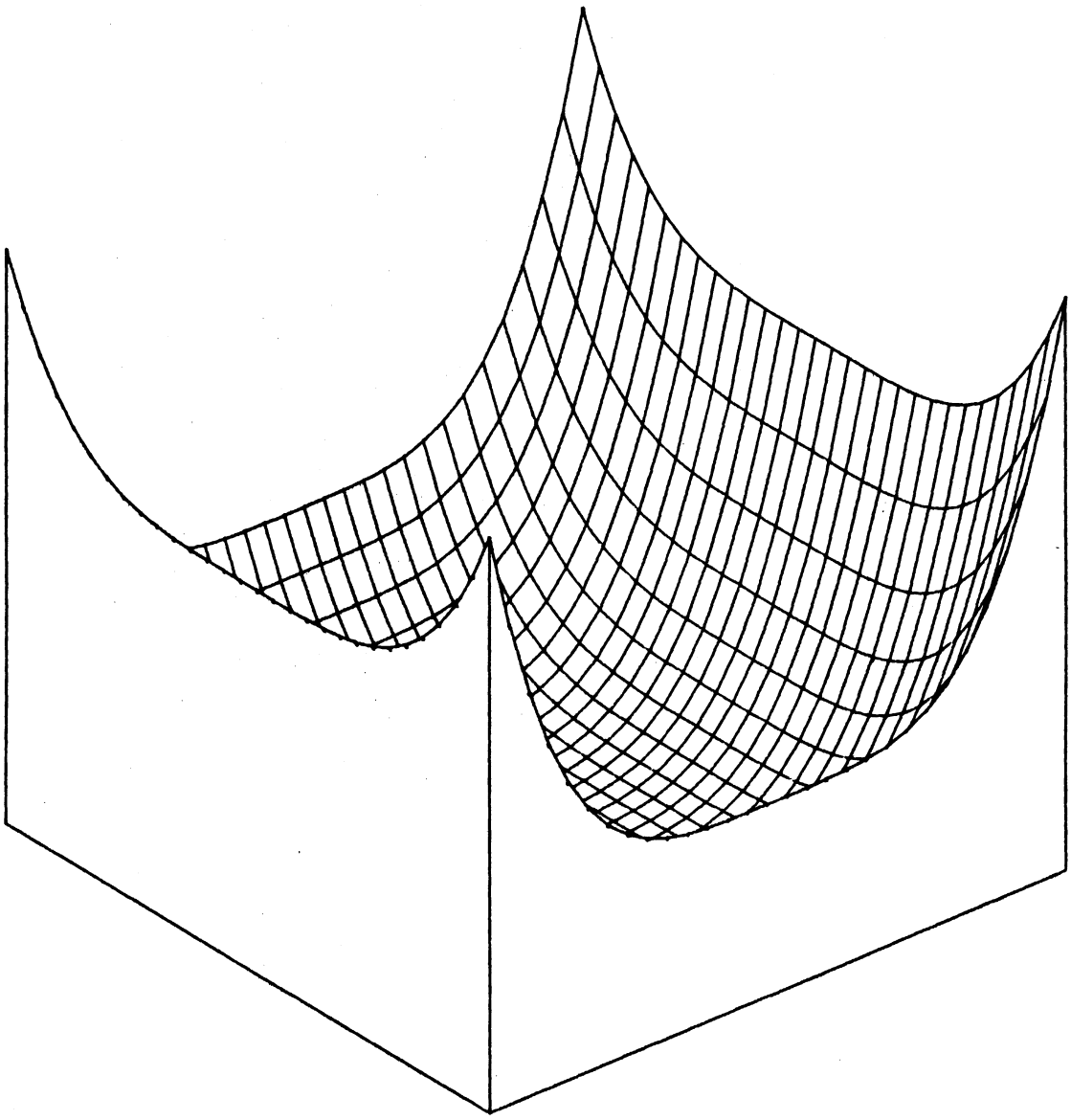


Figure A.4a: Top surface defined by the 2-dimensional version of test function F4.

FIG. A.4B: INVERTED 2-DIMENSIONAL VERSION OF F4

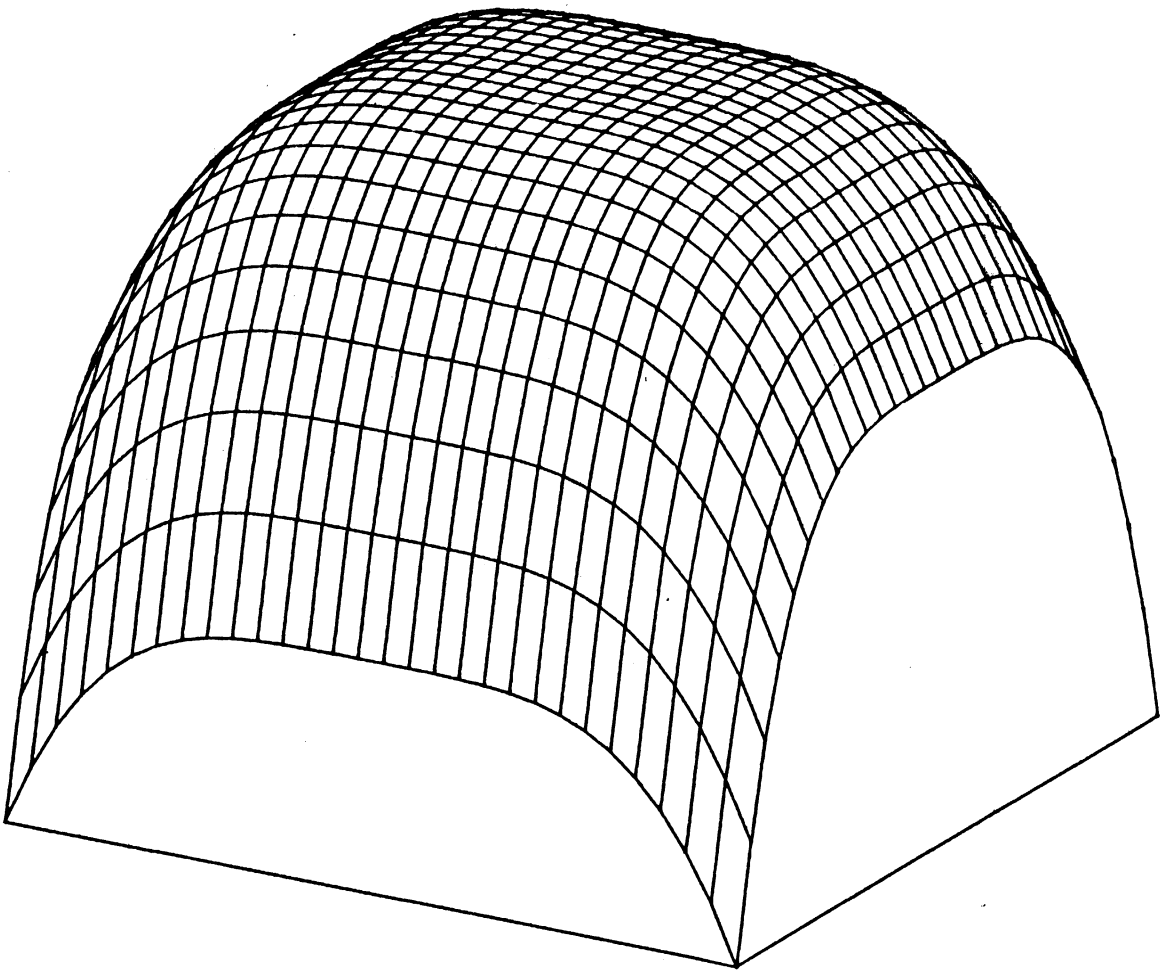


Figure A.4b: Bottom surface defined by the 2-dimensional version of test function F4.

$$\frac{1}{F5(X)} = \frac{1}{K} + \sum_{j=1}^{25} \frac{1}{f_j(X)}$$

where

$$f_j(X) = c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6$$

F5 is an interesting multimodal function synthesized as suggested by Shekel (1971). It is a continuous, non-convex, non-quadratic, two-dimensional function with 25 local minima approximately at the points $\{(a_{1j},a_{2j})\}_{j=1}^{25}$. The function value at the point (a_{1j},a_{2j}) is approximately c_j.

For testing purposes, the a_{ij} were defined by:

$$\left[a_{ij}\right] = \begin{bmatrix} -32,-16, & 0, & 16, & 32,-32,-16,\ldots, & 0,16,32 \\ -32,-32,-32,-32,-32,-16,-16,\ldots,32,32,32 \end{bmatrix}$$

with $c_j = j$ and $K = 500$. F5 was restricted to the space A defined by $-65.536 \leq x_i \leq 65.536$, $i=1,2$ with a resolution factor of $\Delta x_i = .001$ on each axis. So the space A to be searched consisted of $(131,072)^2 \cong 16*10^9$ alternative solutions on which:

$$MAX(F5) \cong 500$$

$$MIN(F5) \cong 1$$

$$AVE(F5) \cong 473$$

Figures A.5a and A.5b illustrate the surface defined by F5. It is essentially a flat surface F5(X) = 500 with 25 deep perforations centered about the points (a_{1j},a_{2j}).

Near the point (a_{1j},a_{2j}), F5 is almost completely dominated by the term $f_j(X)$, i.e.

$$F5(X) = c_j + \sum_{i=1}^{2} (x_i-a_{ij})^6$$

and hence $F5(a_{1j},a_{2j}) \cong c_j \doteq j$. So F5 has 25 local minima at which F5 takes on the values 1,2,...,25.

FIG. A.5A: F5 (25 FOX HOLES)

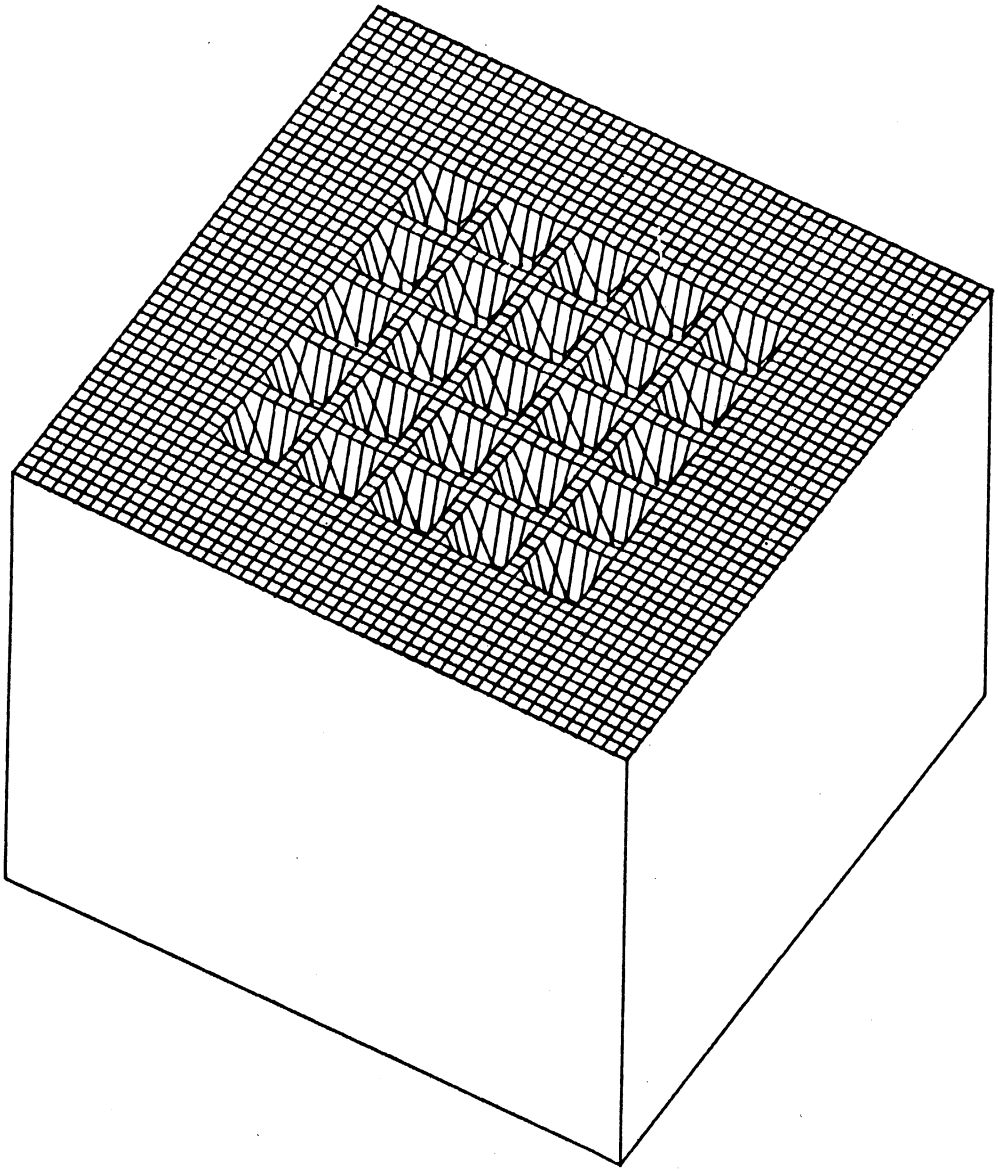


Figure A.5a: Top surface defined by test function F5.

FIG. A.5B: INVERTED F5 (25 FOX HOLES)

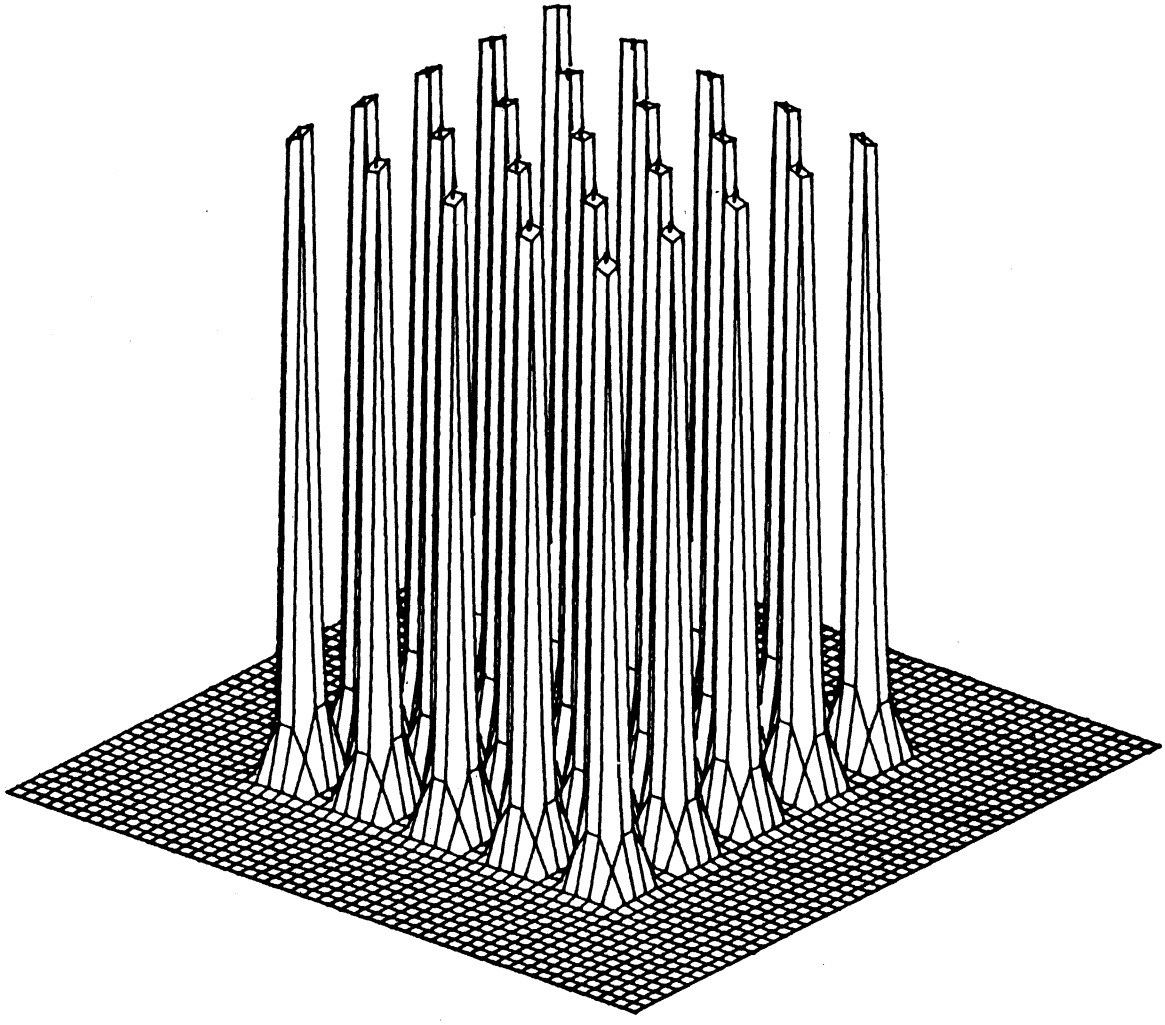


Figure A.5b: Bottom surface defined by test function F5.

