Theory and application of learning automata.

MACKIE, N.J.

1980

THEORY AND APPLICATIONS OF LEARNING AUTOMATA

by

Neil James Mackie

Abstract

Although the theoretical performance of many learning automata has been considered, the practical operation of these automata has received far less attention. This work starts with the construction of two action Tsetlin and Krylov automata. The performance of these automata has been measured in stationary and non-stationary environments. The operation of a hierarchical automaton controlling the memory size of a Tsetlin automaton is also investigated.

Two new automata are proposed with the aim of avoiding the operational disadvantages of the Tsetlin automaton. These automata have been tested using a computer simulation and in addition theoretical performance results have been calculated and compared with results for Tsetlin, Krylov and Lri automata.

A model of a non-autonomous environment is simulated and its operation analysed theoretically. A more accurate model is analysed and its operation with a Lri automaton examined and compared to theoretical predictions. The requirements for learning automata to operate successfully in non-autonomous environments is considered and it is shown that the Lrp and Lri automata do not converge to the optimum for a non-autonomous environment.

Three automata are proposed which are designed to operate in non-autonomous environments and their performances are compared to those of the Lrp and Lri automata.

The operation of automata in a hierarchical learning system and in cooperative and competitive games is considered. In these situations the performance of the new automata is compared to that of the Lrp and Lri automata.

Finally, two applications of learning automata are investigated. The first considers the Tsetlin allocation scheme, gives a modification which increases the performance and makes a comparison with a scheme using other learning automata. The second involves the selection of a processor in a multiprocessor computer system and compares a scheme using learning automata with a fixed scheduling discipline.

THEORY AND APPLICATION OF LEARNING AUTOMATA

by

NEIL JAMES MACKIE B.Sc

with
First Class Honours
in Electronic Engineering

A thesis submitted in partial fulfilment of the requirements of the Council for National Academic Awards for the degree of Doctor of Philosophy (Ph.D.)

School of Electronic and Electrical Engineering,
Robert Gordon's Institute of Technology,
Aberdeen.

October 1980

DECLARATION

I hereby declare that this thesis is a record of work undertaken by myself, that it has not been the subject of any previous application for a degree, and that all sources of information have been duly acknowledged.

In the course of this research the following were included in an approved programme of advanced studies:

(1) SRC Vacation School on 'Stochastic Processes in Control Systems' held at Warwick University, April 1978.

(2) 1st International Conference on 'Stochastic Computing and its Applications' held at the Institute National Polytechnique de Toulouse, France, 29 November - 1 December 1978.

(3) Joint SRC/IMC Symposium for 'Research Students in Control' held at the City University, London, 27-28 March 1980.

Neil J. Mackie
October 1980

THEORY AND APPLICATIONS OF LEARNING AUTOMATA

by

Neil James Mackie

Abstract

Although the theoretical performance of many learning automata has been considered, the practical operation of these automata has received far less attention. This work starts with the construction of two action Tsetlin and Krylov automata. The performance of these automata has been measured in stationary and non-stationary environments. The operation of a hierarchical automaton controlling the memory size of a Tsetlin automaton is also investigated.

Two new automata are proposed with the aim of avoiding the operational disadvantages of the Tsetlin automaton. These automata have been tested using a computer simulation and in addition theoretical performance results have been calculated and compared with results for Tsetlin, Krylov and Lri automata.

A model of a non-autonomous environment is simulated and its operation analysed theoretically. A more accurate model is analysed and its operation with a Lri automaton examined and compared to theoretical predictions. The requirements for learning automata to operate successfully in non-autonomous environments is considered and it is shown that the Lrp and Lri automata do not converge to the optimum for a non-autonomous environment.

Three automata are proposed which are designed to operate in non-autonomous environments and their performances are compared to those of the Lrp and Lri automata.

The operation of automata in a hierarchical learning system and in cooperative and competitive games is considered. In these situations the performance of the new automata is compared to that of the Lrp and Lri automata.

Finally, two applications of learning automata are investigated. The first considers the Tsetlin allocation scheme, gives a modification which increases the performance and makes a comparison with a scheme using other learning automata. The second involves the selection of a processor in a multiprocessor computer system and compares a scheme using learning automata with a fixed scheduling discipline.

Acknowledgements

I would like to express my thanks to my project supervisors Professor P Mars and Dr N Deans for their encouragement and guidance throughout the project.

I am also grateful to the technical staff of the School of Electronic and Electrical Engineering, RGIT and the staff of the Computer Services Unit for their assistance and cooperation. In particular I wish to thank Mr B Davidson for his assistance in producing the diagrams.

Acknowledgements also to Mr A Brown and Dr J Eades of the School of Electronic and Electrical Engineering, RGIT for allowing me to use their stochastic simulation program with graphic input.

Finally I would like to acknowledge the support of an SRC research studentship.

Abbreviations and Symbols

ADDIE	adaptive digital element
α	first parameter of Lrp automaton
α_i	action of learning automaton
a.p.	average penalty
β	second parameter of Lrp automaton
c_i	penalty probability associated with action i
cu	computational units
E()	expected value
ϵ	a small quantity
F	algorithm of learning automaton
f.c.f.s.	first come first served
f.s.d	fixed scheduling discipline
G	output function of learning automaton
γ	parameter of Lrp automaton
i j k	learning automaton states or matrix element subscripts
k_i	a positive constant which specifies c_i in Kumar's non-autonomous environment
$m_{i\,j}$	mean first passage time from state i to j
M	average penalty
N	memory size of Tsetlin automaton
p	probability of counting up in a random walk
Pa	action probability vector of learning automaton
ϕ	state vector of learning automaton
ϕ	parameter in linear or non-linear non-autonomous environment
PID	proportional intergral differential
PRBS	pseudo-random binary sequence

Ps	state probability vector of learning automaton
Pt(x)	transition matrix of learning automaton
q	probability of counting down in a random walk
r	number of actions available to a learning automaton
r_o	number of states in a random walk
s.s.	step size of probabilistic Tsetlin automaton
Θ	parameter in linear or non-linear non-autonomous environment
x	input to automaton from environment

THEORY AND APPLICATIONS OF LEARNING AUTOMATA

CONTENTS

THEORY AND APPLICATIONS OF LEARNING AUTOMATA

CHAPTER 1 REVIEW OF LEARNING AUTOMATA

Introduction

The aim of constructing a machine able to control a variety of processes with little or no prior knowledge of the process being required is an attractive proposition. The use of learning is a method of achieving these aims and has lead to the study of learning automata.

The aim of a learning automaton is to select an optimal action from a set of possible actions. An action, as selected by a learning automaton, can consist of a single action or a number of actions which are performed on an environment. The environment responds to the action or actions with an output from a set of possible outputs which is probabilistically related to the action of the automaton. The automaton in turn learns by using the output of the environment to change its internal state prior to selecting another action. The configuration of automaton and environment is shown in Figure 1.1.

The study of learning automata involves determining the characteristics of learning automata and the environments with which they will be used. This enables automata to be selected to suit different types of environment and allows the performance of automata to be evaluated, compared and hopefully improved. This chapter deals in general with the different types of automaton and environment while the other chapters deal with particular automata and particular environments and types of environment.

The Environment

The process, system or medium in which the learning automaton operates is termed the environment. The environment is defined by the triple (α, C, X) where α represents the input set to the environment in the form of an action and X represents the output set. The environment is assumed to be stochastic so that in response to an input α_i, it is possible to generate any of the elements of the output set according to c_i, an element of C, the penalty probability set.

There are three different schemes for the output of the environment, termed the P, Q and S models. In the S model, the output of the environment can have a continuous range of values in the interval [0,1]. In the Q model, the output can have one of a finite number of values in the interval [0,1]. However it is the P model which is most widely used with learning automata and which will be used in the chapters that follow. In this model there are only two output values, namely 0 representing a reward and 1 representing a penalty. At time t=n c_i is defined as

$$c_i(n) = \text{Probability}(\, x(n)=1 \mid \alpha(n)=\alpha_i \,) \qquad (1.1)$$

Thus c_i represents the probability of a penalty being output in response to input α_i while the probability of a reward is $1-c_i$. The P model has the advantage of simplicity when environments or learning automata are being investigated either through theoretical analysis or practical observation.

The Learning Automaton

Although a learning automaton requires little a priori knowledge about an environment, some information is required. An automaton must know the number of allowable actions for the environment or a number greater than the number of allowable actions. This is so that the actions of the automaton can be matched to the actions of the environment, with any extra automaton actions being made dummies with a penalty probability of unity. The automaton must know the form of the environment output in terms of P, Q or S models. Finally, the operation of the environment and automaton must be synchronised so that action and feedback follow each other in the correct order. Both the automaton and the environment are assumed to operate in discrete time with the input to the environment $\alpha(n)$ being followed by output $x(n)$ to the automaton which after its internal operations produces $\alpha(n+1)$. Apart from the above information, a learning automaton should be able to converge towards selecting the optimal action of the environment by working from an initial condition where each action is regarded as being equally favourable. The information the automaton uses to select its actions is the favorable (reward) or unfavorable (penalty) responses made by the environment to its past actions.

A learning automaton can be described by the quintuple (X,α,ϕ,F,G) and falls within the classification known as the Mealy model [1,2]. The input set X has the allowable inputs to the automaton as its elements. For an automaton operating with a P model environment, the set will have two elements $x_1 = 0$ and $x_2 = 1$. The set α is the output set of the automaton which has as its elements the actions the automaton can take. The set ϕ is the set of states of the automaton. The operation of the automaton is defined by its algorithm F which

relates the state of the automaton to its next state

$$F(\phi(n),x(n)) \rightarrow \phi(n+1) \qquad (1.2)$$

F can be a deterministic or a stochastic function and defines a set of transition matrices, one for each element of X allowing equation (1.2) to be written as

$$Pt(x)\ \phi(n) = \phi(n+1) \qquad (1.3)$$

If the transition matrices have only 0 or 1 as their elements the automaton is called deterministic while if any of the elements are probabilities the automaton is called stochastic. If the elements of the transition matrices are constants the automaton is described as having a fixed structure but if any of the elements is a variable the automaton has a variable structure. G is the output function of the automaton which relates the state of the automaton to the output,

$$G(\phi(n)) \rightarrow \alpha(n) \qquad (1.4)$$

G may be deterministic or stochastic.

There are two more quantities which are often used in describing automata, namely Ps(n) and Pa(n). Ps(n) is the state probability vector and its elements are defined as

$$\text{probability}(\phi(n) = \phi_i) = ps_i(n) \qquad (1.5)$$

the probability that the automaton occupies state ϕ_i at time n. Pa(n) is the action probability vector and its elements are defined as

$$\text{probability}(\alpha(n) = \alpha_i) = pa_i(n) \qquad (1.6)$$

the probability that the action of the automaton will be action α_i at time n.

Types of Learning Automata

A large number of learning automata have been proposed and investigated [3,4,5]. Table 1.1 gives a list of those commonly mentioned in the literature on the subject as well as automata which are of particular interest in later chapters. This list has been divided into four types containing automata which have similarities in the way they operate.

The automata included in Type 1 are all variable structure automata. These automata are best described by their algorithm and best observed via the action probability vector Pa. The naming of these automata is based on the algorithm so that if the algorithm is a linear equation the automaton has an L as the start of its name. If the algorithm is non-linear the first letter is an N while H denotes a hybrid algorithm. The subscripts which follow indicate whether the automaton changes state in response to a reward (r), a penalty (p) or remains inactive (i), while (w) indicates a weighted response.

Although the Type 1 automata are a more recent development than the Type 2 automata, recent investigations have dealt far more with Type 1 automata than Type 2. In particular the Lrp [6] and Lri [7,8] have been the most common automata for study. As these two automata are used in later chapters the Lrp and Lri automata will be described here.

The operation of the Lrp automaton is described in terms of its algorithm as

$$pa_{j \neq i}(n+1) = \alpha pa_j(n) \qquad (1.7)$$

$$pa_i(n+1) = 1 - \Sigma_{j \neq i}\, pa_j(n+1) \qquad (1.8)$$

in response to a reward after action α_i while

$$pa_i(n+1) = \beta pa_i(n) \qquad (1.9)$$

$$pa_{j \neq i}(n+1) = pa_j(n) + (1-\beta)/(r-1)\, pa_i(n) \qquad (1.10)$$

in response to a penalty after action α_i, where α and β are in the interval (0,1).

From equations (1.7)-(1.10) it can be seen that α controls the operation of the automaton in response to a reward while β controls the operation in response to a penalty. Often α and β are combined to give a third parameter γ defined as

$$\gamma = (1-\alpha)/(1-\beta) \qquad (1.11)$$

By making $\beta=1$ the action probability vector does not change in response to a penalty and the Lrp becomes the Lri automaton. Thus the Lri automaton is just a special case of the Lrp automaton with $\beta=1$ but the performance of the Lri automaton is sufficiently different from that of the Lrp automaton for the Lri automaton to be refered to as a distinct automaton.

While the performance of the rest of the automata in Type 1 has been studied [9,10,11,12,13,14], none have had the consistent performance of the Lrp and Lri automata for all penalty probabilities.

The automata included in Type 2 are all fixed structure automata with a deterministic or stochastic algorithm and a deterministic output function. Because of their deterministic output function each state is associated with only one output and these automata can be classed as Moore models [1,2,20]. These automata are best described by a graph showing the state transitions in response to a penalty and reward. Figure 2.1 shows the graph of the Tsetlin automaton as an example. The capacity for changing the performance of these automata is limited as the algorithm is fixed, only the number of states in the automaton is variable.

The first of the Type 2 automata to be studied was the Tsetlin [15] while the automata of Krylov [16], Krinskii [17] and Ponomarev [18] followed the lead given by Tsetlin by devising automata similar to Tsetlin's but with modifications designed to improve performance. All these automata have a series of states corresponding to a single action joined to other series of states corresponding to different actions. The difference between the automata is in the way they attempt to ensure the automaton stays in states corresponding to the optimal action. The $\mathcal{Q}$ model is a modification by Tsetlin to his own multi-action automaton where instead of the action of the automaton changing in a deterministic manner it changes stochastically. These automata have been studied by Langholz [20,21,22] and the G2n,2 has been studied by Narendra et al [19] but nothing further will be said here as the Tsetlin and Krylov automata are studied in Chapter 2.

Although the Type 3 automata are based on the Tsetlin automaton they have a variable structure and so are similar to Type 1, however the Type 1 automata have a stochastic output function while the Type 3 automata have a deterministic output. The Type 3 automata are investigated in Chapter 3.

Although the automata included in Type 4 have a variable structure with either deterministic or stochastic output functions [23] they are not grouped with the Type 1 automata because of their different approach to learning. The Type 4 automata use the output of the environment to estimate the elements of the set C. The automata then generally select the action corresponding to the lowest estimate which should be the optimal action of the environment. These automata have not been as widely studied as Types 1 and 2 though some work has been done by Coutts [24,25].

Type 5 automata are fixed structure automata with deterministic algorithms and stochastic output functions. They are similar to Type 2 automata, consisting of a series of states but cannot be classed as Type 2 because of their stochastic output functions which give them some of the advantages of Type 1 automata. These automata are dealt with in more detail in Chapter 5.

Synthesis of Learning Automata

Using digital techniques, learning automata can be readily and economically developed to run at high speed and automata have been built using this method, for example the Tsetlin and Krylov automata in Chapter 2. However some automata are too complex to be easily synthesised this way and in experimentation, where comparisons have to be made between different automata, more flexibility is required. Microprocessors have been used to provide this without a proliferation of hardware and a great speed penalty [26]. Where speed is not important and the greatest flexibility is required a mainframe computer has been used to simulate the learning automata as in Chapter 6.

Three functions that are required in the study of learning automata are the generation of random numbers, the generation of a random bit with a particular probability and the estimation of a probability from a random sequence. The methods used to obtain these functions can be implemented using any of the synthesis techniques mentioned above.

Random numbers can be obtained using independent segments of a pseudo-random binary sequence (PRBS) as a binary number. Such numbers will be uniformly distributed in the range $0 \rightarrow 2^N - 1$ where N is the number of bits on the binary number. A PRBS can be obtained from a shift register operating with feedback [27,28]. A register when

fitted with the appropriate feedback connections will progress through every possible register state except for the all zeros state in a pseudo-random manner before reentering its initial state. The output from a single bit of the register will be a PRBS with probability 0.5. To obtain a random bit e.g. to obtain a penalty or reward with a particular penalty probability, a random number in the range (0,1) is compared to the probability. If the random number is less than the probability the output is a penalty, otherwise it is a reward. Figure 1.2 shows a digital version of this which was used in Chapter 2 to produce a sequence of bits to represent a penalty probability.

In order to estimate the value of a probability an Adaptive Digital Element (ADDIE) is used [29,30] as shown in Figure 1.3. When operating in the steady state the ADDIE counter contains an estimate of the input probability. If the value in the counter is too low the feedback from the comparator is such that the counter counts up more often than it counts down while, if the value in the counter is too high the inverse is true. In reaction to a sudden change in the input, an ADDIE has a first order response with an error decaying exponentially with time. For a fast response, an ADDIE should have a small counter size but for the estimate of the input to have low variance, the counter size should be large. In practice, a compromise must be reached between these two conflicting criteria.

Stationary Environments and Measures of Performance for Learning Automata

Stationary environments have penalty probabilities c_i which are constant and do not vary with time. One measure of performance is the average penalty M output by the environment. An automaton is said to be expedient if

$$M < 1/r * \sum_{i = 1 \longrightarrow r} c_i \qquad (1.12)$$

that is, if the automaton operates so that it receives an average penalty lower than that which could be obtained by randomly selecting actions. An automaton is said to be absolutely expedient if

$$E(M(n+1) \mid Ps(n)) < M(n) \qquad (1.13)$$

that is, the average penalty can be expected to decrease as the automaton operates. If

$$M(n) < 1/r \sum_{i = 1 \longrightarrow r} c_i \qquad (1.14)$$

absolute expediency implies expediency and in stationary environments absolute expediency implies ϵ optimality [4]. An automaton is said to be optimal if

$$\lim_{n \longrightarrow \infty} M \rightarrow c_i \text{ (min)} \qquad (1.15)$$

and ϵ optimal if

$$\lim_{n \longrightarrow \infty} M \rightarrow c_i \text{ (min)} + \epsilon \qquad (1.16)$$

When operating in a stationary environment an automaton which was optimal would select the action corresponding to c_i (min) with probability 1 and so receive the lowest possible average penalty. An automaton described as having gone optimal is selecting an action with probability 1. In a stationary environment an automaton which is optimal or as near optimal as possible will receive the lowest average penalty and have the best performance.

Non-stationary Environments and Measures of Performance for Learning Automata

Non-stationary environments are defined as environments in which the characteristics of the penalty probabilities change with time. A switched environment is one in which one or more penalty probabilities change instantaneously from one value to another. There are deterministically switched environments in which the changes will

occur at regular time intervals and Markov switched environments in which, at regular time intervals, there is a probability that the penalty probabilities will change.

Unlike the stationary environment an optimal, or near optimal automaton operating in a non-stationary environment will in most cases not achieve the lowest average penalty. To be able to adapt to changes in the environment, an automaton must detect these changes by selecting non-optimal actions. A near optimal automaton will usually take a long time to adapt to changes in the environment and during this time the automaton will not be selecting the action corresponding to the lowest penalty probability. Thus a measure of performance introduced for use in non-stationary environments is the mean adjustment or switching time [31]. This is defined for an automaton selecting between two actions as the average number of epochs after a sudden change in the penalty probabilities from $c_1 < c_2$ to $c_1 > c_2$ until the action probability pa_1 changes from being less than pa_2 to being greater than pa_2.

Non-Autonomous Environments and Measures of Performance for Learning Automata

The environments described so far have been autonomous in that the penalty probabilities associated with an action were unaffected by the operation of an automaton. However in many practical situations the environment would be affected by the actions taken by the automaton. An example would be a telephone system where the available lines would depend on the routing of previous calls. Such environments are described as non-autonomous.

In autonomous environments, a single action corresponds to the minimum penalty probability and so the performance of an automaton can be judged from how nearly optimal the automaton is. In a non-autonomous environment, because the penalty probabilities vary as the action probabilities change, no single action probability can be described as best, and the task of the automaton changes from finding the best action to finding the best distribution of actions. In non-autonomous environments the degree of optimality is not an effective measure of performance and the average penalty received by the automaton is used.

A non-stationary non-autonomous environment is not one in which the penalty probabilities change but one in which the relationship between the penalty probabilities and the action probabilities change. In non-autonomous environments the mean switching time of an automaton is less important than in an autonomous environment. Poor mean switching times in a non-stationary autonomous environment are caused by an automaton selecting a single action and not detecting changes in the second. However in a non-autonomous environment, where the best policy is to select both actions in a particular ratio, changes are quickly detected.

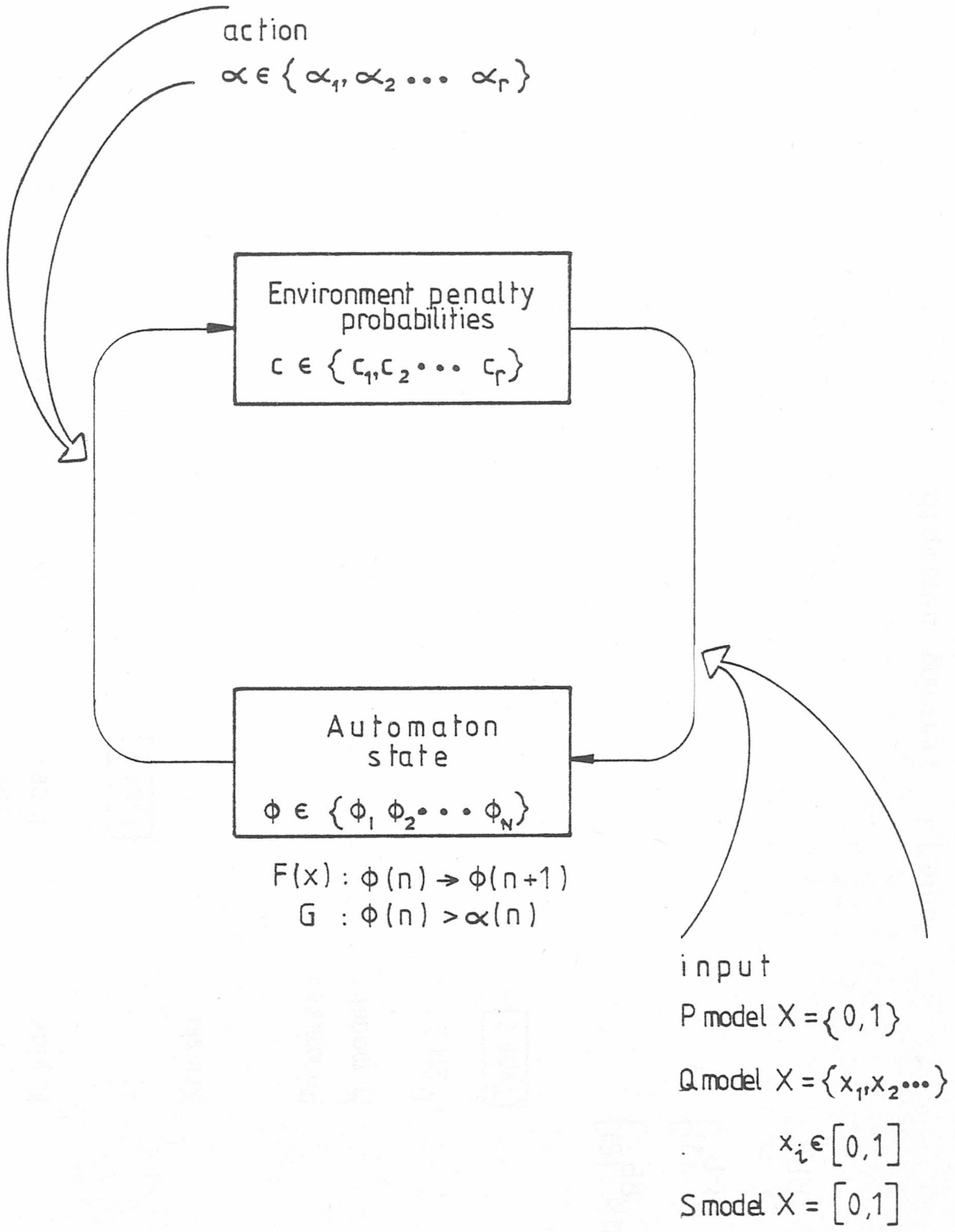


Figure 1-1 Automaton - environment feedback configuration

L_{RP}	Tsetlin	Modified Tsetlin	Sample Meon	T_{RP}
		Type 1		
L_{RI}	Krylov	Type 2	Modified Estimating	T_{IP}
L_{RR}		Type 3	Type 4	T_{RI}
L_{IP}	Krinski			Type 5
L_{PP}				
L_{R-WP}	Ponomarev			
	β model			
L_{WR-I}				
$N_{RP}^{(1)}$	$G_{2N,2}$			
•				
•	Type 2			
$N_{RP}^{(6)}$				
$H\left[N_{RP}^{(2)} N_{RP}^{(5)}\right]$				
$H\left[L_{RI}, N_{R-P}^{(2)}\right]$				
$H\left[L_{R-I}, \hat{L}_{RP}\right]$				
Type 1				

Table 1.1 Learning automata

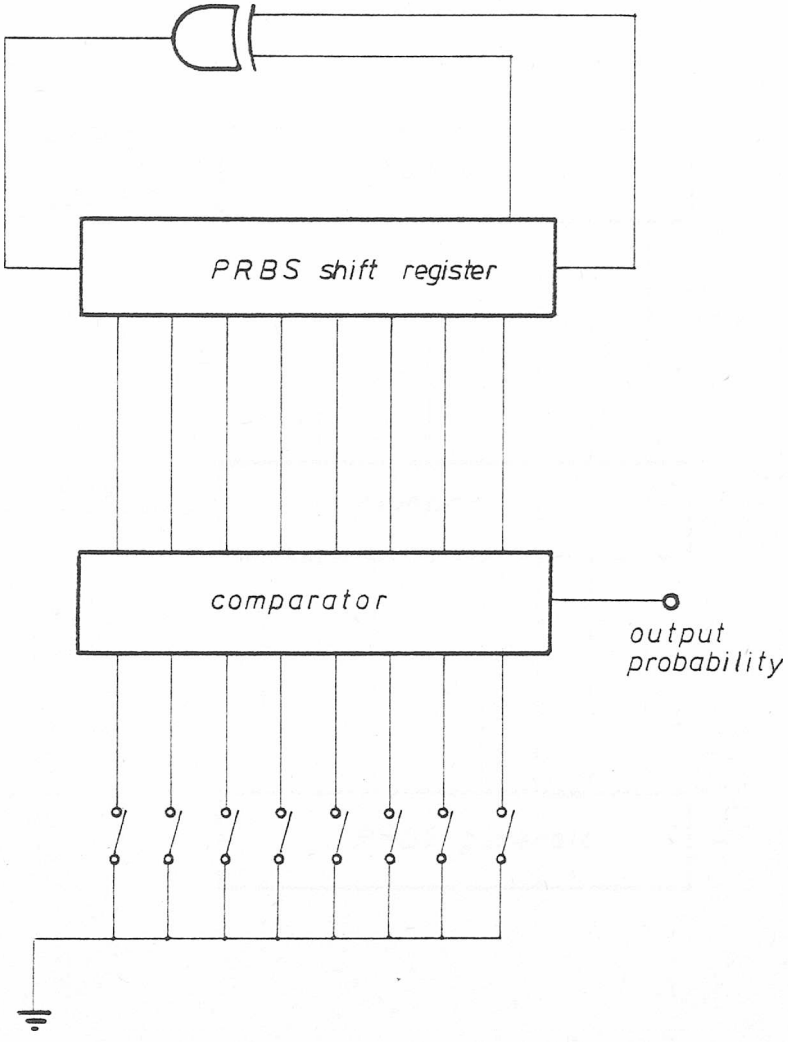


Figure 1.2 Schematic diagram of a PRBS probability generator

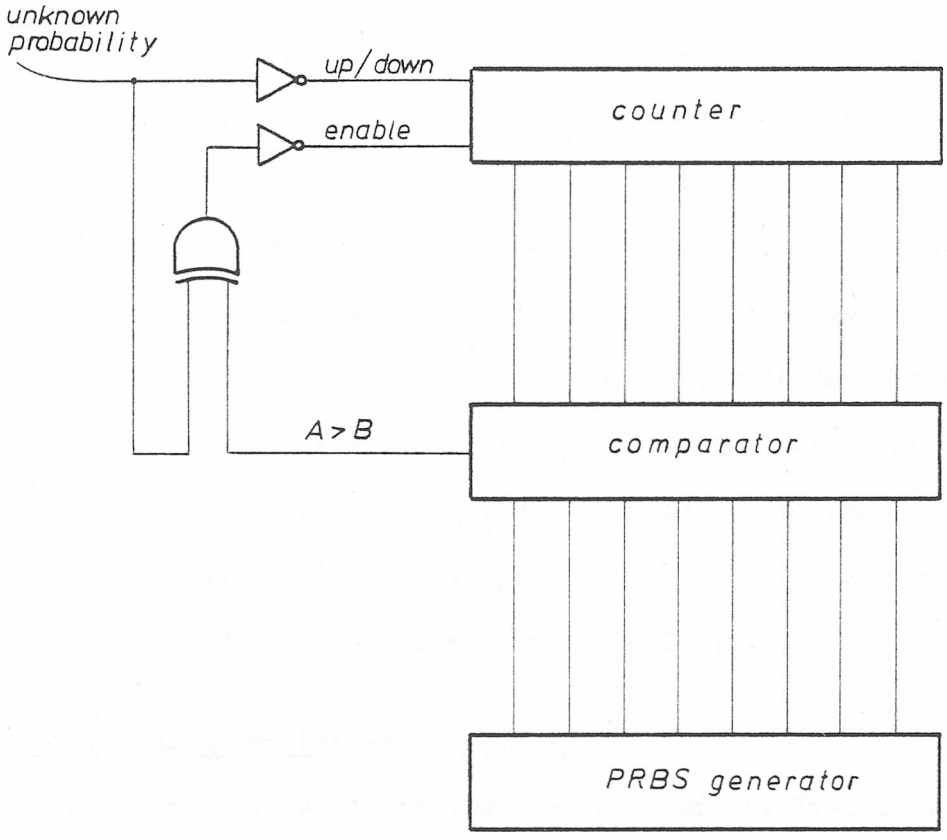


Figure 1.3 Schematic diagram of an ADDIE

CHAPTER 2 TSETLIN AND KRYLOV AUTOMATA

Tsetlin Automaton-Operation

In 1961 Tsetlin described a fixed structure learning automaton with deterministic algorithm and output function [15]. The Tsetlin automaton has been considered theoretically [32,33] and by simulation [19,30] but a Tsetlin automaton has not been built and its practical operation considered. The operation of the automaton is best explained with the aid of Figure 2.1. This shows a two action automaton with 2N states and a memory size of N. States 1 to N correspond to one action while states N+1 to 2N correspond to the other. When the automaton receives a penalty it moves towards states N and N+1 while, in response to a reward, the automaton moves towards end state 1 or 2N. Thus the automaton performs a random walk determined by the penalty probabilities of the environment with reflecting barriers beyond states 1 and 2N [34].

Tsetlin Automaton-Hardware Synthesis

In order to investigate the operation of the Tsetlin automaton the automaton was built using digital electronics. A block diagram of the circuitry used is shown in Figure 2.2 with more detailed circuit diagrams of the combinational logic used shown in Figures 2.3 and 2.4.

The heart of the automaton was a 12 bit binary counter allowing up to 4096 states or memory sizes up to 2048 with two actions. The most significant bit of the counter was taken as the action of the automaton and was connected directly to the environment. In response the environment output a penalty or reward according to the appropriate penalty probability. The output of the environment and

the action of the automaton were fed into combinational logic to convert them into an up/down control signal for the counter. The up/down control was in turn fed into more combinational logic with the state of the automaton and signals representing the memory size to provide a disable signal to prevent the counter exceeding the required memory size.

An environment was constructed using the method shown in Figure 1.3 using shift registers of length 23 and 31 bits generating maximal length sequences. The two penalty probabilities were then fed to the circuitry shown in Figure 2.5 which was used to produce a switched environment if required and to select the action probability corresponding to the automaton action. To monitor the operation of the automaton, the state of the 12 bit binary counter was converted to an analogue signal and displayed on an oscilloscope.

Tsetlin Automaton-Experimental Results

The performance of a Tsetlin automaton with a memory size of 2048 was investigated in both stationary and switched environments. Figure 2.6 shows learning curves for the automaton with action 2 the output and c_2 changed from 0 to 7/16 in steps of 1/16. The results show what is basically a linear movement from the central states of the automaton to the end state. For low penalty probabilities the movement to the end state is faster giving a shorter learning time.

Figure 2.7 shows the operation of the automaton in a deterministically switched environment with the central trace indicating the switching instants when c_1 was changed to the previous value of c_2 and c_2 changed to the previous value of c_1. Figure 2.7(a) shows the automaton operating with c_i's of 15/16 and 1/16. It can be seen that the automaton operates well and starts to move

towards states associated with the other action as soon as the environment switches. Figure 2.7(b) shows the same as Figure 2.7(a) initially but then the c_i 's are changed to 15/16 and 3/4. The performance of the automaton changes, it remains near its central states N and N+1 and frequently changes its output between action 1 and action 2. Figure 2.7(c) also shows the same as Figure 2.7(a) initially but the c_i 's are then changed to 3/16 and 1/16. In this case the operation of the automaton also changes. The automaton operates poorly as its action remains the same regardless of the changes in the environment.

The results shown in Figure 2.7 dèmonstrate that the operation of the Tsetlin automaton will fall into one of three modes depending on the environment. If the c_i 's are about the value of 0.5, one action will tend to make the automaton move towards states associated with the other action, while the other action will tend to make the automaton move towards the corresponding end state. Thus one action is stable while the other is unstable and the automaton works well. If the c_i 's are both greater than 0.5, both actions will tend to make the automaton move towards states associated with the other action. Thus both states are unstable, the automaton moves between states N and N+1 frequently and works poorly. If the c_i 's are both less than 0.5, both actions will tend to make the automaton move towards the end state associated with that action. Thus both actions are stable, with the automaton only moving from one action to another due to variance in the penalty probability causing it to be temporarily greater than 0.5 over a long enough time to allow the automaton to move from one action to the other. If the largest penalty probability is not close to 0.5, or if the memory size is large, the automaton can output the

wrong action for long periods of time and the automaton works poorly.

Tsetlin Automaton-Action Probability Results

Though the Tsetlin automaton is a deterministic automaton with a deterministic output function, over a long period of time a two action Tsetlin automaton will output both actions. If these are recorded the overall probability of selecting the optimal action can be calculated. It was found that slight differences were present between the measured and expected results which became apparent as the difference between the penalty probabilities was reduced or the penalty probabilities approached low values. It was found that the positioning of the connections from the individual bits of the PRBS shift registers affected the penalty probabilities. Rather than build a new random number generator the best positioning of the connections was selected and used for the later results.

Tsetlin Automaton-Mean Switching Time Results

Measurement were made of the mean switching time of the Tsetlin in switched environments with penalty probabilities equally spaced about 0.5. For the Tsetlin and Krylov automata the mean switching time can be defined as the average number of epochs, after a sudden reversal of the penalty probabilities till the first output of the correct action, assuming the automaton had rightly output the previously correct action immediately prior to the switch in the environment. Figures 2.8(a)-(g) show the mean switching time results for various penalty probabilities plotted against memory size with the corresponding theoretical results. In general the results are in good agreement but it can be seen that as the difference between the c_i's is reduced the measured results differ more from the theoretical results. This is

due to the deficiencies in the generation of the penalty probabilities.

Krylov Automaton-Operation

The Krylov automaton [16] was proposed as an automaton which became more nearly optimal as its memory size increased in any environment instead of only in environments with one or both c_i's less than or equal to 0.5 as for the Tsetlin automaton. The Krylov automaton is very similar to the Tsetlin automaton in that it has a series of states 1 to 2N, with states 1 to N being associated with one action and states N+1 to 2N being associated with the other. It is in the movement between states that the Krylov and Tsetlin automata differ. As Figure 2.9 shows, in response to a reward the Krylov automaton acts as the Tsetlin and moves deterministically towards an end state but, in response to a penalty, the automaton acts stochastically and moves either towards states N and N+1 or towards the end states with probability 0.5.

The operation of the Krylov automaton can be related to that of the Tsetlin automaton. If an automaton performs an action such that it receives a penalty with probability c_1 then

$$\text{penalty probability} = c_1$$

$$\text{reward probability} = 1-c_1$$

If a reward response is taken as a movement towards states 1 or 2N and if a penalty response is taken as a movement towards states N and N+1 then for the Krylov automaton

$$\text{penalty response probability} = c_1/2$$

$$\text{reward response probability} = (1-c_1)+(c_1/2) = 1-c_1/2$$

and for the Tsetlin automaton

$$\text{penalty response probability} = c_1$$

reward response probability = $1-c_1$

and a similar argument applies to c_2 .

Equating response probabilities it is seen that a Krylov automaton receiving penalty probabilities in the range [0,1] is equivalent to a Tsetlin automaton receiving penalty probabilities in the range [0,0.5]. However previous results showed that the Tsetlin automaton did not operate well with penalty probabilities which were both less than 0.5, and so it was expected that the Krylov automaton would not work well.

Krylov Automaton-Hardware Synthesis

Because of the similarity between the Krylov and Tsetlin automata the circuitry used in constructing the two automata was identical except for combinational logic block 1, as shown in Figure 2.2.

This circuitry, which is shown in Figure 2.10, instead of deterministically converting a penalty response from the environment into an up/down signal for the counter, as in the Tsetlin automaton, sampled a stochastic sequence of probability 0.5 and used this as the control signal for the counter.

Krylov Automaton-Experimental Results

The performance of the Krylov automaton was investigated whilst operating in both stationary and switched environments. Figure 2.11(a) shows a Krylov automaton of memory size 2045, initially with output action 1, operating in a switched environment with penalty probabilities of 0 and 15/16. As predicted the result is similar to a Tsetlin automaton working with both c_i 's less than 0.5 with the automaton action remaining unchanged even though the environment switches. This inability to change is a function of memory size. The

automaton has two stable actions, with the action corresponding to the lower c_i being more stable than the other and with stability increasing as the memory size increases. Variance in the penalty probabilities causes movement between the actions and the time spent in an action depends on its stability. Thus while both actions are stable, for small memory sizes, variance should cause movement between the actions with the automaton spending more time in the most stable action. This can be seen in Figure 2.11(b) which shows a Krylov automaton, with memory size of 8, and operating with $c_1 = 7/8$ and $c_2 = 5/8$ moving from states corresponding to c_2 to states corresponding to c_1, remaining in those states for a time and then moving back. Figure 2.11(c) shows a Krylov automaton, with memory size of 8, operating in a switched environment with c_i's of 3/4 and 5/8. Since when the switching trace is high the automaton trace should be low it can be seen that the automaton works poorly.

Tsetlin and Krylov Automata-Theoretical Action Probability Results

In order to calculate how optimal a Tsetlin or Krylov automaton is the steady state probabilities of the states of the automaton are required.