Appendix B

RANDOM SEARCH ON E

B.1 Introduction

As a first attempt at evaluating genetic adaptive plans, their performance was compared with pure random search on the environment described in appendix A. Since pure random search takes advantage of none of the information accumulating over a sequence of trials, its performance serves as a lower bound on the performance of an adaptive plan. Any plan which claims to dynamically exploit sampling information had better show improved performance over random search.

In order to evaluate the performance of pure random search on the environment E, a plan RANDOM was implemented in PL/I to simulate a uniformly-distributed random search over the discrete spaces A associated with the test functions. Considerable difficulty was encountered in validating the uniform randomness of RANDOM on the multi-dimensional spaces associated with the test functions. Initially, RANDOM called the standard system library random number generator URAND N times to produce a point in N-space. However, this resulted in a non-uniform distribution biased toward a hypersphere in the center of the space. This approach was modified to maintain N separate pseudo-random streams in parallel, one for each axis. This improved the randomness in N-space somewhat,

although considerable difficulties arose in determining a method for selecting N seeds so that the streams were in fact independent.

At this point I began reading about random number generators and discovered Marsaglia's article (1968) pointing out the difficulties in generating uniform distributions in N-space with multiplicative congruential random number generators, of which URAND was an instance. This led me to papers by Tootill, etc. (1971, 1973) describing the properties of several classes of Tausworthe generators, first suggested by Tausworthe (1965), which are based on primitive polynomials over GF(2) and are implemented by linear shift register sequences. These generators can be shown analytically to generate uniformly random distributions in N-space when $N \leq K$. The value of K depends on the particular member of the class chosen, and the cost of generating the random numbers increases with K. A member of the class for which K=30 was chosen and implemented as the subroutine TRAND for use with RANDOM. This resulted in a considerable improvement in the uniform randomness of RANDOM and led to very little difficulty in validating the performance of RANDOM on the environment E.

B.2 Validating RANDOM on E

In order to provide a comparison with genetic algorithms, both the off-line and on-line performance

of RANDOM was measured for each of the test functions. Care was taken to attempt to validate the performance curves obtained from the simulated random searches. That is, I attempted to verify that performance curves approximated the expected performance and did not in fact represent an artifact of the pseudo-uniform distribution over the space. The verification took two forms. Whenever possible, I derived an analytic expression for the expected performance curve assuming a uniform distribution. If this was not possible, the space was rotated and inverted during the sequence of simulations used to generate a performance curve in an attempt to counteract any pseudo-random bias.

Expected off-line performance was derived as follows. If we consider a test function F as a random variable on the space A, we can ask: what is the expected number of trials required to find a point X in A such that $F(X) \leq \epsilon$. This is a standard waiting time problem for a sequence of Bernouli trials. That is, let p be the probability of a success ($F(X) \leq \epsilon$). Then the expected number of trials required to generate the first success is simply $f = \frac{1}{p}$. So the problem reduces to one of computing PROB $[F(X) \leq \epsilon]$. Whether or not this is easily computed depends, of course, on the particular function.

For on-line performance, we need to know the expected value of a sample at time t. For uniform random

search, this is simply the average value of F on the space A. Whether or not this is easily computed depends again on the particular test function.

B.3 RANDOM on F1

F1 provides a good validation test for RANDOM since it is amenable to mathematical analysis. For off-line performance, we need to compute PROB $\left[F1(X) \leq \epsilon\right]$. We note that, for F1, the points in A satisfying this constraint lie in the sphere defined by $x_1^2+x_2^2+x_3^2 = \epsilon$. Moreover, the search space A is bounded by constraints of the form $|x_i| \leq b$. Hence, for $\sqrt{\epsilon} \leq b$, we have:

$$\text{PROB}\left[F1(X) \leq \epsilon\right] = \frac{V_s(\sqrt{\epsilon})}{V_c(2b)}$$

where $V_s(\sqrt{\epsilon})$ represents the volume of a sphere of radius $\sqrt{\epsilon}$ and $V_c(2b)$ represents the volume of a cube with sides of length 2b. The volume of a sphere is given by:

$$V_s(\sqrt{\epsilon}) = \frac{4}{3}\pi\epsilon^{3/2}$$

so that

$$\text{PROB}\left[F1(X) \leq \epsilon\right] = \frac{4\pi\epsilon^{3/2}}{3(2b)^3} = \frac{\pi\epsilon^{3/2}}{6b^3}$$

and hence the expected number of trials required to locate a point satisfying the constraint is given by:

$$f = \frac{1}{\text{PROB}[F1(X) \leq \epsilon]} = \frac{6b^3}{\pi \epsilon^{3/2}}$$

Expected on-line performance is obtained by computing:

$$\text{AVE}(F1(X)) = \frac{1}{(2b)^3} \int_A F1(X)\, dX$$

$$= \frac{1}{(2b)^3} \int_{-b}^{b}\int\int x_1^2+x_2^2+x_3^2 \, dx_1 dx_2 dx_3$$

$$= \frac{1}{(2b)^3} * 8b^5$$

$$= b^2$$

Figures B.1a and B.1b compare the expected performance curves with those generated by RANDOM. As can be seen, the values agree closely.

B.4 RANDOM on F2

Deriving an analytic expression for the off-line performance of RANDOM on F2 is very difficult, if not impossible. As a consequence, no direct validation of the performance curve illustrated in figure B.2a was done. However, the space A was rotated and inverted during the generation of the performance curve in an

FIG. B.1A: OFF-LINE PERFORMANCE ON F1

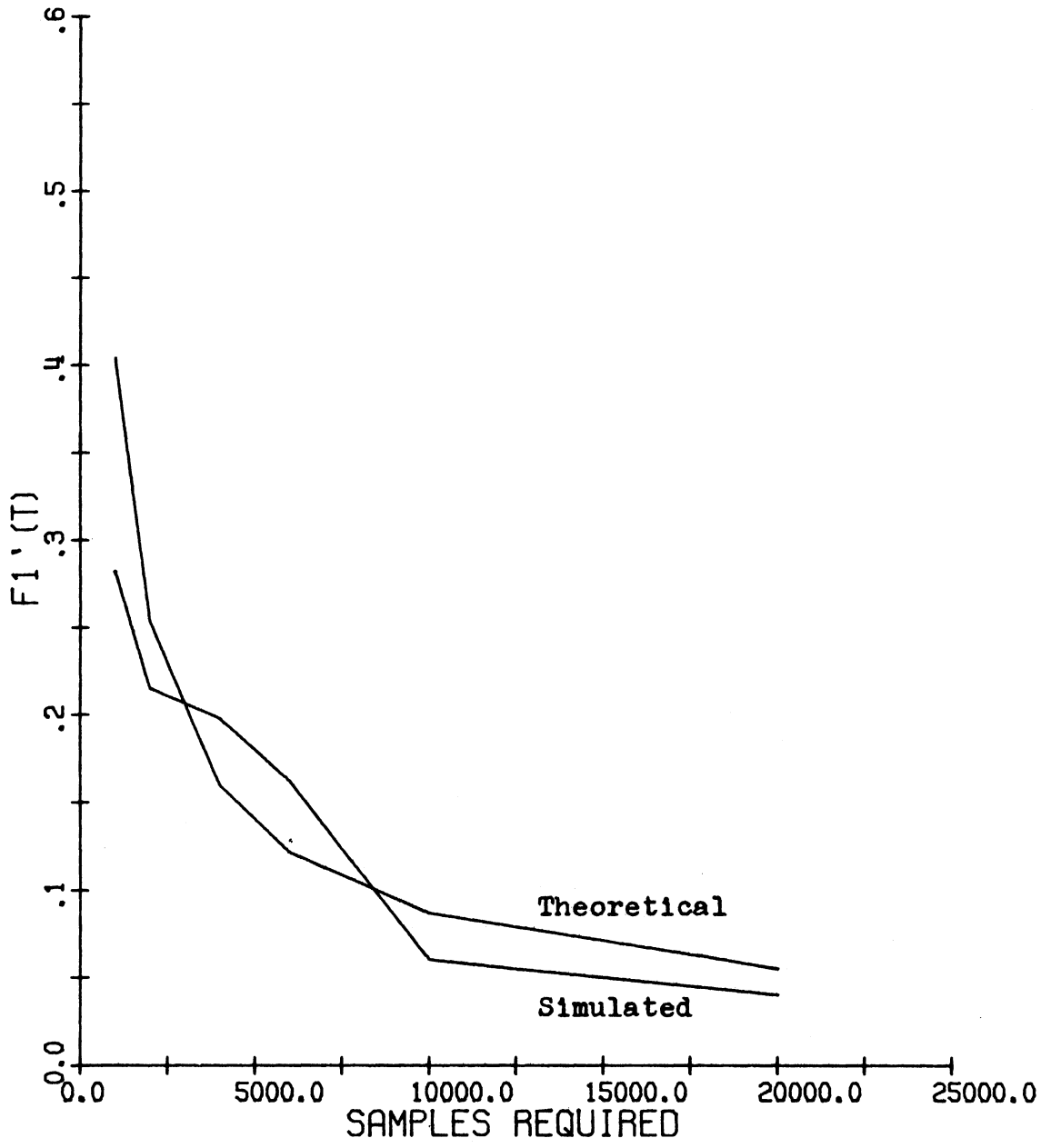


Figure B.1a: Off-line performance curves for random search on test function F1.

FIG. B.1B: ON-LINE PERFORMANCE ON F1

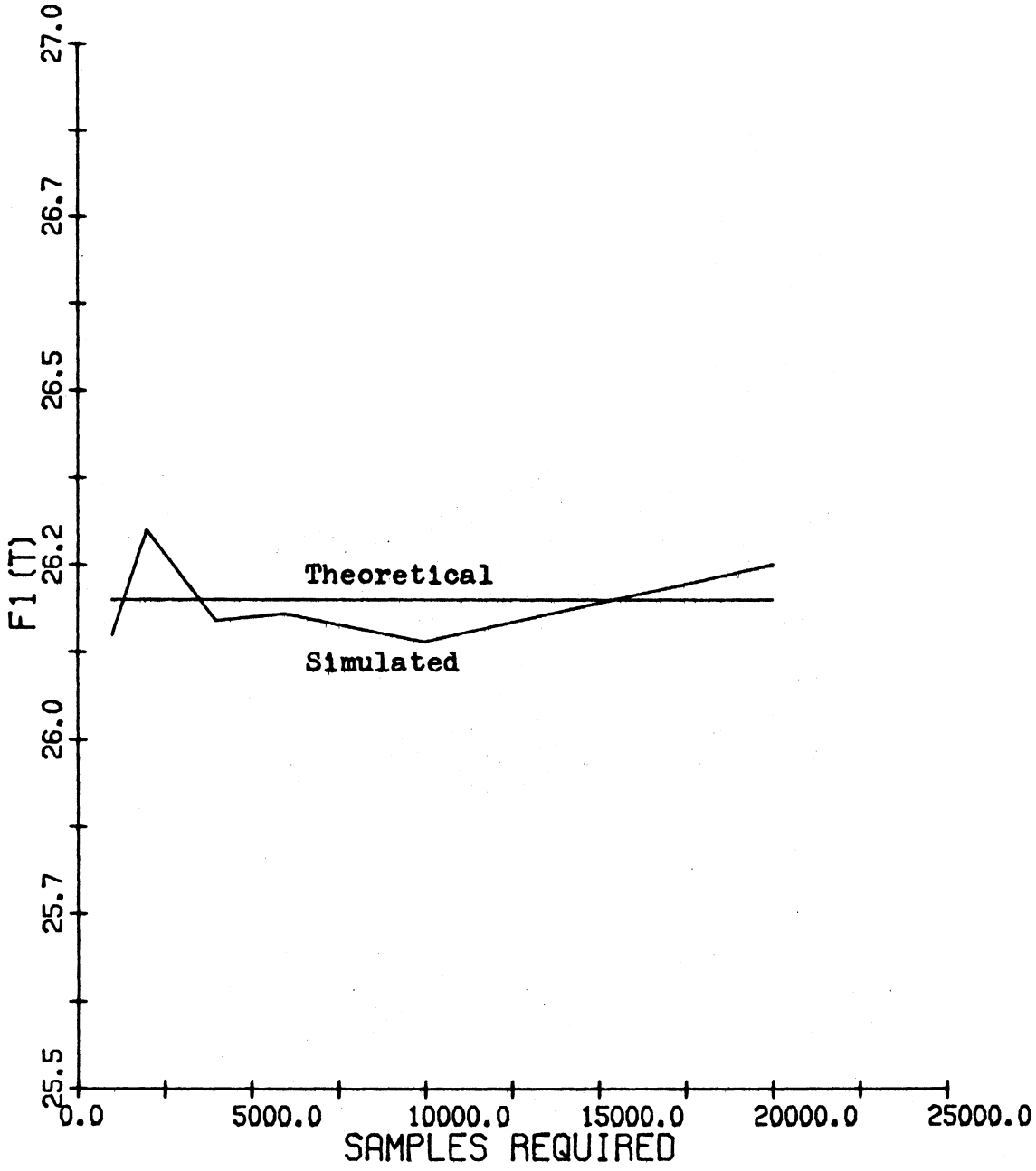


Figure B.1b: On-line performance curves for random search on test function F1.

FIG. B.2A: OFF-LINE PERFORMANCE ON F2

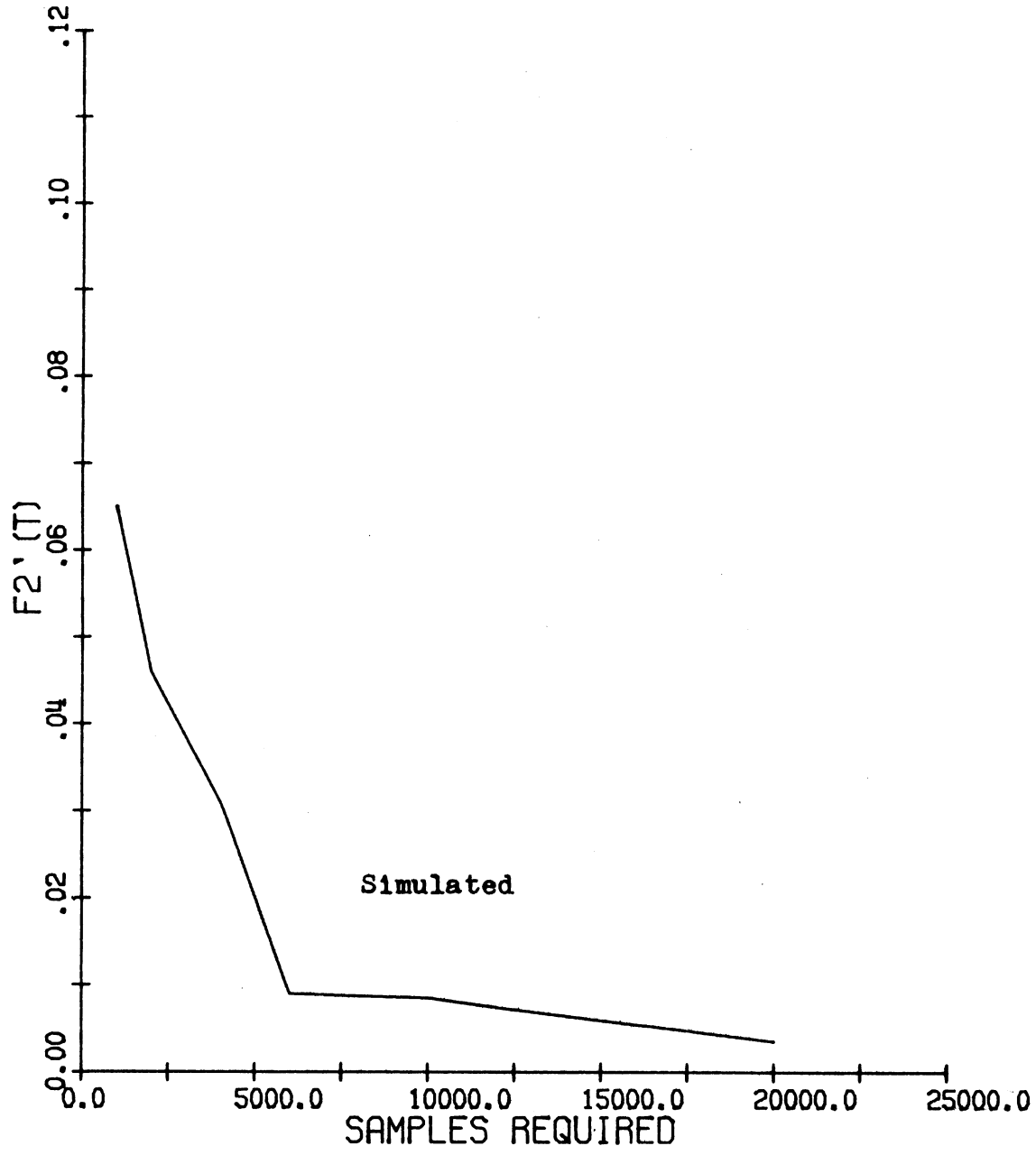


Figure B.2a: Off-line performance curve for random search on test function F2.

attempt to minimize any bias.

Expected on-line performance on F2 is computed as follows:

$$\text{AVE}(F2(X)) = \frac{1}{(2b)^2} \int_A F2(X)\, dX$$

$$= \frac{1}{(2b)^2} \int_{-b}^{b}\int_{-b}^{b} 100(x_1^2-x_2)^2+(1-x_1)^2\, dx_1 dx_2$$

$$= \frac{1}{4b^2} \left(80b^6+\frac{404}{3}b^4+4b^2\right)$$

$$= 20b^4 + \frac{101}{3} b^2 + 1$$

Figure B.2b compares the theoretical on-line performance with that generated by RANDOM. The agreement is quite good.

B.5 RANDOM on F3

Deriving the expected off-line performance on F3 is a straightforward, but tedious, problem in combinatorics. The probability of landing on a particular plateau is given by:

$$p = \frac{1}{(2b)^5} \prod_{i=1}^{5} \ell_i$$

where ℓ_i are the lengths of the sides of the plateau. It remains only to count the number of plateaus satis-

FIG. B.2B: ON-LINE PERFORMANCE ON F2

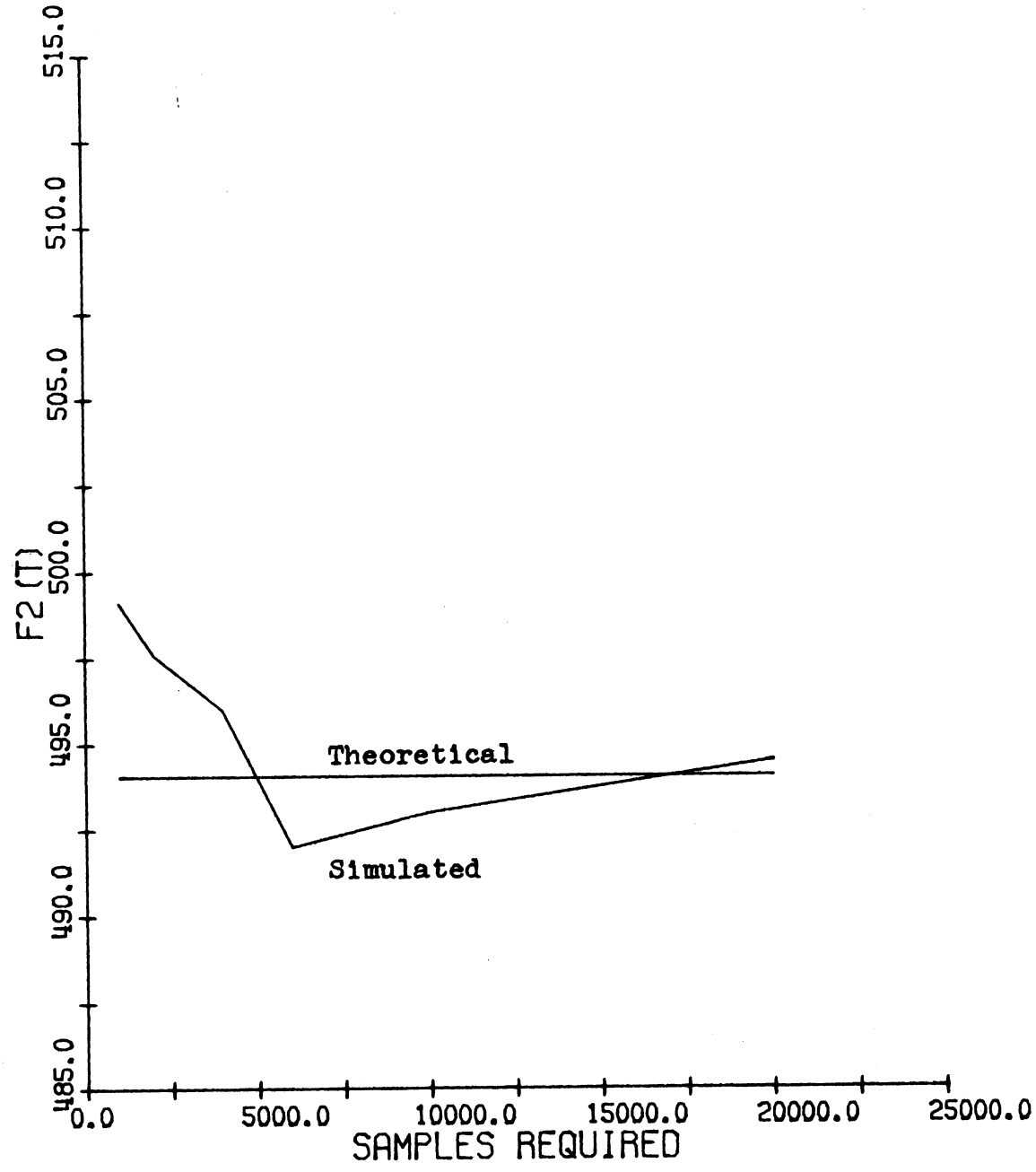


Figure B.2b: On-line performance curves for random search on test function F2.

fying the constraint $F3(X) \leq \epsilon$ and sum their associated probabilities. For example, the PROB $\left[F3(X) \leq -29\right]$ when $|x_1| \leq 5.12$ is computed as follows:

Since $F3(X) = \sum_{1}^{5} \left[x_1\right]$, the minimum value $F3(X)=-30$ can only be obtained by $\left[x_1\right] = 6$ for all i. Hence,

$$\text{PROB}\left[F3(X)=-30\right] = \left(\frac{.12}{10.24}\right)^5$$

Similarly, $F3(X)=-29$ is obtained by summing

$$(-6)+(-6)+(-6)+(-6)+(-5)$$

so that

$$\text{PROB}\left[F3(X)=-29\right] = \binom{5}{1}\left(\frac{.12}{10.24}\right)^4\left(\frac{1}{10.24}\right)$$

and summing the two yields PROB $\left[F3(X) \leq -29\right]$.

Expected on-line performance on F3 is given by:

$$\text{AVE}(F3(X)) = \text{AVE} \sum_{1}^{5} \left[x_1\right]$$

$$= \sum_{1}^{5} \text{AVE}\left[x_1\right]$$

$$= 5 * \left(\left[-b\right] + \left[b\right]\right)/2$$

Figures B.3a and B.3b compare the expected performance curves with those obtained from RANDOM. As can be seen, the values are very close.

FIG B.3A: OFF-LINE PERFORMANCE ON F3

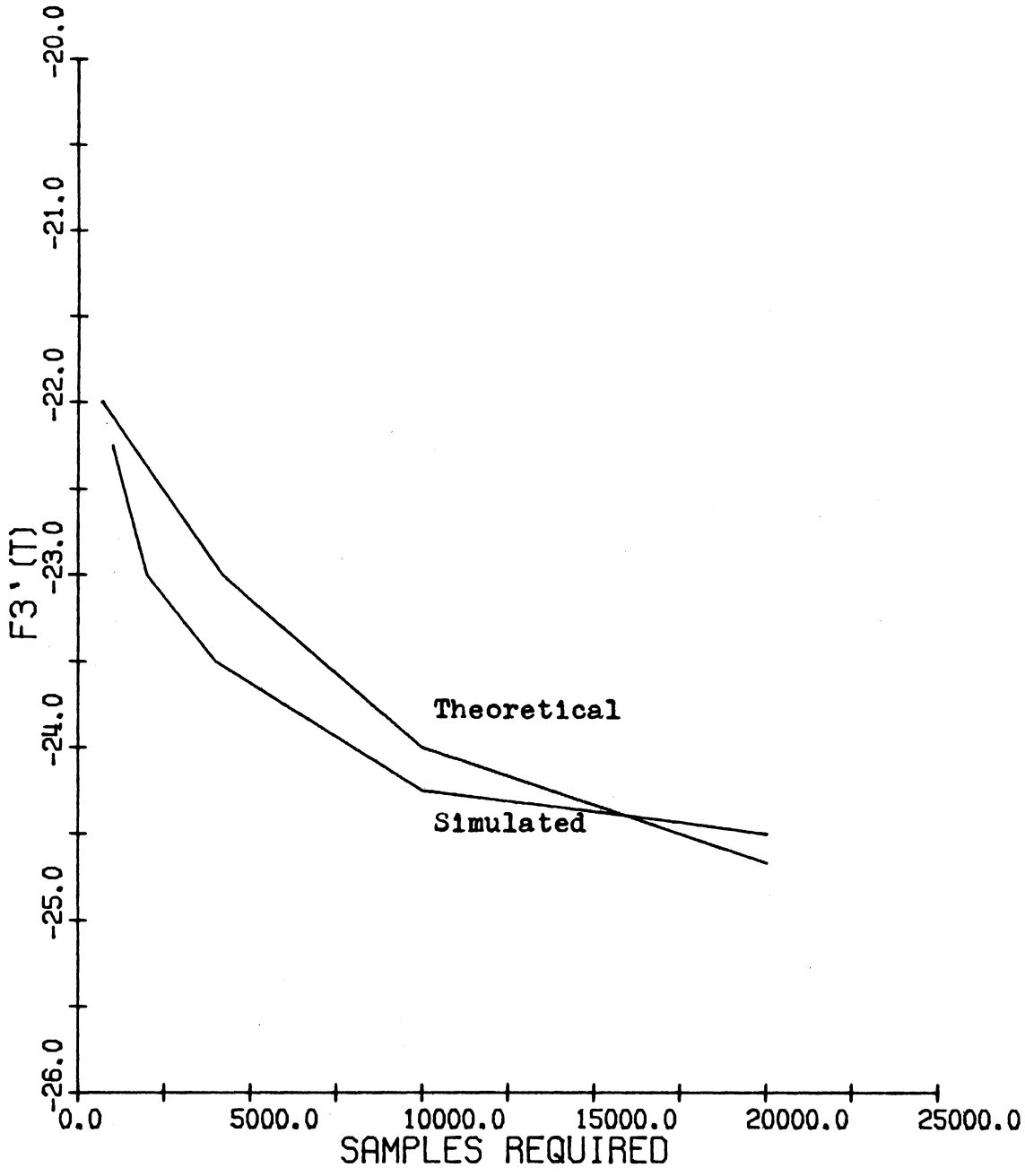


Figure B.3a: Off-line performance curves for random search on test function F3.

FIG B.3B: ON-LINE PERFORMANCE ON F3

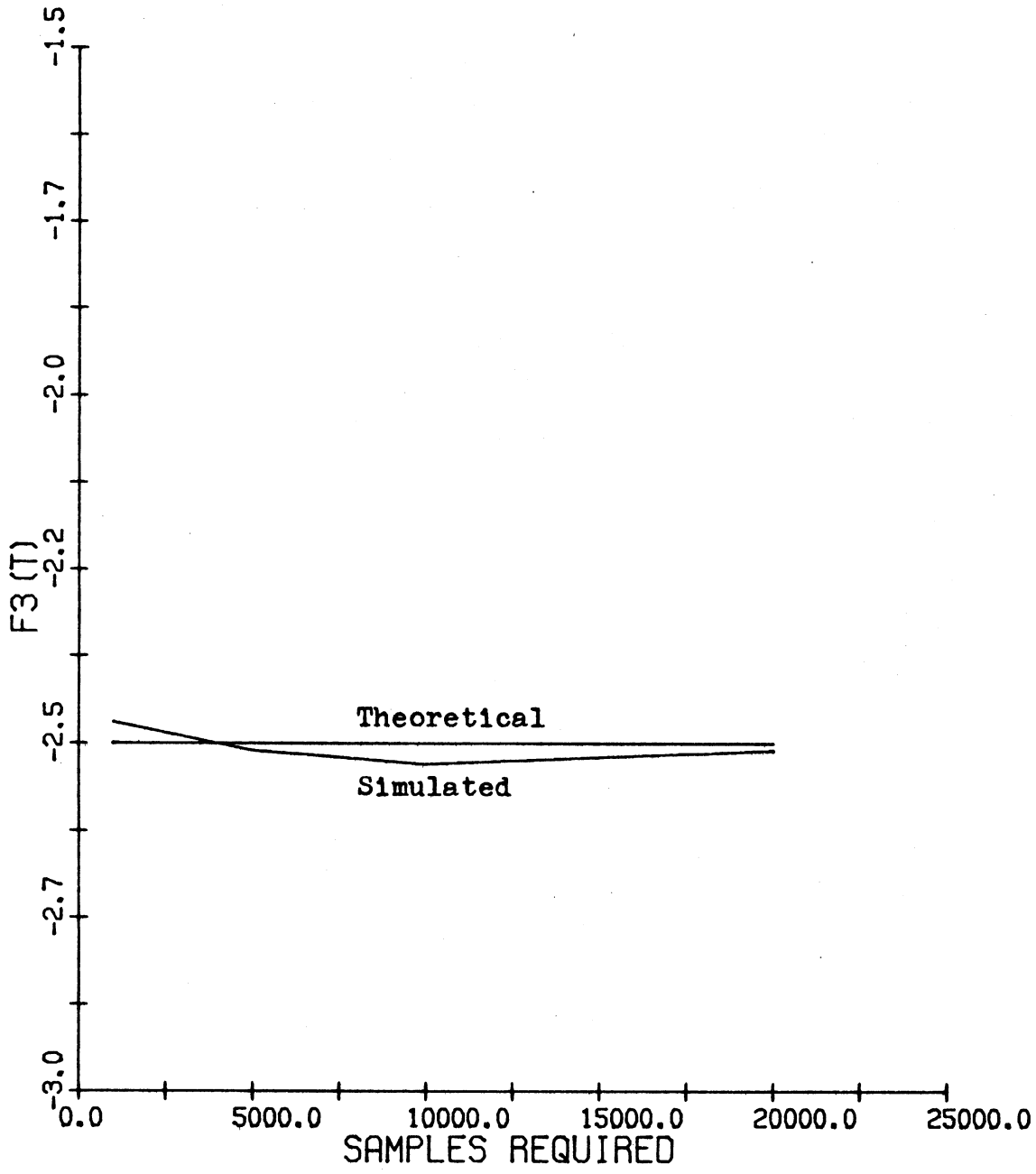


Figure B.3b: On-line performance curves for random search on test function F3.