For a Tsetlin automaton if the environment is such that $c_2 = 1-c_1$, and if the automaton is not at an end state, when the input is action 1 the probability of the automaton counting up is c_1 and the probability of counting down is $1-c_1$ It is action 2 the probability of counting up is $1-c_2 = c_1$ and the probability of counting down is $c_2 = 1-c_1$. Thus for all states 2 to 2N-1 the probability of counting up is c_1 and the probability of counting down is $1-c_1$. For a random walk with reflecting boundaries [34] at 1 and a and with a probability of going up of p and of going down of q the

probability of being in state k after a long time is

$$(1-p/q) \; / \; (1-(p/q)^{r_o} \;) \; * \; (p/q)^{k-1} \qquad (2.1)$$

where r_o is the number of states in the random walk.

Thus for the Tsetlin automaton the steady state probability of state k is

$$(1-(c_1 \; /(1-c_1 \;)))/(1-(c_1 \; /(1-c_1 \;))^{2N})*(c_1 \; /(1-c_1 \;))^{k-1} \qquad (2.2)$$

where k = 1 -> 2N

If c_2 does not equal $1-c_1$ then the calculation of the steady state probabilities is more difficult. The method used to calculate the steady state probabilities is given in Appendix 1 and was used to calculate the results given in Chapter 3.

Figures 3.3(a)-(d) give the theoretical degree of optimality for the Tsetlin and Krylov automata against memory size for various environments with penalty probabilities about 0.5. Results for the Lrp automaton have been included as this automaton was used as a reference. The results show that the Tsetlin and Krylov automata become nearly optimal as the memory size approaches 10. The Krylov automaton is also more optimal than the Tsetlin for the same memory size.

Figures 3.5(a)-(f) show results for the Tsetlin automaton in environments where the penalty probabilities are not constrained about 0.5. For penalty probabilities both greater than 0.5 the optimality of the automaton levels out and does not increase to one as the memory size is increased. When one of the penalty probabilities falls below 0.5 this measure of performance begins to approach 1 as the memory size in increased. As the penalty probabilities are decreased further the optimality for a particular memory size increases.

Tsetlin and Krylov Automata-Theoretical Mean Switching Time Results

The method used to calculate the mean switching time of the Tsetlin automaton used for Figure 2.8 and the results in Chapter 3 is given in Appendix 2.

In a switched environment there is a probability that the automaton will not be selecting the action corresponding to the lowest penalty probability when the environment switches. When this occurs the switching time of the automaton is zero. In taking results for the graphs shown in Figure 2.8, switching times of zero were ignored. In order to have the calculated mean switching times correspond to the results the steady state probability vector was modified so that

$$\sum_{i = 1 \longrightarrow n} ps_i = 1$$

i.e. the automaton action is always correct immediately prior to the switch in the environment. This definition gives slightly longer mean switching times compared to the definition given in Chapter 1.

Figures 3.4(a)-(d) give mean switching time results for Krylov and Tsetlin automata in a variety of environments corresponding to those in Figures 3.3(a)-(d). Again the Lrp automaton has been included as a reference. It should be noted that the definition of mean switching time for the Lrp automaton is that given in Chapter 1 and differs slightly from that used for the Tsetlin and Krylov automata. Even for widely spaced penalty probabilities the Krylov automaton has very long switching times. As the difference between the penalty probabilities is reduced the performance of the Krylov automaton worsens dramatically. In Figures 3.6(a)-(f) which correspond to the environments of Figure 3.5 the results for the Krylov automaton have been omitted so the results for the Tsetlin automaton can be examined. For penalty probabilities above 0.5 where the optimality of the

Tsetlin automaton is relatively poor the mean switching times are low. For penalty probabilities below 0.5 where the optimality is high the mean switching times are high, so much so that results for the Tsetlin have been excluded from Figures 3.6(e)-(f).

The theoretical results for the Tsetlin and Krylov automata confirmed the conclusions drawn from the experimental work with the automata. The Krylov automaton has a near optimal performance in all environments but has switching times so large that its use is impractical. The Tsetlin automaton has relatively poor optimality in environments with high penalty probabilities compared to its performance with low penalty probabilities. However with low penalty probabilities the switching times of the Tsetlin automaton are high and it is only with penalty probabilities about 0.5 that the Tsetlin has a high degree of optimality and low mean switching times.

Tsetlin Automaton-Average Penalty

In his paper [15] Tsetlin considers the operation of his automaton in a Markov switched environment and derives an equation for finding the average penalty as

$$M=1/2-(a-1)^2/2 * \frac{\cosh(ny)-1}{2Nd/(1-2d)*((a+1)**2)*\cosh(Ny)+((a-1)**2)*\coth(y/2)*\sinh(Ny)} \quad (2.3)$$

where

$$\cosh(y)=((1+a)**2)/2a * (1-d)/(1-2d) -1$$

$$a=p/(1-p),\ c_1 = p,\ c_2 = 1-p$$

d = probability of environment switching

N = memory size

Figure 2.12 shows results from this equation which show there is a memory size which corresponds to a minimum average penalty and that

this memory size decreases as the switching rate increases. Figure 2.13 shows that as the penalty probabilities move toward 0.5 the best memory size increases while the minimum of the curve becomes less distinct.

Tsetlin Automaton-Average Penalty Results

Measurements were made of the average penalty received by the Tsetlin automaton in both deterministically and Markov switched environments. In order to produce a Markov switched environment the switching circuitry shown in Figure 2.5 was used, connected to a Markov switching clock. The switching clock was arranged to switch every time a penalty was present on a signal representing the switching probability.

Measurements were then taken of the measured average penalty of a Tsetlin automaton operating in a Markov switched environment with penalty probabilities of 1/4 and 3/4 and varying the memory size and switching rate. Figure 2.14 shows the measured results compared with theoretical results. The measured average penalties show there is a memory size which corresponds to a minimum average penalty but disagree in some cases with the theoretical results on the value of the memory size. The differences between measured and theoretical results increase as the switching rate increases until the measured results indicate that at very fast switching rates the automaton is receiving an average penalty greater than the mean of the two penalty probabilities. These differences are due to the deficiencies in the generation of the penalty probabilities but also indicate how sensitive the Tsetlin automaton is to the nature of the penalty probabilities, a factor which should be borne in mind if the automaton is used in real environments.

Figure 2.15 shows measurements of the average penalty but obtained with the automaton operating in a deterministically switched environment. Comparison with Figure 2.14 shows that the curves are steeper with the best memory size being more clearly defined. This is to be expected since a Markov switching rate is a mixture of a range of deterministic switching rates.

Optimal Memory Size Automaton-Criteria

Having shown that for a given switching rate there is an optimal memory size it was decided to build circuitry to automatically control the memory size in order to minimise the average penalty received by the Tsetlin automaton. This would create a hierarchical structure with a secondary automaton adjusting a parameter of the primary, Tsetlin automaton. Whilst the Tsetlin automaton would be operating in a switched environment, the secondary automaton would, if the switching rate remained constant, operate in a stationary environment. The secondary automaton would be working with penalty probabilities like those of Figure 2.14. These have a single global minimum with no local minima so stochastic hill-climbing methods could be used as an alternative to stochastic automata methods [4]. Since the curves of Figure 2.14 are relatively flat the automaton would have to be slow in order to distinguish between penalty probabilities which were near the same value. However, because the curves were flat an action which was non-optimal, but near the optimal value for memory size could be tolerated since the difference in average penalty between the two would be small. A non-optimal automaton was also favoured so that it could adjust the memory size to changing switching rates. At first sight it seemed that a multi-action automaton with an action corresponding to each particular memory size would be needed. This

would have required a large structure but it was realised that in the simple environment with no local minima a two action, deterministic, gradient following automaton could be used.

Optimal Memory Size Automaton-Operation

The automaton designed to control the memory size of the Tsetlin automaton was like those of type 4 in Table 1.1 in that it estimated the penalty probabilities, and selected an action on the basis of those estimates. It consisted of two counters, a comparator, a memory size counter and some control circuitry as shown in Figure 2.16. In operation the automaton measures the penalty probability, in this case the average penalty received by the Tsetlin automaton, at a memory size and the penalty probability at the next highest memory size. The automaton then compares the two measurements and, on the basis of which is the smaller, either increments or decrements the memory size counter by one and repeats the operation. In this way the automaton moves down the gradient of the average penalty curves towards the optimal memory size. The automaton can never output the optimal memory size with probability greater than 0.5 since comparisons will always be made with the memory sizes above and below the optimal value. Thus the automaton will respond to changes in the environment due to changes in the switching rate relatively quickly while the increased penalty probability caused by selecting the memory sizes about the optimal size is not great. The size of the measuring counters is a compromise between the smoothing effect required to obtain the average penalty received by the Tsetlin automaton over a number of switches in the environment, speed of operation and construction considerations. A value of 12 bits giving a counter size of 4096 was selected. The automaton was also limited to operate with

memory sizes in the range 1 to 16.

Optimal Memory Size Automaton-Results

For the optimal memory size automaton, measurements were made of the steady state probability of each memory size in both Markov and deterministically switched environments for penalty probabilities of 1/4 and 3/4. Figures 2.17(a)-(f) and 2.18(a)-(d) show these results with the optimal memory size shown as a solid line. It can be seen that the memory size favoured by the automaton changes with the switching rate though the most frequent memory size does not always correspond to the optimal size, there being a tendency to favour a higher memory size. This is because the gradients of the average penalty curves are steeper below the optimal memory size than above it. If the automaton is below the optimal memory size it will be forced back towards the optimal action relatively quickly whilst, if it is above the optimal memory size, the average penalty does not rise so steeply so the automaton will be forced back towards the optimal action more slowly.

Figure 2.17(b) shows a consequence of limiting the memory size to 16 which results in an increased probability of the higher memory sizes. If the range of memory sizes was larger, memory sizes above 16 would occasionally be selected but, because there is a limit of 16, the distribution of memory sizes is distorted, resulting in increased probability of states just below the limit.

Figures 2.18(a)-(d) show the results for deterministic switching. These are much more compressed because of the steeper gradients of the average penalty curves. Figures 2.19 (a)-(b) show more results obtained in a deterministically switched environment, this time for the speed of operation of the automaton starting initially at a memory

size of one. The results give the average number of measurements made before getting to the optimal memory size. The automaton is relatively slow but this is due to it having to try each memory size twice, e.g. in moving to memory size 6 the automaton would have to make at least 2 measurements at memory sizes 2,3,4 and 5.

Conclusions

Most learning automata have a high trade off between degree of optimality and mean switching time so that reducing the mean switching time also significantly reduces the optimality. The Tsetlin seems to provide good mean switching times and a high degree of optimality but with a severe limitation on the environment, the c_i 's having to be about 0.5. When operating in a switched environment with penalty probabilities about 0.5 the Tsetlin automaton does not have to sample the wrong state in order to determine whether the environment has switched or not. Because the penalty probabilities are about 0.5 when the switch occurs, a c_i which was less than 0.5 is now greater than 0.5 and the automaton moves towards states associated with the other action no matter the degree of optimality and so a good steady state performance does not imply a poor transient response as in most automata.

The Krylov automaton has been shown to operate for all penalty probabilities like the Tsetlin automaton with penalty probabilities less than 0.5. It works poorly in a switched environment, relying on a small memory size and variance in the penalty probabilities to cause movement between the actions.

The optimal memory size automaton was designed for a specific task and was made as simple as possible. The limitations of its design became apparent in operation as regards speed and degree of optimality but nevertheless it was found satisfactory in controlling the memory size of the Tsetlin automaton.

These investigations highlighted the desirable and undesirable characteristics if the Tsetlin automaton. The difference in operation of the Tsetlin and Krylov is small but the effect on performance is large. Having noticed these changes and their effect it was thought that an automaton could be developed that would retain the good qualities of the Tsetlin automaton but avoiding some of its disadvantages.

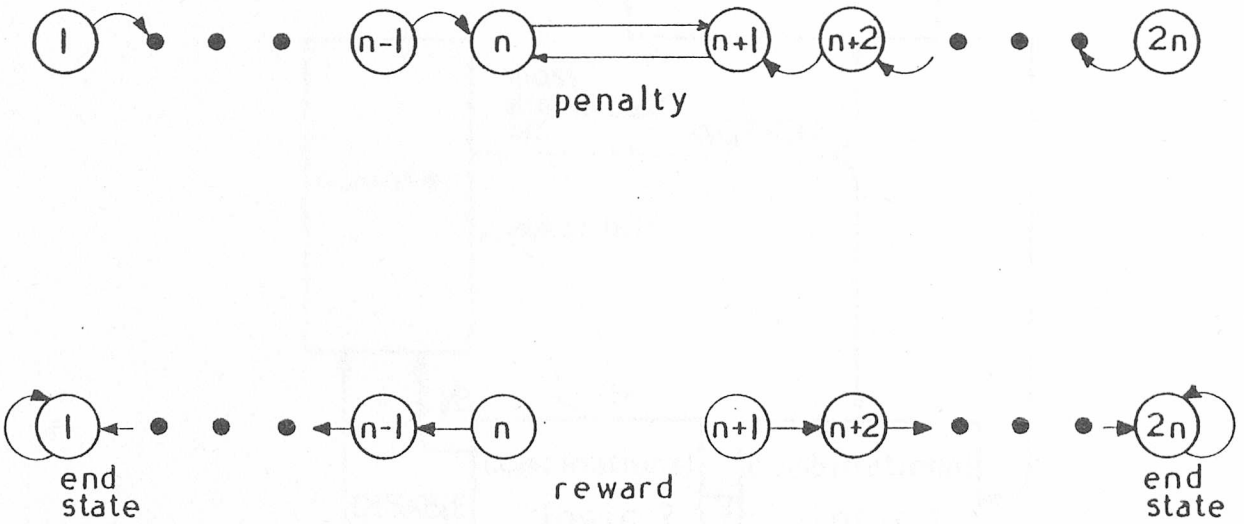


Figure 2.1 State diagram of Tsetlin automaton

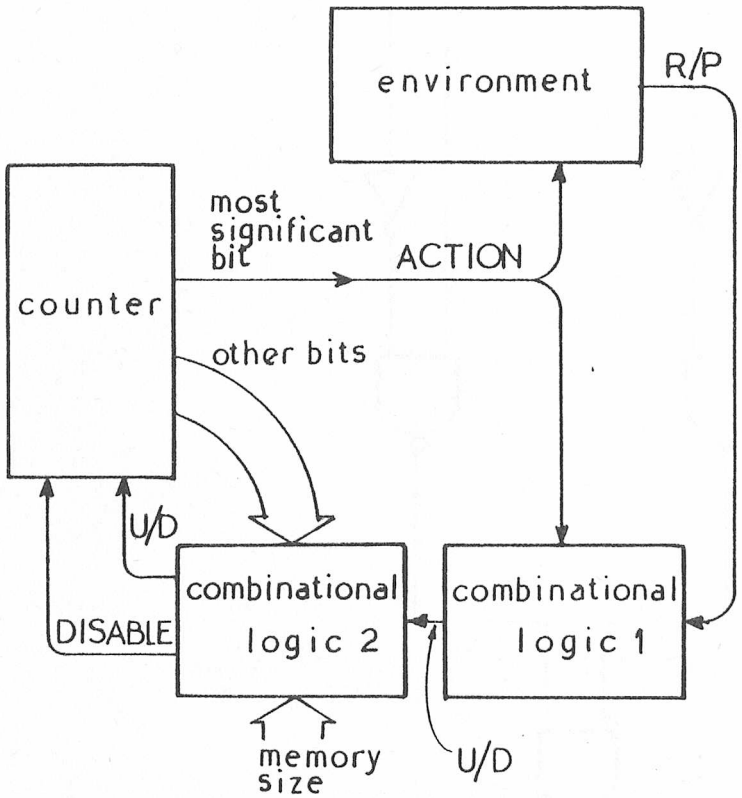


Figure 2.2 Schematic diagram of Tsetlin automaton

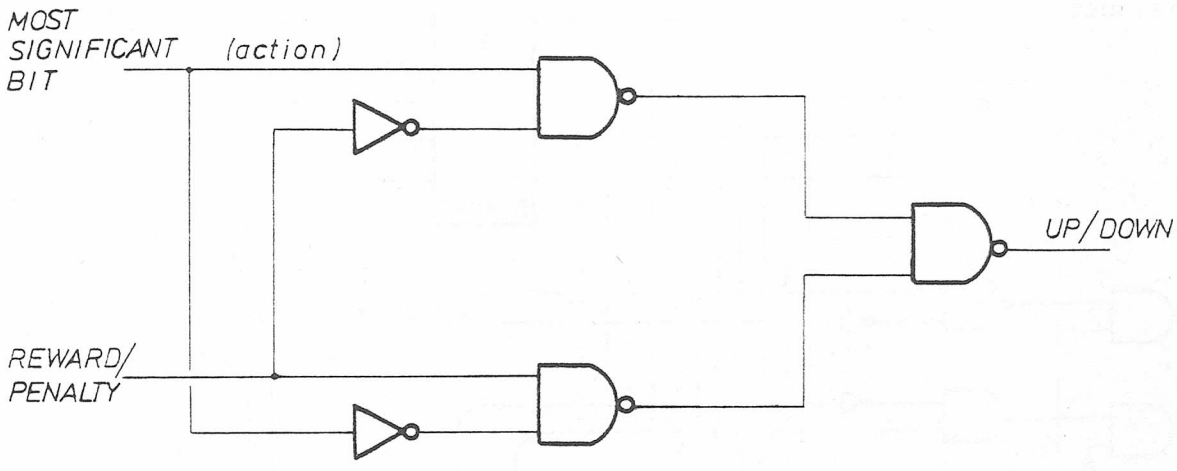


Figure 2.3 Circuit diagram of Tsetlin automaton - 1

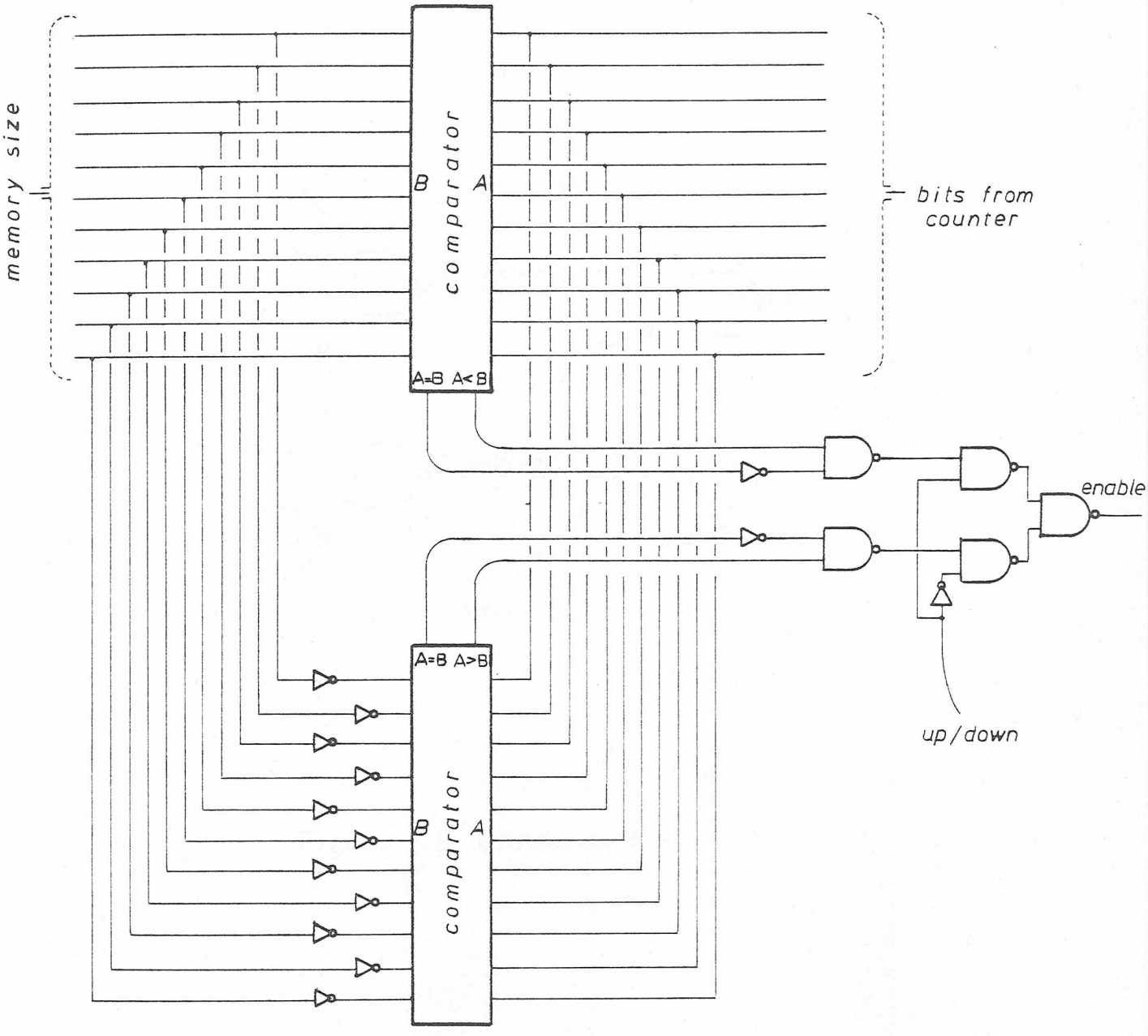


Figure 2.4 Circuit diagram of Tsetlin automaton - 2

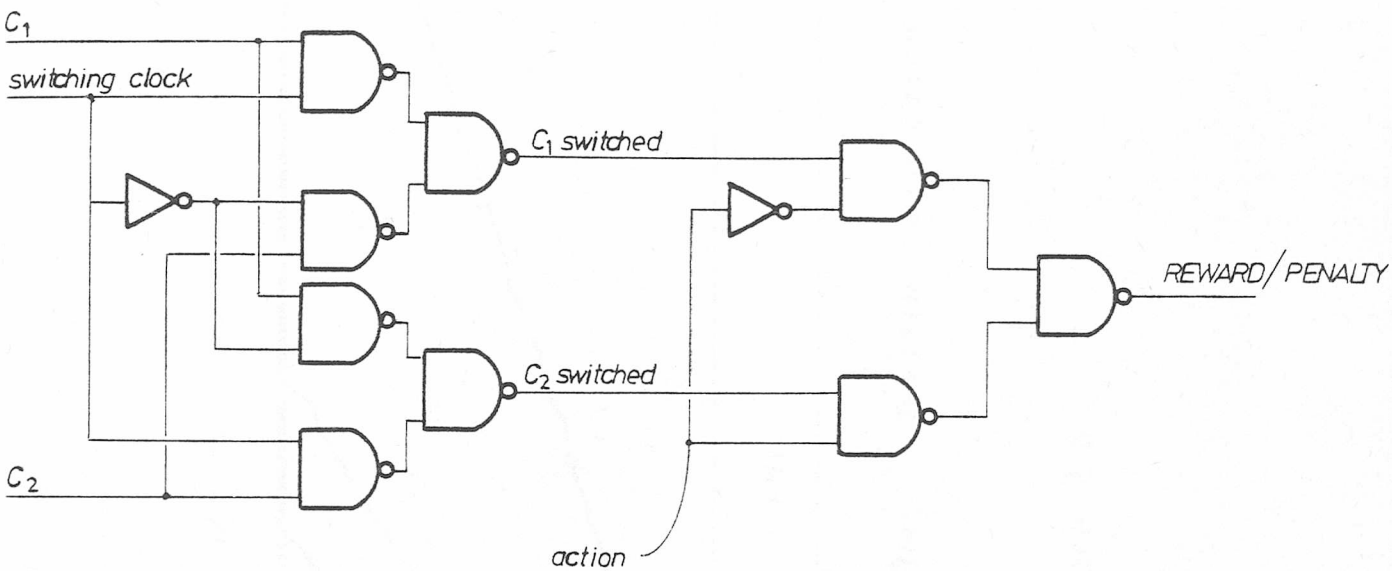
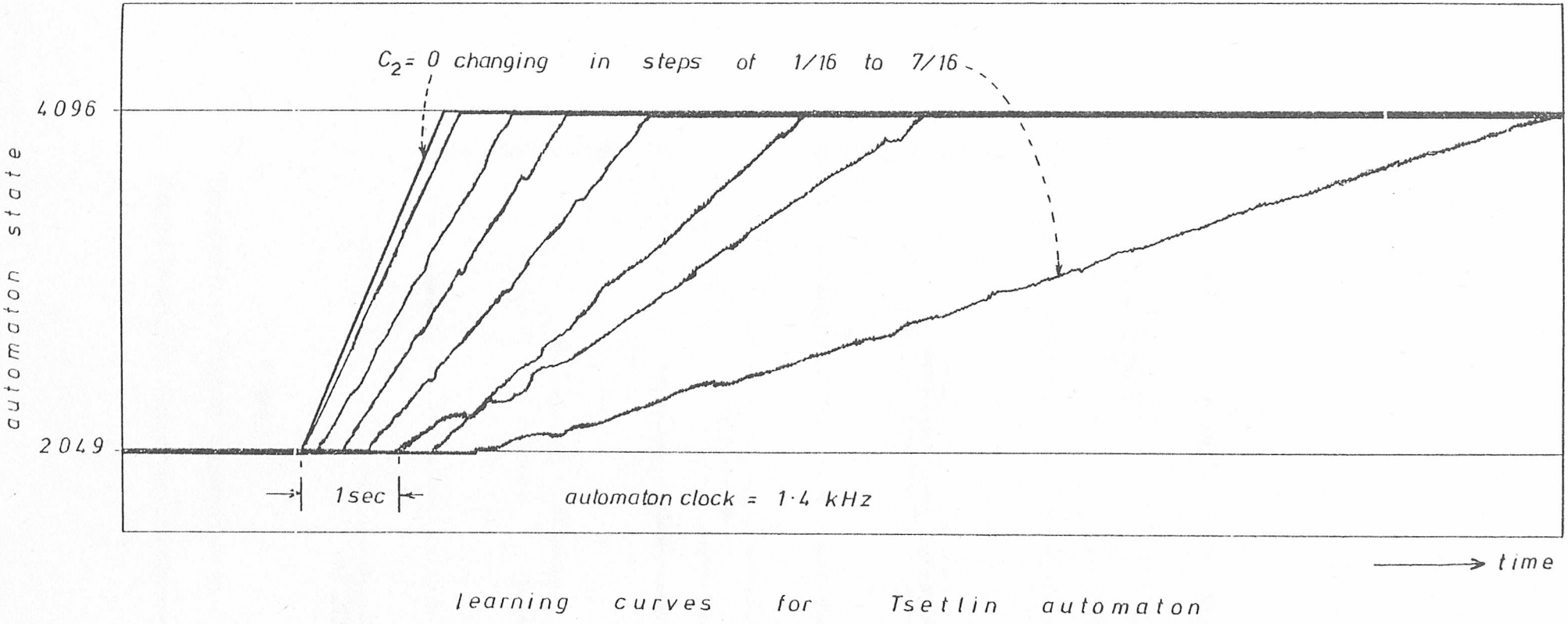


Figure 2.5 Environment switching



learning curves for Tsetlin automaton

Figure 2.6

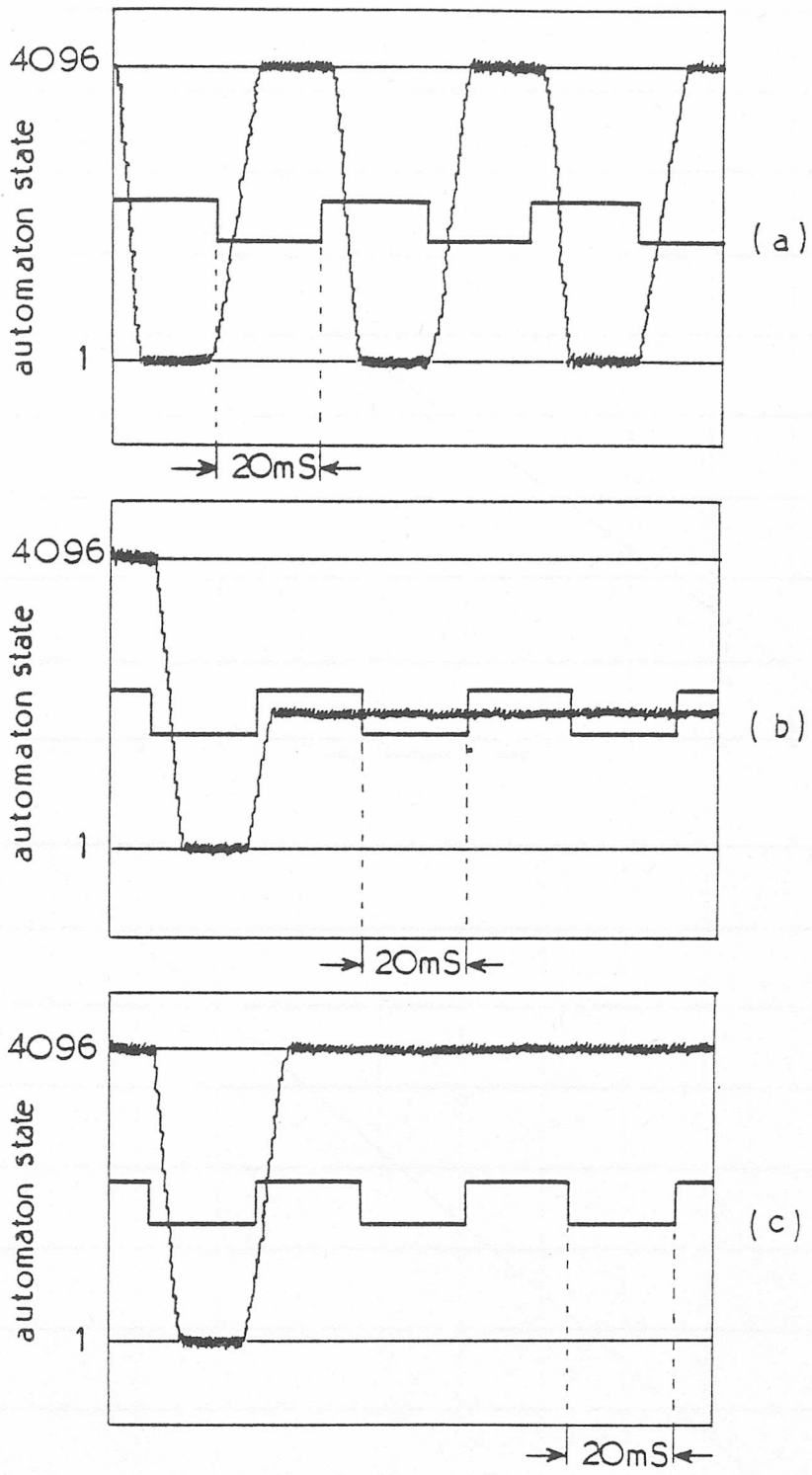
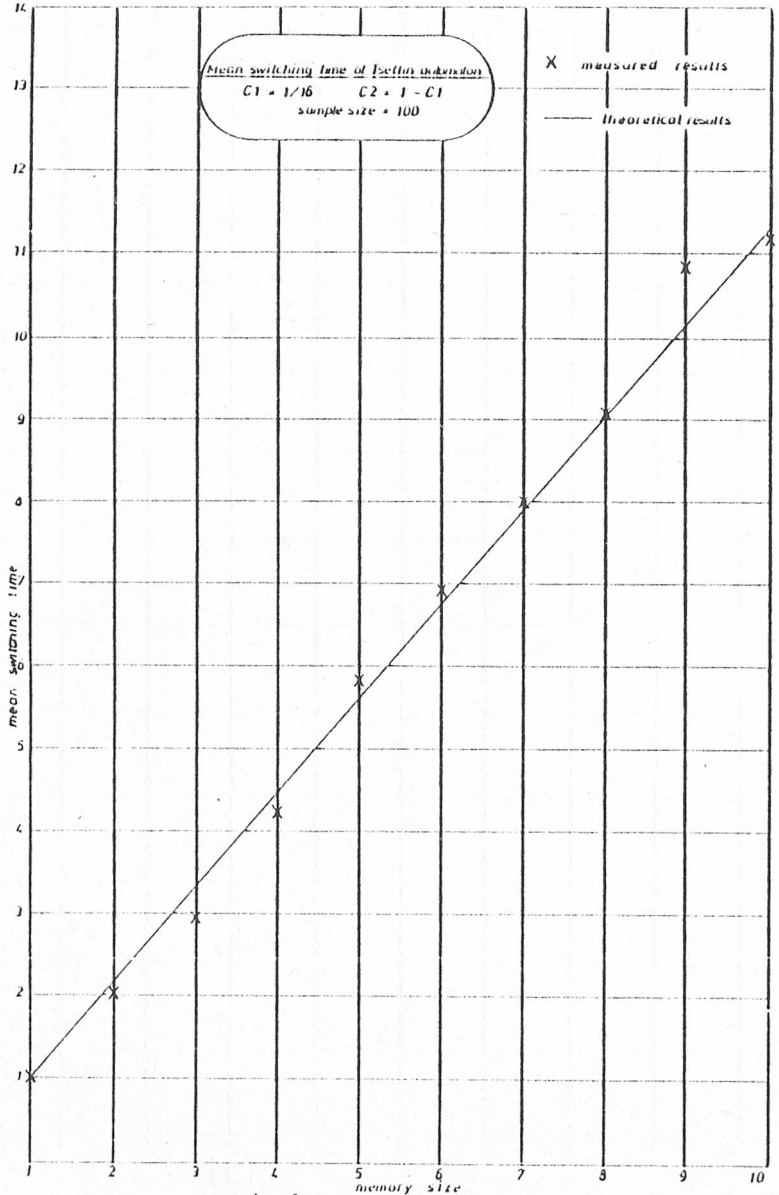


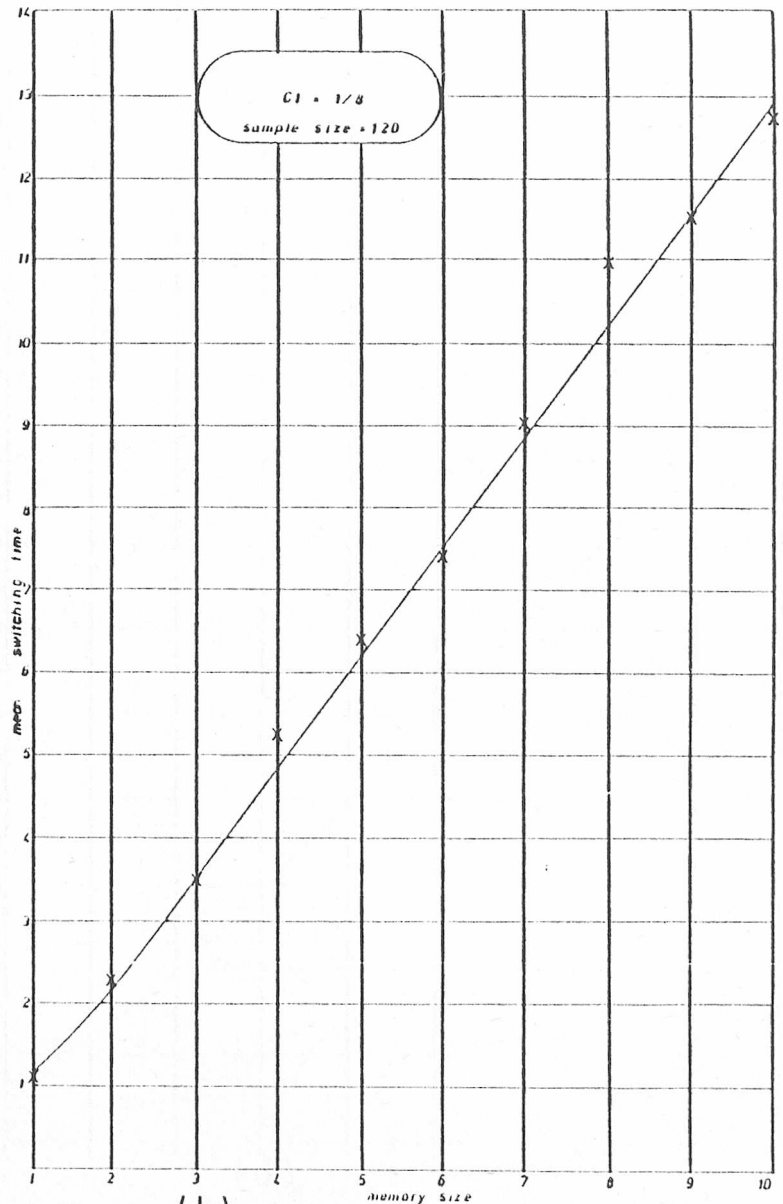
Figure 2·7

Operation of Tsetlin Automaton in

Deterministically Switched Environment

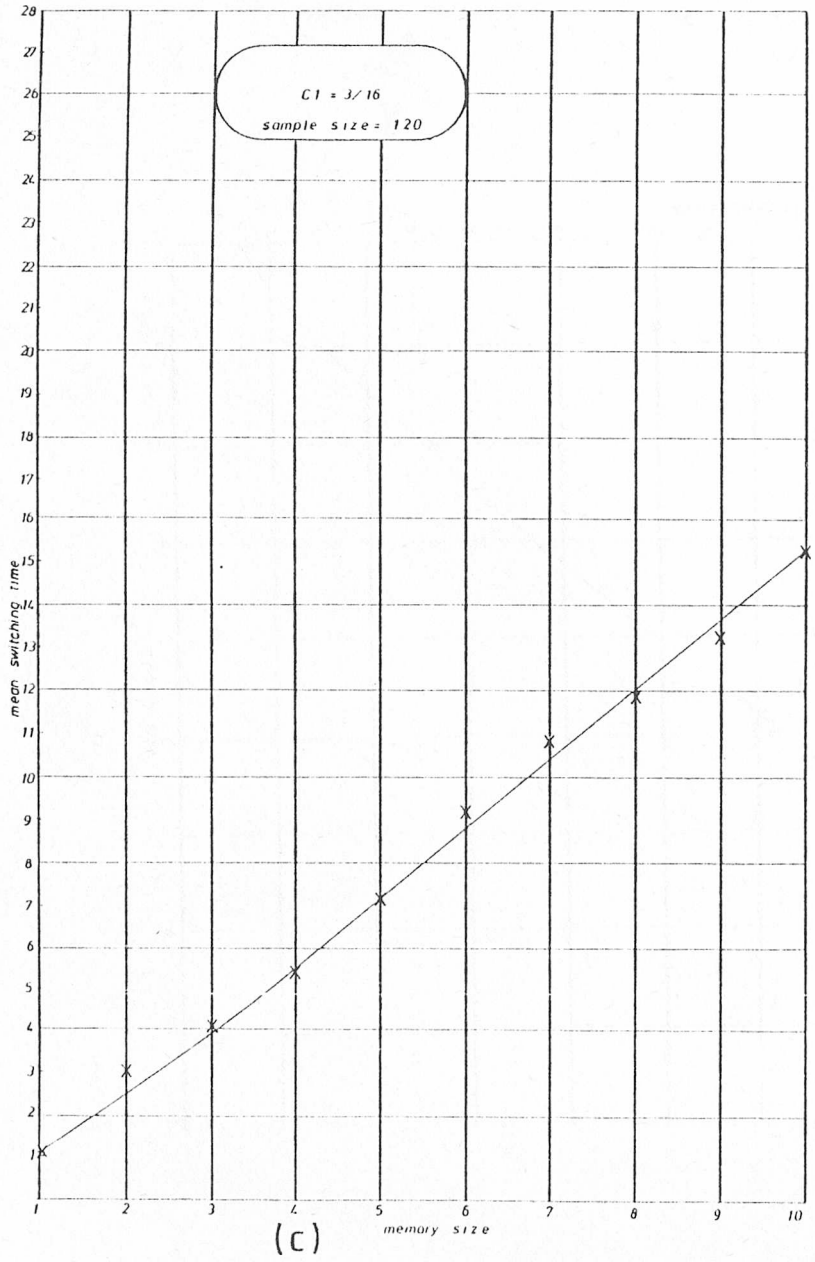

Mean switching time of Feller extension
C1 = 1/16 C2 = 1 - C1
sample size = 100
X measured results
theoretical results
mean switching time
memory size
1
2
3
4
5
6
7
8
9
10
11
12
13