B.6 RANDOM on F4

Deriving expected off-line performance for F4 is difficult. We need to compute PROB$\left[\sum_{1}^{30} ix_i^4+\text{GAUSS}(0,1)\leq\epsilon\right]$ The problem can be simplified somewhat by noting that over the interval of observation, RANDOM was unable to reduce F4*(t) below 50. This suggests that a reasonable approximation is obtained by ignoring the Gaussian noise, that is, PROB $\left[\sum_{1}^{30} ix_i^4 \leq\epsilon\right]$. As in the case of F1, we have that

$$\text{PROB}\left[\sum_{1}^{30} ix_i^4\leq\epsilon\right] = \frac{V_s(\epsilon^{1/4})}{V_c(2b)}, \quad \epsilon^{1/4}\leq b$$

Note, however, that b=1.28 for F4, limiting the above computation to $\epsilon<3$ which is far beyond any practical interval of observation.

Alternatively, we can attempt to bound the PROB $\left[\sum_{1}^{30} ix_i^4 \leq\epsilon\right]$ by noting that:

$$\left\{X:\ |x_1| \leq \left(\frac{\epsilon}{30i}\right)^{1/4}\right\} C \left\{X:\ \sum_{1}^{30} ix_i^4\leq\epsilon\right\} C \left\{X:\ |x_1|\leq\left(\frac{\epsilon}{1}\right)^{1/4}\right\}$$

However, because of the high dimensionality of F4, these bounds are extremely crude, effectively bounding the probability by 0 and 1. As a consequence no direct validation of the off-line performance curve for RANDOM on F4 (shown in figure B.4a) was made. Rather, an attempt was made to minimize any bias due to pseudo-randomness by rotating and inverting the space as the performance

FIG. B.4A: OFF-LINE PERFORMANCE ON F4

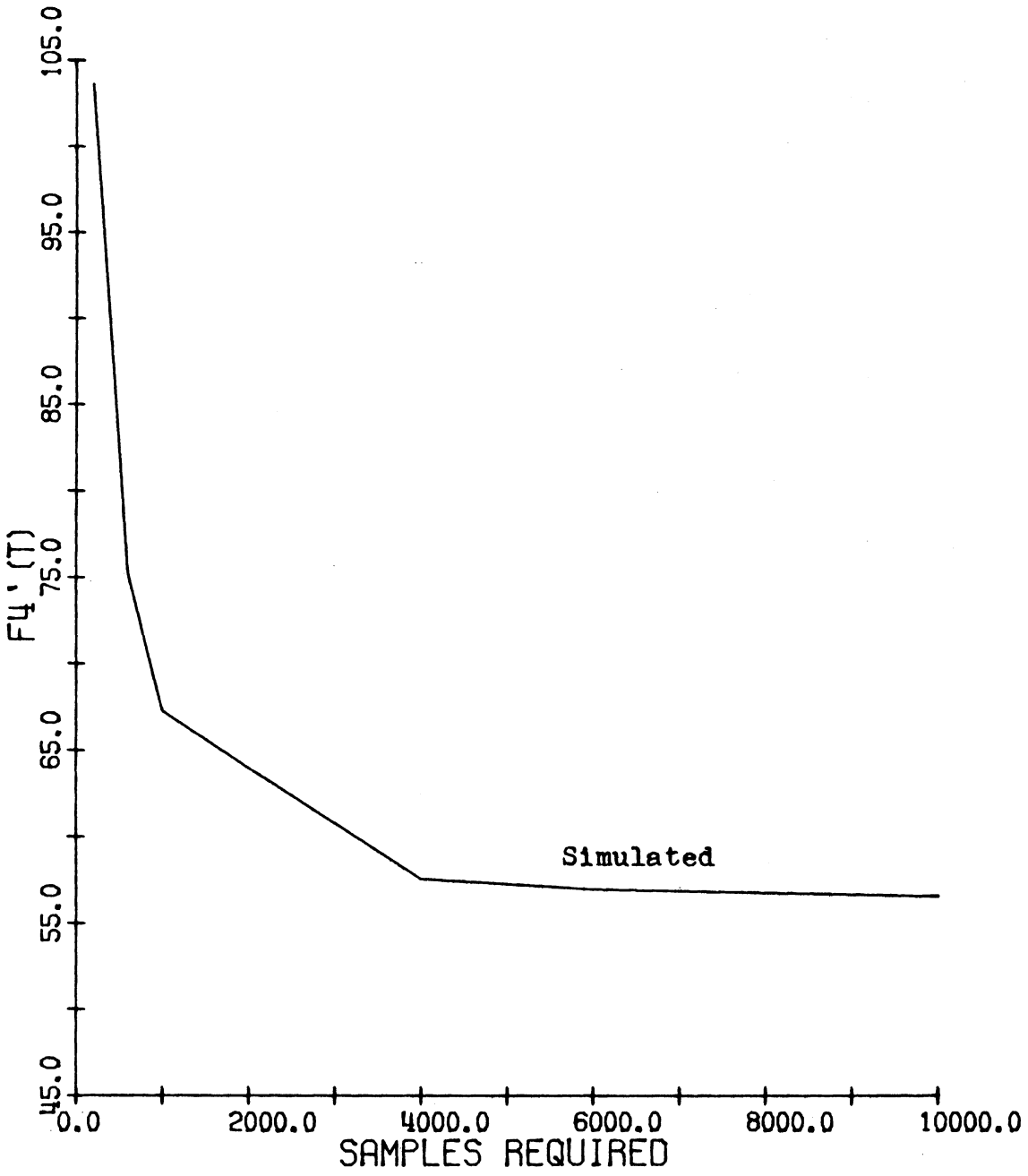


Figure B.4a: Off-line performance curves for random search on test function F4.

curve was generated.

Expected on-line performance on F4 is given by:

$$\text{AVE}(F4(X)) = \frac{1}{(2b)^{30}} \int_{-b}^{b}\int\cdots\int \sum_{1}^{30} ix_i^4 \, dx_1 \ldots dx_{30}$$

$$= \frac{1}{(2b)^{30}} \sum_{1}^{30} \int_{-b}^{b}\int\cdots\int ix_i^4 \, dx_1 \ldots dx_{30}$$

$$= \frac{1}{(2b)^{30}} \sum_{1}^{30} i \cdot \frac{b^4}{5} \cdot (2b)^{30}$$

$$= \frac{b^4}{5} \sum_{1}^{30} i$$

$$= 93b^4$$

Figure B.4b compares the expected on-line performance of random search on F4 with the curve generated by RANDOM. The results closely match.

B.7 RANDOM on F5

To analytically derive the expected off-line performance for random search on F5 is difficult. However, because of the composite nature of F5, a reasonable approximation is not difficult. Since near the j^{th} local minimum $F5(X) \cong f_j(X)$ and elsewhere $F5(X) \cong 500$, we have

FIG. B.4B: ON-LINE PERFORMANCE ON F4

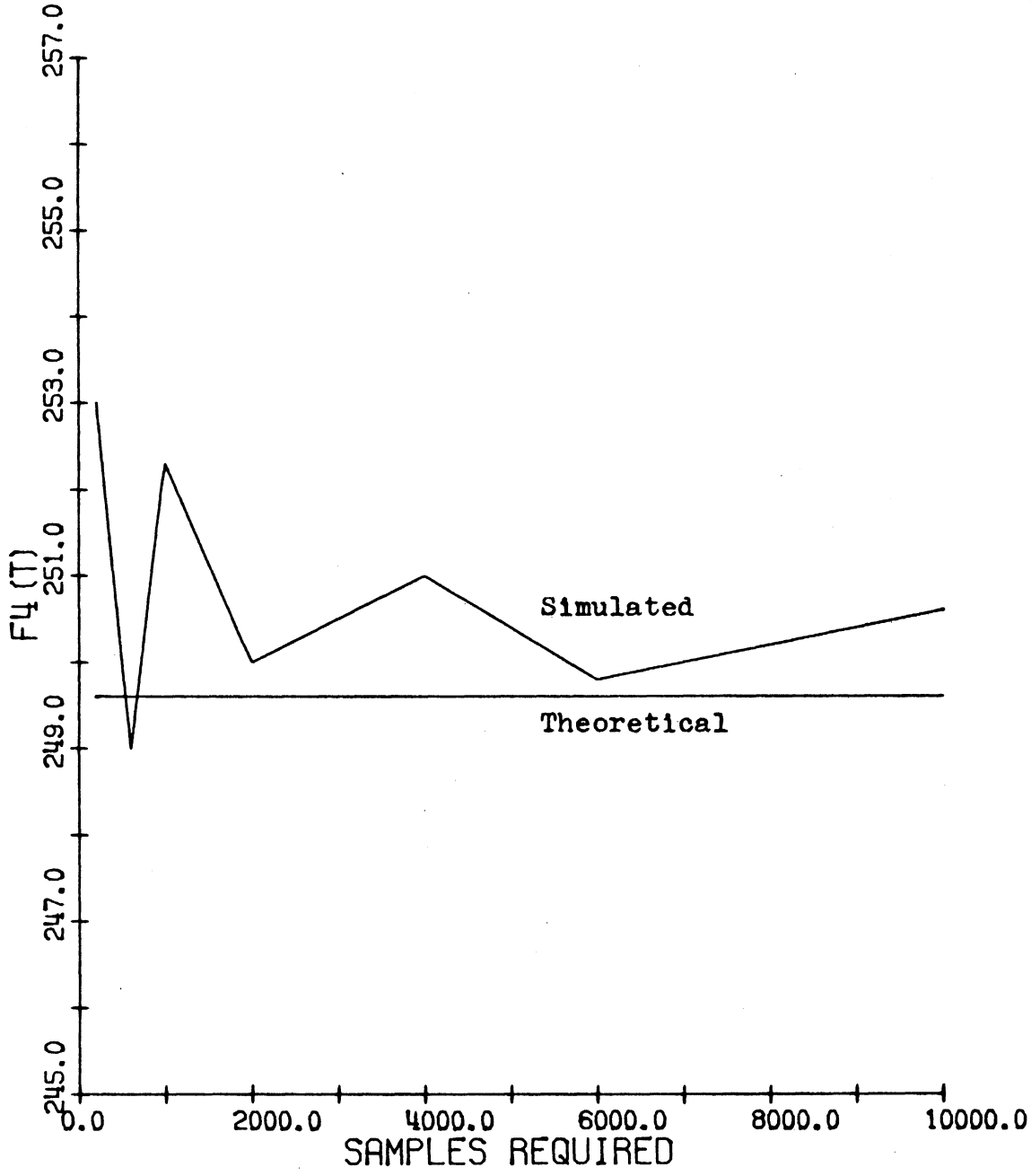


Figure B.4b: On-line performance curves for random search on test function F4.

$$\text{PROB}\left[F5(X) \leq \epsilon\right] = \sum_{j=1}^{25} \text{PROB}\left[f_j(X) \leq \epsilon\right] \quad , \epsilon << 500$$

For each j we have:

$$\text{PROB}\left[f_j(X) \leq \epsilon\right] = \text{PROB}\left[c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6 \leq \epsilon\right]$$

$$= \text{PROB}\left[\sum_{i=1}^{2} (x_i - a_{ij})^6 \leq \epsilon - c_j\right]$$

$$= \text{PROB}\left[\sum_{i=1}^{2} x_i^6 \leq \epsilon - c_j\right]$$

$$= \begin{cases} 0 \text{ if } \epsilon < c_j \\ \dfrac{V_s((\epsilon - c_j)^{1/6})}{V_c(2b)} \text{ if } 0 \leq (\epsilon - c_j)^{1/6} \leq b \end{cases}$$

where

$$V_s(k^{1/6}) = \int_{-k^{1/6}}^{k^{1/6}} \int_{-(k-x_2^6)^{1/6}}^{(k-x_2^6)^{1/6}} 1 \; dx_1 dx_2$$

$$= 4 \int_{0}^{k^{1/6}} (k-x_2^6)^{1/6} \, dx_2$$

$$= 4k^{1/3} \int_0^1 (1-y^6)^{1/6}\, dy$$

which is beyond my powers of integration. However, numerical integration yields:

$$\int_0^1 (1-y^6)^{1/6}\, dy = .963688$$

If we assume this is accurate enough for our purposes, we have:

$$\text{PROB}\left[f_j(X) \leq \epsilon\right] \simeq \begin{cases} 0 \text{ if } \epsilon < c_j \\ \dfrac{3.854752}{(2b)^2} * (\epsilon - c_j)^{1/3}, \; 0 \leq (\epsilon - c_j)^{1/6} \leq b \end{cases}$$

This approximation was used to estimate the expected off-line performance of random search on F5. Figure B.5a compares this approximation with the curve generated by RANDOM. The two agree surprisingly well.

The expected on-line performance of random search on F5 is also difficult to derive. In fact, even a reasonable approximation is difficult to find. As a consequence, no direct validation of the on-line performance curve for RANDOM shown in figure B.5b was done. Rather, the search space was rotated and inverted while

FIG. B.5A: OFF-LINE PERFORMANCE ON F5

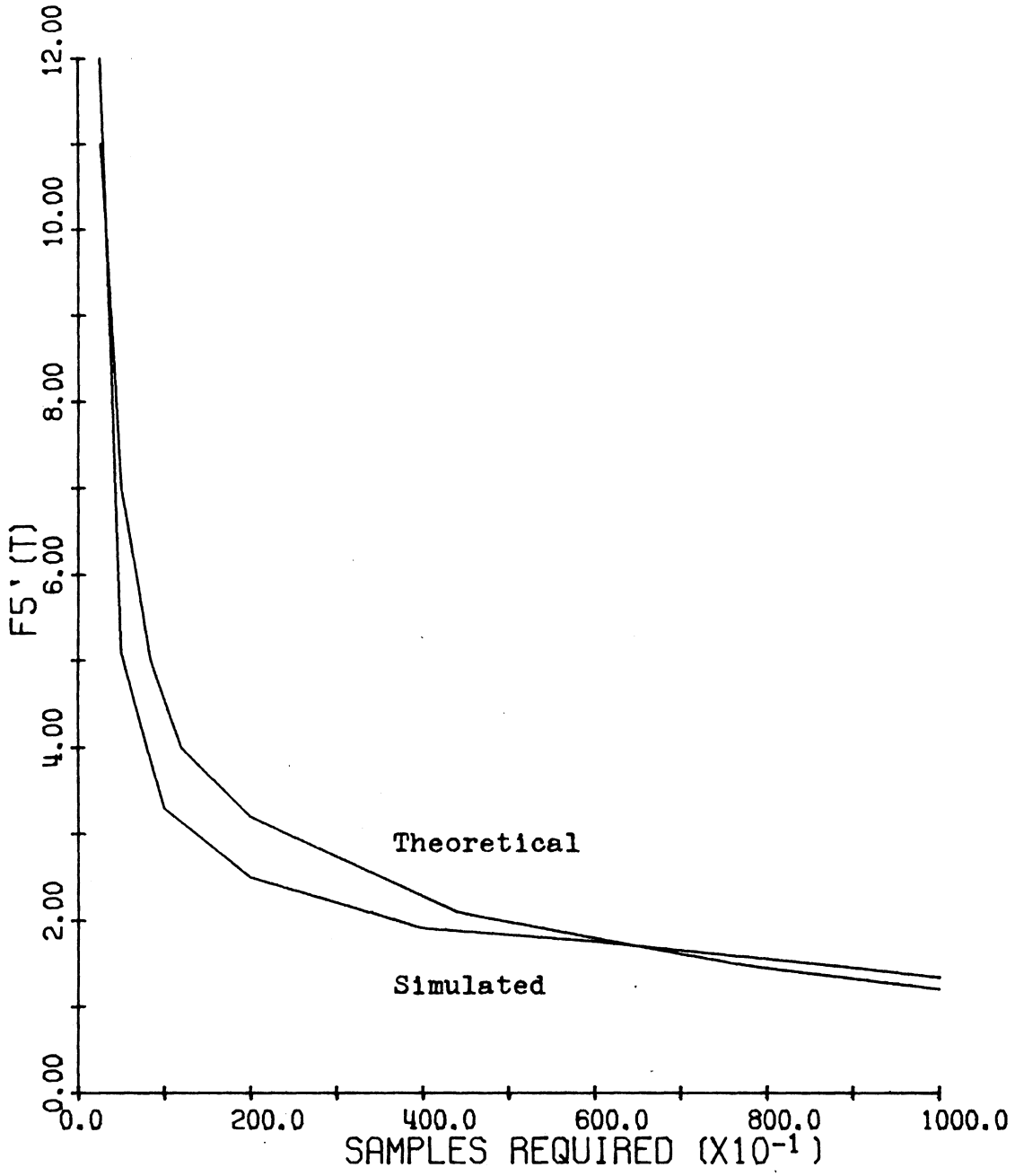


Figure B.5a: Off-line performance curves for random search on test function F5.

FIG. B.5B: ON-LINE PERFORMANCE ON F5

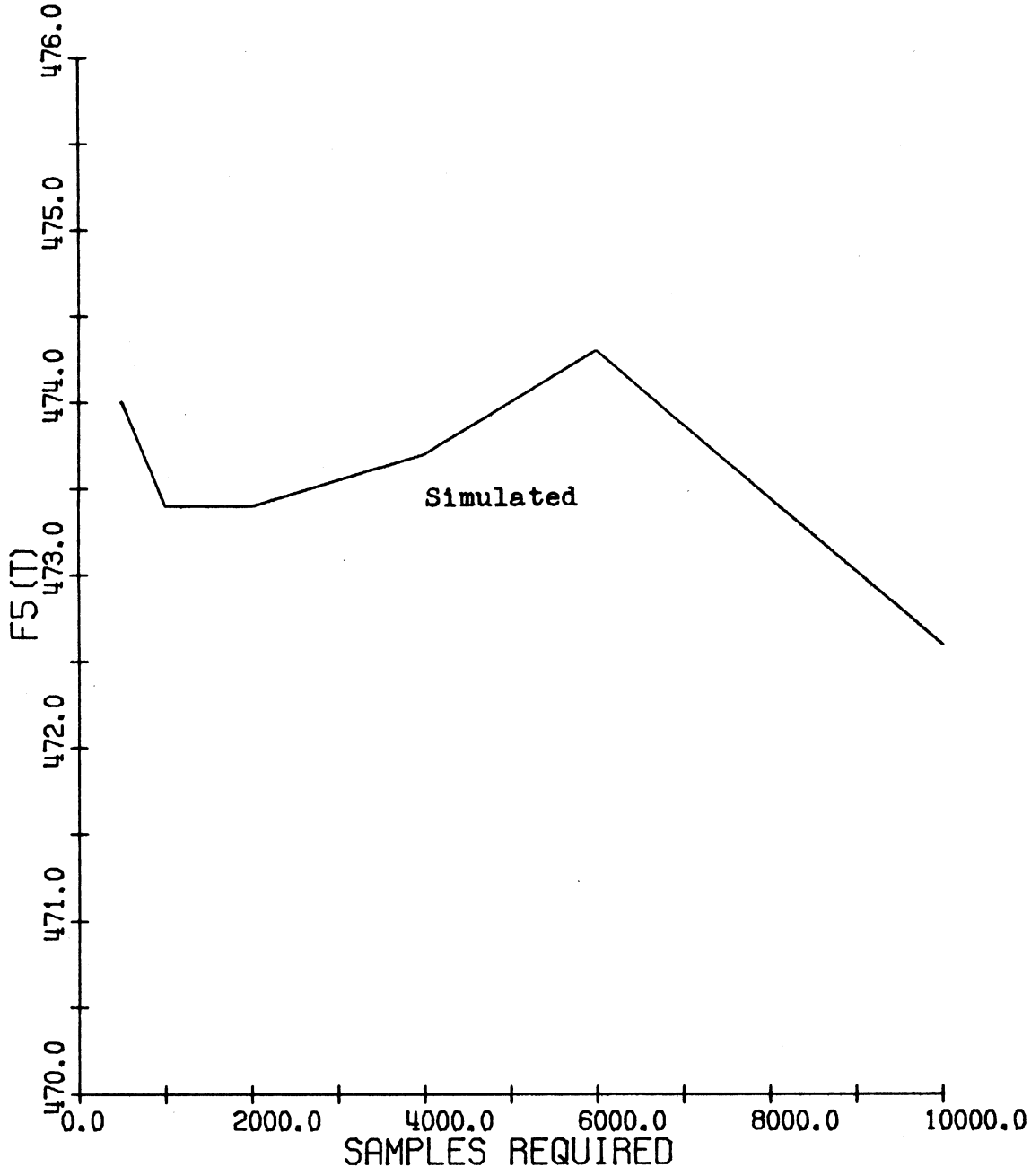


Figure B.5b: On-line performance curve for random search on test function F5.

generating the performance curve in an attempt to minimize any bias due to pseudo-randomness.

Appendix C

PLAN R1 ON E

C.1 Introduction

The idea of simulating the information processing mechanisms of heredity and evolution as strategies for adaptation in artificial systems was first introduced by Holland. Since then considerable experimentation and analysis has been done in such areas as the kind of genetic operators to be applied, the form of the representation space, the mating rules to be used, and so on.

Bagley has compared the behavior of correlation algorithms and genetic algorithms with respect to environments exhibiting first and second order non-linearities. He showed that comparable or superior behavior could be generated by genetic algorithms using population sizes of 200, standard genetic operators applied at fixed levels, and random mating.

Cavicchio has studied the behavior of genetic algorithms in a pattern-matching environment. He was able to show the negative effects of a small population size (less than 20) and experimented with several forms of genetic operators applied at varying levels. He was able to generate excellent adaptive behavior using a scheme for modifying the frequency with which genetic operators are applied during adaptation.

Hollstien has studied the effects that represent-

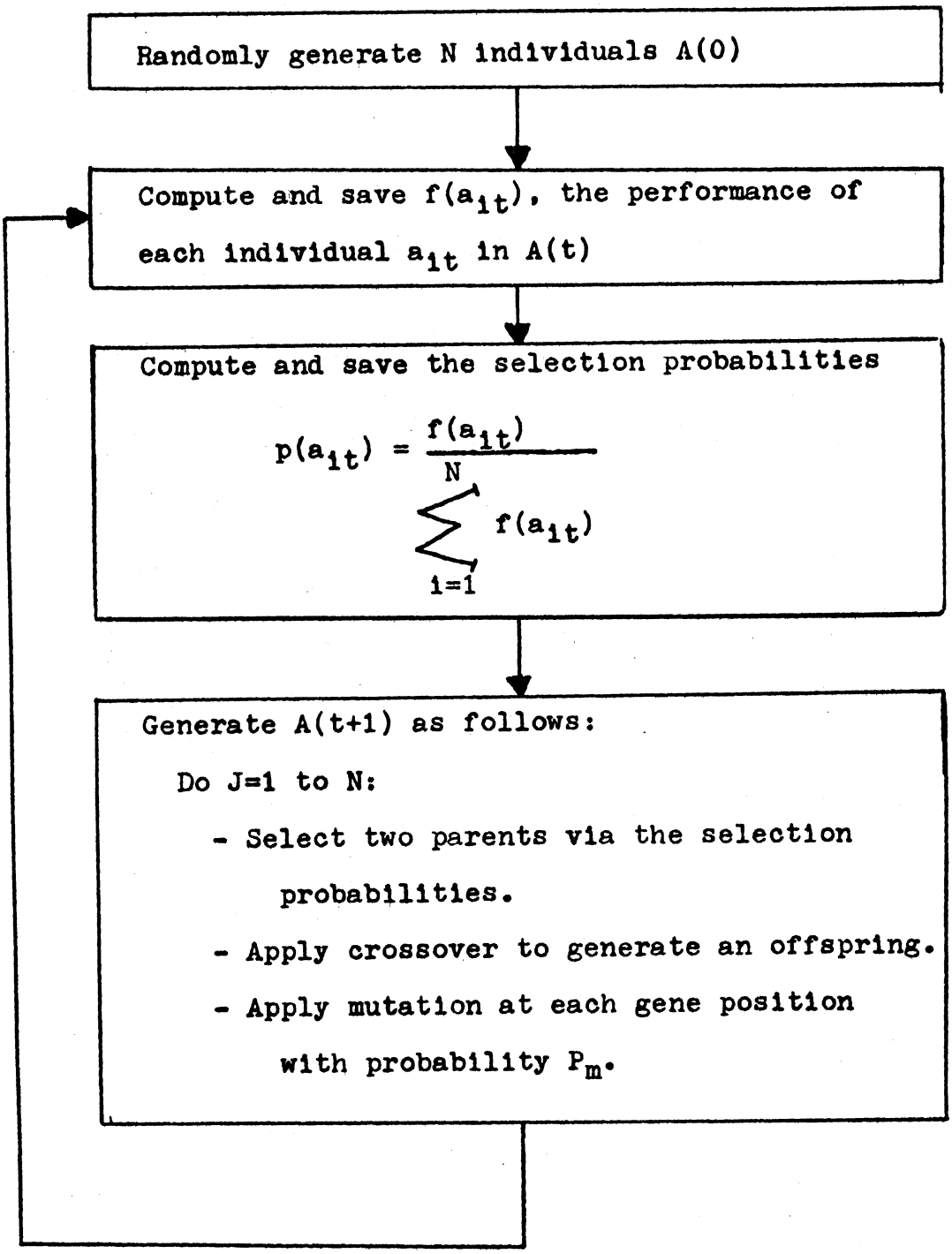
ation and breeding plans have on the behavior of genetic algorithms in a variety of continuous and discontinuous two-variable environments. Using mutation and crossover at fixed rates and a population of size 16, he exhibited robust behavior for both random mating and recurrent inbreeding-crossbreeding. He showed the negative effects of such breeding plans as linebreeding and inbreeding. He also examined the effect of several representations for genes including binary, gray code, and polygene forms.

Frantz has studied the effects of the genetic operators on the composition of the resultant population in the face of highly non-linear environments. He was able empirically to verify hypotheses about the effects of crossover and mutation, but had difficulty in verifying inversion effects.

Foo and Bosworth have attempted mathematical analyses of the basic genetic operators. Bosworth, Foo, and Zeigler have experimented with incorporating gradient information as a mutation operator in the context of function optimization. In the same context Zeigler, Bosworth, and Bethke have shown genetic algorithms to be relatively insensitive to noisy environments when compared with classical gradient optimization techniques.

The cumulative experience of these studies provided the framework for the definition of the initial elementary reproductive plan R1 introduced in chapter 2 and

which operates as follows:



This actually specifies a class of genetic algorithms which differ in the parameters N and P_m. Previous experience suggested that population sizes smaller than 30 were subject to severe stochastic effects which made performance measurements difficult. For these measurements, a population size of N=50 was chosen, and a mutation rate of .001 which is an upperbound on the estimate of the mutation rate in nature.

Because R1 is a stochastic process, a minimum of 5 trials was made on each test function. More were made if required to reduce the standard 95% confidence intervals associated with the performance measurements. The results are presented in several ways. Graphs are given to compare the off-line and on-line performance curves $f^*(t)$ and $f(t)$ of R1 and random search on each of the test functions described in appendix A. Tables are given to compare the performance indices $x_e^*(T)$ and $x_e(T)$ for R1 and random search for various values of T on each of the test functions. Finally, the robustness of R1 and random search on E is compared using the measures defined in Chapter 1.

C.2 Plan R1 on F1

Figures C.1a and C.1b compare the performance curves for R1 and random search on F1 over the interval of observation. The associated performance ratings $x_{F1}^*(T)$ and $x_{F1}(T)$ are tabularized below for various values of T:

FIG. C.1A: OFF-LINE PERFORMANCE ON F1

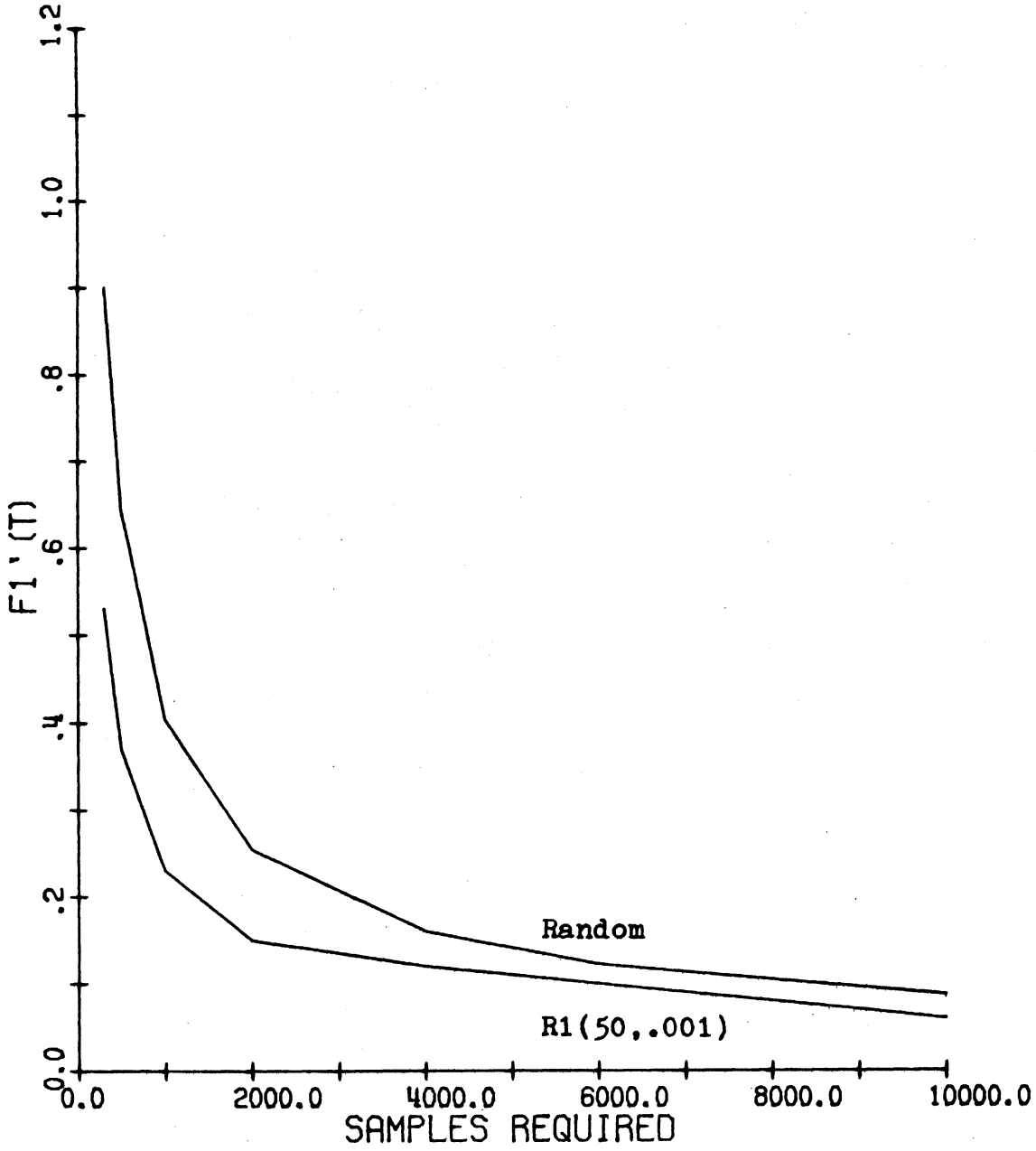


Figure C.1a: Off-line performance curve for plan R1 on test function F1.