(a)


C1 = 1/8
sample size = 120
mean switching time
memory size
1
2
3
4
5
6
7
8
9
10
11
12
13

(b)

Figure 2·8


C1 = 3/16
sample size = 120
mean switching time
memory size
(c)

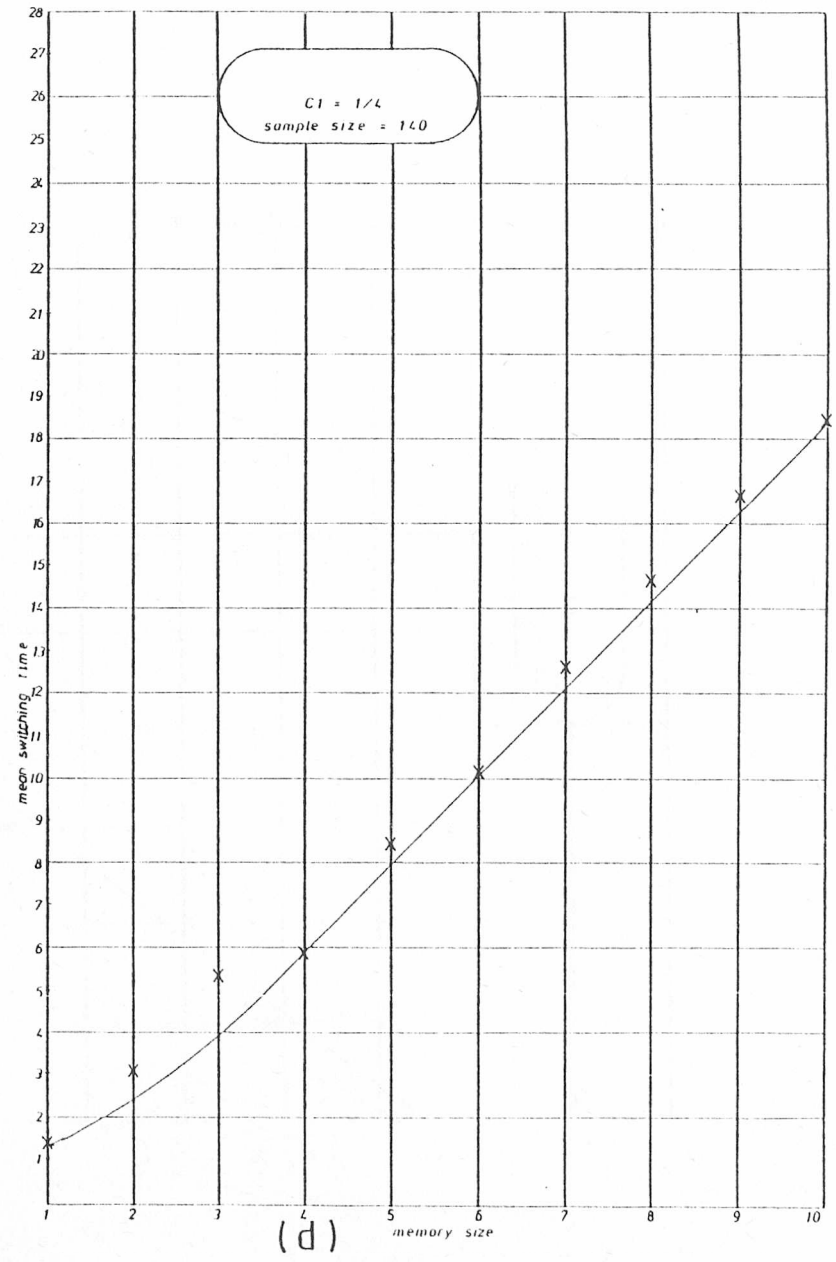

C1 = 1/4
sample size = 140
mean switching time
memory size
(d)

Figure 2·8

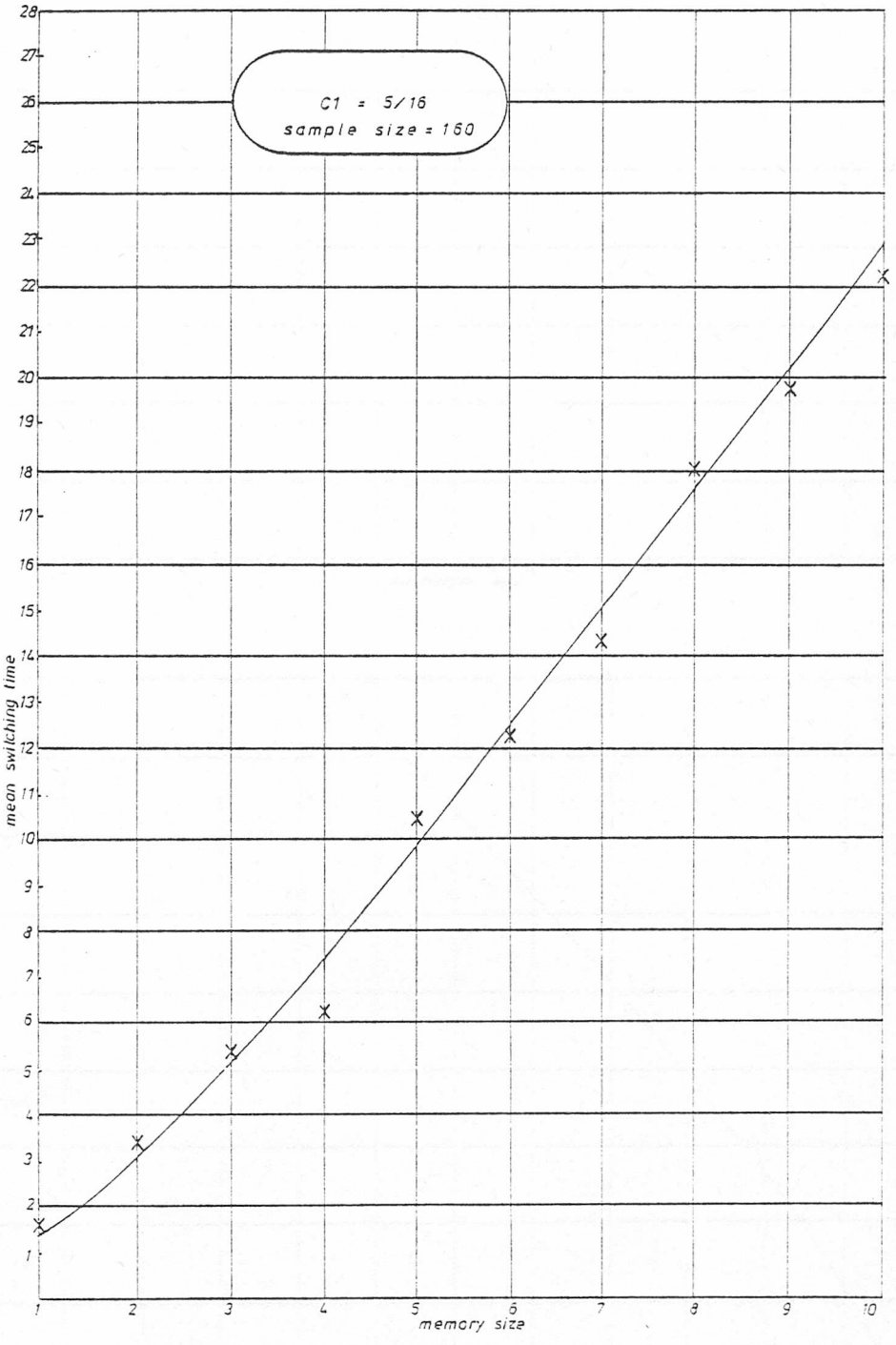
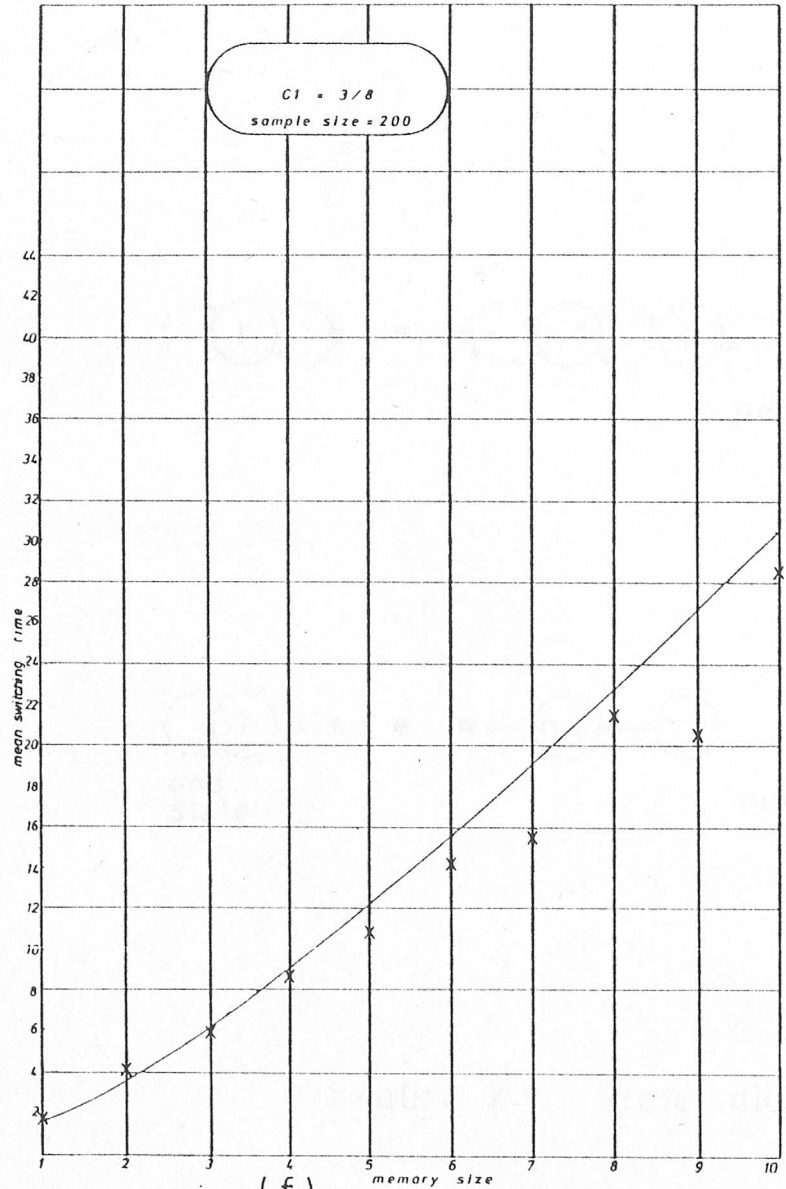
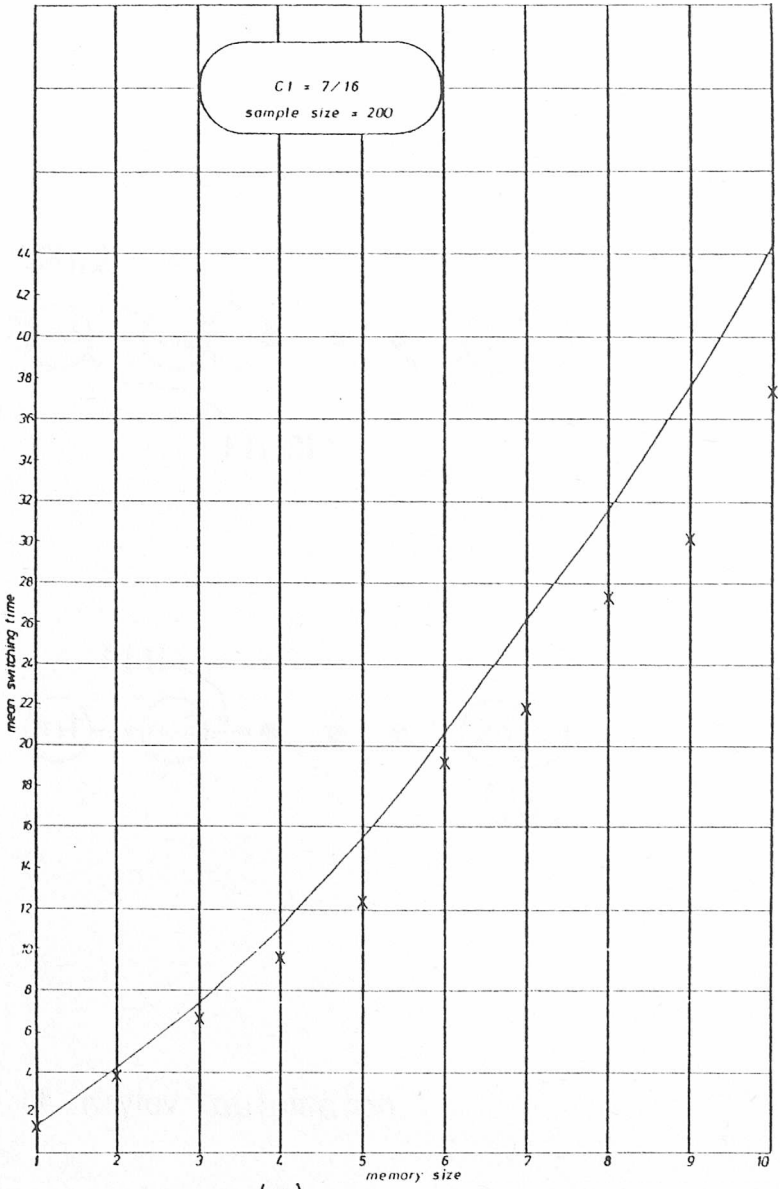

C1 = 5/16
sample size = 160
mean switching time
memory size
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

Figure 2·8 (e)


C1 = 3/8
sample size = 200
mean switching time
memory size
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
1
2
3
4
5
6
7
8
9
10

(f)

Figure


C1 = 7/16
sample size = 200
mean switching time
memory size
1
2
3
4
5
6
7
8
9
10

(g)

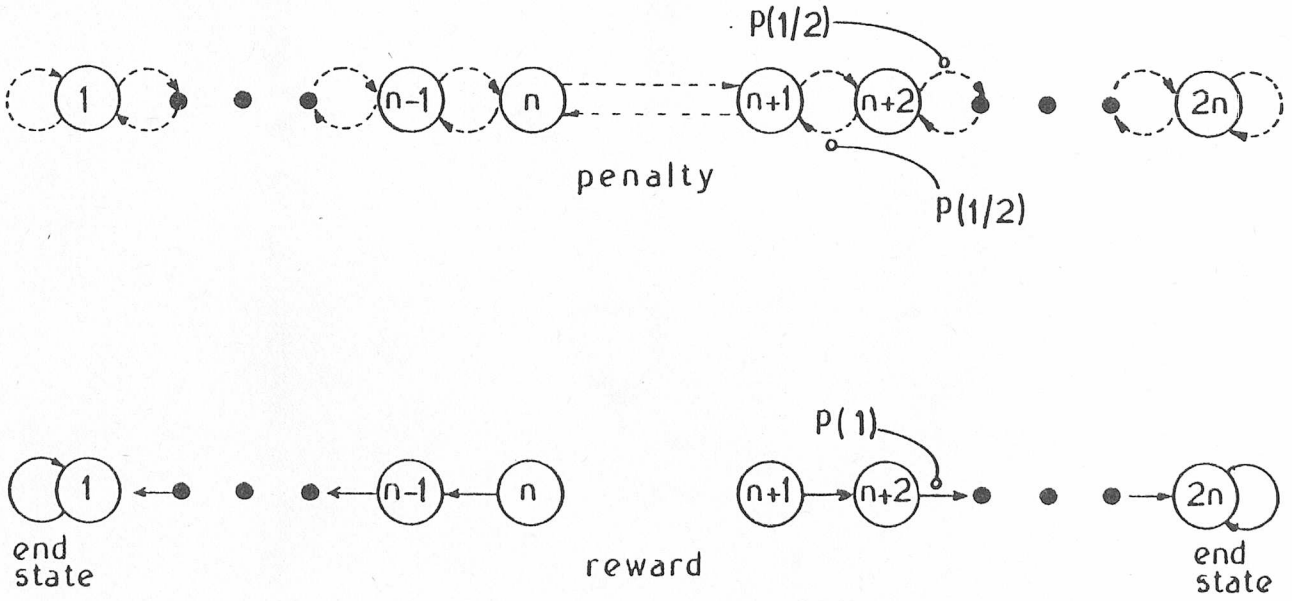


Figure 2.9 State diagram of Krylov automaton

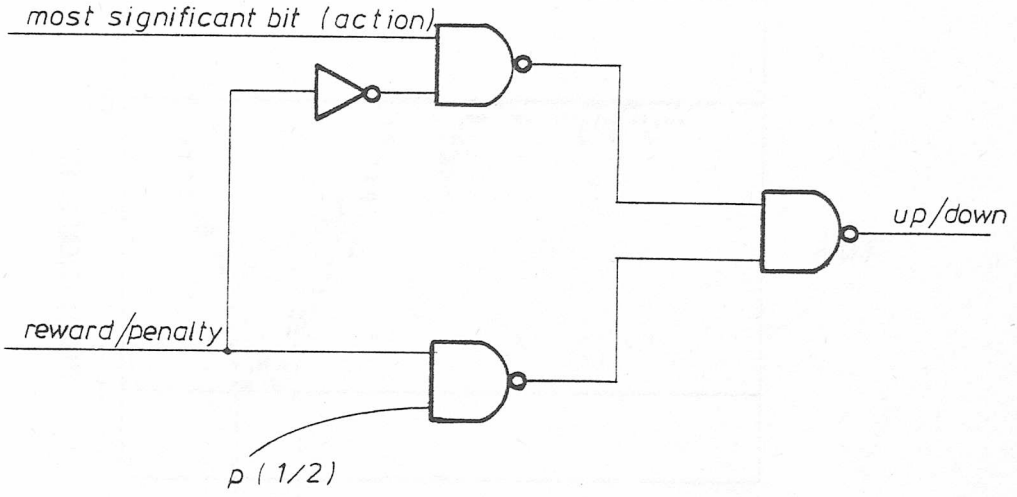


Figure 2.10 Circuit diagram of Krylov automaton

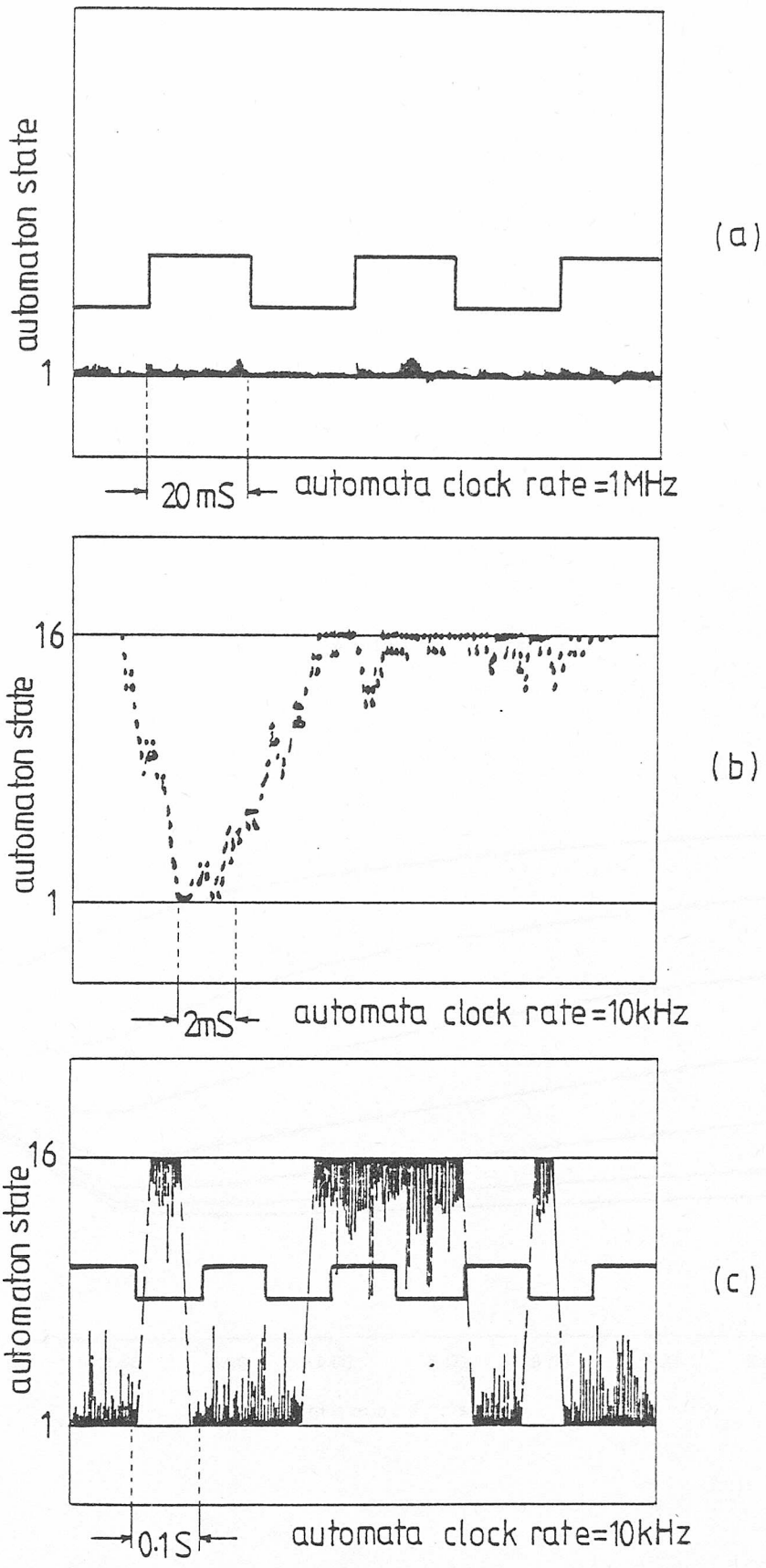


Figure 2.11 Operation of Krylov automaton in switched environment

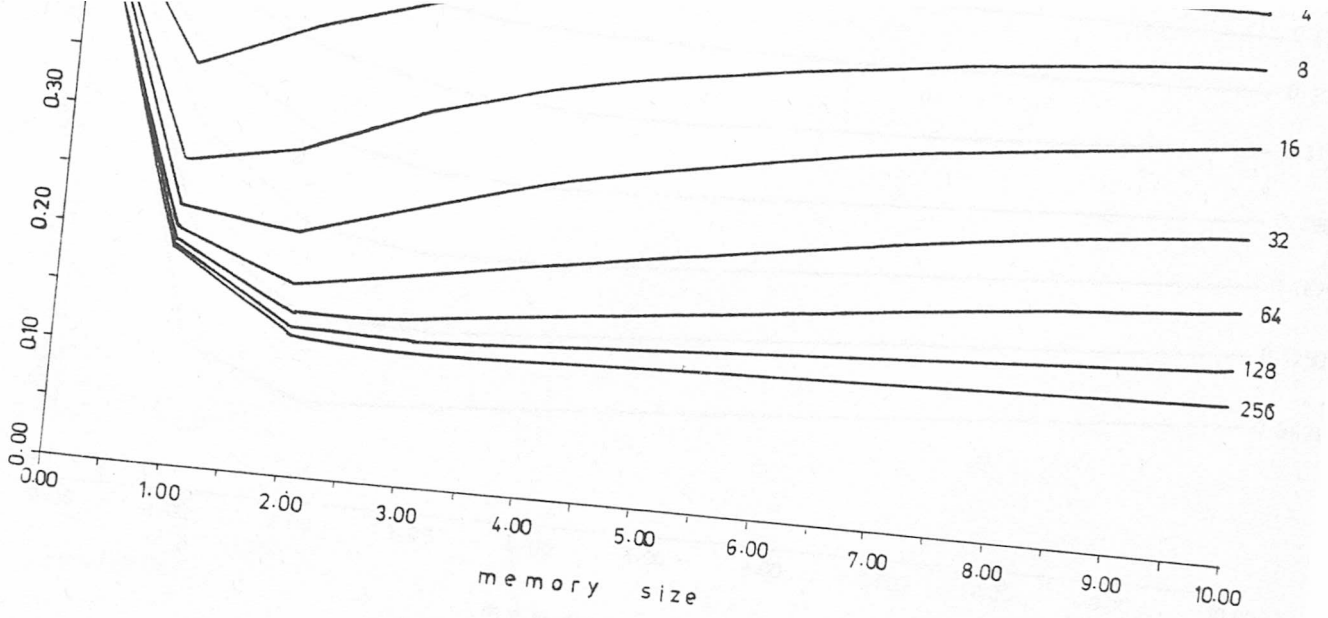
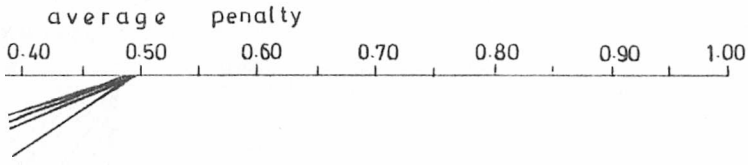
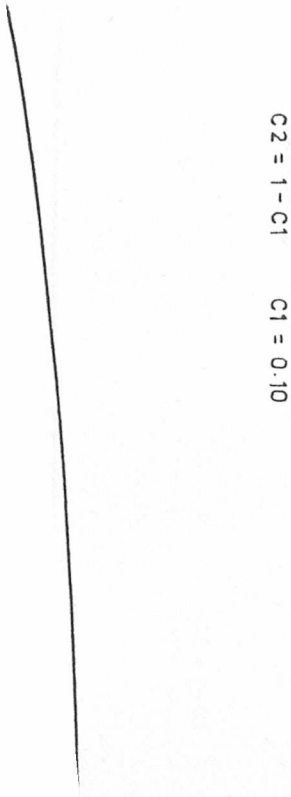


Figure 2.12 Average penalty of Tsetlin automaton





average number of clock pulses between switches

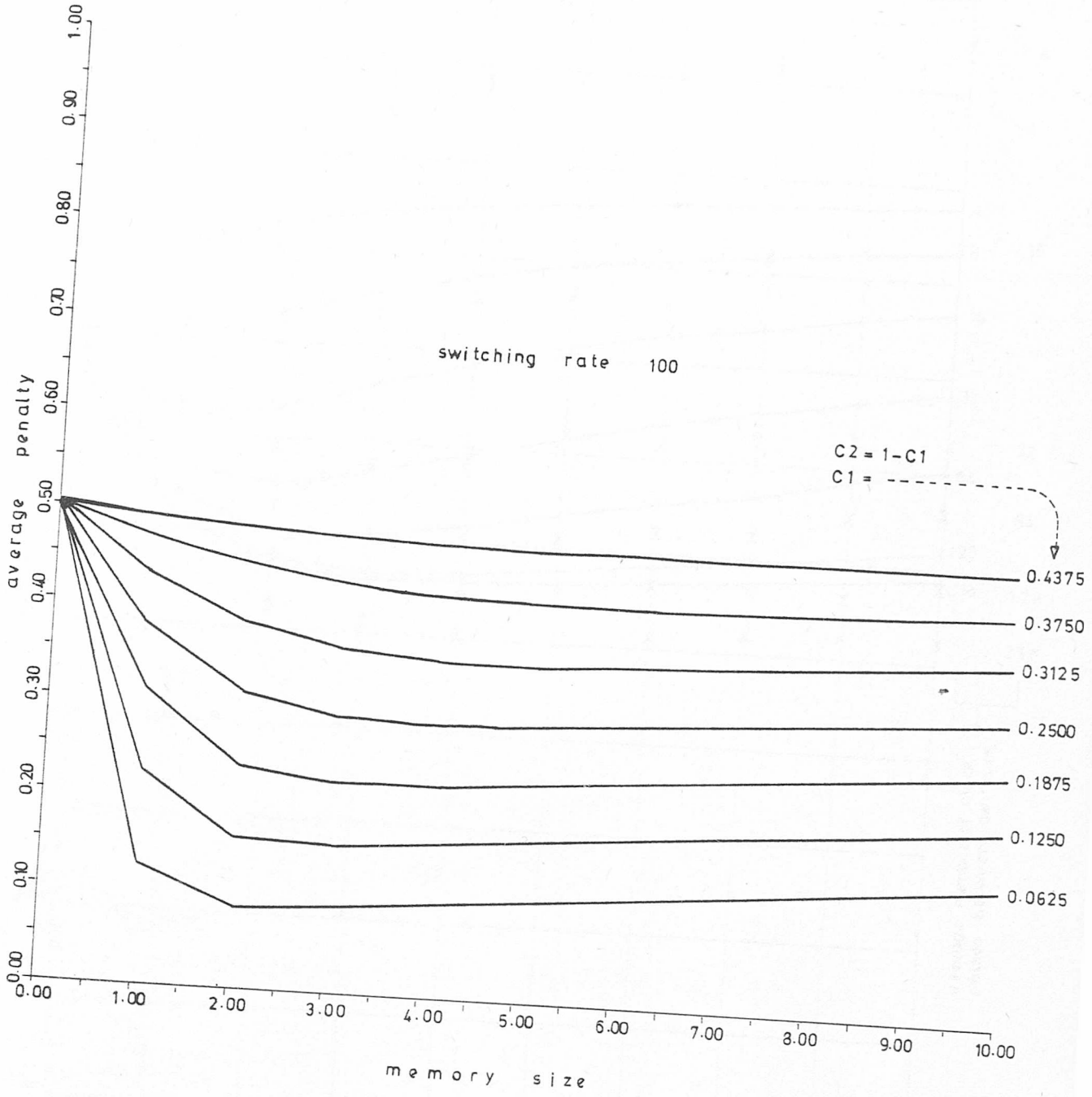
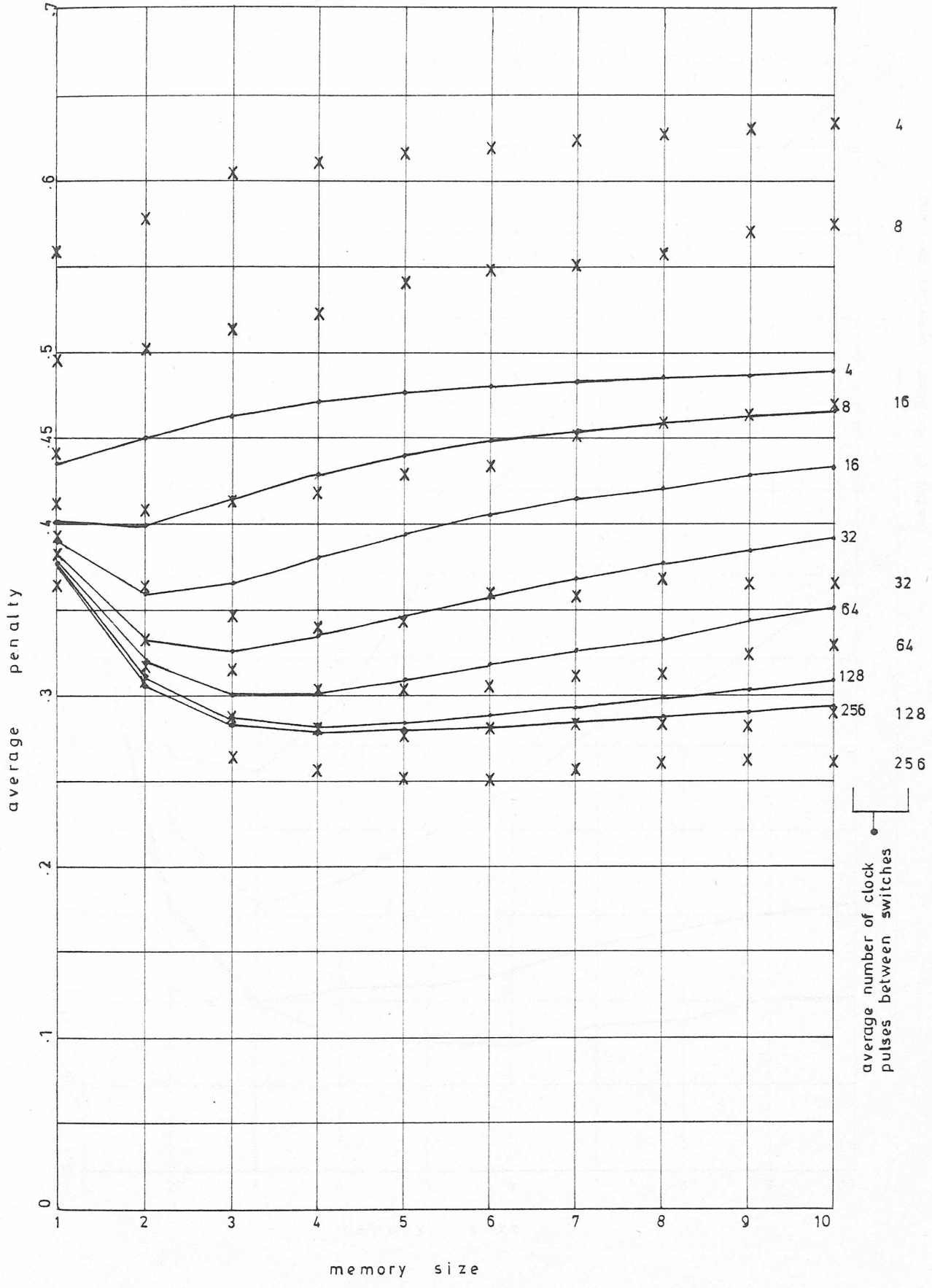


Figure 2.13 Average penalty of Tsetlin automaton



X are measured results

Figure 2.14 Average penalty results

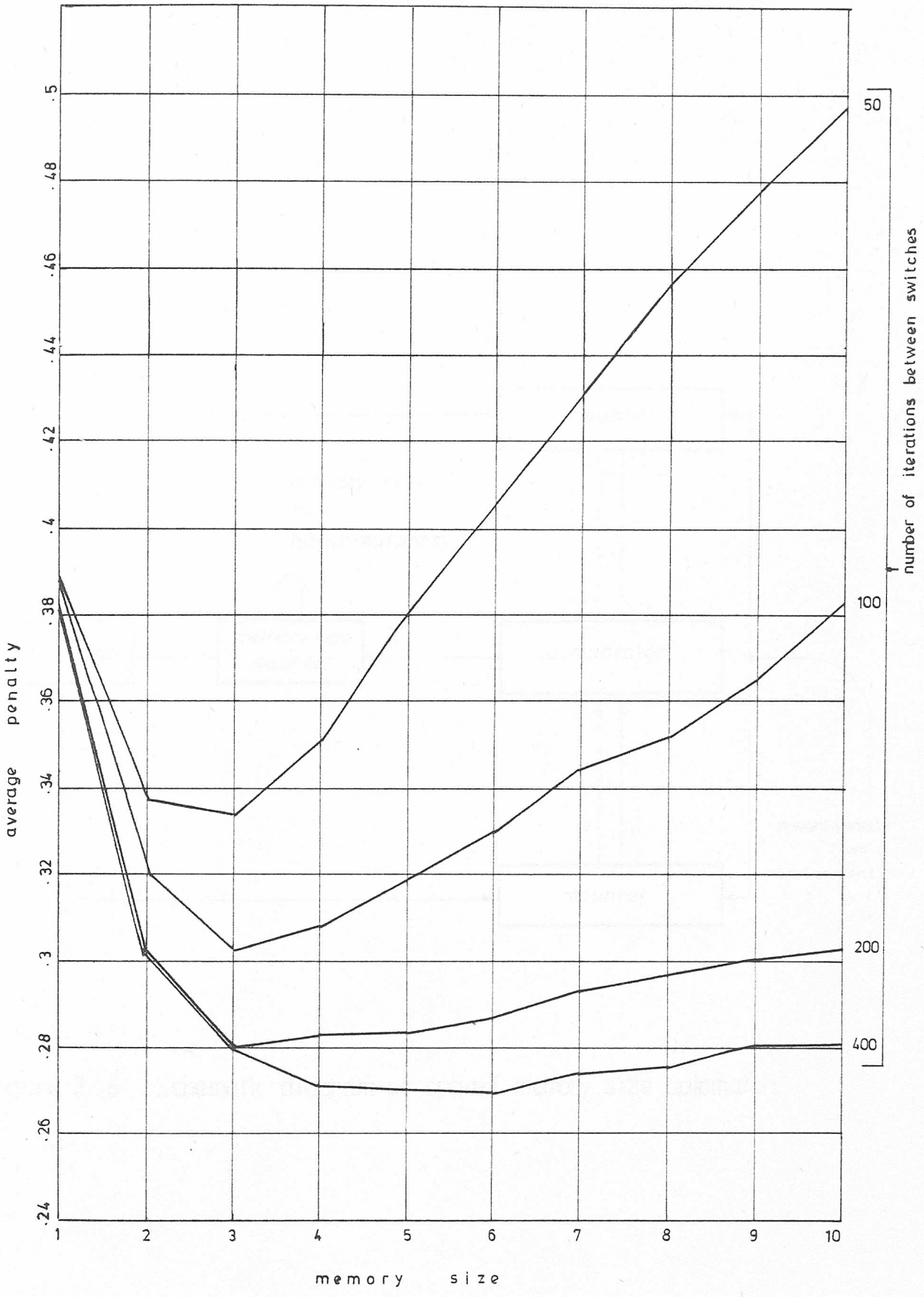


Figure 2.15 Average penalty for determinate switching

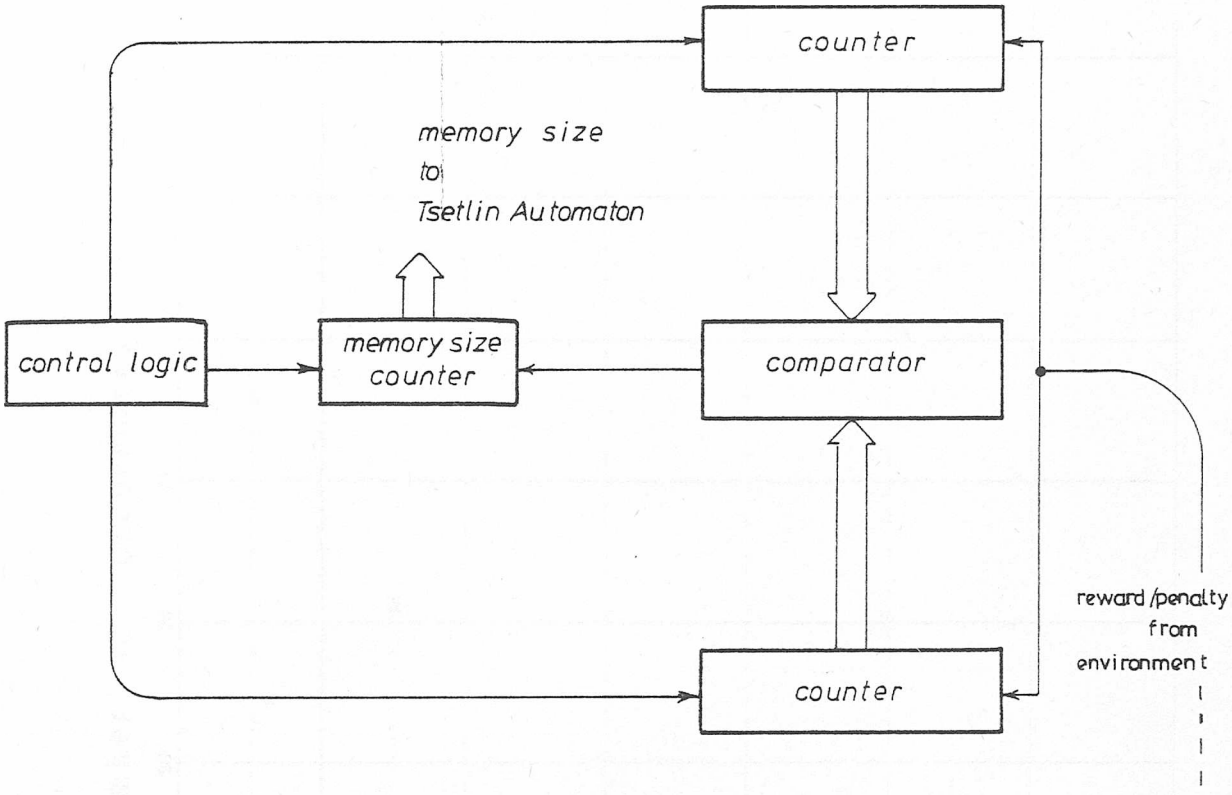


Figure 2.16 Schematic diagram of optimal memory size automaton

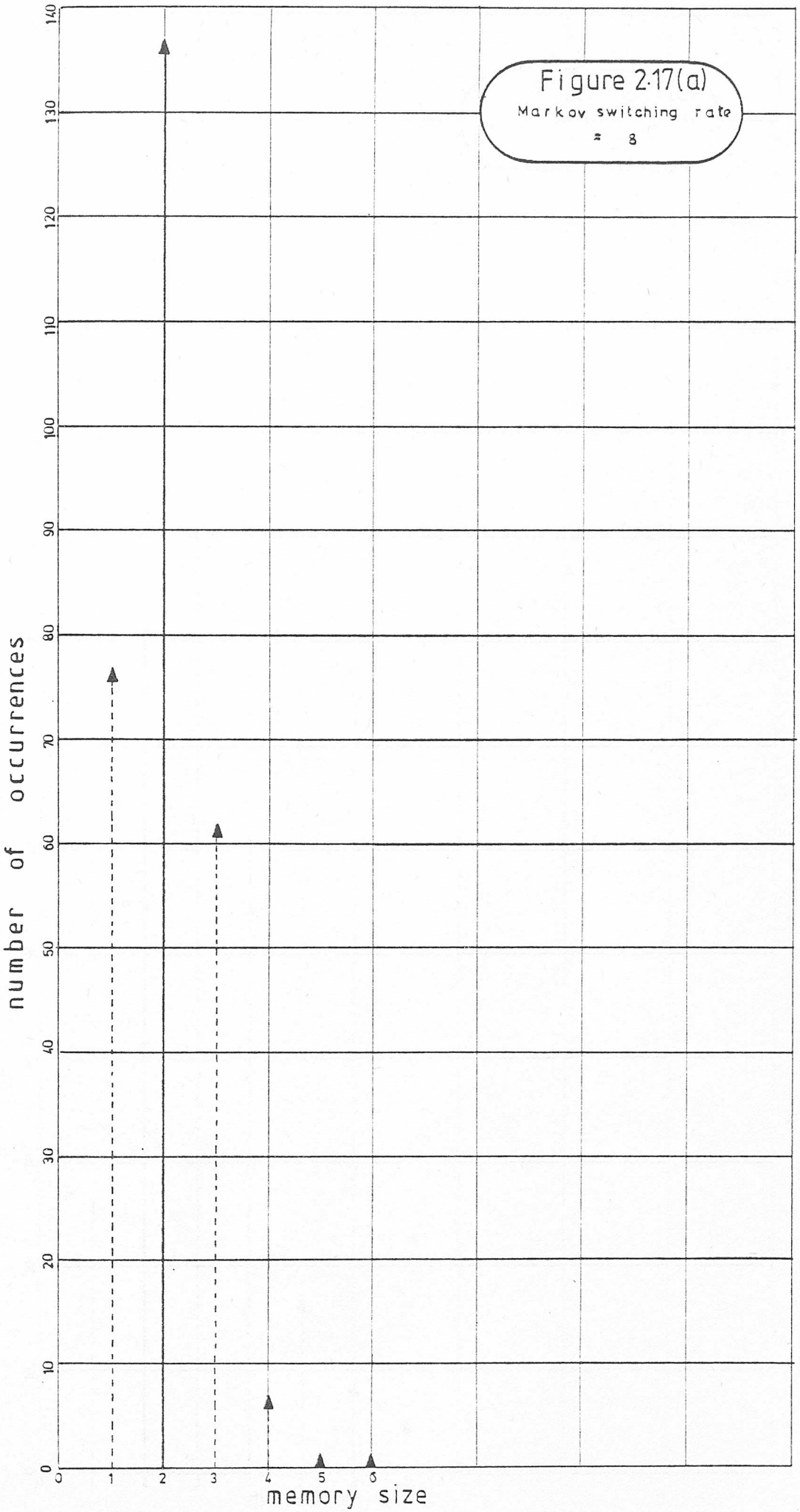


Figure 2.17(a)
Markov switching rate
= 8
number of occurrences
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
0
1
2
3
4
5
6
memory size