FIG. C.1B: ON-LINE PERFORMANCE ON F1

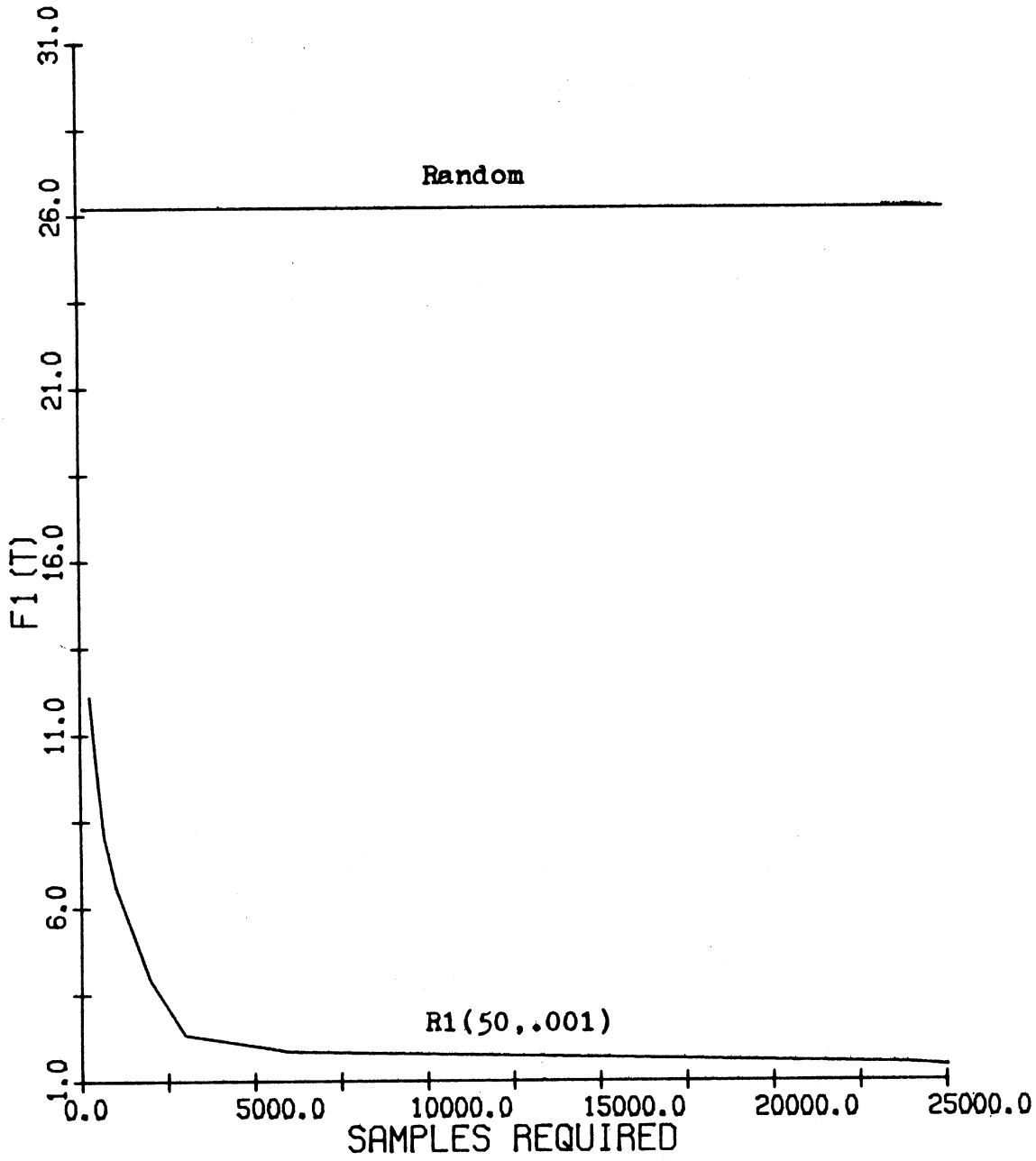


Figure C.1b: On-line performance curve for plan R1 on test function F1.

$x^*_{F1}(T)$

T	R1	Random
500	1.25	3.0
1000	.73	.94
2000	.48	.75
4000	.28	.44
6000	.22	.36
10000	.125	.27

$x_{F1}(T)$

T	R1	Random
500	14.0	26.2
1000	10.6	26.2
2000	7.8	26.2
4000	5.3	26.2
6000	4.6	26.2
10000	3.1	26.2

So we see that R1 performs better than random search on F1 over the interval of observation. This, of course, was expected for on-line performance. Notice that the off-line performance of R1 is not strikingly better; it gets off to a good start and then seems to plateau. This observation is explored more fully in chapter 3.

C.3 Plan R1 on F2

Figures C.2a and C.2b compare the performance curves of R1 and random search on F2. The associated performance

FIG. C.2A: OFF-LINE PERFORMANCE ON F2

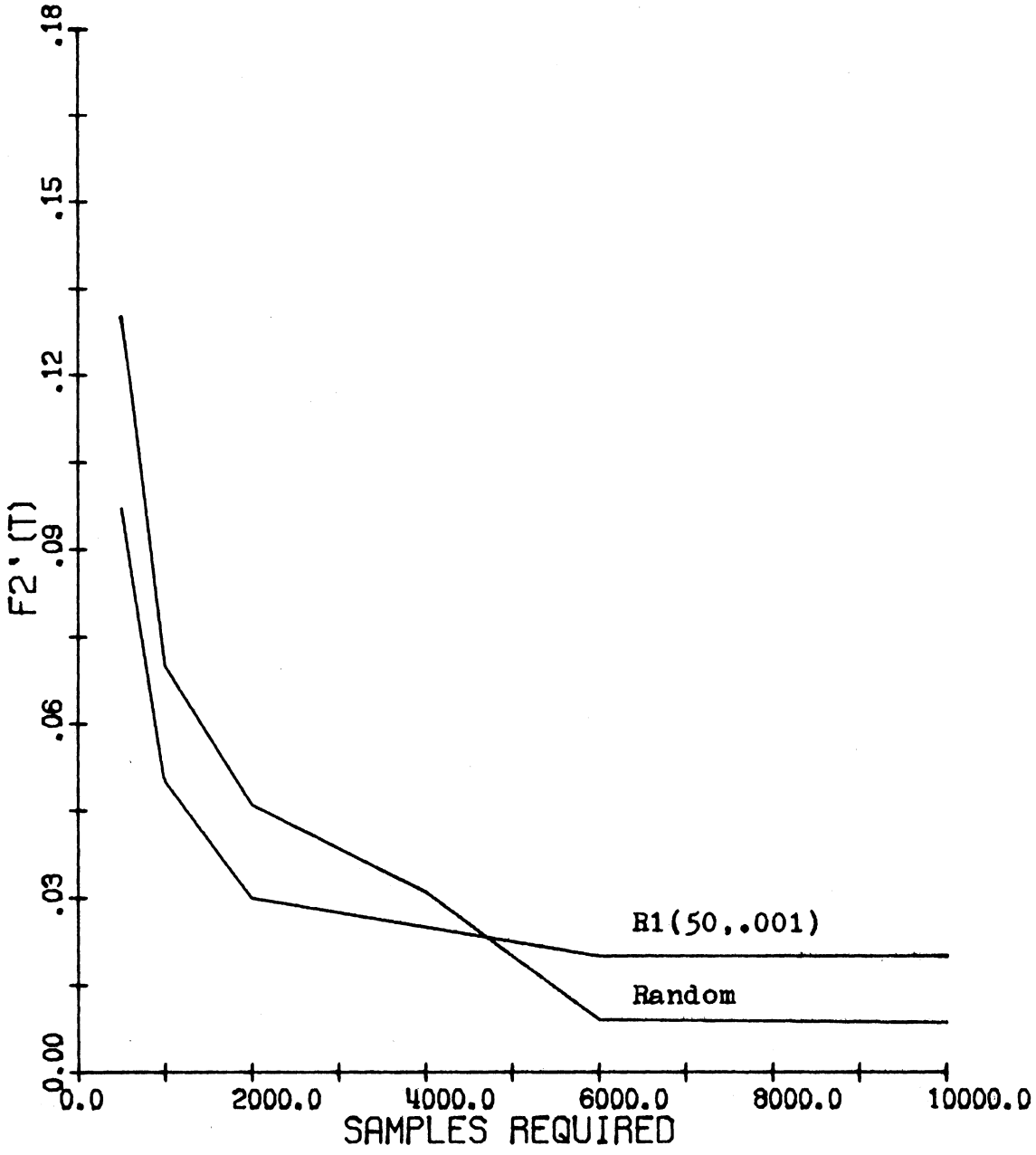


Figure C.2a: Off-line performance curve for plan R1 on test function F2.

FIG. C.2B: ON-LINE PERFORMANCE ON F2

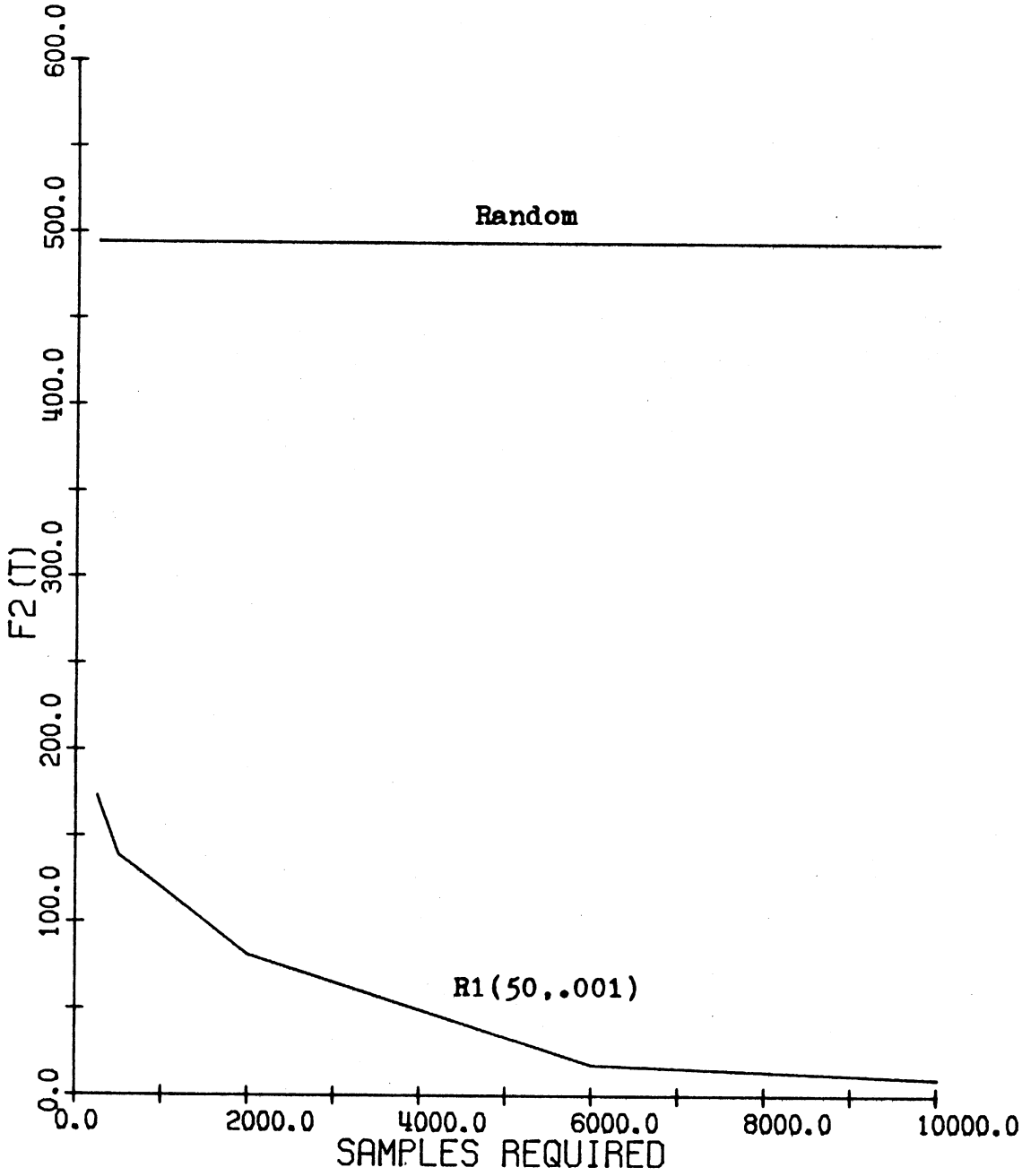


Figure C.2b: On-line performance curve for plan R1 on test function F2.

ratings $x^*_{F2}(T)$ and $x_{F2}(T)$ are tabularized below for various values of T:

$x^*_{F2}(T)$

T	R1	Random
500	3.72	3.97
1000	1.84	1.93
2000	.93	1.02
4000	.47	.53
6000	.34	.35
10000	.25	.20

$x_{F2}(T)$

T	R1	Random
500	219.3	494.05
1000	170.0	494.05
2000	133.3	494.05
4000	100.2	494.05
6000	78.4	494.05
10000	67.7	494.05

So we see that R1 generally outperforms random search on F2 over the interval of observation. This, of course, was expected for on-line performance. However, note that off-line performance is initially better, but then actually falls behind random search. This observation is explored more fully in chapters 3 and 4.

C.4 Plan R1 on F3

Figures C.3a and C.3b compare the performance curves of R1 and random search on F3. The associated performance ratings $x^*_{F3}(T)$ and $x_{F3}(T)$ are tabulated below for various values of T:

$x^*_{F3}(T)$

T	R1	Random
500	-23.1	-18.0
1000	-23.8	-19.7
2000	-24.6	-21.1
4000	-25.5	-22.3
6000	-26.2	-22.7
10000	-27.1	-23.3

$x_{F3}(T)$

T	R1	Random
500	-10.8	-2.5
1000	-14.4	-2.5
2000	-18.3	-2.5
4000	-21.0	-2.5
6000	-22.1	-2.5
10000	-23.0	-2.5

So we see that R1 performed considerably better on F3 than random search, indicating that discontinuous functions pose no problem for genetic plans.

FIG C.3A: OFF-LINE PERFORMANCE ON F3

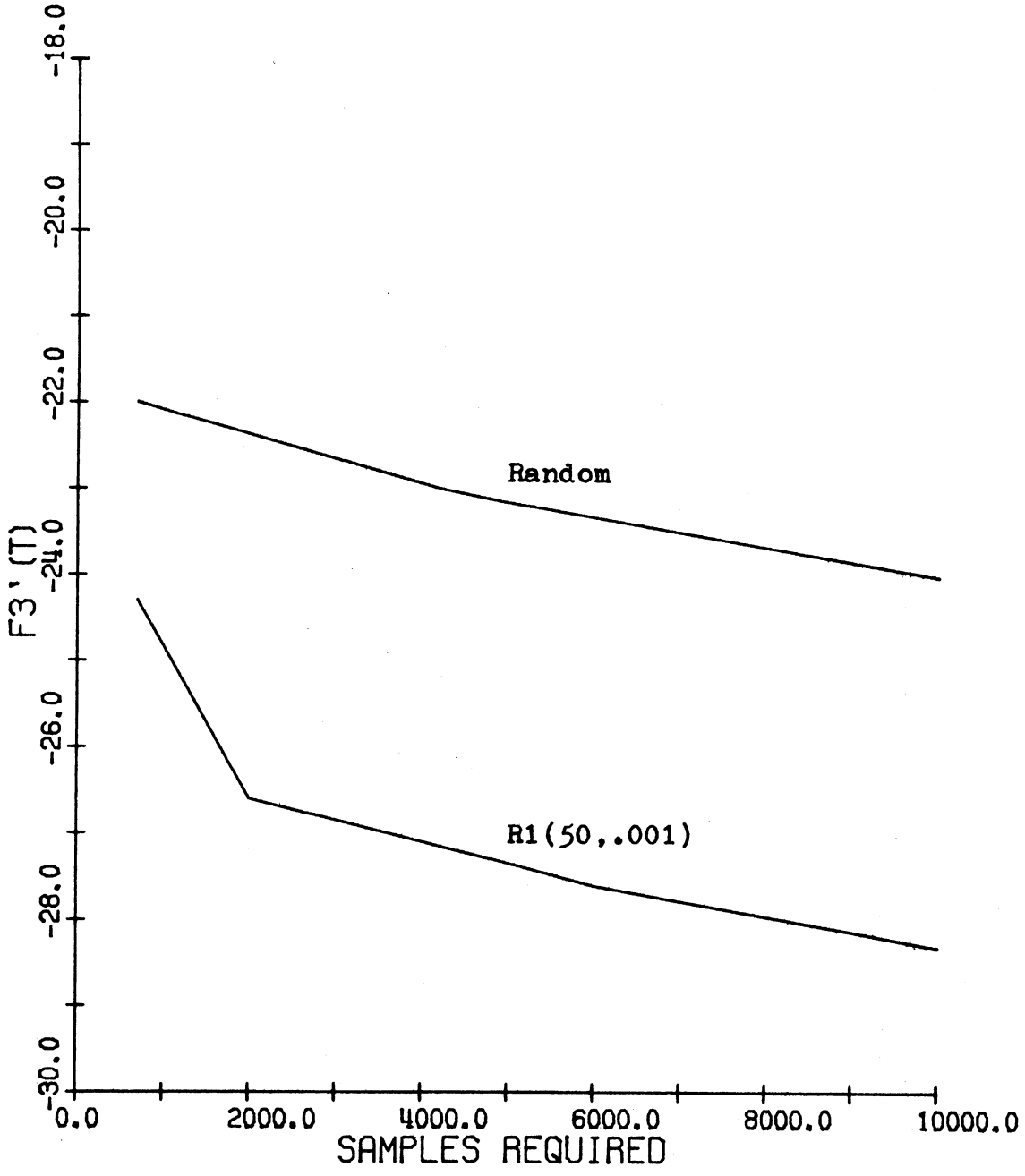


Figure C.3a: Off-line performance curve for plan R1 on test function F3.

FIG C.3B: ON-LINE PERFORMANCE ON F3

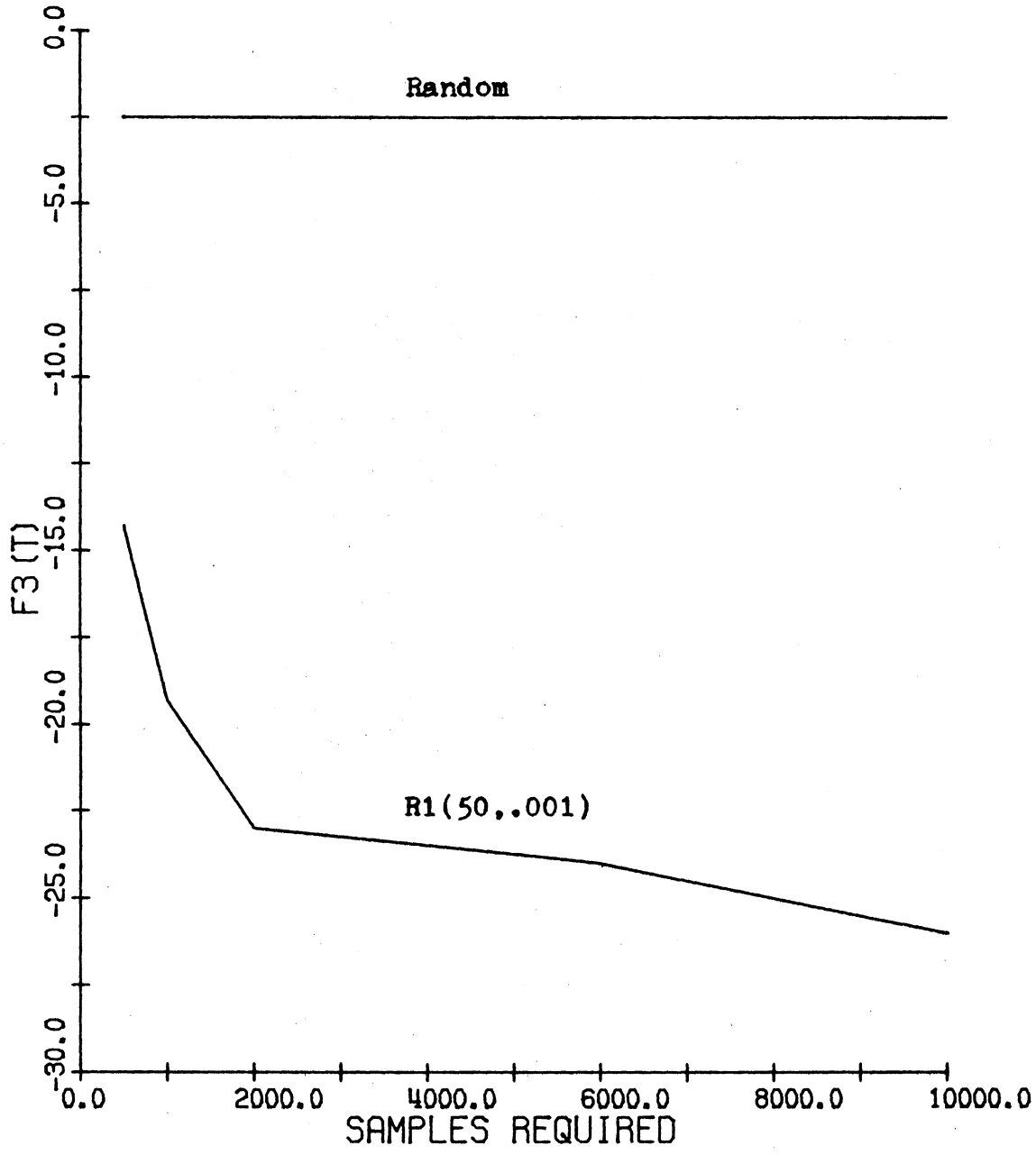


Figure C.3b: On-line performance curve for plan R1 on test function F3.

C.5 Plan R1 on F4

Figures C.4a and C.4b compare the performance curves for R1 and random search on F4. The associated performance indices computed for several values of T are given below:

$x^*_{F4}(T)$

T	R1	Random
500	80.9	105.0
1000	61.2	89.2
2000	46.7	78.6
4000	38.5	70.6
6000	35.9	66.3
10000	31.6	63.7

$x_{F4}(T)$

T	R1	Random
500	180.1	249.6
1000	149.5	249.6
2000	120.9	249.6
4000	105.4	249.6
6000	97.3	249.6
10000	75.6	249.6

So we see that R1 performed considerably better on F4 than random search, indicating that high-dimensionality and noise pose no particular problems for genetic algorithms.

FIG. C.4A: OFF-LINE PERFORMANCE ON F4

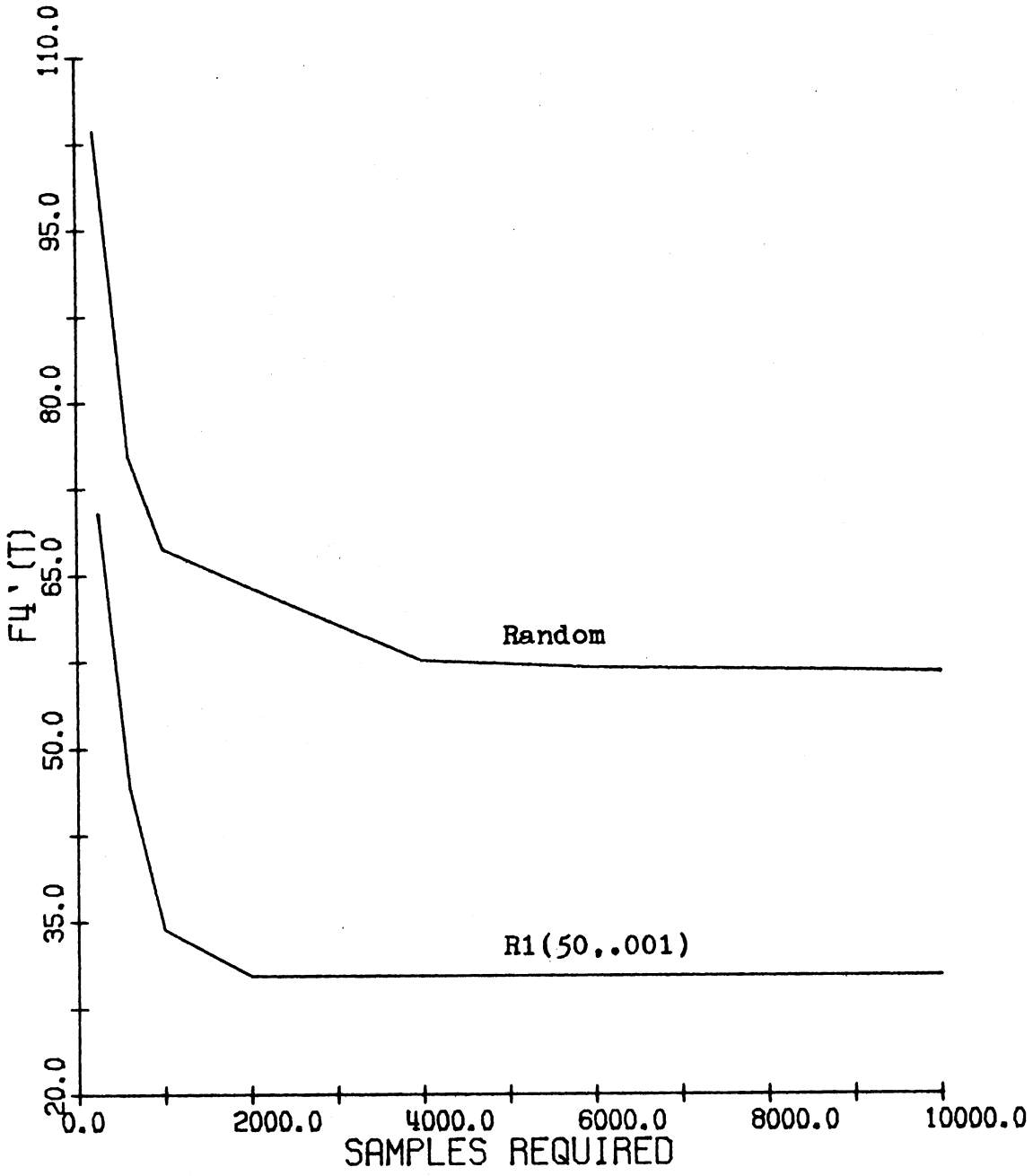


Figure C.4a: Off-line performance curve for plan R1 on test function F4.

FIG. C.4B: ON-LINE PERFORMANCE ON F4

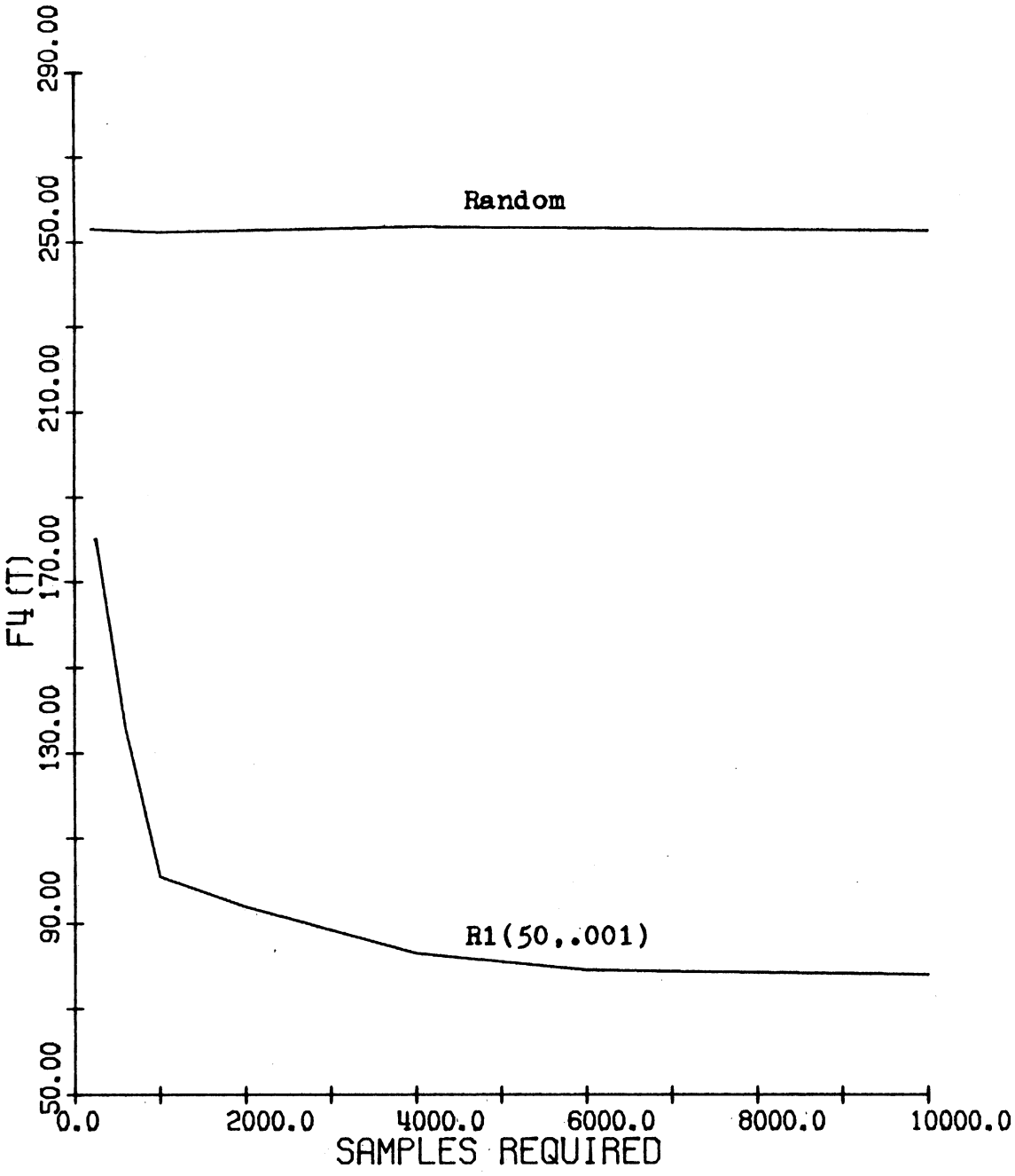


Figure C.4b: On-line performance curve for plan R1 on test function F4.