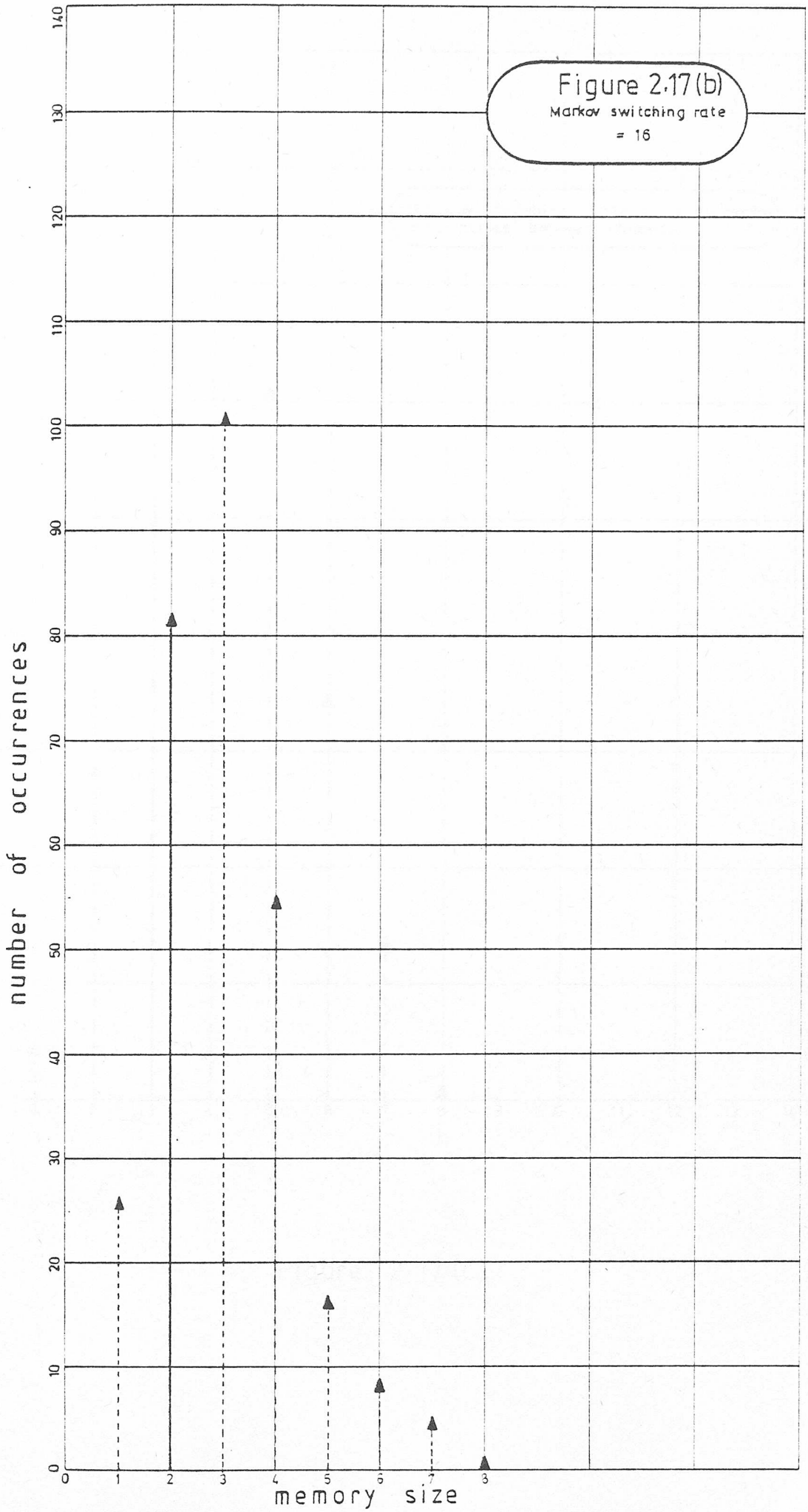


Figure 2.17(b)
Markov switching rate
= 16
number of occurrences
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
0
1
2
3
4
5
6
7
8
memory size

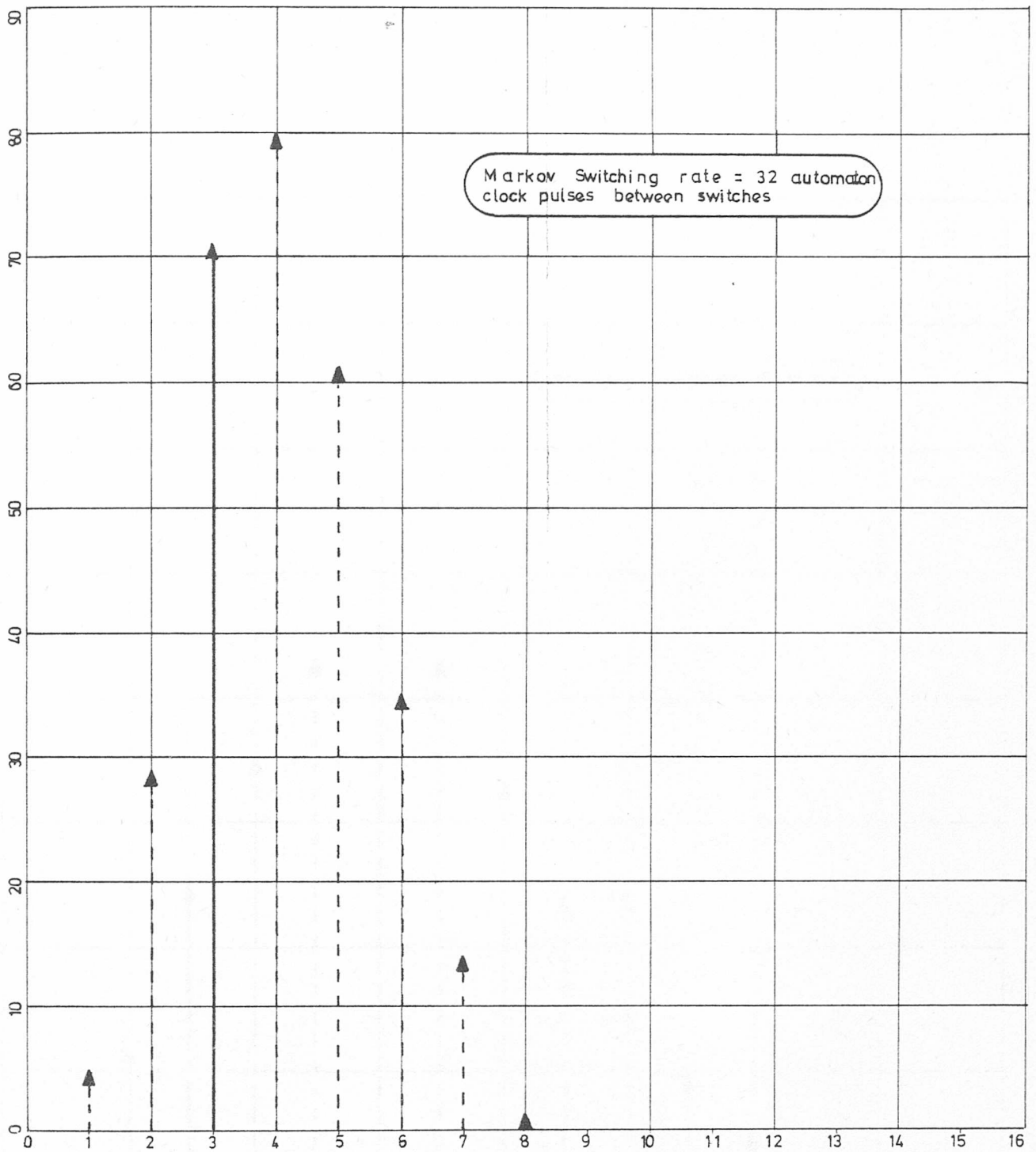

Markov Switching rate = 32 automaton clock pulses between switches
0
10
20
30
40
50
60
70
80
90
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Figure 2·17 (c)

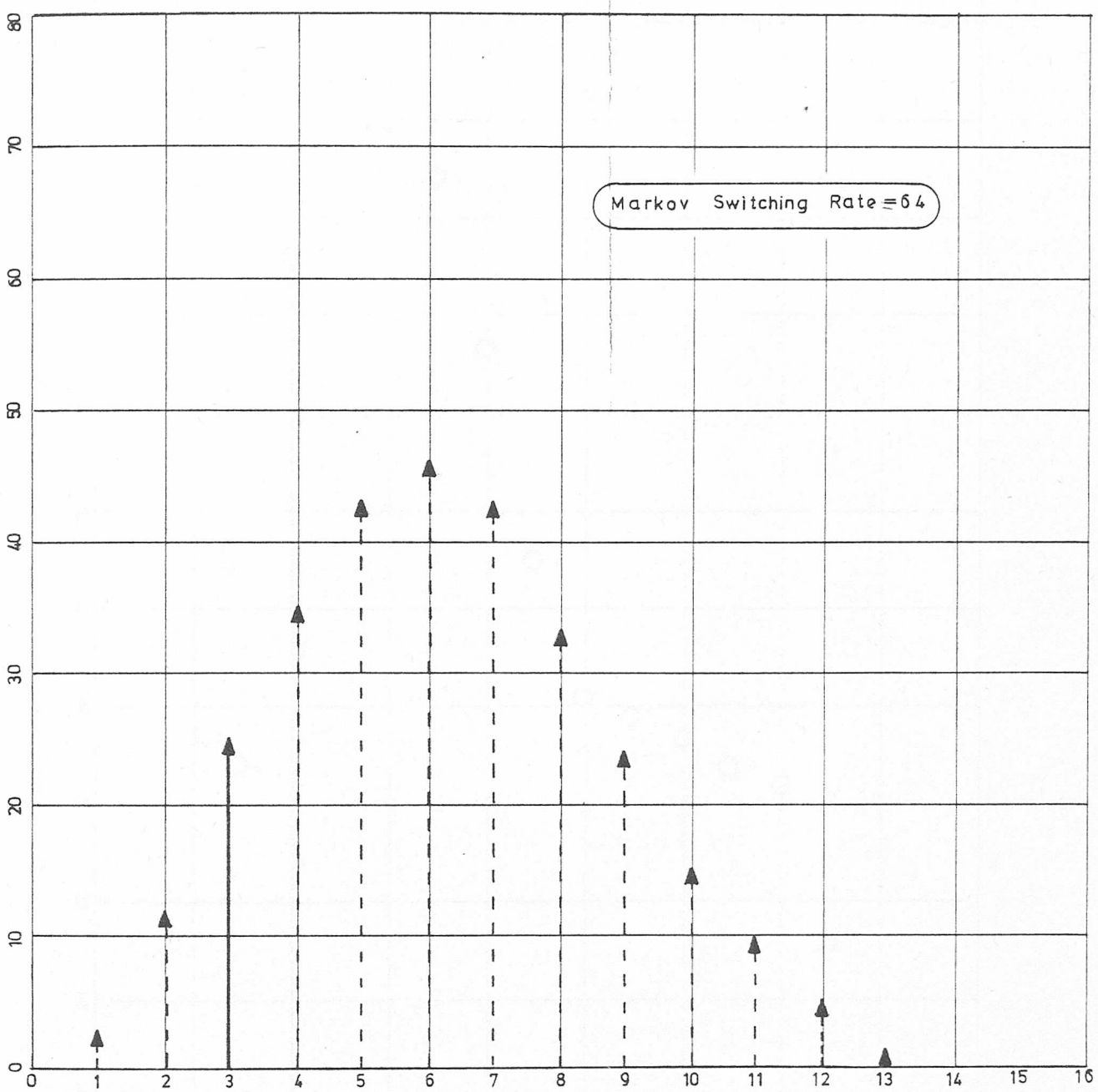

Markov Switching Rate = 64
0
10
20
30
40
50
60
70
80
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Figure 2·17 (d)

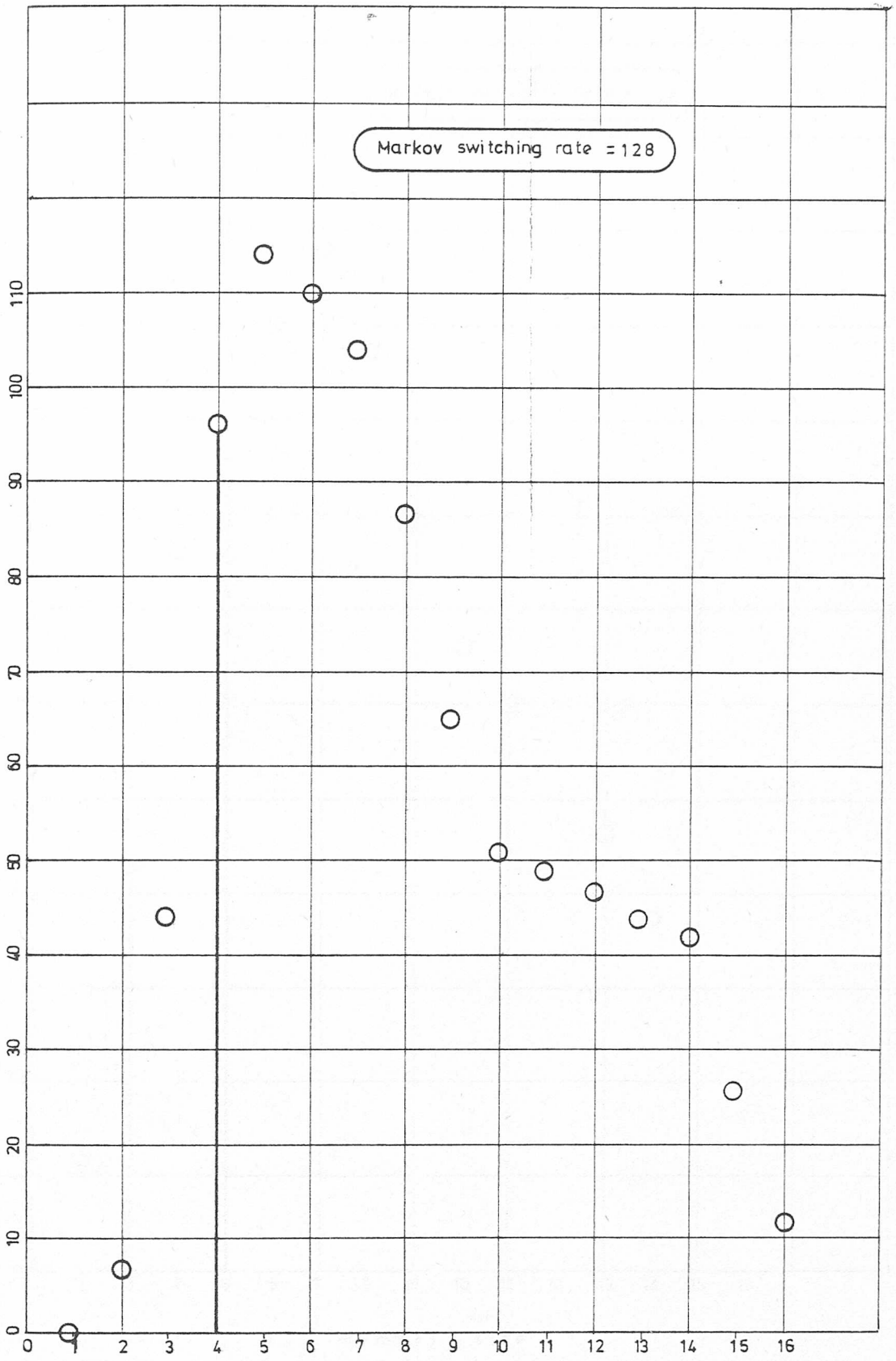

Markov switching rate = 128
0
10
20
30
40
50
60
70
80
90
100
110
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Figure 2·17 (e)

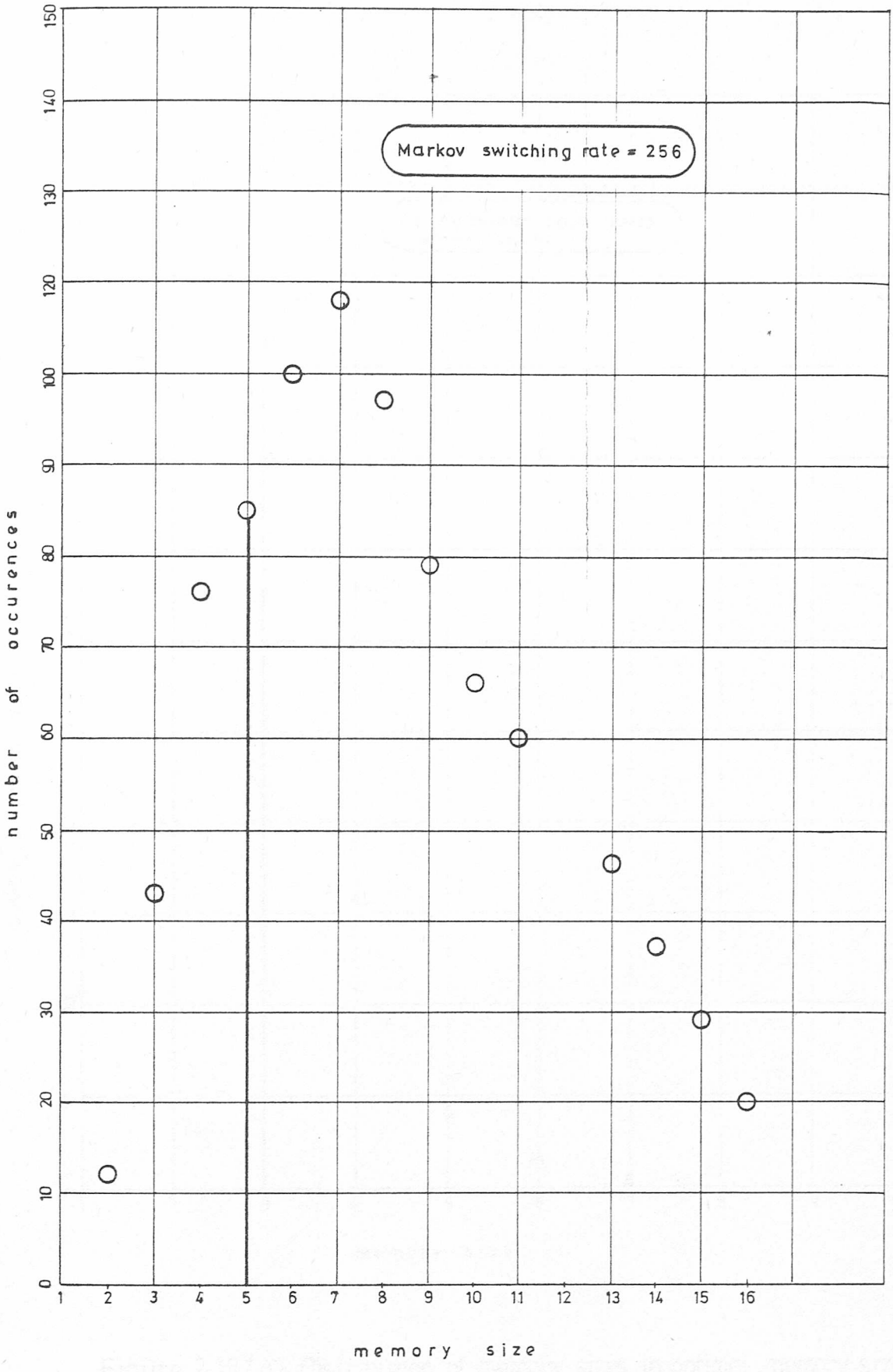

Markov switching rate = 256
number of occurences
memory size
0
10
20
30
40
50
60
70
80
90
100
120
130
140
150
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Figure 2.17 (f)

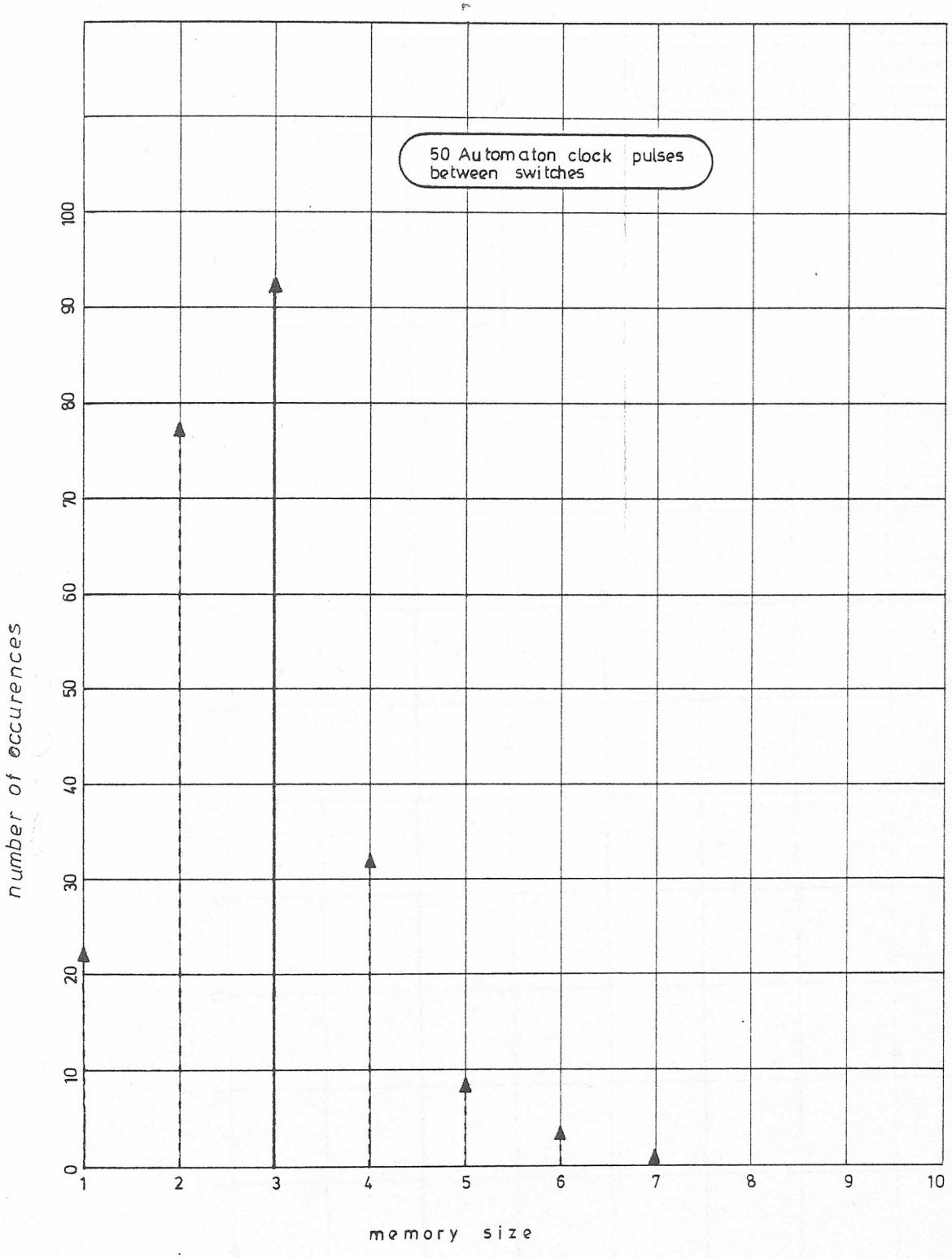


Figure 2·18 (a) Distribution of memory sizes in optimal memory size automaton

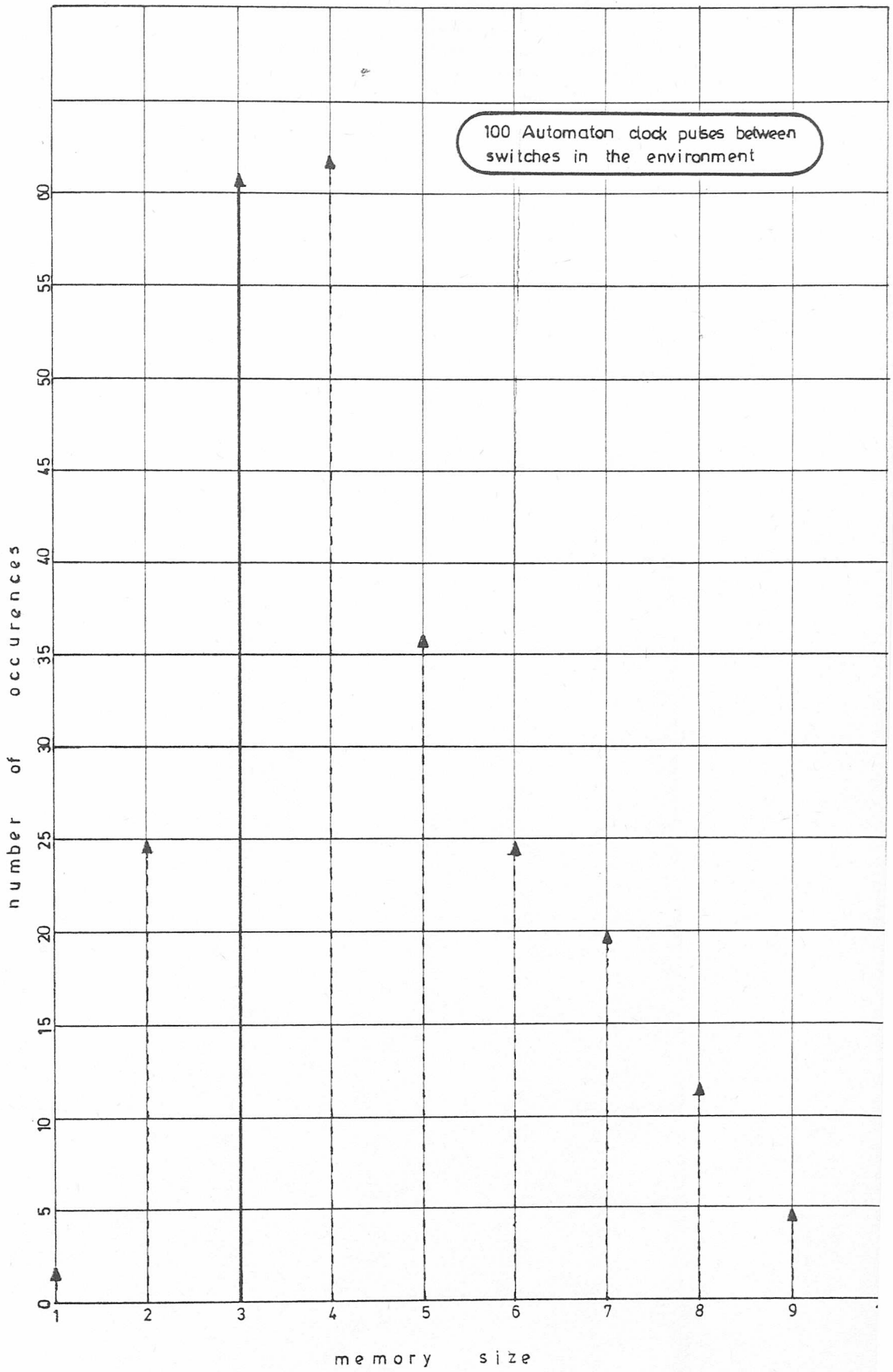

100 Automaton clock pulses between switches in the environment
number of occurences
0
5
10
15
20
25
30
35
40
45
50
55
60
1
2
3
4
5
6
7
8
9
memory size

Figure 2·18 (b)

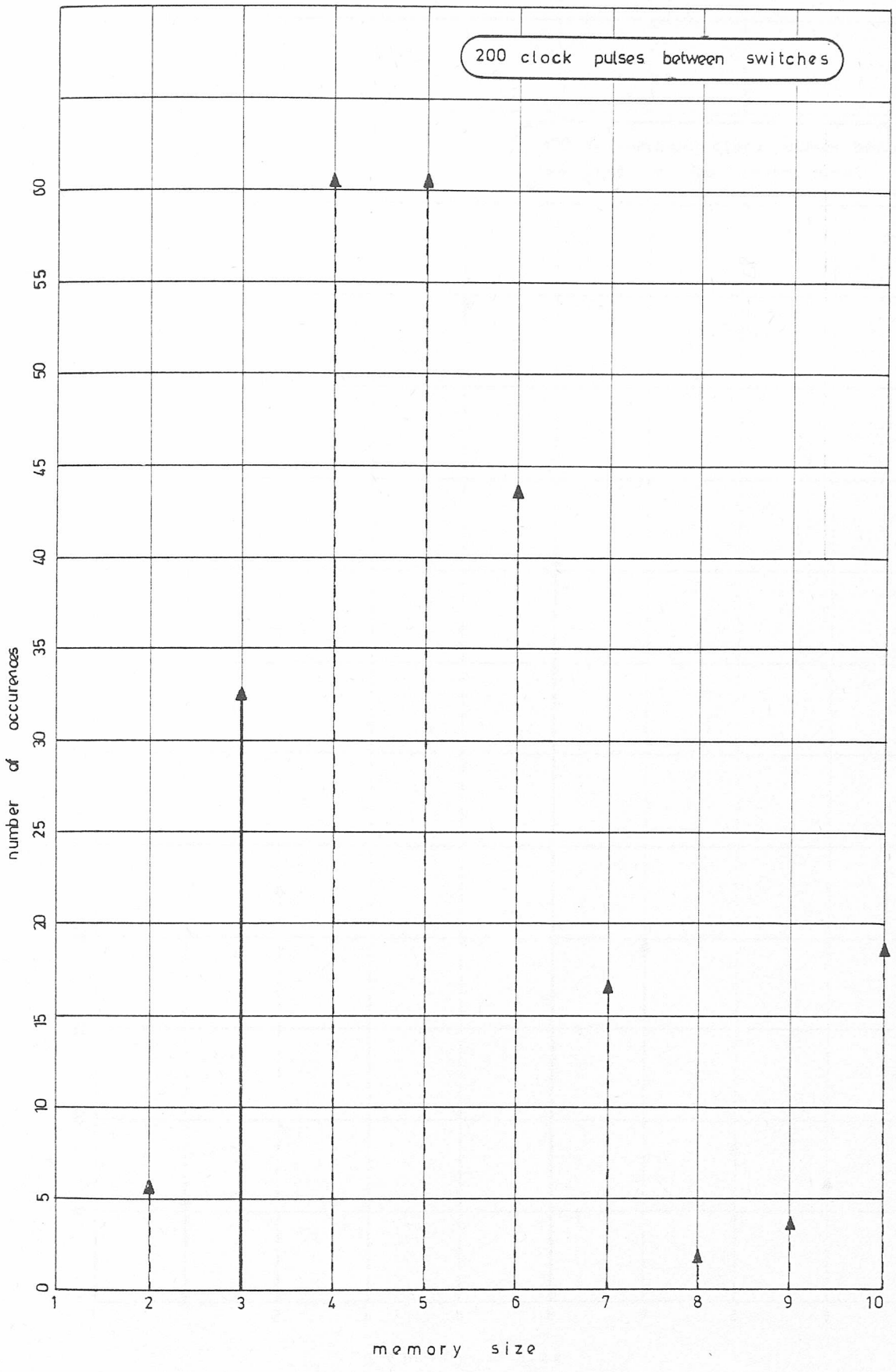

200 clock pulses between switches
number of occurences
0
5
10
15
20
25
30
35
40
45
50
55
60
1
2
3
4
5
6
7
8
9
10
memory size

Figure 2·18(c)

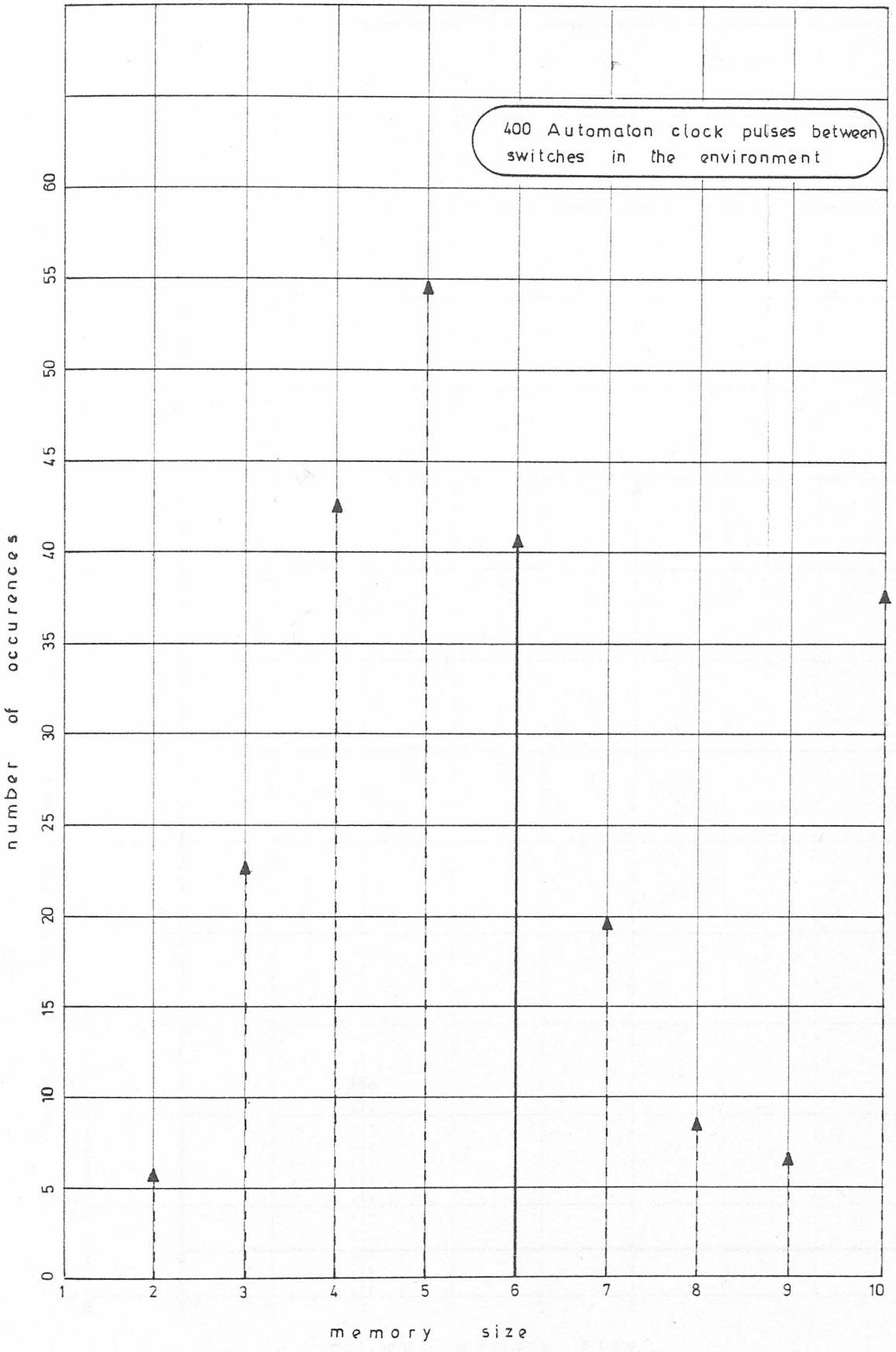

400 Automaton clock pulses between switches in the environment
number of occurences
0
5
10
15
20
25
30
35
40
45
50
55
60
1
2
3
4
5
6
7
8
9
10
memory size