C.6 Plan R1 on F5

Figures C.5a and C.5b compare the performance curves for R1 and random search on F5. The associated performance indices computed for various values of T are given below:

$x^*_{F5}(T)$

T	R1	Random
500	11.4	35.7
1000	6.21	19.2
2000	3.74	10.6
4000	2.15	6.25
6000	1.83	4.82
10000	1.61	3.45

$x_{F5}(T)$

T	R1	Random
500	127.0	474
1000	99.7	473.4
2000	71.9	473.4
4000	51.6	473.7
6000	36.2	473.3
10000	14.7	472.6

So we see that R1 outperforms random search on F5, indicating that multimodal surfaces pose no particular problems for genetic algorithms.

C.7 Robustness of Plan R1

Robustness is defined in chapter 1 as the average

FIG. C.5A: OFF-LINE PERFORMANCE ON F5

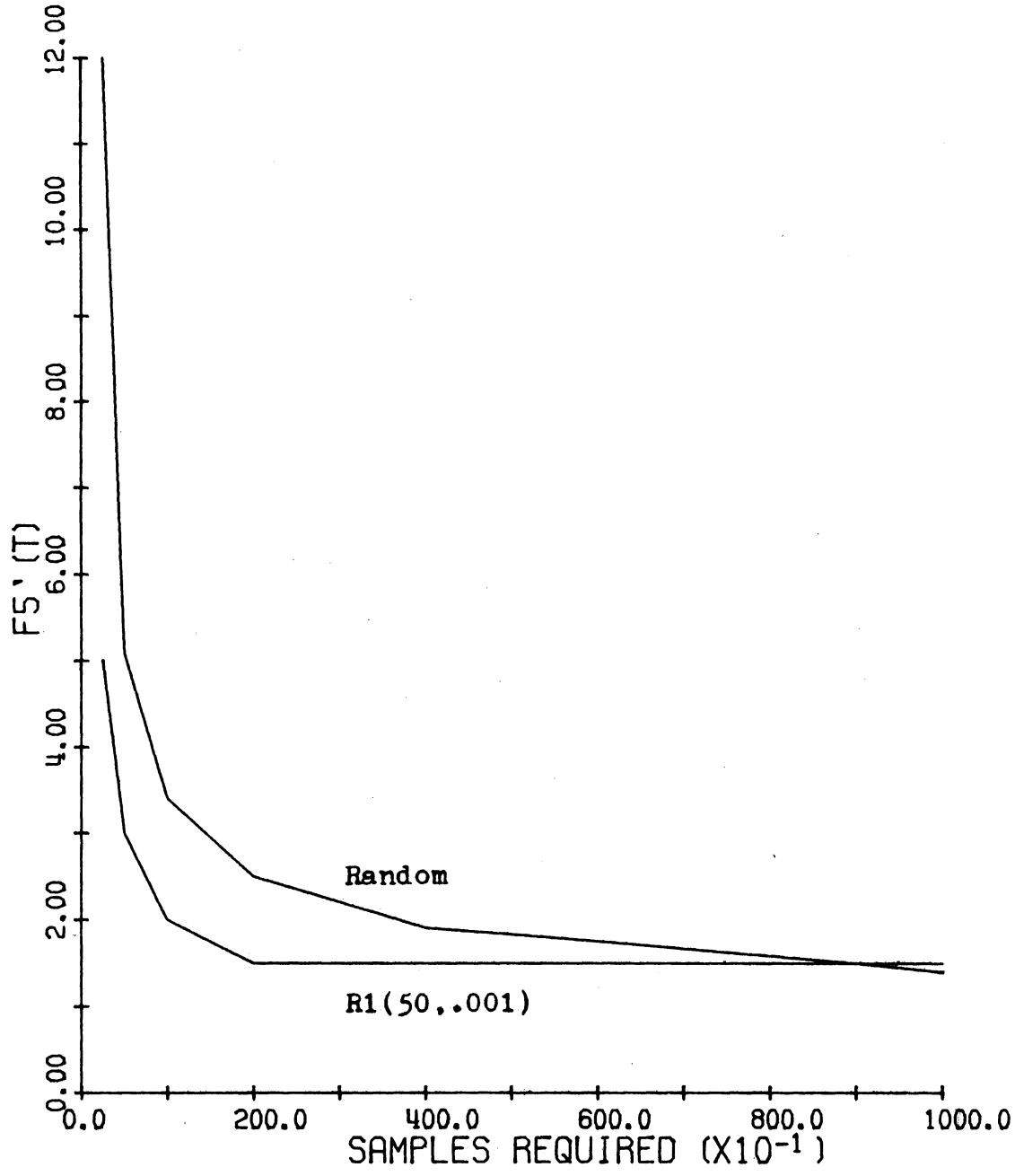


Figure C.5a: Off-line performance curve for plan R1 on test function F5.

FIG. C.5B: ON-LINE PERFORMANCE ON F5

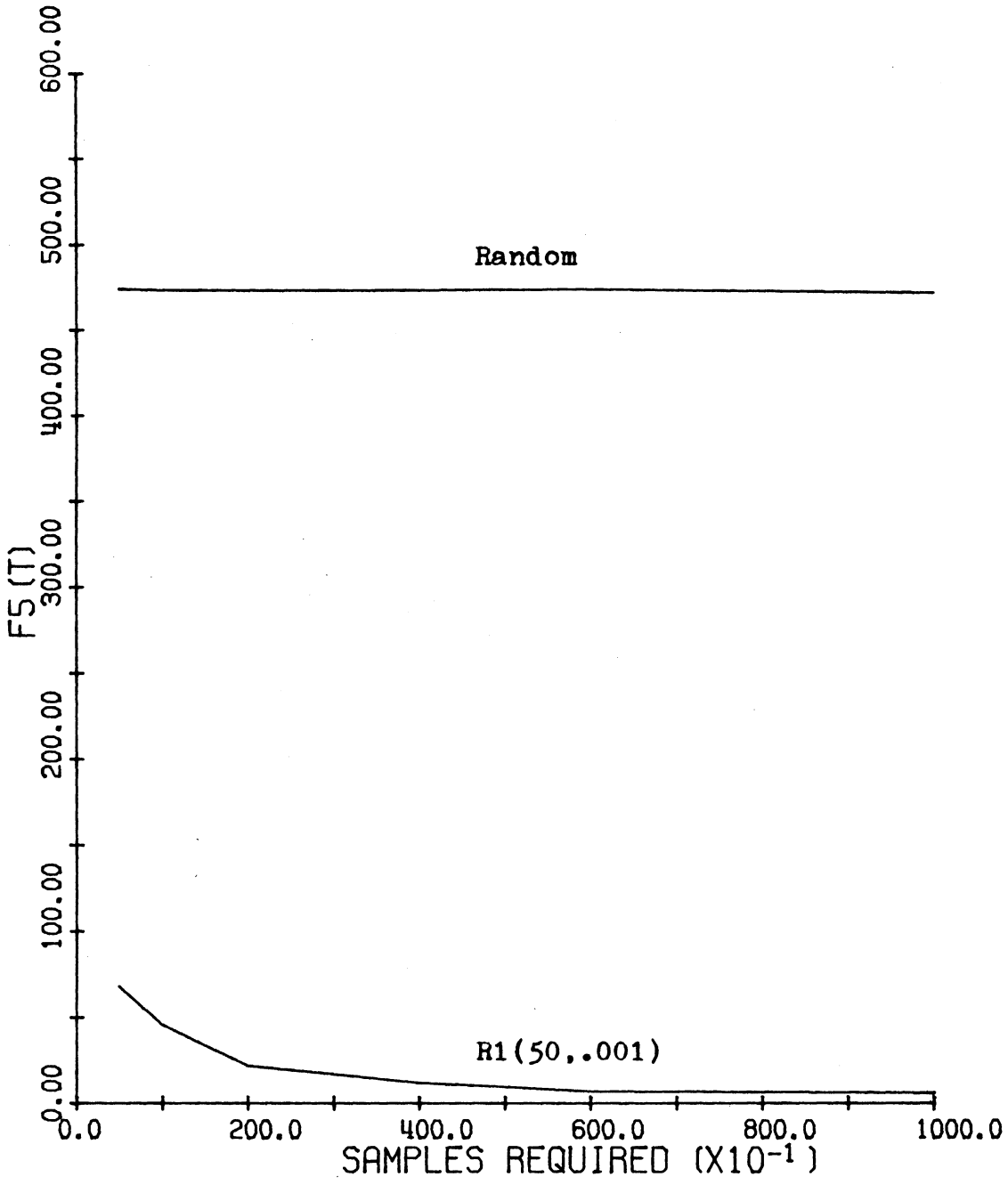


Figure C.5b: On-line performance curve for plan R1 on test function F5.

performance rating over E. This is computed for various values of T below:

$X_E^*(T)$

T	R1	Random
500	14.83	25.93
1000	9.24	18.34
2000	5.45	13.97
4000	3.18	11.10
6000	2.48	9.83
10000	1.29	8.86

$\bar{X}_E(T)$

T	R1	Random
500	105.92	147.66
1000	83.08	147.77
2000	63.12	147.57
4000	48.30	147.59
6000	38.88	147.61
10000	27.62	147.55

So we see that R1 is definitely more robust on E than random search is.

BIBLIOGRAPHY

Aigner, D. J. 1968. Principles of Statistical Decision Making. New York: MacMillan.

Bagley, J. D. 1967. The behavior of adaptive systems which employ genetic and correlation algorithms. Doctoral Thesis, Department of Computer and Communication Sciences, University of Michigan.

Bauer, M. J. 1974. A simulation approach to the design of dynamic feedback scheduling algorithms for time-shared computer systems. Simuletter, ACM SIGSIM Quarterly. 514: 23-31.

Bellman, R. 1959. Adaptive Control Processes: A Guided Tour. Princeton University Press.

Bosworth, J. L.; Foo, N.; and Zeigler, B. P. 1972. Comparison of genetic algorithms with conjugate gradient methods. University of Michigan Technical Report No. 00312-1-T.

Bremermann, H. 1970. A method of unconstrained global optimization. Mathematical Biosciences. 9:1-15.

Brent, R. P. 1971. Algorithms for finding zeros and extrema of functions without calculating derivatives. Doctoral Thesis, Computer Science Department, Stanford.

Cavicchio, D. J. Jr. 1970. Adaptive search using simulated evolution. Doctoral Thesis, Department of Computer and Communication Sciences, University of Michigan.

Crow, J. F., and Kimura, M. 1970. An Introduction to Population Genetics Theory. Harper & Row.

Feller, W. 1950. An Introduction to Probability Theory and its Applications, Volume I. New York: John Wiley & Sons.

Fletcher, R. and Powell, M. J. D. 1963. A rapidly convergent descent method for minimization. The Computer Journal. 6: 163.

Fletcher, R. and Reeves, C. M. 1964. Function minimization by conjugate gradients. The Computer Journal. 7: 149.

Fletcher, R. 1970. A new approach to variable metric algorithms. The Computer Journal. 13: 317.

Foo, N. Y. and Bosworth, J. L. 1972. Algebraic, geometric, and stochastic aspects of genetic operators. University of Michigan Technical Report No. 003120-2-T.

Frantz, D. R. 1972. Non-linearities in genetic adaptive search. Doctoral Thesis, Department of Computer and Communication Sciences, University of Michigan.

Hill, J. D. 1969. A search technique for multimodal surfaces. IEEE Transactions on System Science and Cybernetics, SSC-3,1.

Hogg, R. V. and Craig, A. T. 1965. Introduction to Mathematical Statistics. New York: MacMillan.

Holland, J. H. 1962. Outline for a logical theory of adaptive systems. Journal of the Association for Computing Machinery (ACM). 3: 297-314.

_____. 1967. Nonlinear environments permitting efficient adaptation. Computer and Information Sciences-II. New York: Academic Press. 147-164.

_____. 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press.

Hollstien, R. B. 1971. Artificial genetic adaptation in computer control systems. Doctoral Thesis, Department of Computer and Communication Sciences, University of Michigan.

Howard, R. A. 1971. Dynamic Probabilistic Systems, Volume I. New York: John Wiley & Sons.

Huang, H. Y. 1970. Unified approach to quadratically convergent algorithms for function minimization. Journal of Optimization Theory and Applications. 5: 405.

Jacoby, S. L. S.; Kowalik, J. S.; and Pizzo, J. T. 1972. Iterative Methods for Non-linear Optimization Problems. Englewood Cliffs, New Jersey: Prentice-Hall.

John, P. W. M. 1971. Statistical Design and Analysis of Experiments. New York: MacMillan.

Marsaglia, G. 1968. Random numbers fall mainly in the planes. Proc. Nat. Acad. Sci. 61,1: 25-28.

Mettler, L. E. and Gregg, T. G. 1969. Population Genetics and Evolution. Foundations of Modern Genetics Series. Prentice-Hall.

Powell, M. J. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal. 7: 155.

Rastrigin, L. A. 1960. External control by the method of random scanning. Automatika i Telemakhanika. 21: 1264-1271.

Rosenbrock, H. H. 1960. An automatic method for finding the greatest or least value of a function. The Computer Journal. 3: 175.

Schumer, M. A. and Steiglitz, K. 1968. Adaptive step size random search. IEEE Transactions of Automatic Control. AC-13: 270.

Shekel, J. 1971. Test functions for multimodal search techniques. Fifth Annual Princeton Conference on Information Science and Systems.

Summerville, D. M. Y. 1958. An Introduction to the Geometry of N Dimensions. New York: Dover.

Sworder, D. 1966. Optimal Adaptive Control Systems. New York: Academic Press.

Tausworthe, R. C. 1965. Random numbers generated by linear recurrence modulo two. Math. Comput. 19. 90: 201-209.

Toothill, J. P. R.,; Robinson, W. D.; and Adams, A. G. 1971. The runs up-and-down performance of Tausworthe pseudo-random number generators. J. ACM 18. 3: 381-399.

Toothill, J. P. R.; Robinson, W. D.; and Eagle, D. J. 1973. An asymptotically random Tausworthe sequence. J. ACM 20. 3: 469-481.

Tsypkin, Y. Z. 1971. Adaptation and Learning in Automatic Systems. New York: Academic Press.

Wilde, D. J. and Beightler, C. S. 1969. Foundations of Optimization. Englewood Cliffs, New Jersey: Prentice-Hall.

Yahowitz, S. J. 1969. Mathematics of Adaptive Control Processes. New York: American Elsevier.

Zeigler, B. P; Bosworth, J. L.; and Bethke, A. D. 1973. Noisy function optimization by genetic algorithms and conjugate gradient methods. Logic of Computers Technical Report No. 143, University of Michigan.