Figure 2·18 (d)

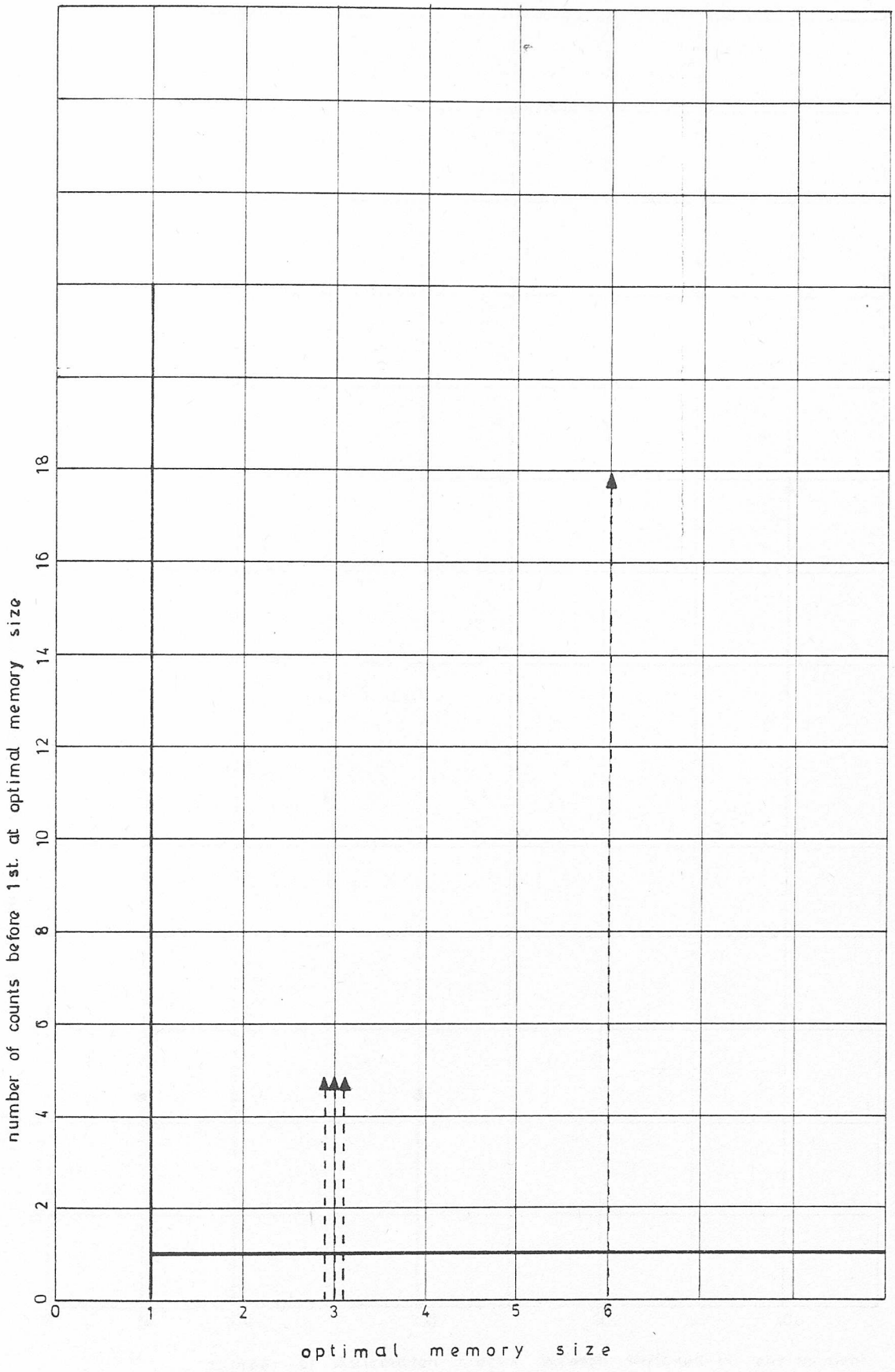


Figure 2·19(a) Speed of operation of optimal memory size automaton

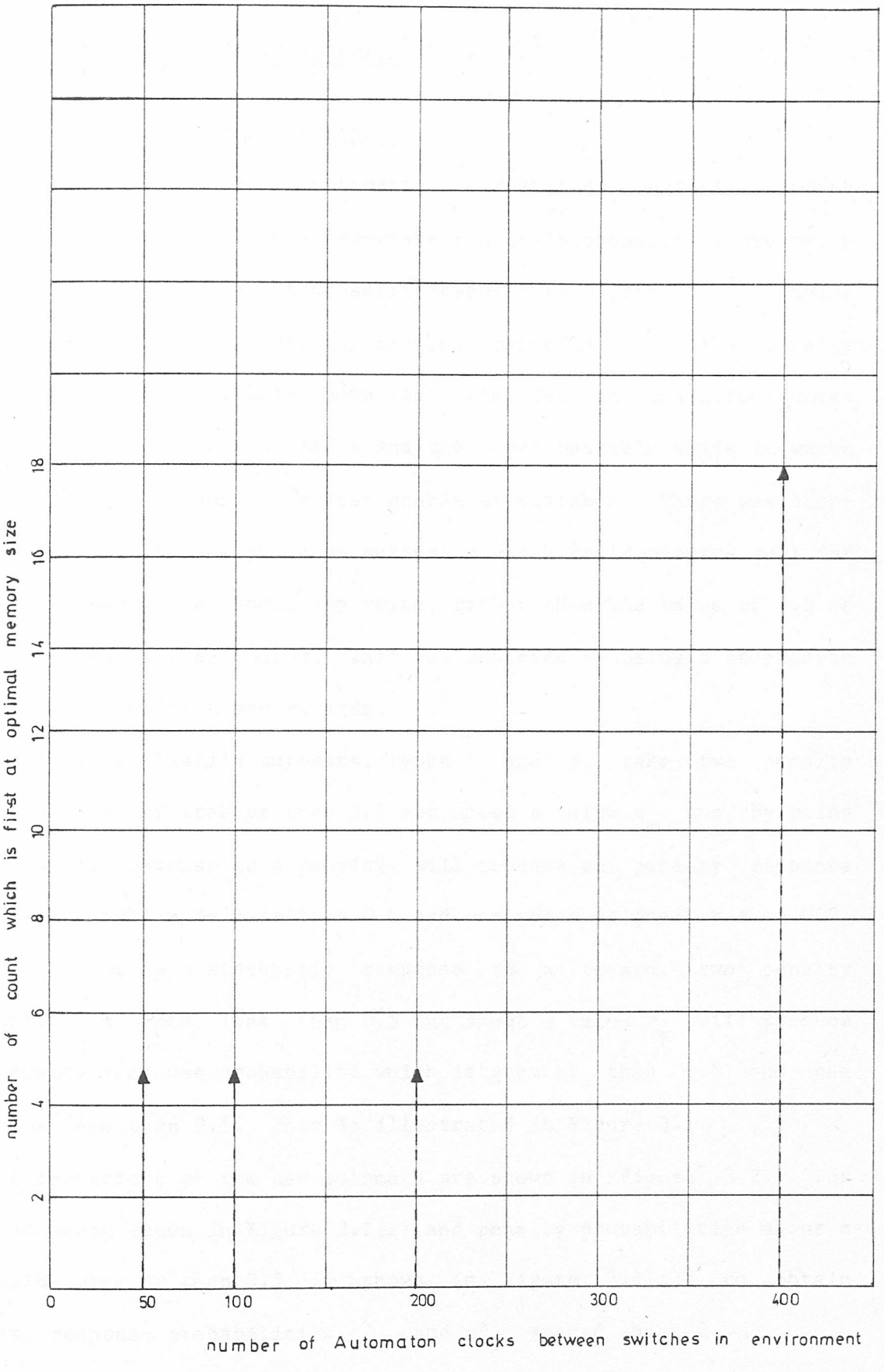

number of count which is first at optimal memory size
2
4
6
8
10
12
14
16
18
0
50
100
200
300
400
number of Automaton clocks between switches in environment

Figure 2.19(b)

CHAPTER 3 MODIFIED TSETLIN AUTOMATA

Modified Tsetlin Automata-Operation

In operation the Tsetlin automaton has stable or unstable actions depending on whether the appropriate penalty probabilities are below or above 0.5. By using a stochastic response to a penalty the Krylov automaton always has stable actions regardless of the penalty probabilities. It has been shown that the Tsetlin automaton works well if one action is stable and the other unstable while it works poorly if both actions are either stable or unstable. There was scope for improvement by designing an automaton which could operate well for penalty probabilities about any value, rather than the value of 0.5 as for the Tsetlin automaton. This was achieved by using a stochastic response to penalties and rewards.

The modified Tsetlin automata, types 1 and 2, take two penalty probabilities of greater than 0.5 but about a value c_m and, by using a stochastic response to a penalty, will produce one penalty response probability which is less than 0.5 and one which is greater than 0.5. Further, by using a stochastic response to a reward, two penalty probabilities both less than 0.5 but about a value c_m will produce one penalty response probability which is greater than 0.5 and one which is less than 0.5. This is illustrated in Figure 3.1.

The operations of the new automata are shown in Figure 3.2. For the automaton shown in Figure 3.2(a) and penalty probabilities about a c_m value greater than 0.5 as shown in Figure 3.1(a), to obtain penalty response probabilities c'_1 and c'_2 spaced about 0.5

$$c'_m = 0.5 \qquad (3.1)$$

Using a stochastic response to a penalty with probability W_p of moving towards states N and N+1 and assuming a deterministic response to a reward then

$$c`_m = c_m * W_p \qquad (3.2)$$

Substituting equation (3.2) into equation (3.1) gives

$$W_p = 1/(2c_m)$$

W_p is to be a stochastic variable and so has a maximum value of 1 thus

$$W_p = 1/(2c_m) \text{ if } 1/(2c_m) < \text{or} = 1$$
$$= 1 \text{ if } 1/(2c_m) > 1 \qquad (3.3)$$

For c_m less than or equal to 0.5 $W_p = 1$.

For penalty probabilities about a c_m value less than 0.5, as shown in Figure 3.1(c), to obtain penalty response probabilities $c`_1$ and $c`_2$ spaced about 0.5

$$c`_m = 0.5 \qquad (3.4)$$

Using a stochastic response to a reward with probability W_r of moving towards the end state associated with the action output by the automaton and assuming a deterministic response to a penalty, an assumption justified by equation (3.3), then

$$c`_m = c_m + (1-W_r)(1-c_m) \qquad (3.5)$$

substituting equation (3.5) into equation (3.4)

$$W_r = 1/2(1-c_m) \text{ if } 1/2(1-c_m) < \text{or} = 1$$
$$= 1 \text{ if } 1/2(1-c_m) > 1 \qquad (3.6)$$

For c_m greater than 0.5, $W_r = 1$, so justifying the assumption made in forming equation (3.2).

For the automaton shown in Figure 3.2(b) in addition to penalty and reward responses there is an inaction response. If an inaction response is counted as half a penalty response, for penalty probabilities about a c_m value greater than 0.5, as shown in Figure 3.1(a), to obtain penalty response probabilities c'_1 and c'_2 spaced about 0.5

$$c'_m = 0.5 \qquad (3.7)$$

Using a stochastic response to a penalty with probability W_p of moving towards states N and N+1 and $(1-W_p)$ of remaining in the same state, and assuming a deterministic response to a reward then

$$c'_m = c_m W_p + 1/2\, c_m (1-W_p) \qquad (3.8)$$

Substituting equation (3.8) into equation (3.7)

$$W_p = (1-c_m)/c_m \text{ if } (1-c_m)/c_m < \text{or} = 1$$
$$= 1 \text{ if } (1-c_m)/c_m > 1 \qquad (3.9)$$

For c_m less than or equal to 0.5 $W_p = 1$.

For penalty probabilities about a c_m value less than 0.5, as shown in Figure 3.2(c), to obtain penalty response probabilities c'_1 and c'_2 spaced about 0.5

$$c'_m = 0.5 \qquad (3.10)$$

Using a stochastic response to a reward with probability W_r of moving towards the end state associated with the action output by the automaton and $(1-W_r)$ of remaining in the same state, and assuming a deterministic response to a penalty, an assumption justified by equation (3.9) then

$$c'_m = c_m + 1/2\, (1-c_m)(1-W_r) \qquad (3.11)$$

Substituting equation (3.11) into equation (3.10)

$$W_r = c_m/(1-c_m) \text{ if } c_m/(1-c_m) < \text{or} = 1$$
$$= 1 \text{ if } c_m/(1-c_m) > 1 \qquad (3.12)$$

For c_m greater than 0.5, W_r =1 so justifying the assumption made in forming equation (3.8).

Using equations (3.3) and (3.6) or (3.9) and (3.12) the automata should be able to operate with c_i 's about any value and retain the qualities of the Tsetlin automaton when operating with c_i 's about 0.5. It was thought that the type 2 automaton with the inaction response would have less variance and so could be more optimal than the type 1 automaton for the same memory size.

Modified Tsetlin Automata-Steady State Probability and Mean Switching Time

The steady state probabilities of the states of the automata and the mean switching times may be calculated for the modified Tsetlin using the same methods as used for the Tsetlin automaton by simply substituting the appropriate Markov transition matrix as given in Appendix 3.

The state probability and mean switching time results were calculated by computer and the corresponding results for the Tsetlin, Krylov and Lrp automata were also calculated for the purpose of comparison. Figures 3.3(a)-(d) show the sum of the steady state probabilities for states corresponding to the optimal action, for various penalty probabilities about 0.5. The corresponding measure for the Lrp automaton is the action probability of the optimal action and these two measures have collectively been described as the optimality of the automaton. It can be seen that the results for the Tsetlin and modified Tsetlin automata are identical. Figures 3.4(a)-(d) show corresponding results for the mean switching times of the automata. Like the Tsetlin and Krylov automata this is defined for the modified Tsetlin automata as the average time from the switch

in the environment till the first output of the new correct action assuming the automaton was selecting the correct action before the switch.

Figures 3.5(a)-(g) and 3.6(a)-(g) show results for penalty probabilities that are not limited to be about the value of 0.5 so showing the performance of the modified Tsetlin automata with c_m values other than 0.5. It can be seen from Figures 3.5(a)-(g) that for penalty probabilities greater than 0.5 the probability of the modified Tsetlin automaton selecting the correct action is far higher than the corresponding Tsetlin automaton. For penalty probabilities both less than 0.5 Figures 3.6(a)-(g) show that the modified Tsetlin automata have mean switching times which are reasonably constant compared to those of the Tsetlin. The differences betwen the two modified Tsetlin automata become apparent in Figures 3.5 and 3.6. For similar memory sizes the type 2 automaton is more nearly optimal while the type 1 automaton has a shorter mean switching time. However over the complete range of penalty probabilities the results show that the modified Tsetlin automata maintain near optimal behavior and have short mean switching times indicating that they will operate well in non-stationary environments.

Having seen that the modified Tsetlin automata can operate without restrictions on the penalty probabilities they can be compared with the Lrp automaton. The values of α and β chosen for the automaton represent values which in practice would give a very high performance. The results for the optimality of the modified Tsetlin automata are better than the corresponding results for the Lrp. However the mean switching time results are poorer. In practice the memory sizes for the Lrp automaton are rather small and more states would be used in

order to gain a degree of optimality comparable to the modified Tsetlin results. This would in turn increase the mean switching time results to a level nearer the values for the modified Tsetlin automata.

Modified Tsetlin Automata-Simulation

Because of the relatively complicated calculation involved in finding W_p and W_r required for the operation of the modified Tsetlin automata it was decided to carry out investigations using a simulation on a computer rather than build a hardware synthesis. A set of stochastic simulation programs was already in existence and so additions were made to these to include the modified Tsetlin automata and also the Tsetlin and Krylov automata.

The first program that was modified created, from a graphical schematic diagram input by the user, a data file which was used by a second program to simulate the system. An example of a schematic diagram produced by this program is shown in Figure 3.7 which shows the Tsetlin, Krylov and both modified Tsetlin automata in a typical test circuit, connected to two probability generators which provide the penalty probabilities. The simulation facilities were limited to two action automata but the memory size of the automata was variable as was the initial state.

To operate the modified Tsetlin automata a value is required for c_m in order to calculate W_p and W_r. This was done by using two ADDIEs to estimate the penalty probabilities, c_1 and c_2, input to the automata and c_m was taken as the arithmetic mean of these. When the automata had as their output, action 1, the penalty probability was input to the ADDIE estimating c_1 while for action 2 the penalty probability was input to the ADDIE estimating c_2. This method of

obtaining c_m was included in the second program to be modified which carried out the stochastic simulation and allowed the inspection and modification of circuit element parameters.

Modified Tsetlin Automata-Simulation Results

The results obtained from the stochastic simulation program were in graphical form showing automaton state against iterations with states 1 to N, corresponding to action 1, below the axis and states N+1 to 2N, corresponding to action 2, above the axis. At the end of each simulation it was possible to examine and modify circuit element parameters and in this way, by changing the penalty probabilities, switches in the environment could be simulated.

Figures 3.8, 3.9 and 3.10 show the operation of the type 1 modified Tsetlin automaton in a range of environments. In each case the memory size is 10 and the ADDIEs have 5 bits while c_1 and c_2 are initially 0.4 and 0.1, 0.65 and 0.35 and 0.9 and 0.6 with the environment being switched between the (a) and (b) figures. Figures 3.11, 3.12 and 3.13 show corresponding results for the type 2 modified Tsetlin automaton. It can be seen that the automata learn in all environments, though the learning times are longer if the penalty probabilities are high. It can also be seen that the automata respond quickly to a switch in the environment, the switching time being smaller than the learning time. Figure 3.11 and to a lesser extent Figures 3.12 and 3.13 show the lower variance of the type 2 automaton as compared with the type 1 automaton.

When the automata first operate, the ADDIEs are in their initial state and hold c_i estimates of zero giving a c_m value of zero. Because of this, both actions of the automaton are unstable and the automaton moves frequently between the two actions so providing an

input for both ADDIEs. As the c_i estimates rise the actions of the automata become less unstable until the value of c_m becomes larger than the lower c_i. At this point the automata have one action that is unstable and one that is just stable. The automaton will spend most time in states associated with the stable action allowing the corresponding ADDIE to rise to its steady state value. Any movement into states associated with the wrong action will tend to make the corresponding ADDIE rise towards its steady state value, increasing the value of c_m and making the correct action more stable. The results show that the initial learning time is longer than the switching time. This is because, within the automaton, the response time of the ADDIEs is longer than that of the counter. When a switch in the environment occurs the action that was stable becomes unstable and the automaton moves to change its action. The ADDIE also receives a new penalty probability and begins to move towards a new steady state value, however, before it has time to change significantly the automaton moves from the unstable to the new stable action.

Figure 3.14 shows the response of the automata to a change in the environment from 0.4 and 0.1 to 0.6 and 0.9. It can be seen that the automata respond with a learning type behaviour, moving frequently between states N and N+1, before moving to the correct action. When the environment changes, both actions are made unstable. Time is required for the ADDIEs to respond and produce a c_m value high enough to result in one stable and one unstable action.

Figure 3.15 shows the response of the two automata to a change in the environment from 0.9 and 0.6 to 0.1 and 0.4. In this situation the automata continue to output the same action because of their near optimal behavior. When the environment switches, both actions become

more stable because of the high value of c_m in relation to the new c_i's. The automata continue to output the same action and the corresponding ADDIE falls to a new steady state value but this is not sufficient to give a c_m value low enough to make the action unstable. In this case the solution, as for other learning automata, is to make the automaton less optimal. This causes the automaton to enter states which correspond to the now smaller penalty probability allowing the ADDIE to fall. This decreases c_m and makes both actions less stable until there is again a stable and an unstable action.

In Figures 3.16(a) and (b) the effect of making the automata less optimal can be seen. Conditions in Figure 3.16 are the same as those in Figure 3.15 except that the memory size has been decreased. It can be seen that the automata respond better to the change in the environment though of course the steady state performance has been reduced.

Though the use of ADDIEs with a small number of bits may seem desirable, in that it decreases the learning time of the automaton, it introduces an undesirable effect. Figure 3.12(b) is an example of this and shows a significant delay between the switch and a change in the action of the automaton. This can occur with either of the modified Tsetlin automata and has two causes. The first is the value of c_m which is too large causing the actions of the automata to be too stable so increasing the mean switching time. The large c_m is due to inaccurate penalty probability estimates at the time of the switch in the environment. This can be caused either by variance in the ADDIEs or as a result of a short ADDIE response time. When the environment switches the ADDIE associated with the action the automaton is taking receives a higher penalty probability and starts

to move towards this higher value. If the memory size is large, relative to the ADDIE size, before the automaton has time to change the action it is taking the ADDIE will have moved significantly upwards. This raises the value of c_m, making both actions more stable and increasing the mean switching time. The solution to both these causes, the high variance and the short response time, is to increase the number of bits in the ADDIEs sufficiently to reduce the variance and increase the response time.

Conclusions

It has been shown that the two modified Tsetlin automata do operate well for all penalty probabilities. The automata retain the short mean switching times and near optimal behaviour like that of the Tsetlin automaton but without the limitations on the values of the penalty probabilities. The most striking feature is the ability in many environments to change actions in response to a switch without having to sample the non-optimal action. The operational difficulties of the automata have also been discovered in the form of the relationship between the ADDIE and counter response times and the environments in which the switching times are longer. This latter problem is one shared by all automata operating in non-stationary autonomous environments.

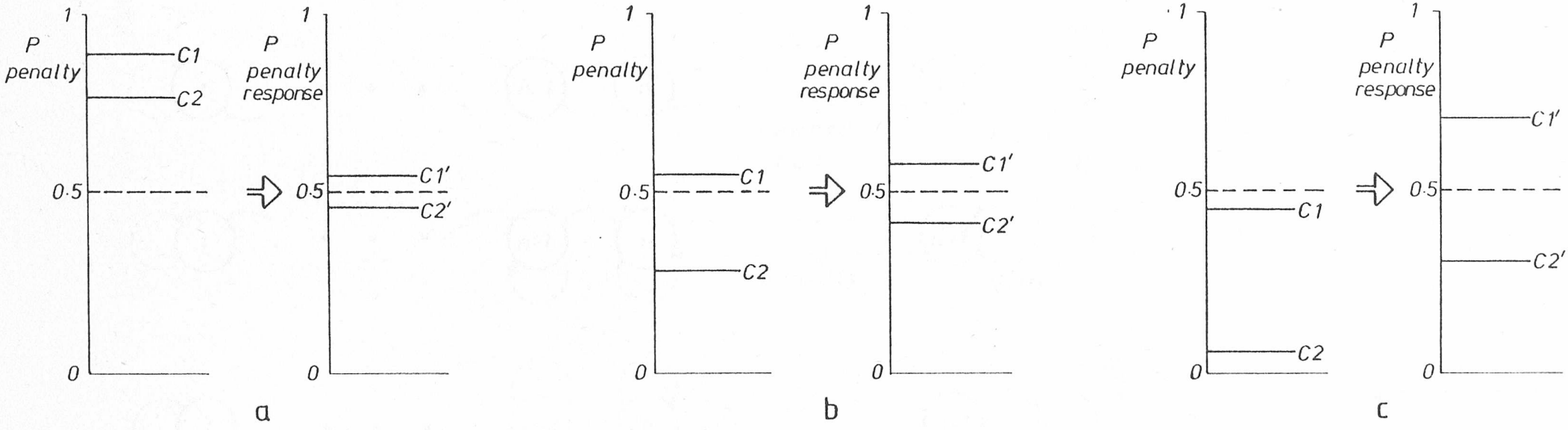


Figure 3·1 The effect of the modified Tsetlin automata on penalty probabilities

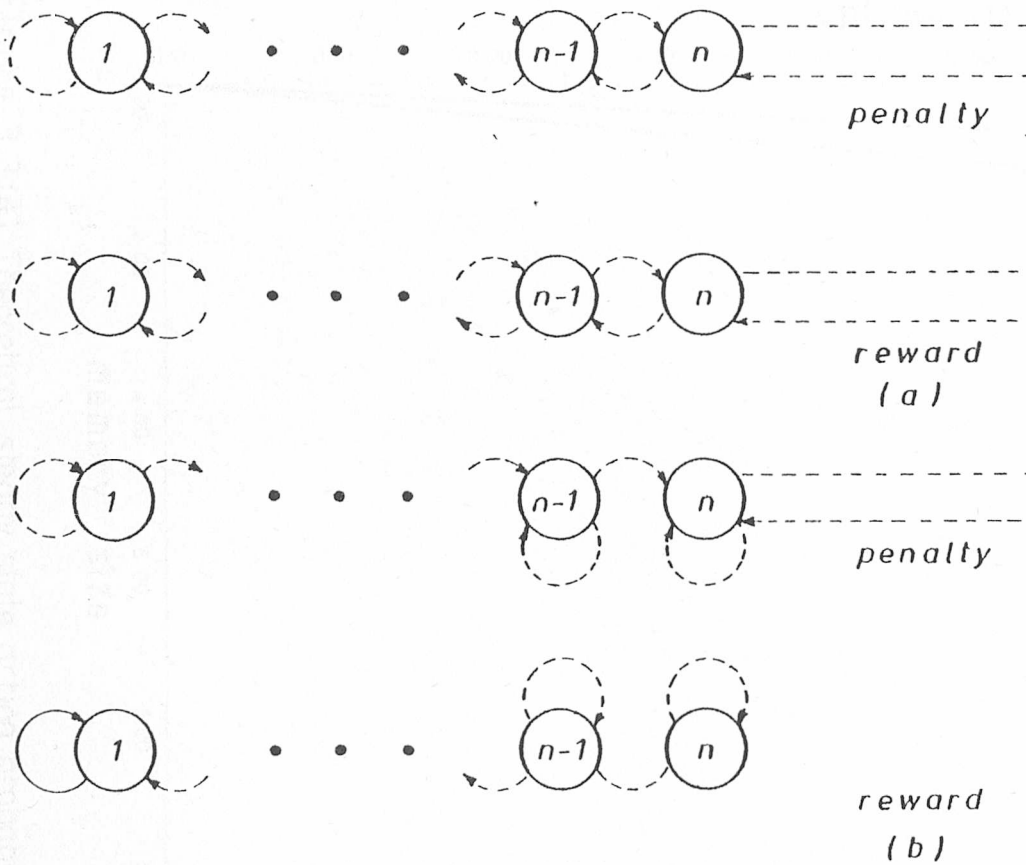
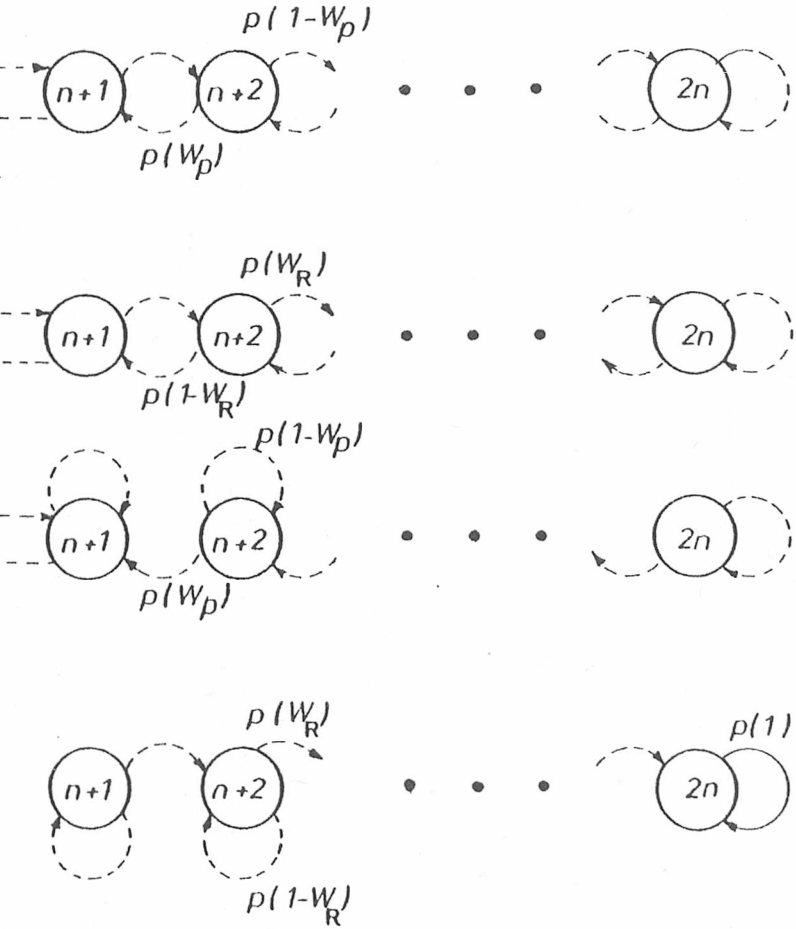


Figure 3.2 State diagrams of



modified Tsetlin automata

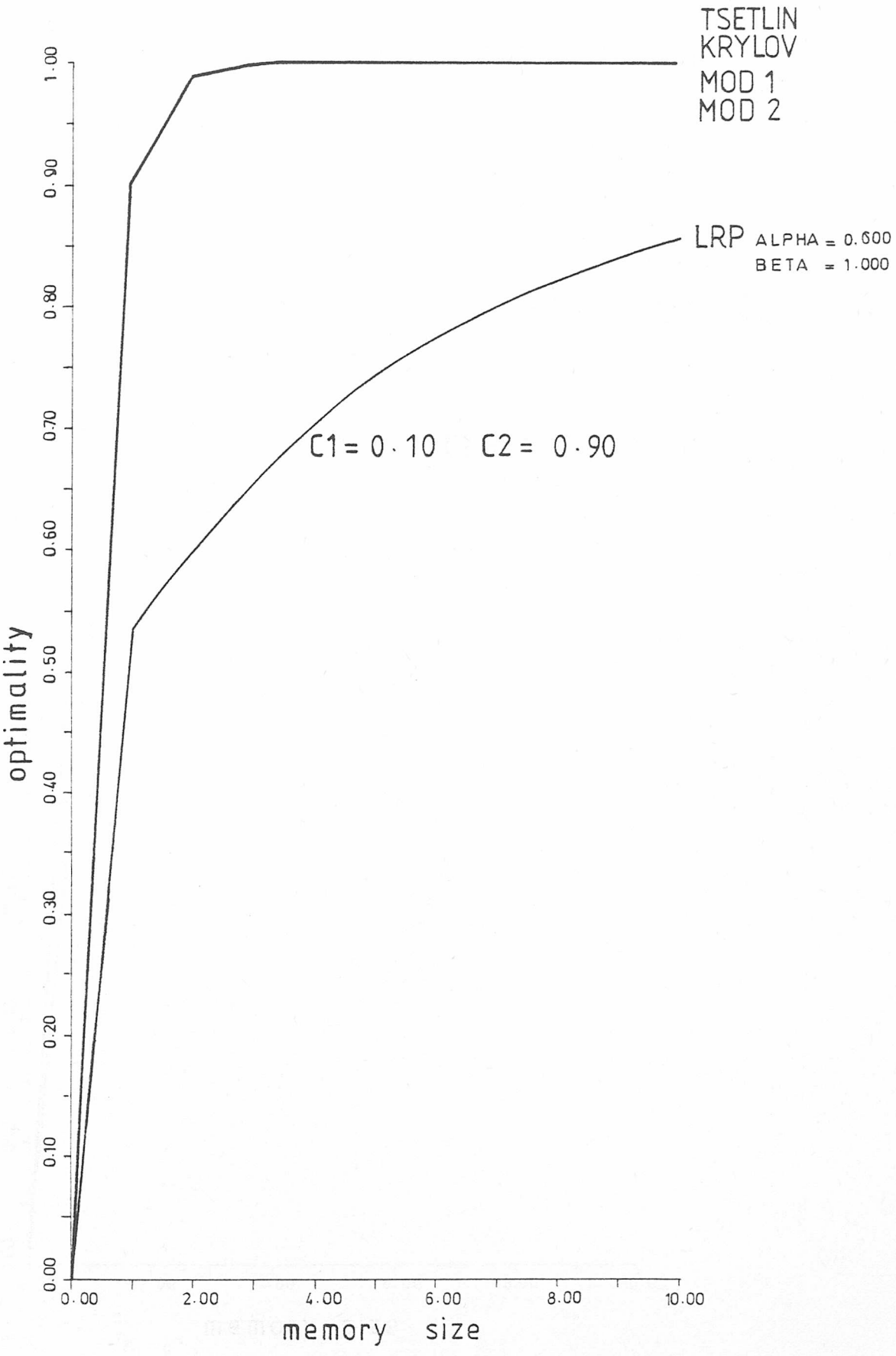


Figure 3.3(a) Theoretical steady state action probabilities

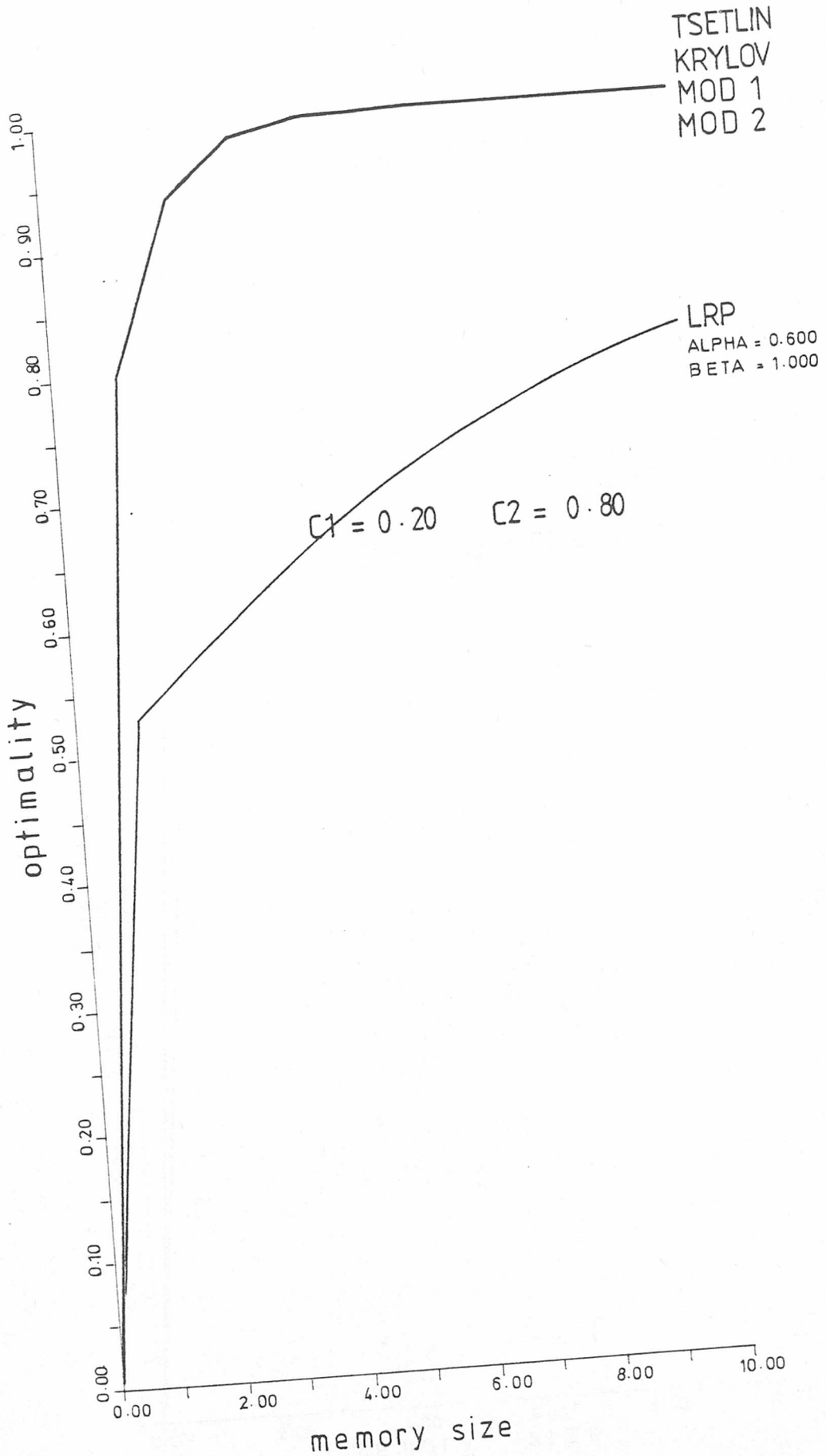

TSETLIN
KRYLOV
MOD 1
MOD 2
LRP
ALPHA = 0.600
BETA = 1.000
C1 = 0.20 C2 = 0.80
optimality
memory size
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 3.3(b)

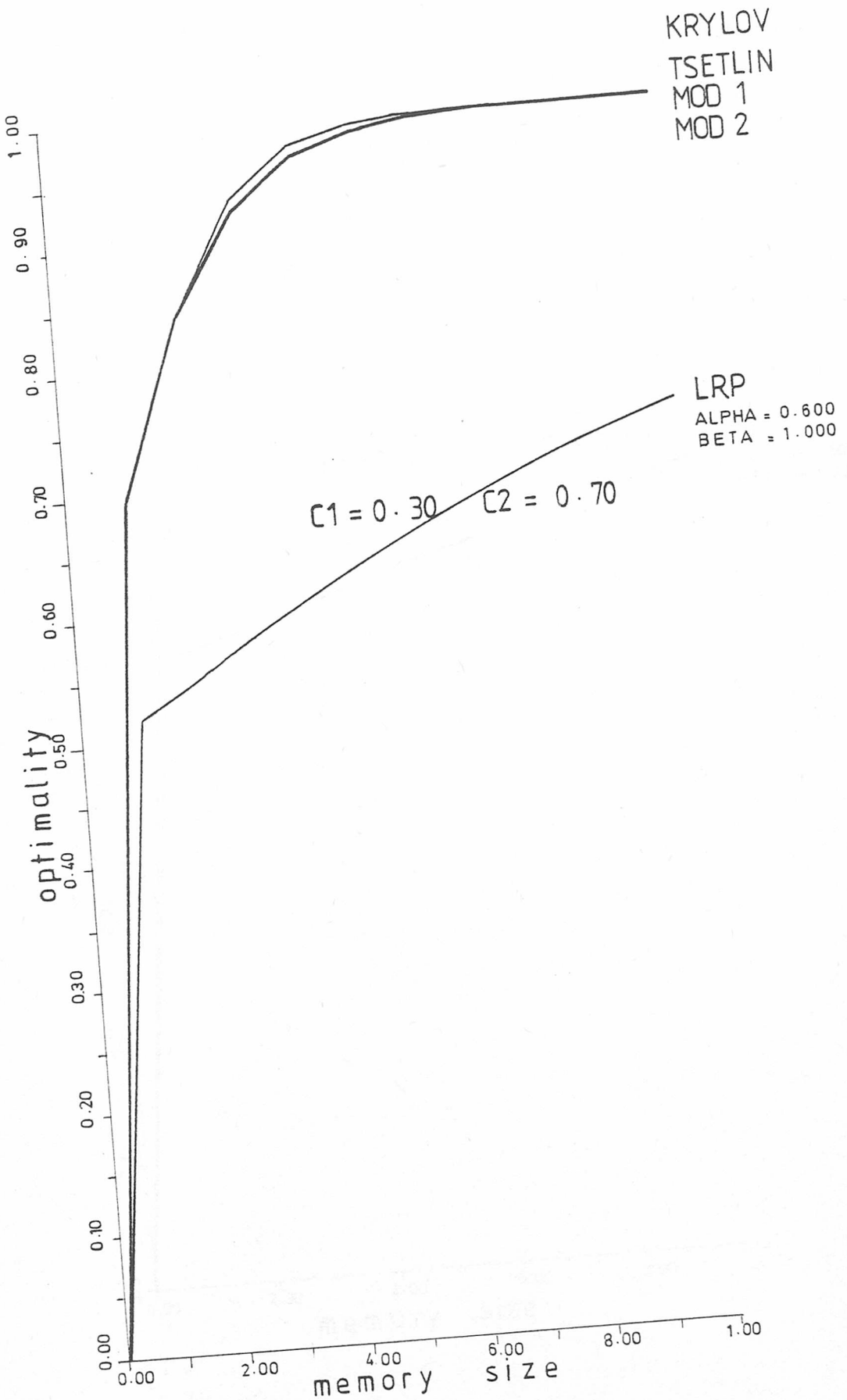

KRYLOV
TSETLIN
MOD 1
MOD 2
LRP
ALPHA = 0.600
BETA = 1.000
C1 = 0.30
C2 = 0.70
optimality
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
2.00
4.00
6.00
8.00
1.00
memory size

Figure 3.3(c)

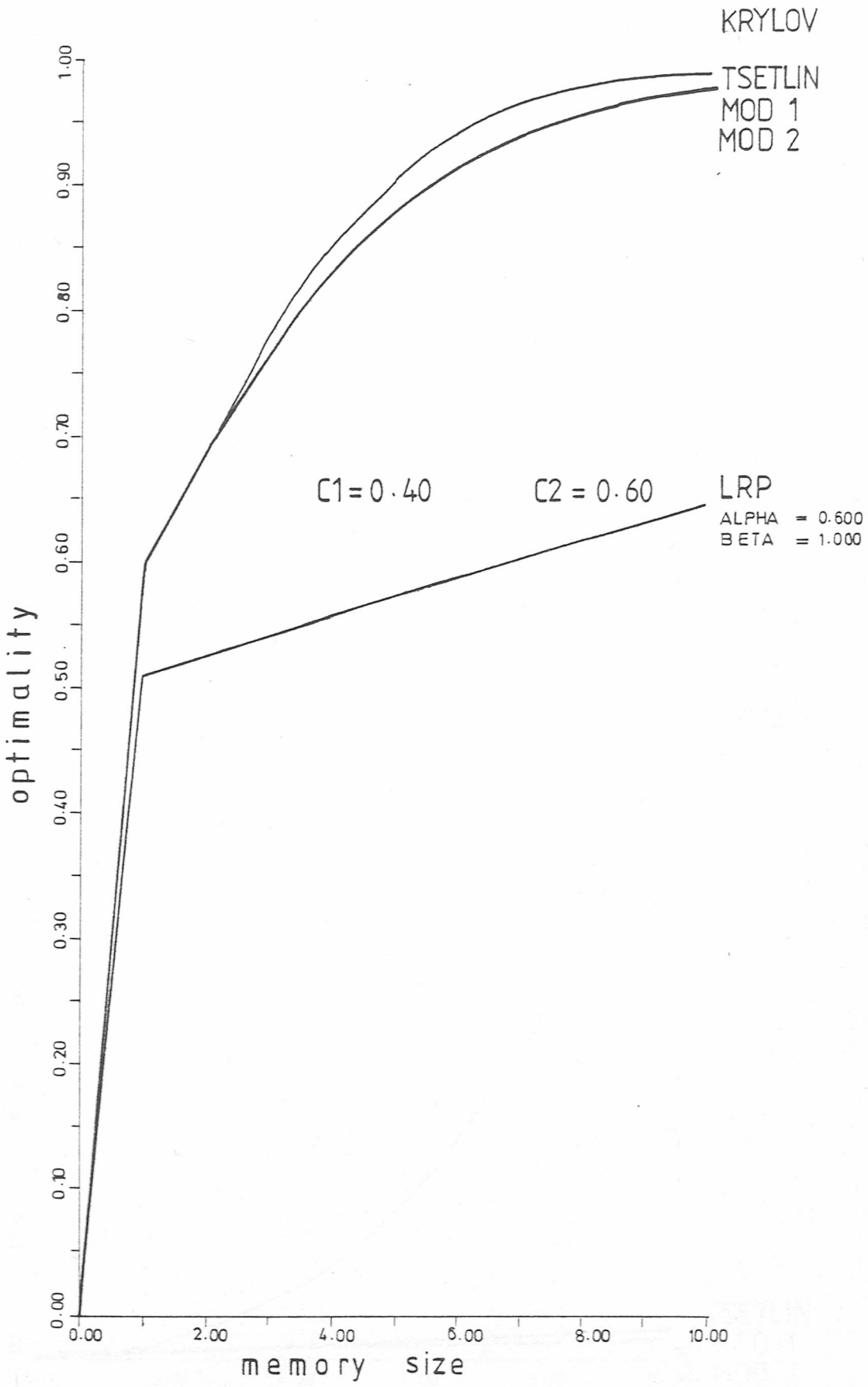

KRYLOV
TSETLIN
MOD 1
MOD 2
C1 = 0.40
C2 = 0.60
LRP
ALPHA = 0.600
BETA = 1.000
optimality
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 3.3(d)

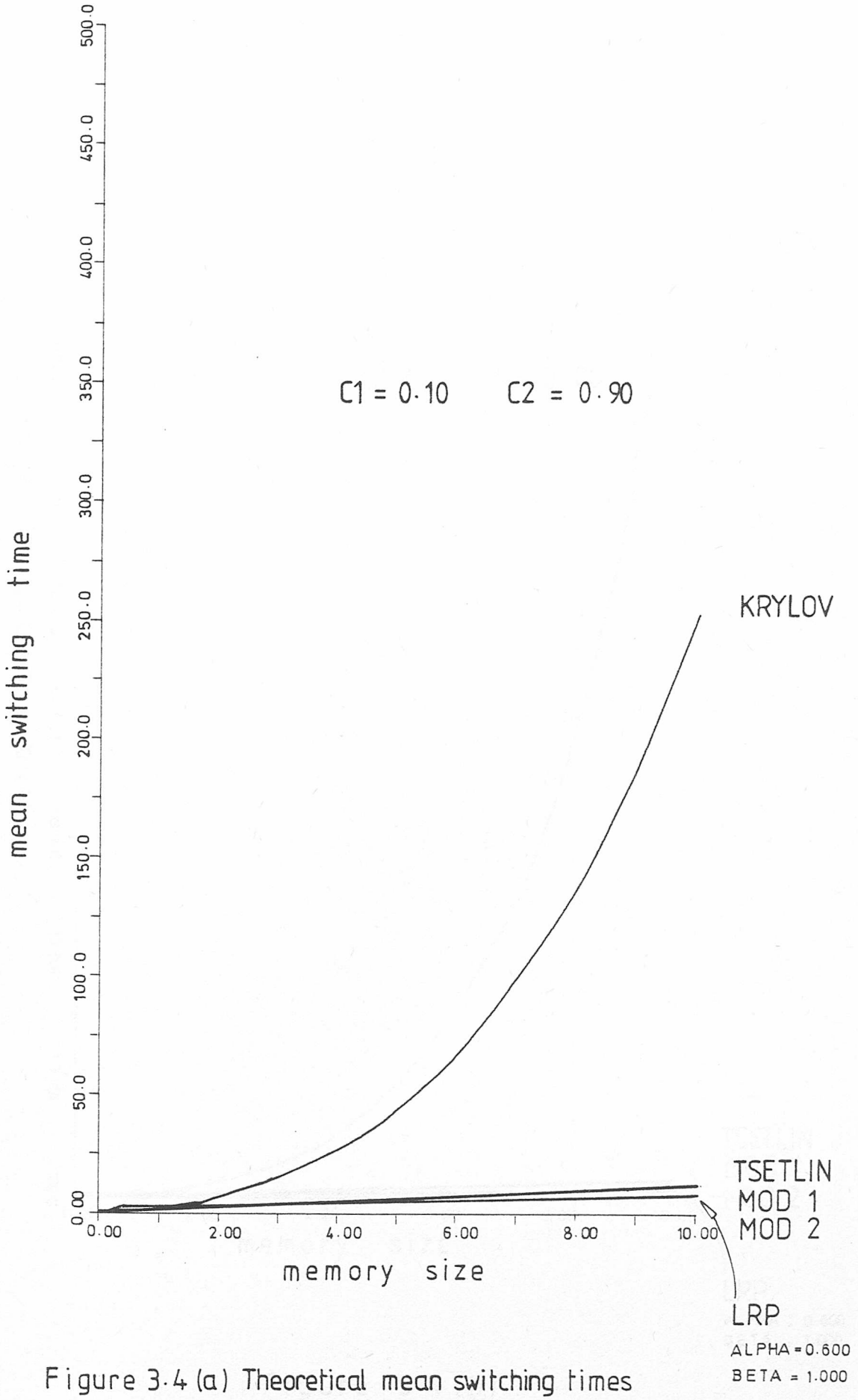


Figure 3.4 (a) Theoretical mean switching times

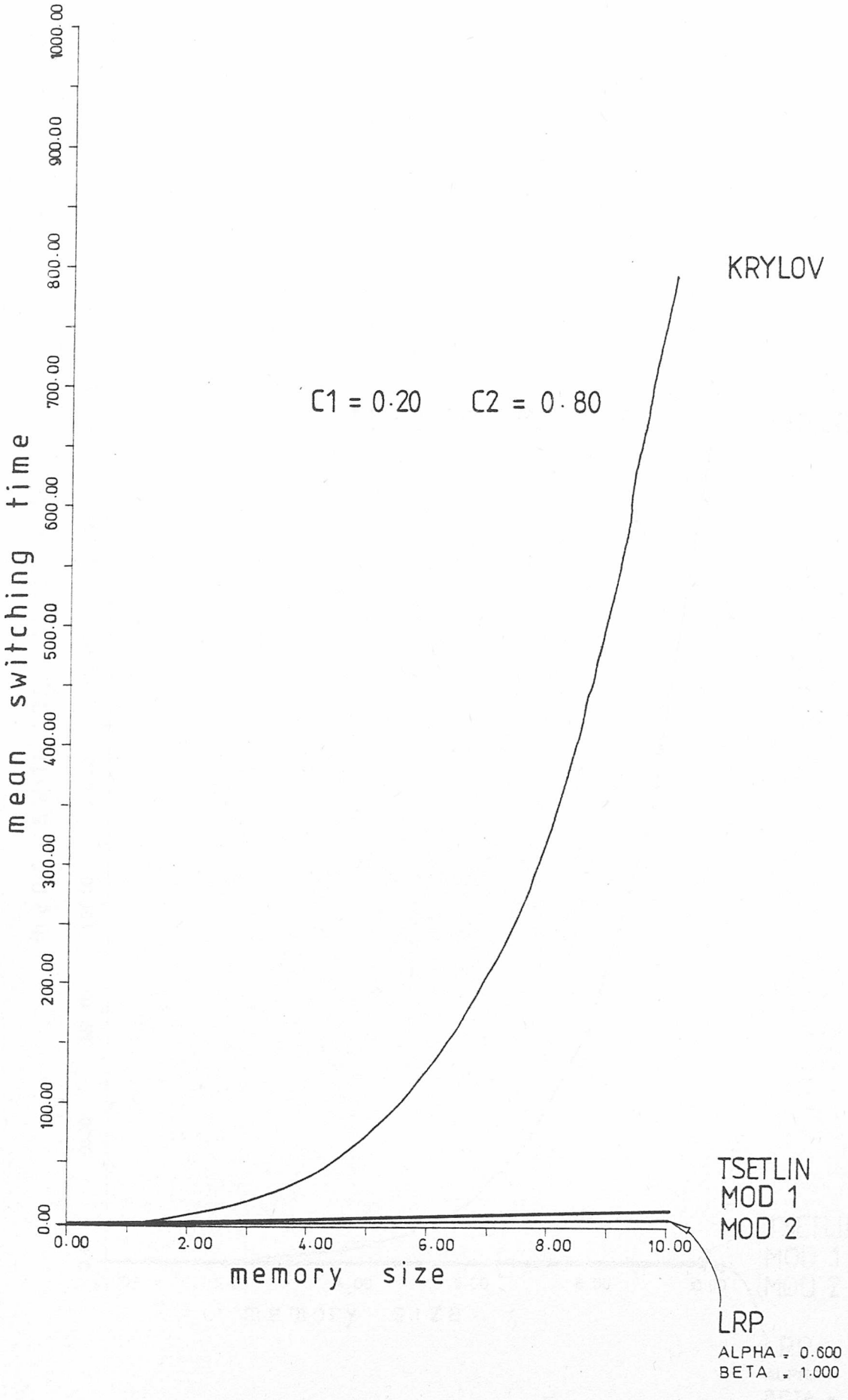

mean switching time
1000.00
900.00
800.00
700.00
600.00
500.00
400.00
300.00
200.00
100.00
0.00
C1 = 0·20 C2 = 0·80
KRYLOV
TSETLIN
MOD 1
MOD 2
LRP
ALPHA = 0.600
BETA = 1.000
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 3·4(b)

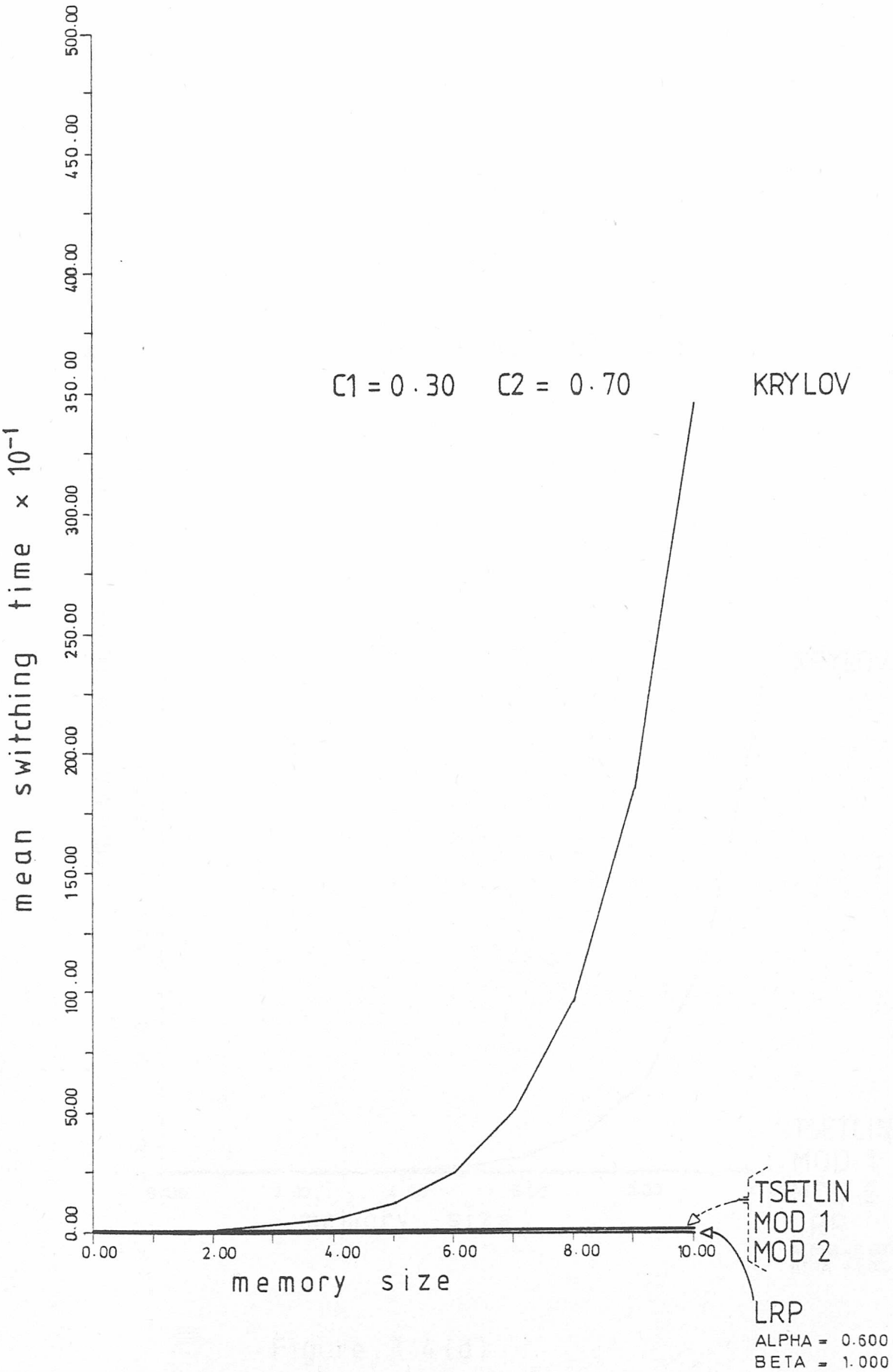

C1 = 0.30 C2 = 0.70
KRYLOV
mean switching time × 10^{-1}
0.00
50.00
100.00
150.00
200.00
250.00
300.00
350.00
400.00
450.00
500.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size
TSETLIN
MOD 1
MOD 2
LRP
ALPHA = 0.600
BETA = 1.000

Figure 3.4(c)

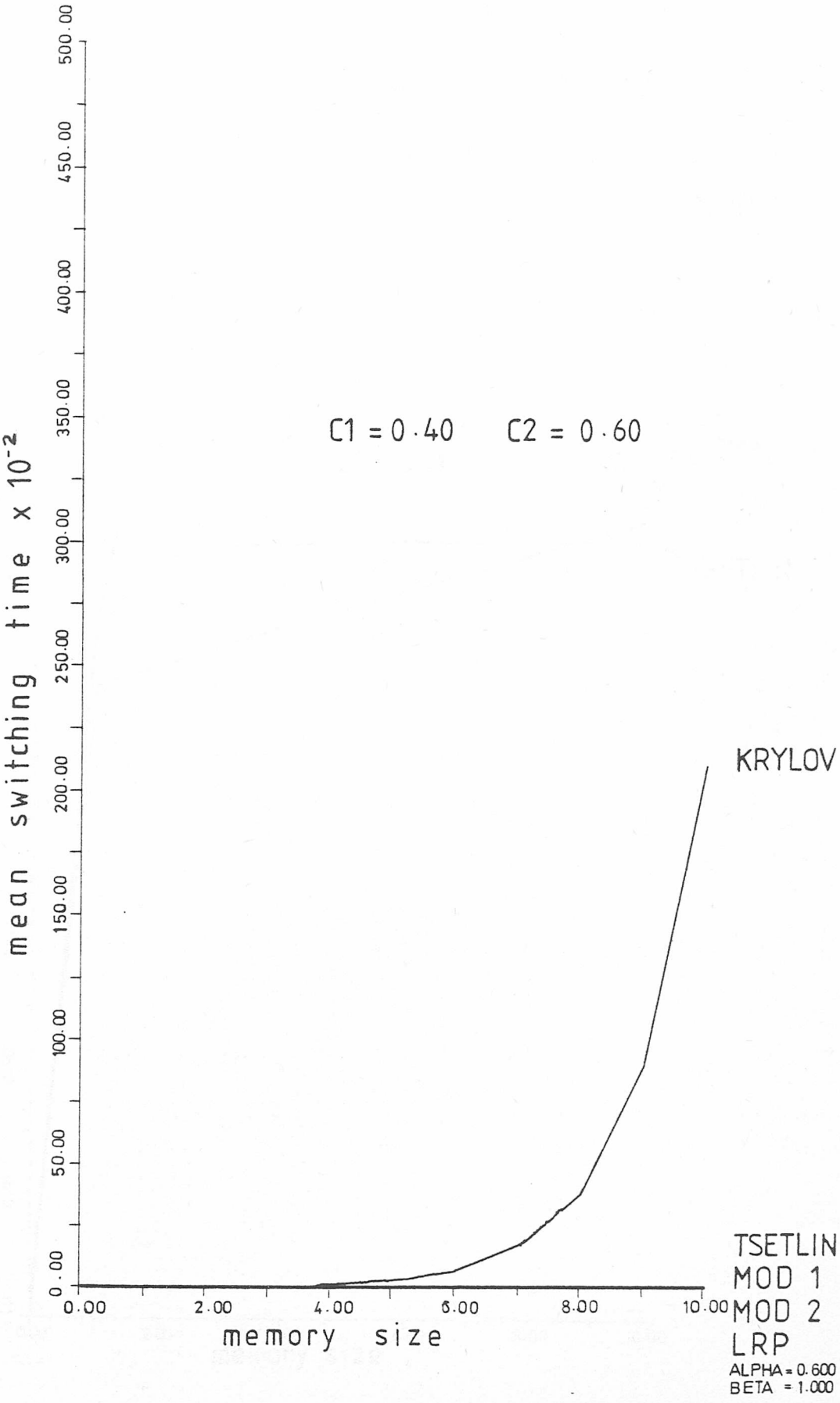

C1 = 0·40 C2 = 0·60
mean switching time x 10^{-2}
0.00
50.00
100.00
150.00
200.00
250.00
300.00
350.00
400.00
450.00
500.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size
KRYLOV
TSETLIN
MOD 1
MOD 2
LRP
ALPHA = 0.600
BETA = 1.000

Figure 3·4(d)

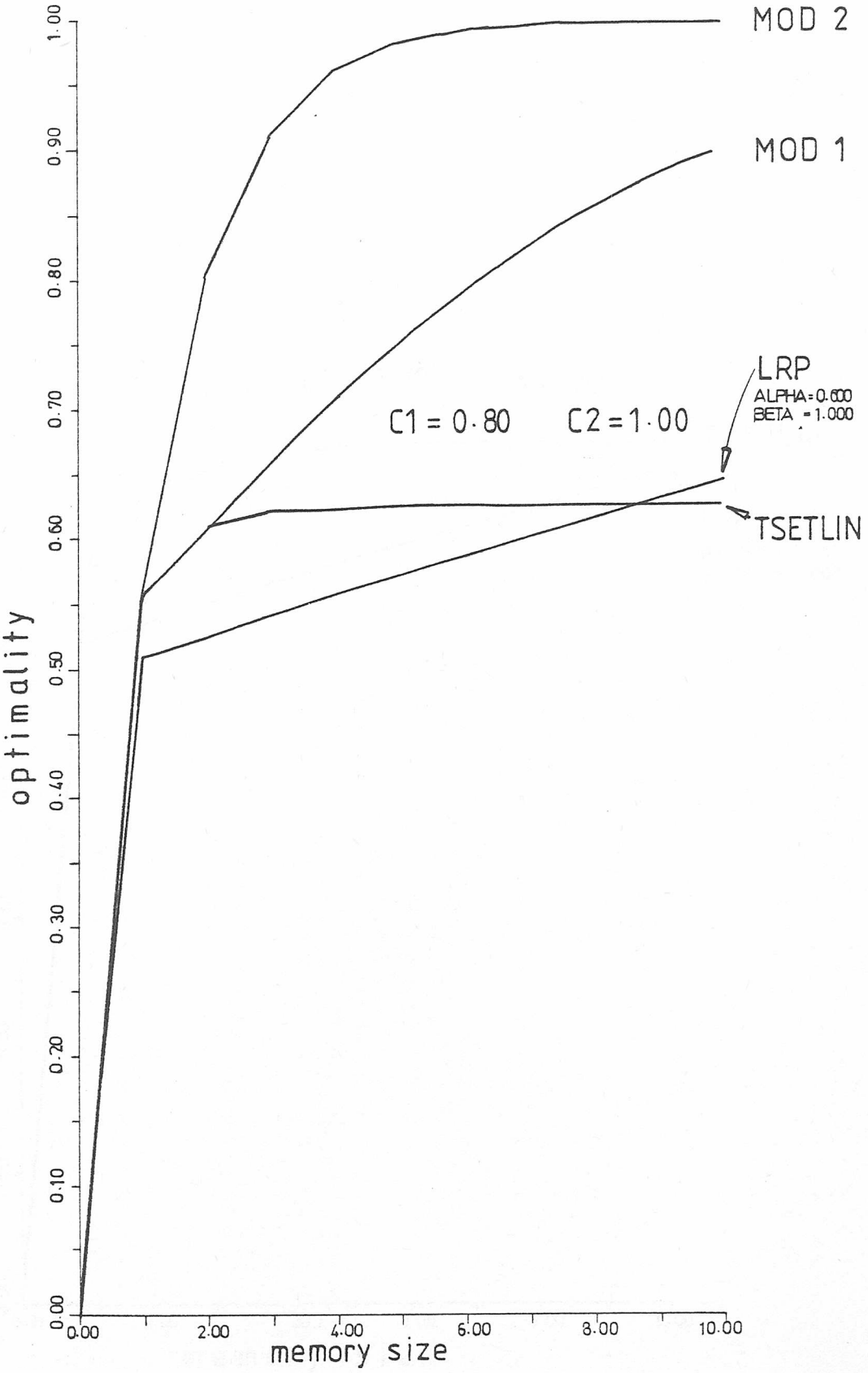


Figure 3.5 (a) Theoretical steady state action probabilities

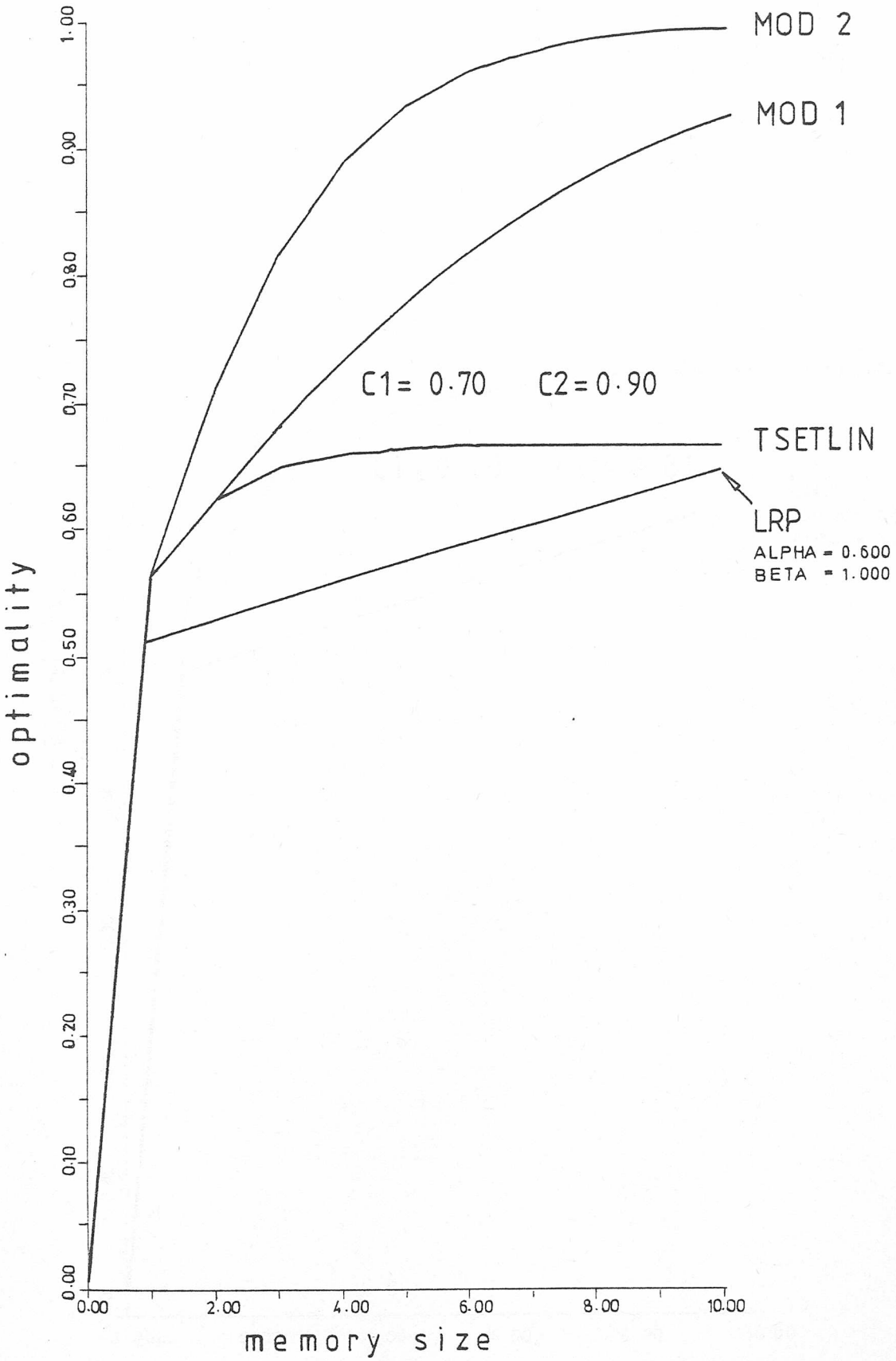

MOD 2
MOD 1
C1= 0.70 C2=0.90
TSETLIN
LRP
ALPHA = 0.600
BETA = 1.000
optimality
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 3.5(b)

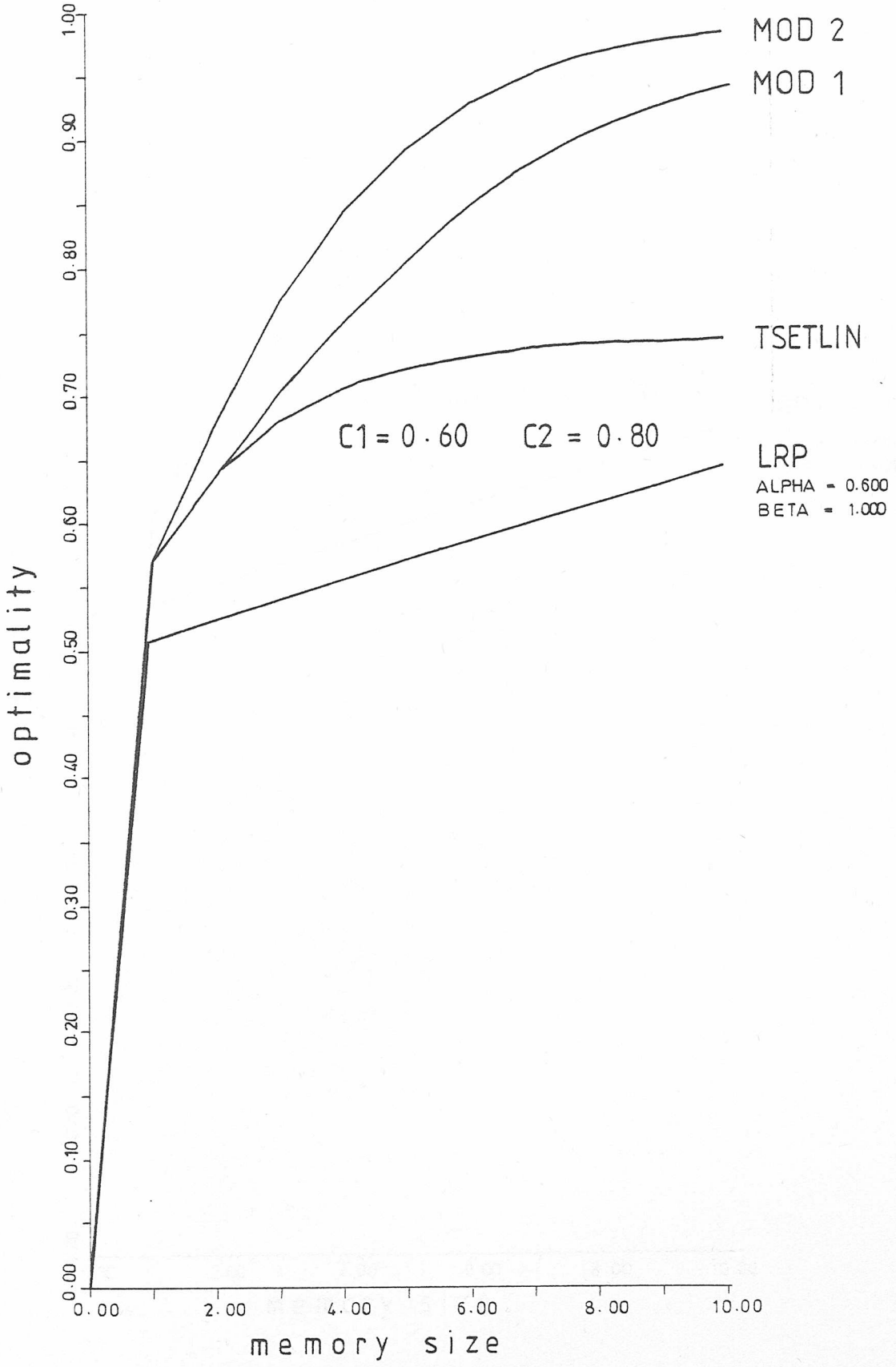

MOD 2
MOD 1
TSETLIN
C1 = 0·60 C2 = 0·80
LRP
ALPHA = 0.600
BETA = 1.000
optimality
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 3·5(c)

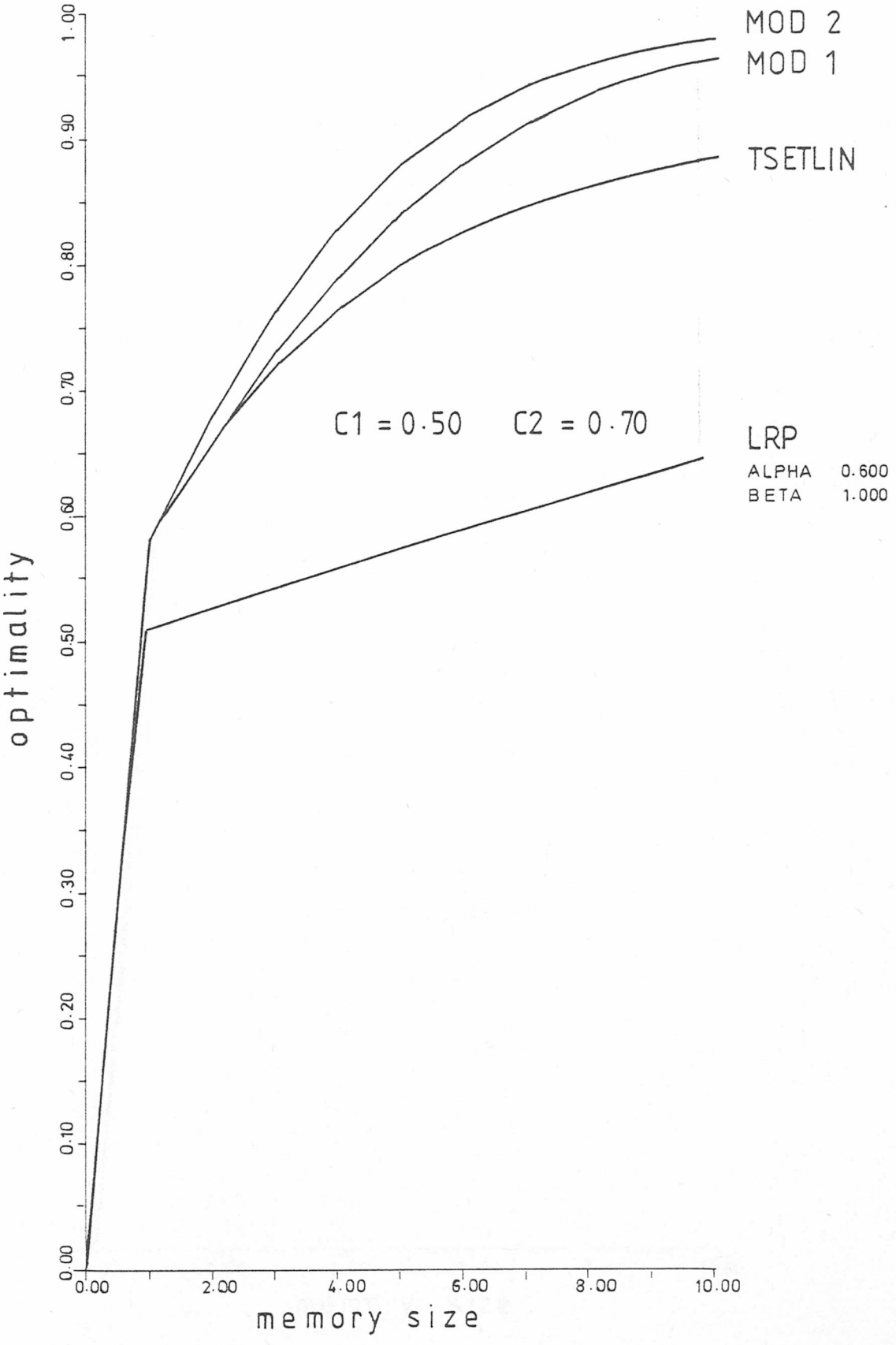

MOD 2
MOD 1
TSETLIN
C1 = 0·50 C2 = 0·70
LRP
ALPHA 0.600
BETA 1.000
optimality
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 3·5(d)

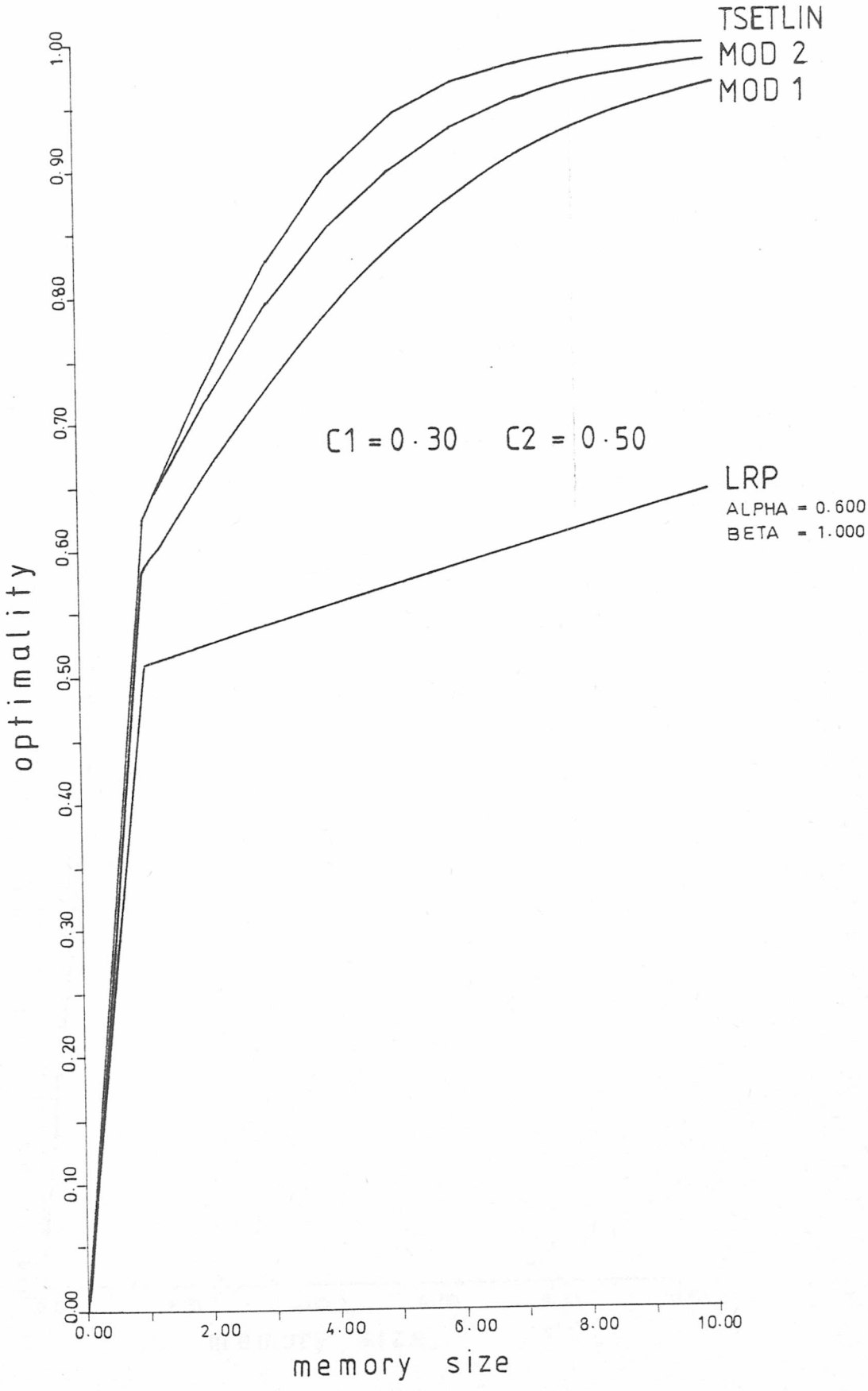

TSETLIN
MOD 2
MOD 1
C1 = 0·30 C2 = 0·50
LRP
ALPHA = 0.600
BETA = 1.000
optimality
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 3·5(e)

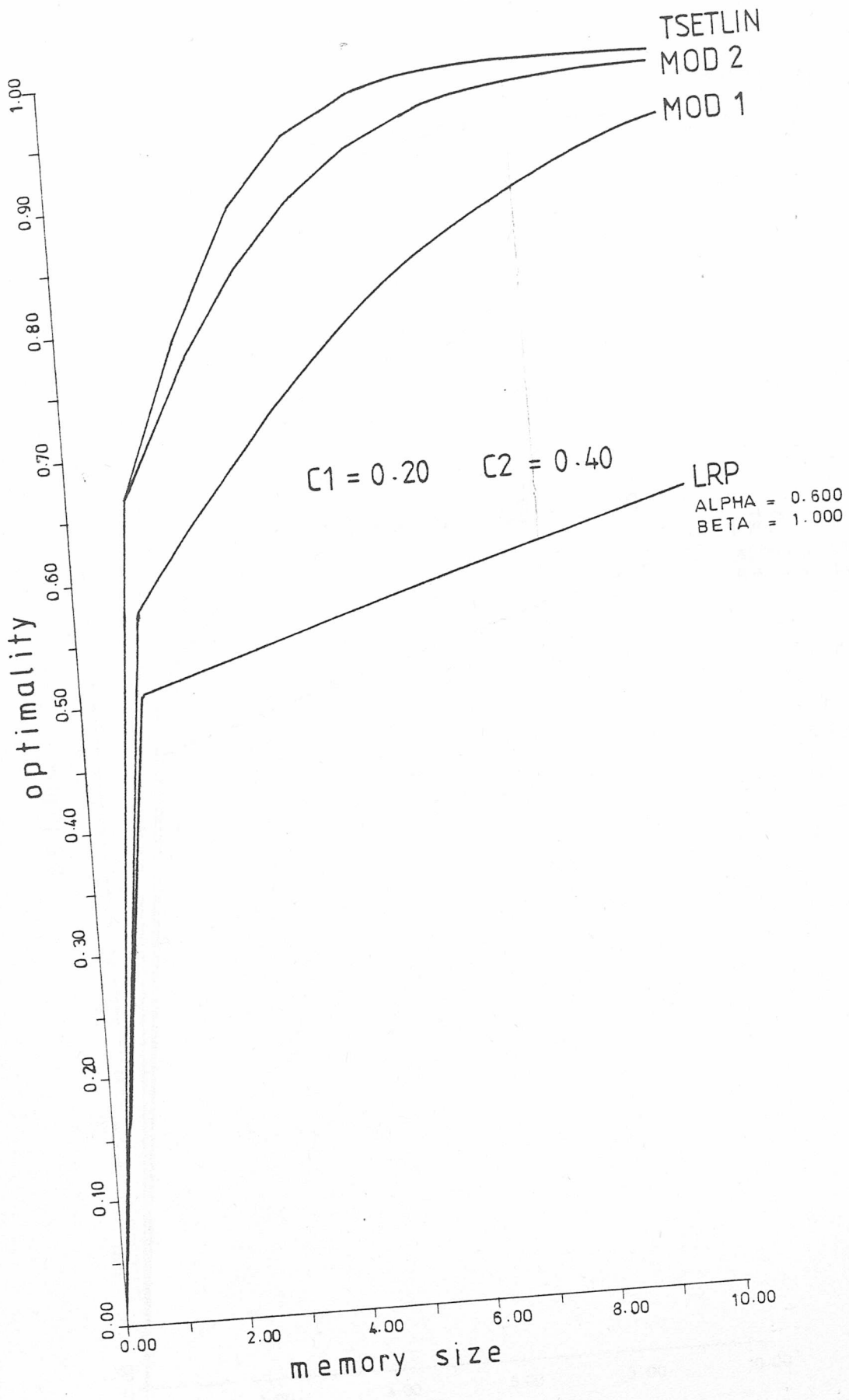

TSETLIN
MOD 2
MOD 1
C1 = 0.20 C2 = 0.40
LRP
ALPHA = 0.600
BETA = 1.000
optimality
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 3.5(f)

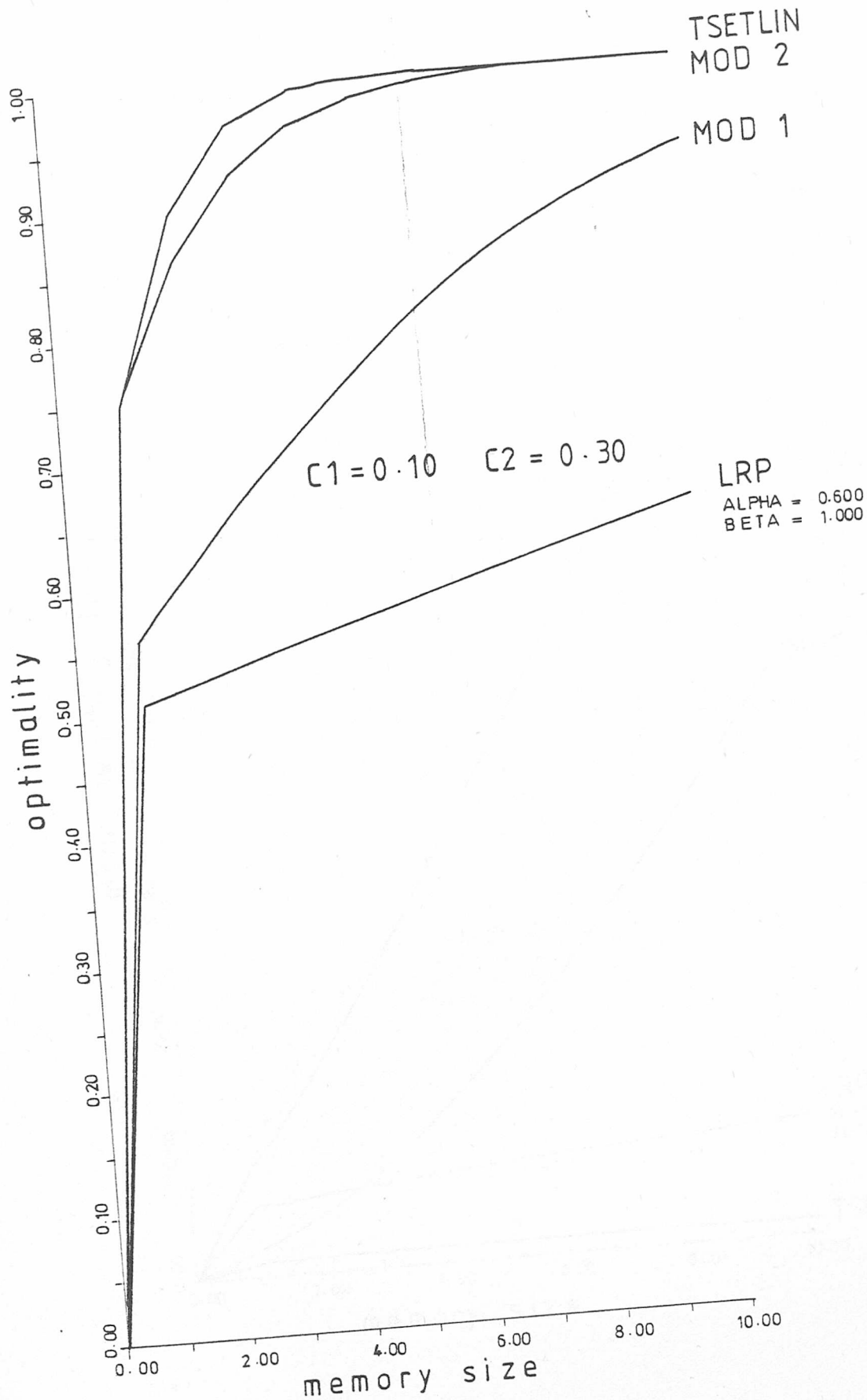

TSETLIN
MOD 2
MOD 1
C1 = 0·10 C2 = 0·30
LRP
ALPHA = 0.600
BETA = 1.000
optimality
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 3.5(g)

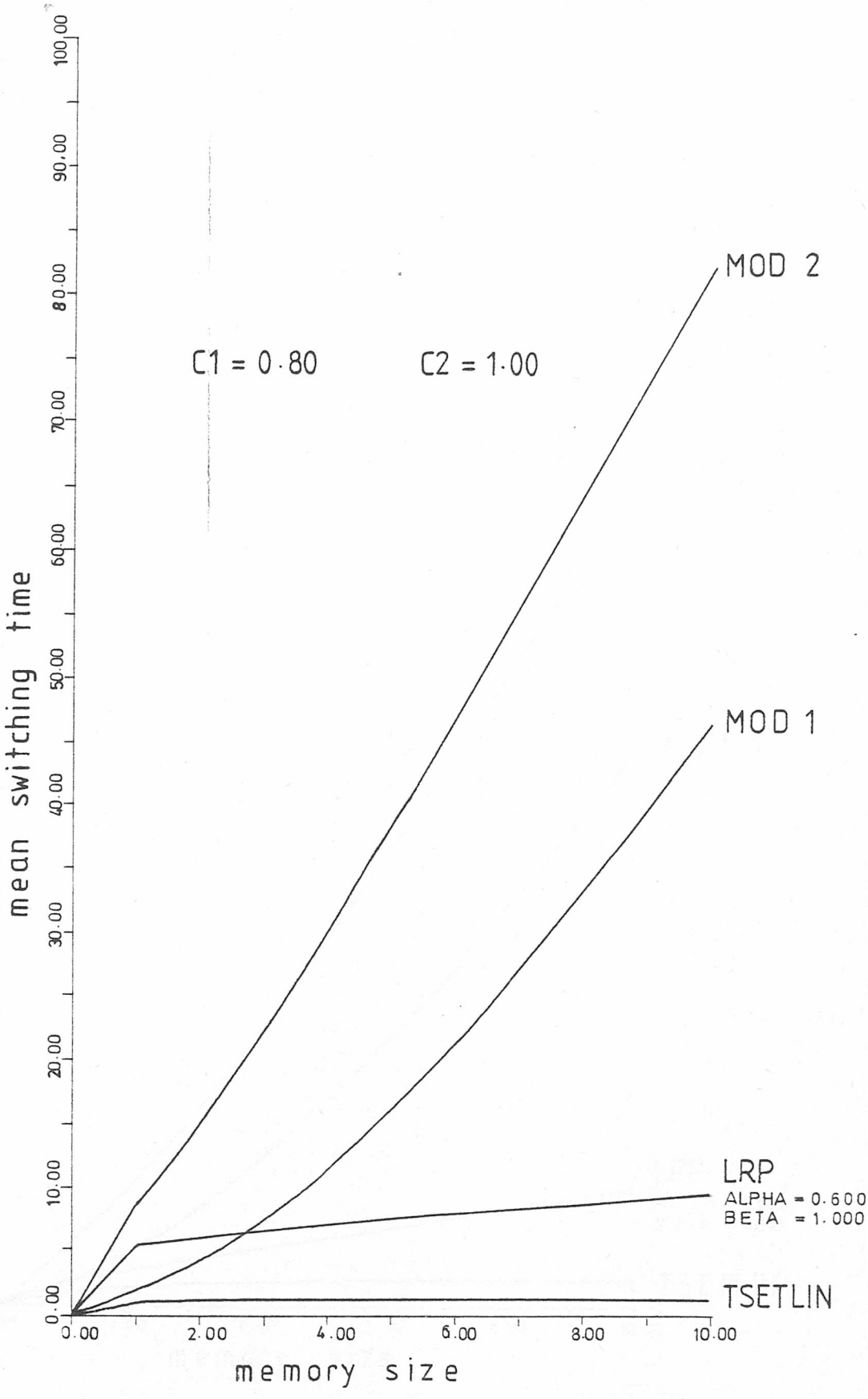


Figure 3·6(a) Theoretical mean switching times

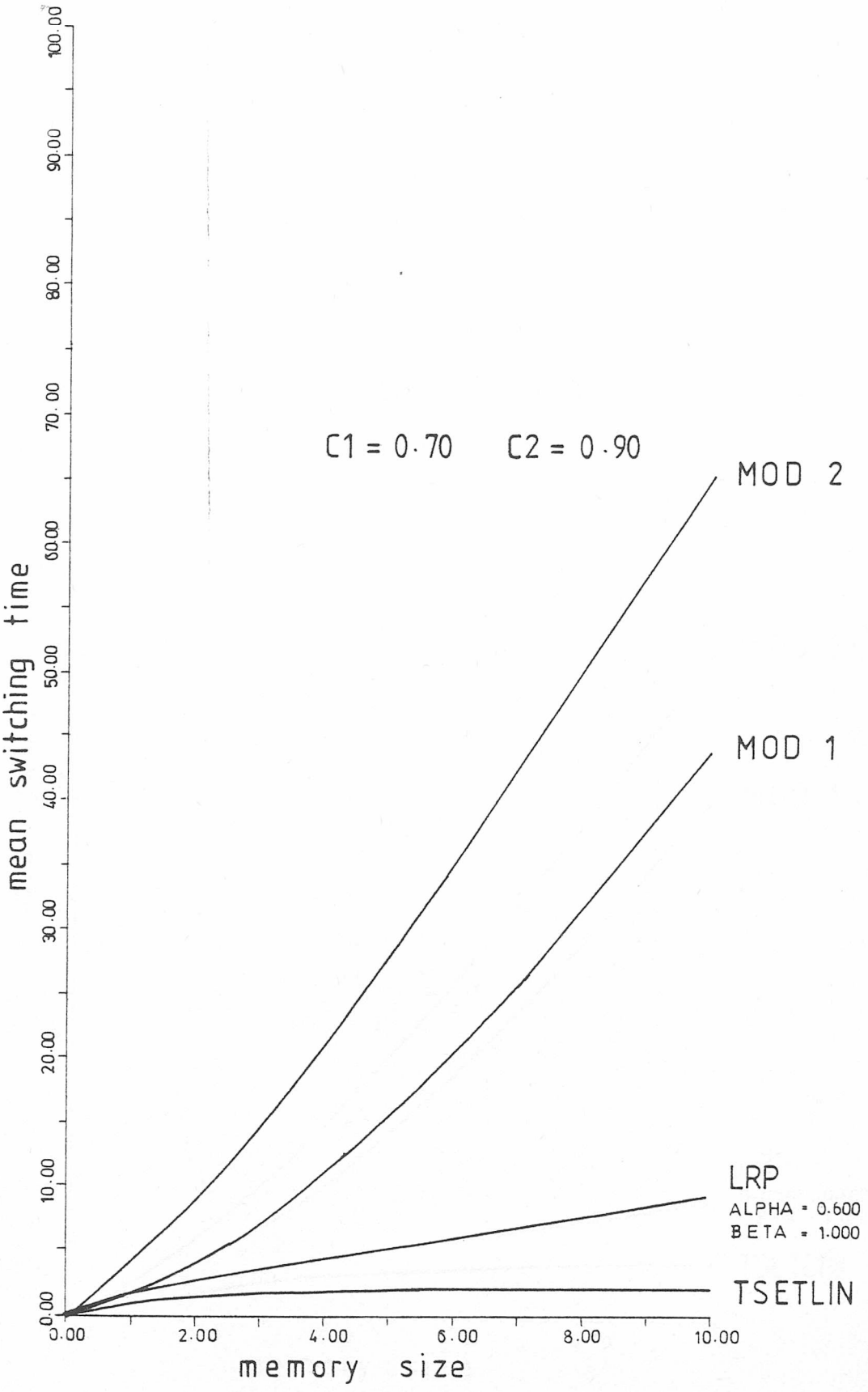

C1 = 0·70 C2 = 0·90
MOD 2
MOD 1
LRP
ALPHA = 0.600
BETA = 1.000
TSETLIN
mean switching time
memory size
0.00
10.00
20.00
30.00
40.00
50.00
60.00
70.00
80.00
90.00
100.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 3·6(b)

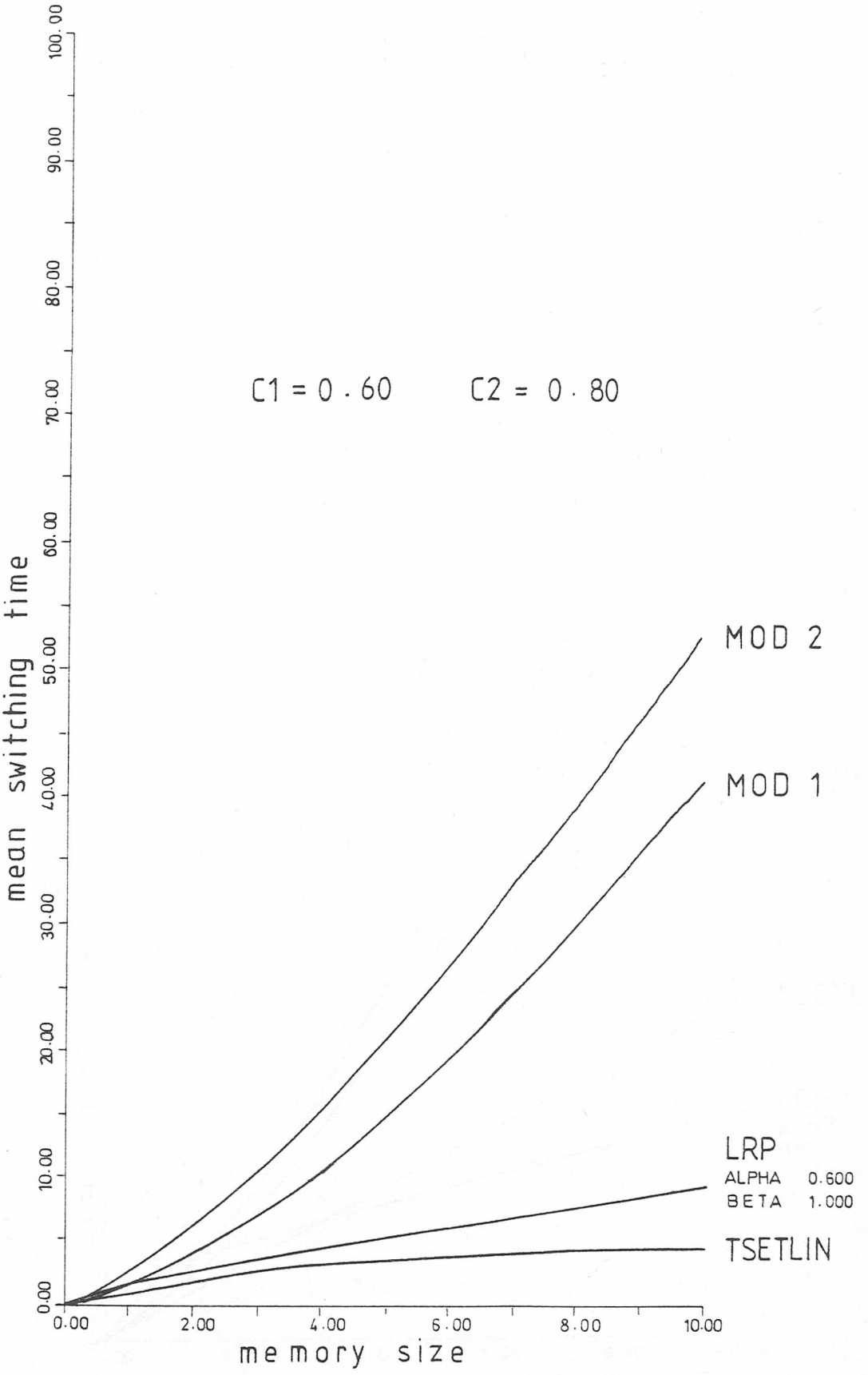

C1 = 0.60
C2 = 0.80
mean switching time
memory size
0.00
10.00
20.00
30.00
40.00
50.00
60.00
70.00
80.00
90.00
100.00
0.00
2.00
4.00
6.00
8.00
10.00
MOD 2
MOD 1
LRP
ALPHA 0.600
BETA 1.000
TSETLIN

Figure 3.6(c)

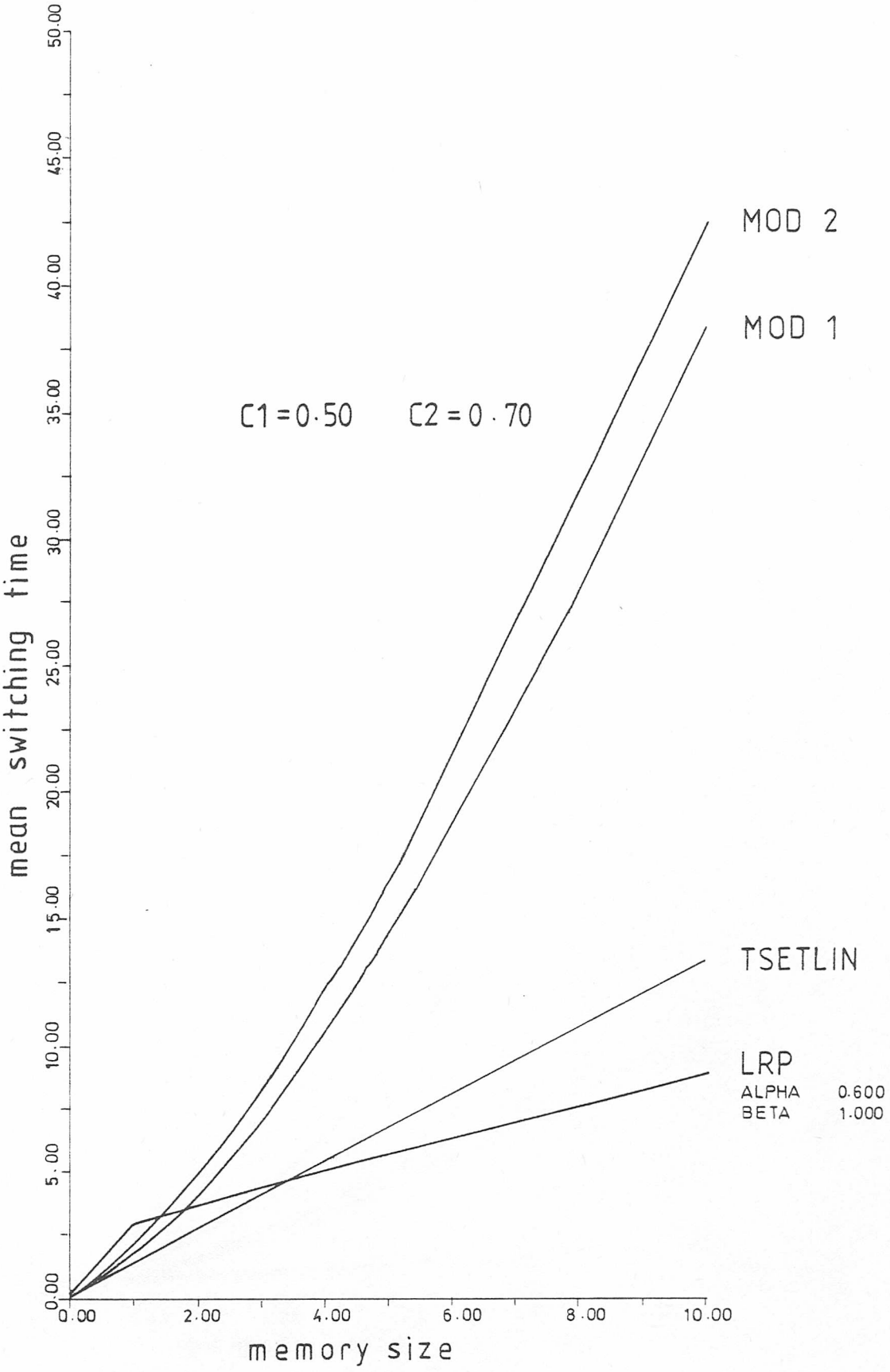

mean switching time
memory size
0.00
5.00
10.00
15.00
20.00
25.00
30.00
35.00
40.00
45.00
50.00
0.00
2.00
4.00
6.00
8.00
10.00
C1 = 0·50
C2 = 0·70
MOD 2
MOD 1
TSETLIN
LRP
ALPHA 0.600
BETA 1.000

Figure 3·6 (d)

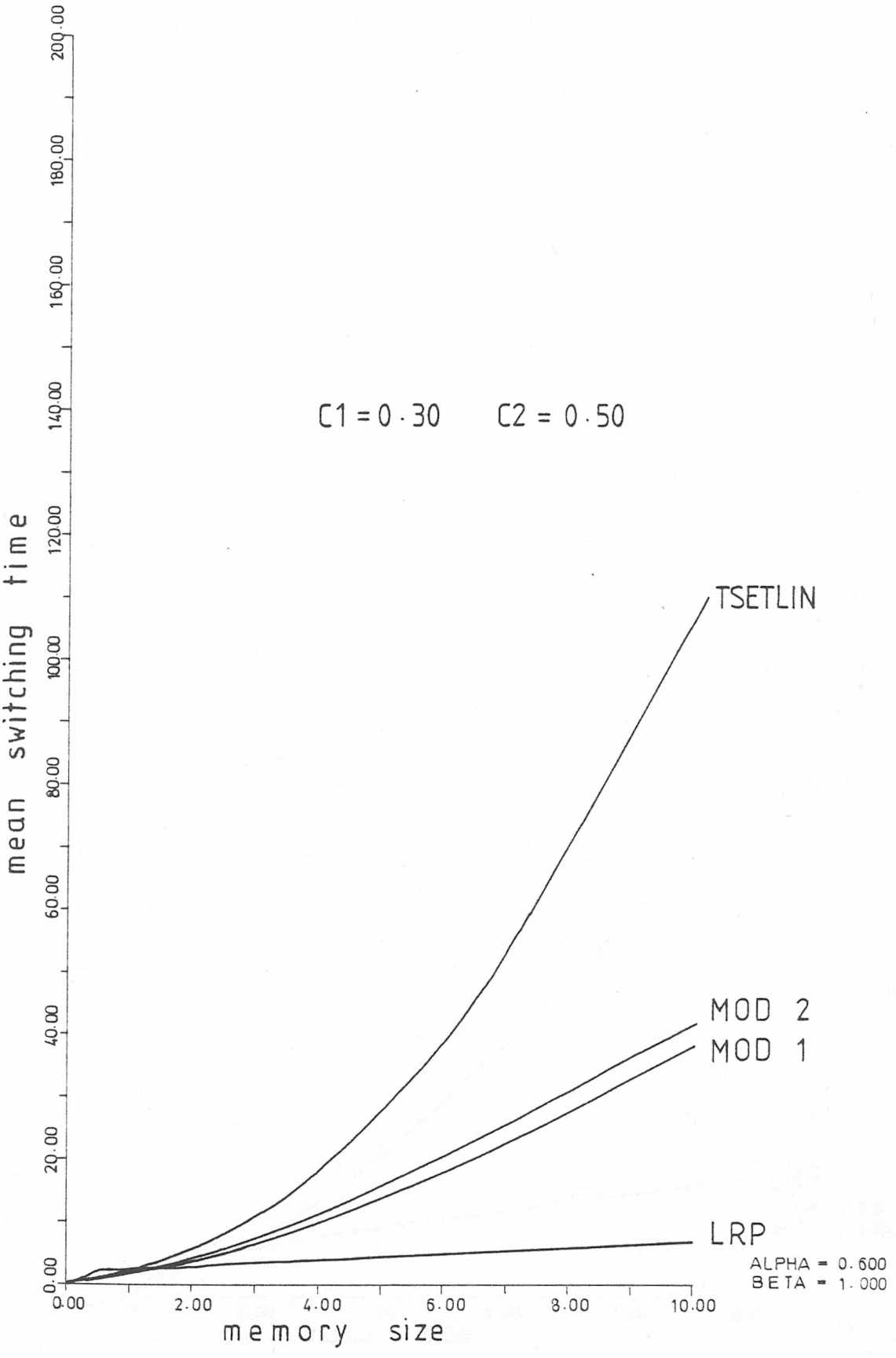

C1 = 0·30 C2 = 0·50
mean switching time
memory size
0.00
20.00
40.00
60.00
80.00
100.00
120.00
140.00
160.00
180.00
200.00
2.00
4.00
6.00
8.00
10.00
TSETLIN
MOD 2
MOD 1
LRP
ALPHA = 0.600
BETA = 1.000

Figure 3.6 (e)

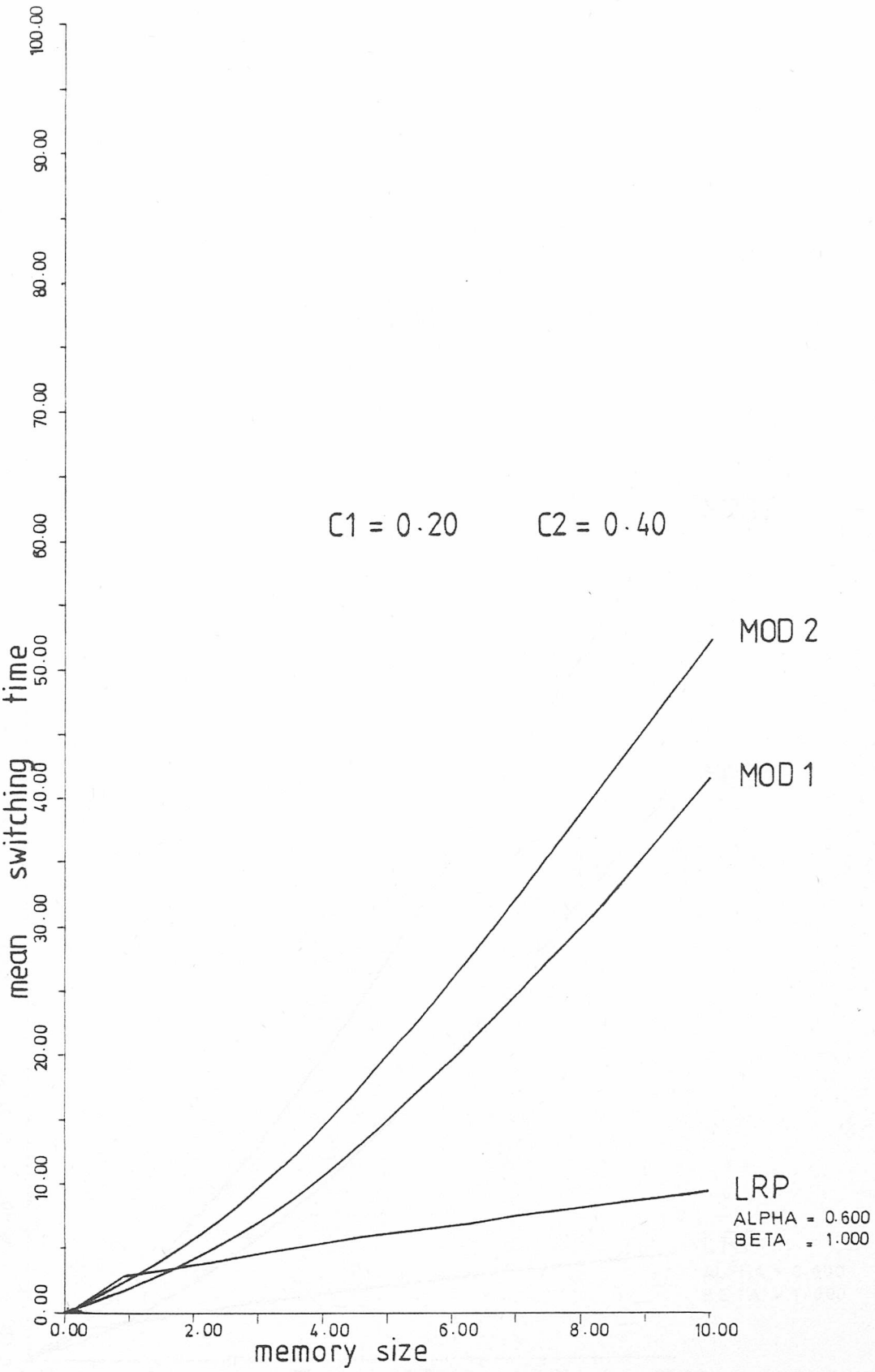

C1 = 0·20
C2 = 0·40
MOD 2
MOD 1
LRP
ALPHA = 0.600
BETA = 1.000
mean switching time
memory size
0.00
10.00
20.00
30.00
40.00
50.00
60.00
70.00
80.00
90.00
100.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 3·6(f)

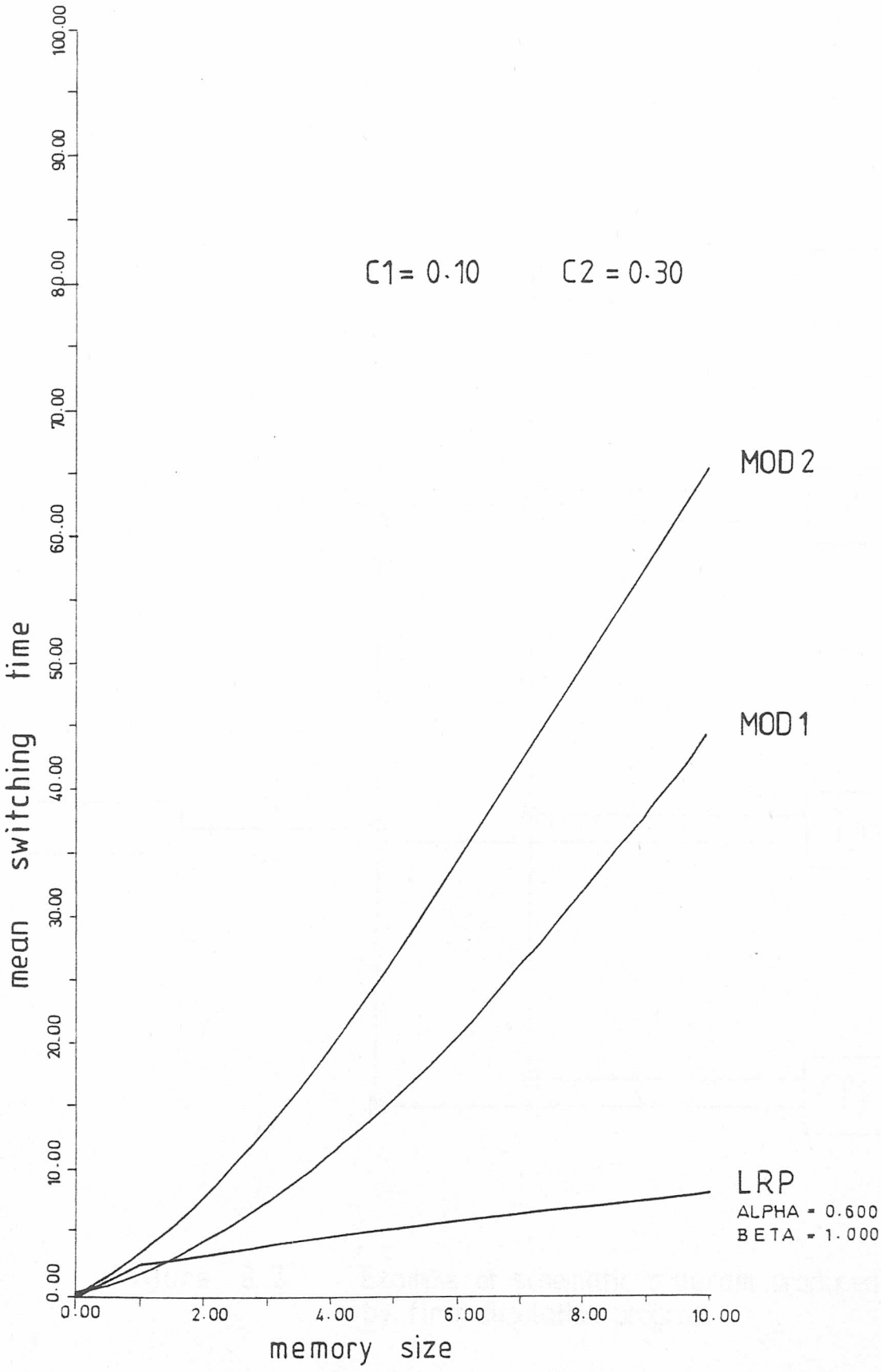

C1 = 0·10
C2 = 0·30
MOD 2
MOD 1
LRP
ALPHA = 0.600
BETA = 1.000
mean switching time
memory size
0.00
10.00
20.00
30.00
40.00
50.00
60.00
70.00
80.00
90.00
100.00
0·00
2.00
4.00
6.00
8.00
10.00

Figure 3.6 (g)

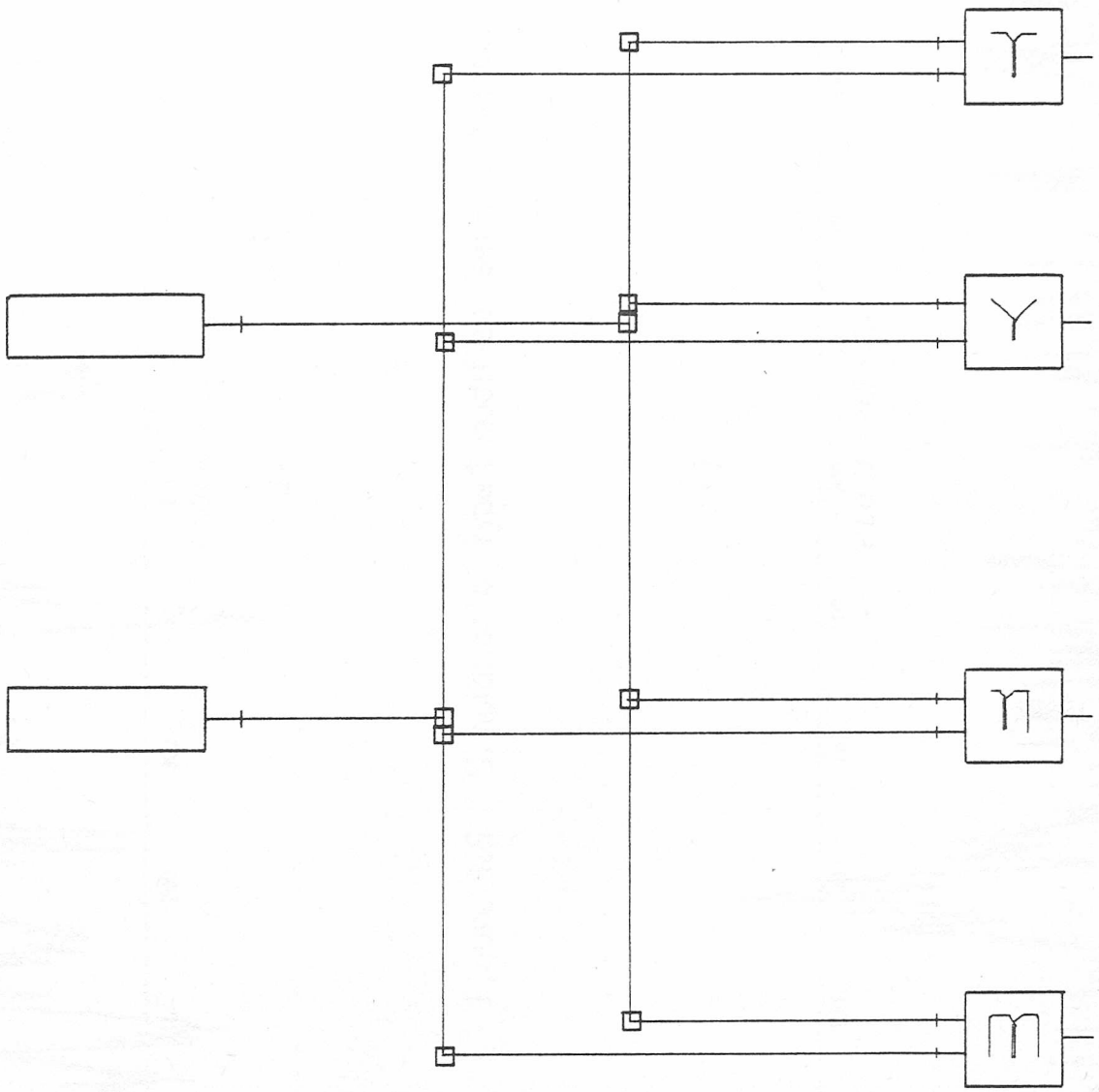
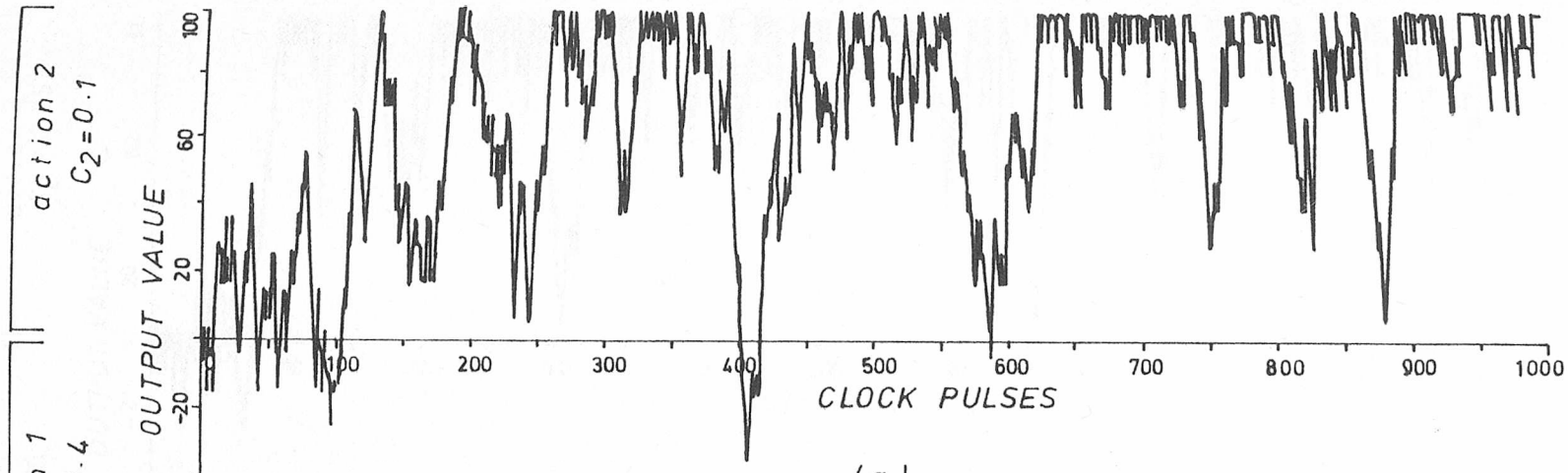
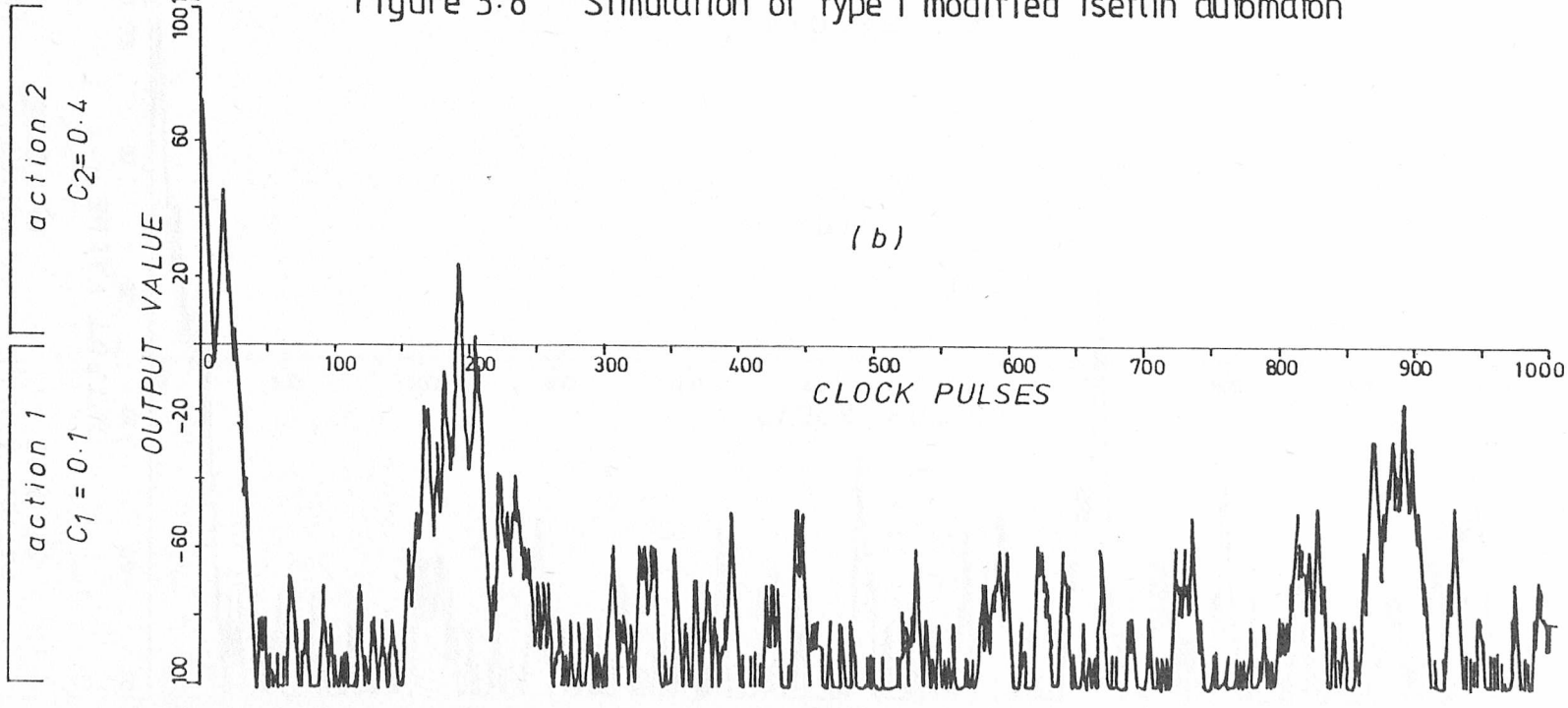


Figure 3·7 Example of schematic diagram produced by first simulation program

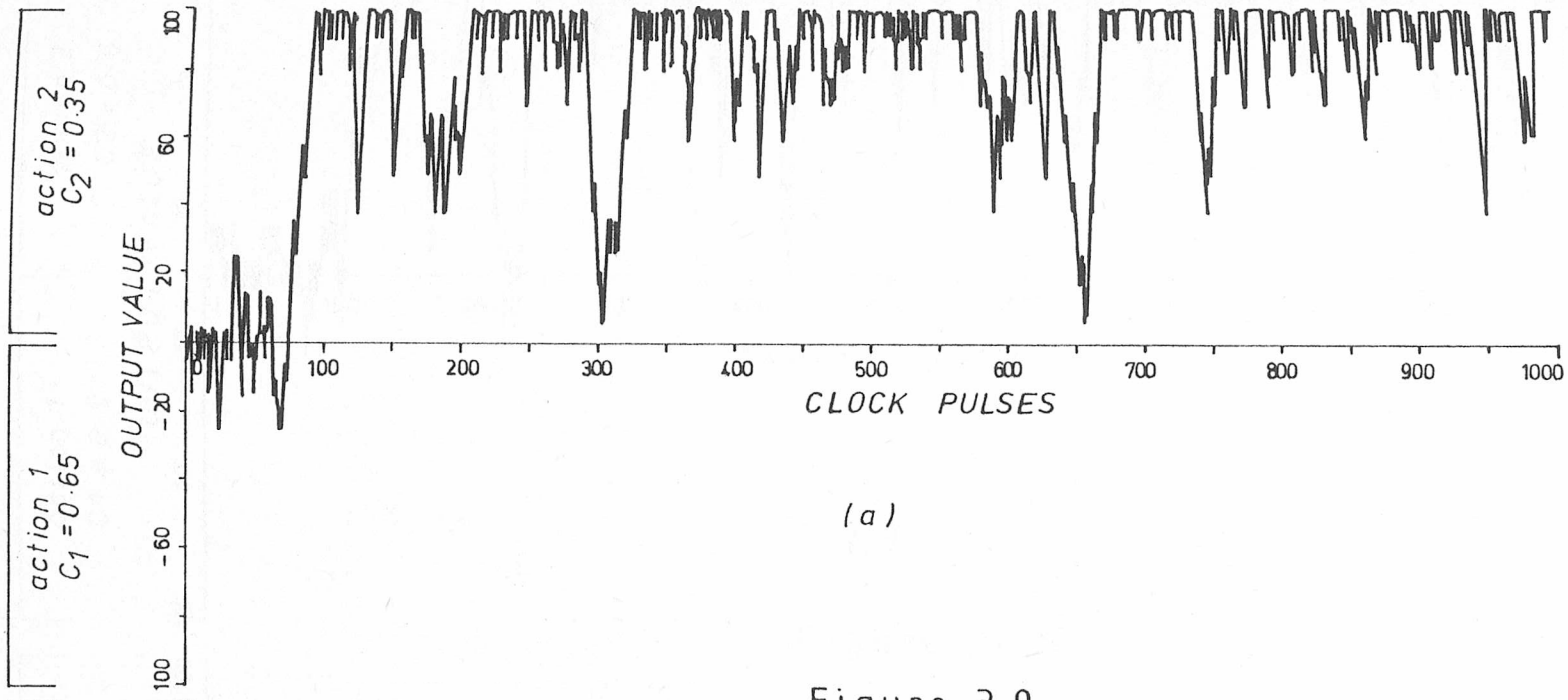


(a)

Figure 3.8 Simulation of type 1 modified Tsetlin automaton

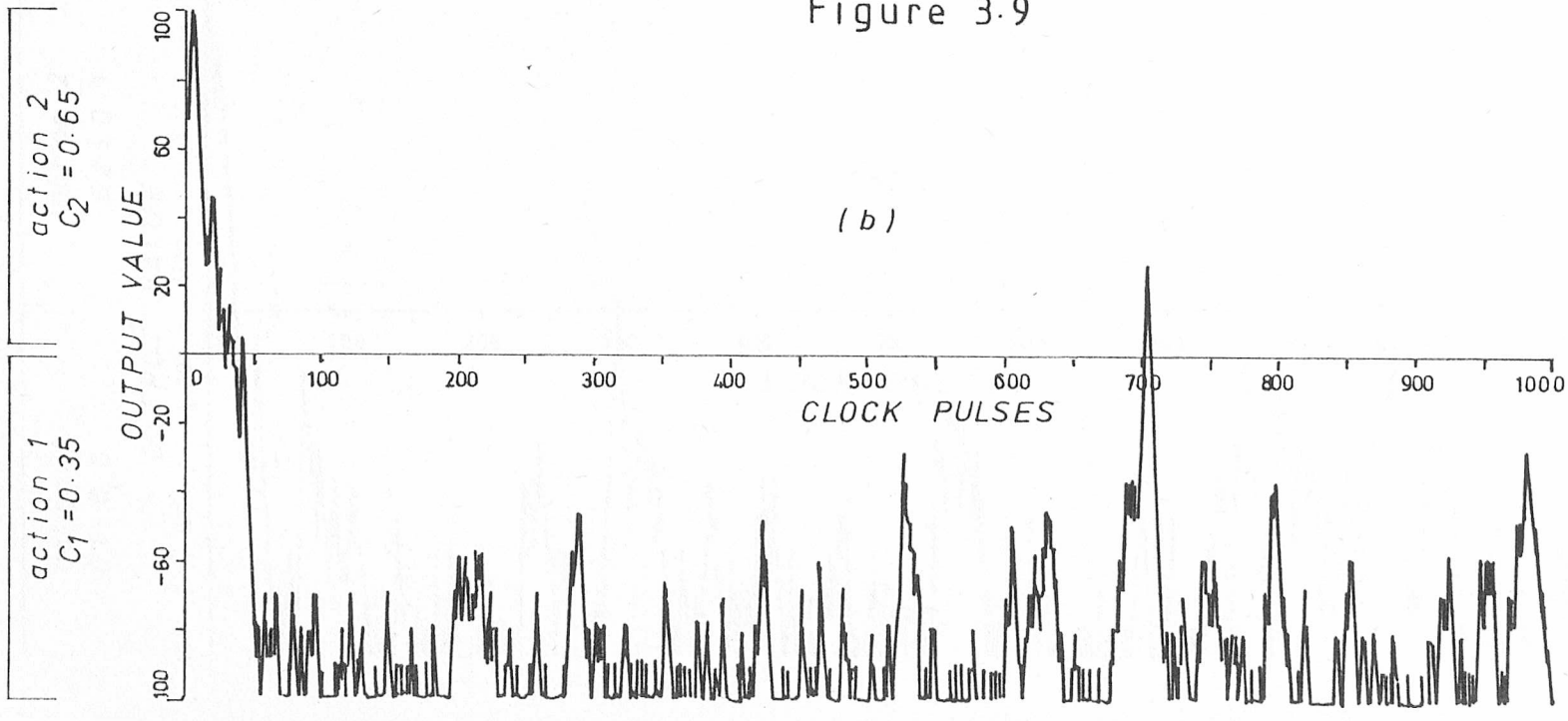


(b)

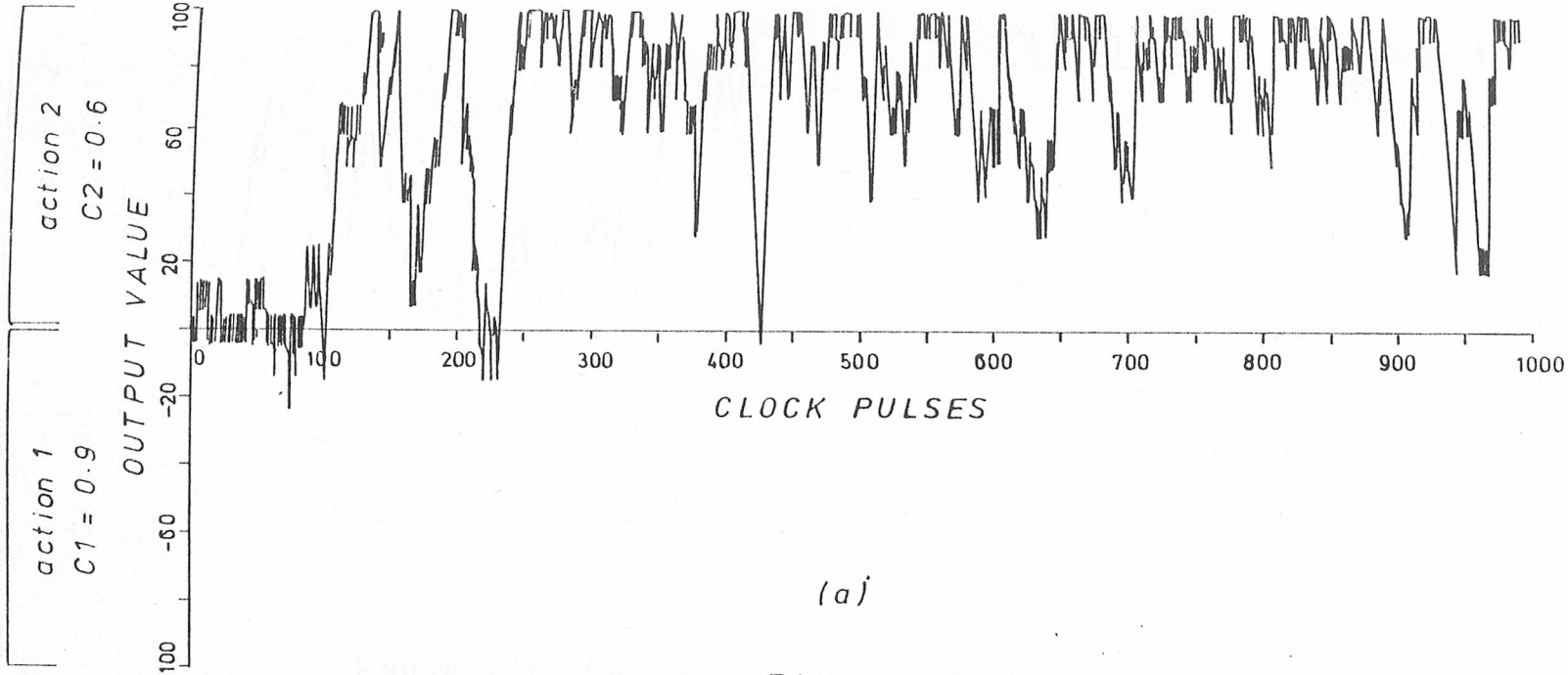

action 2
$C_2 = 0.35$
action 1
$C_1 = 0.65$
OUTPUT VALUE
100
60
20
-20
-60
-100
0
100
200
300
400
500
600
700
800
900
1000
CLOCK PULSES

(a)

Figure 3.9

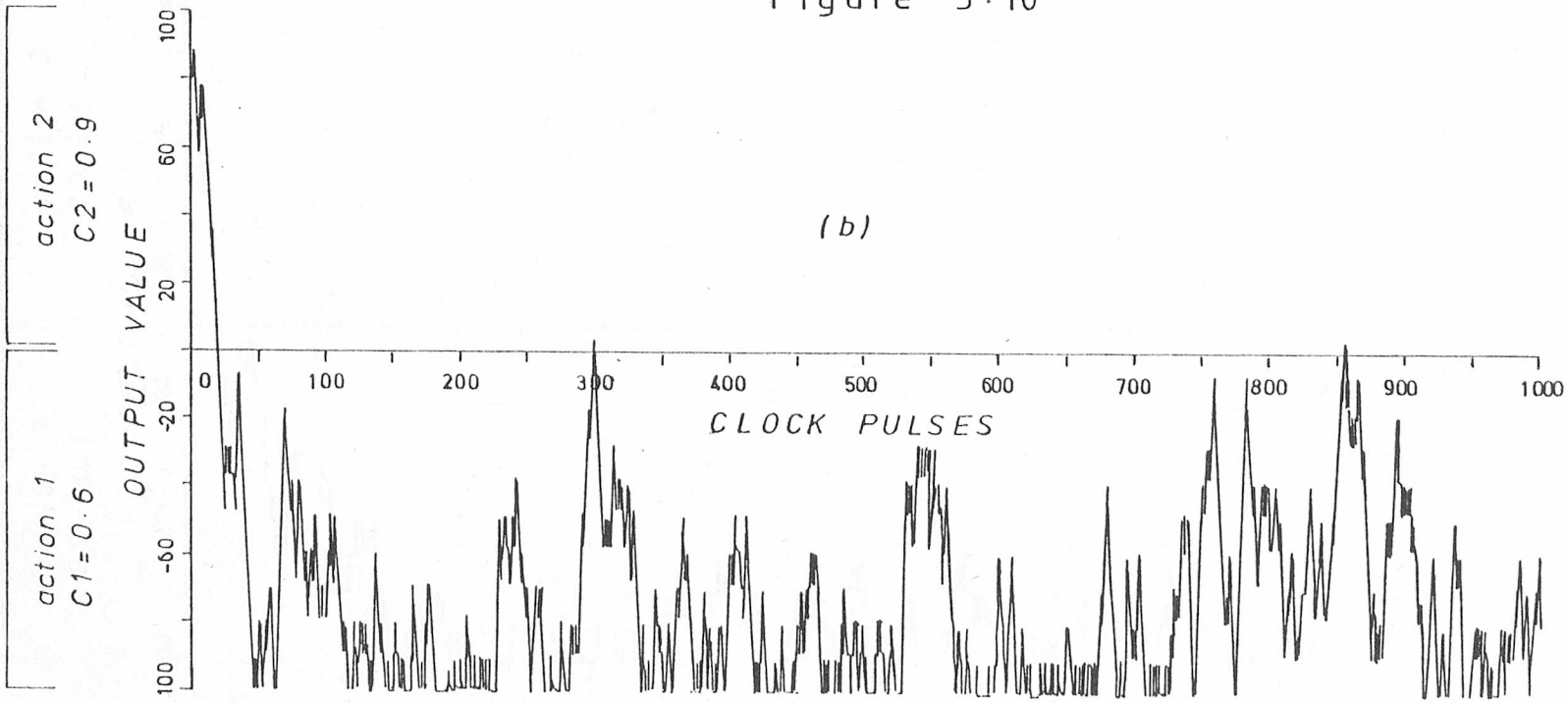

action 2
$C_2 = 0.65$
action 1
$C_1 = 0.35$
OUTPUT VALUE
100
60
20
-20
-60
-100
0
100
200
300
400
500
600
700
800
900
1000
CLOCK PULSES

(b)

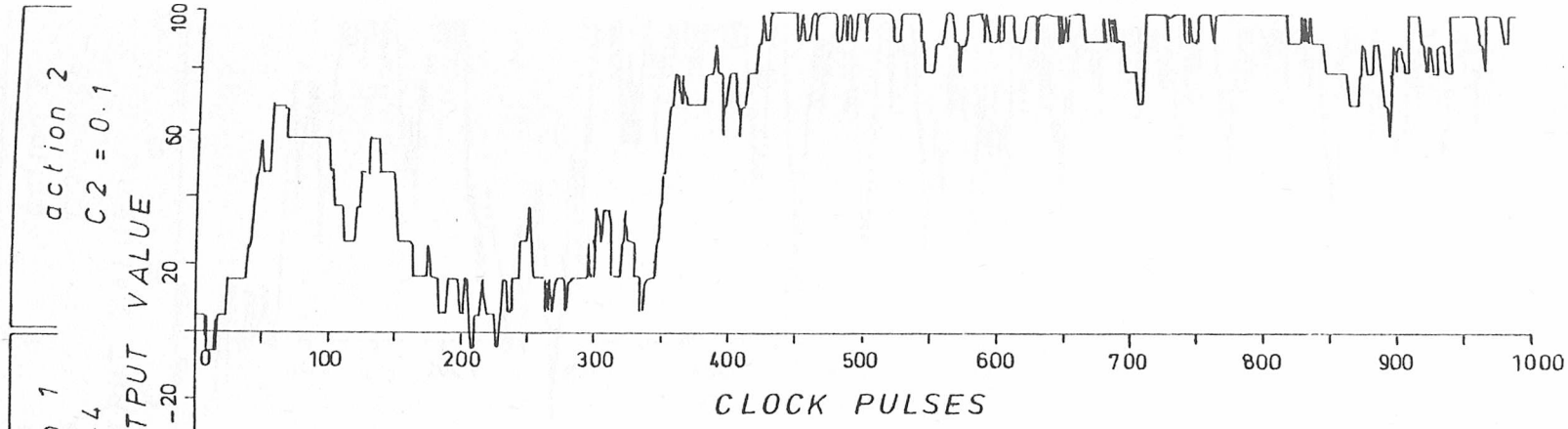


(a)

Figure 3·10

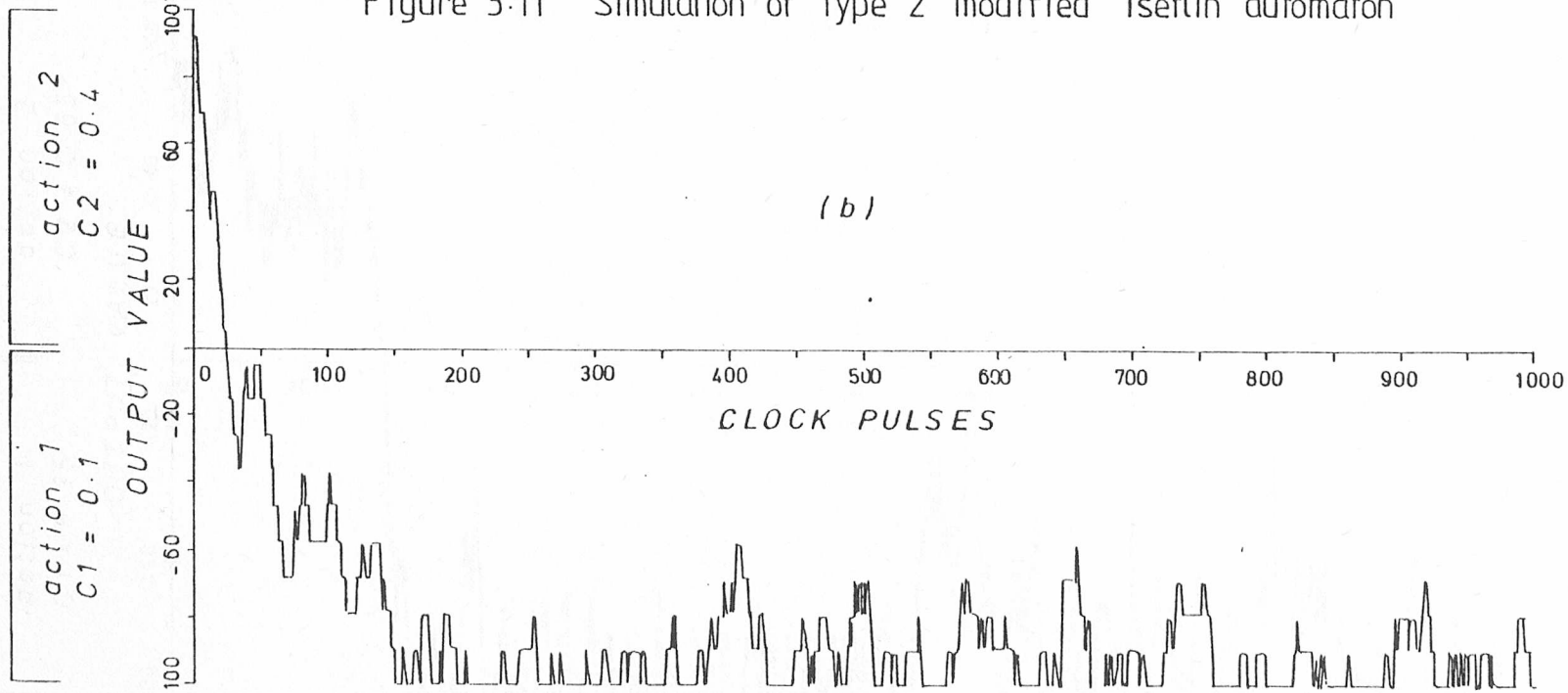


(b)

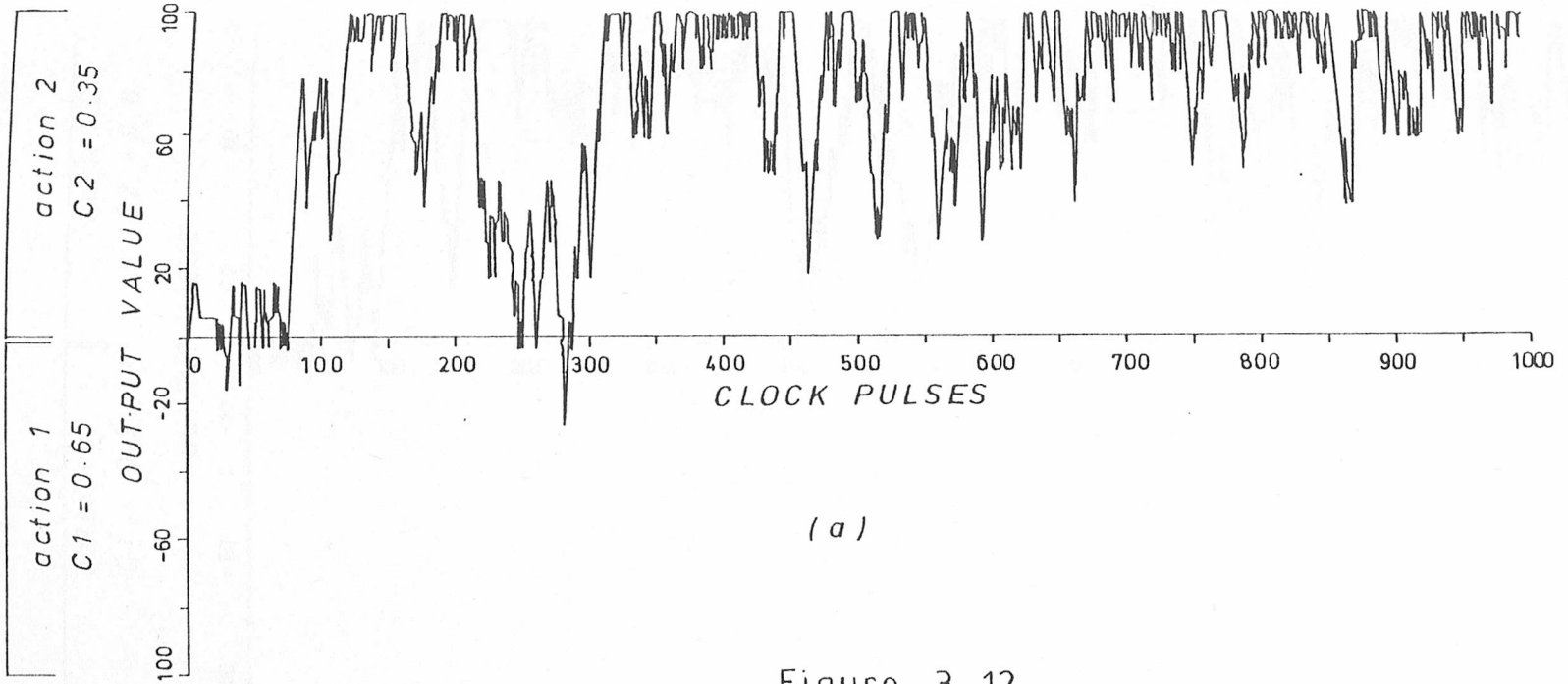


(a)

Figure 3.11 Simulation of type 2 modified Tsetlin automaton

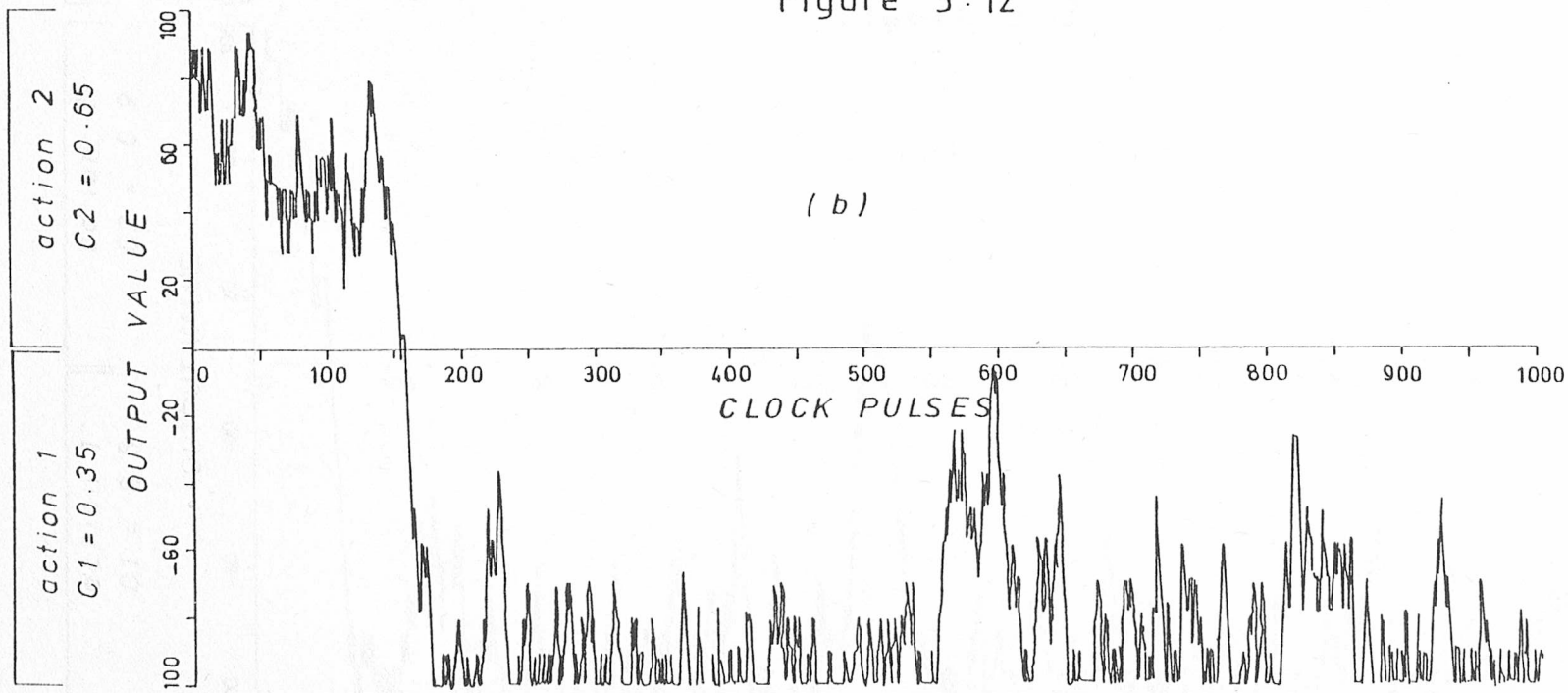


(b)

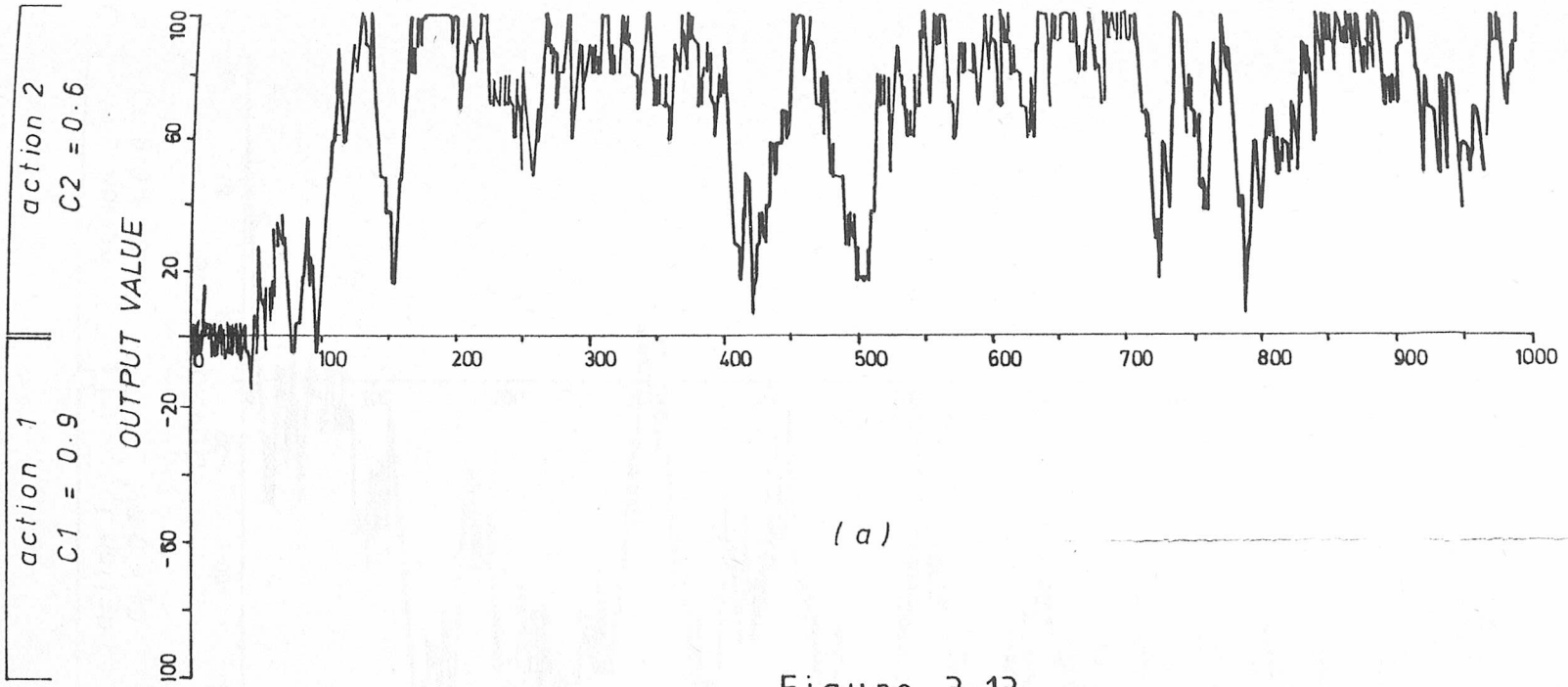


(a)

Figure 3.12

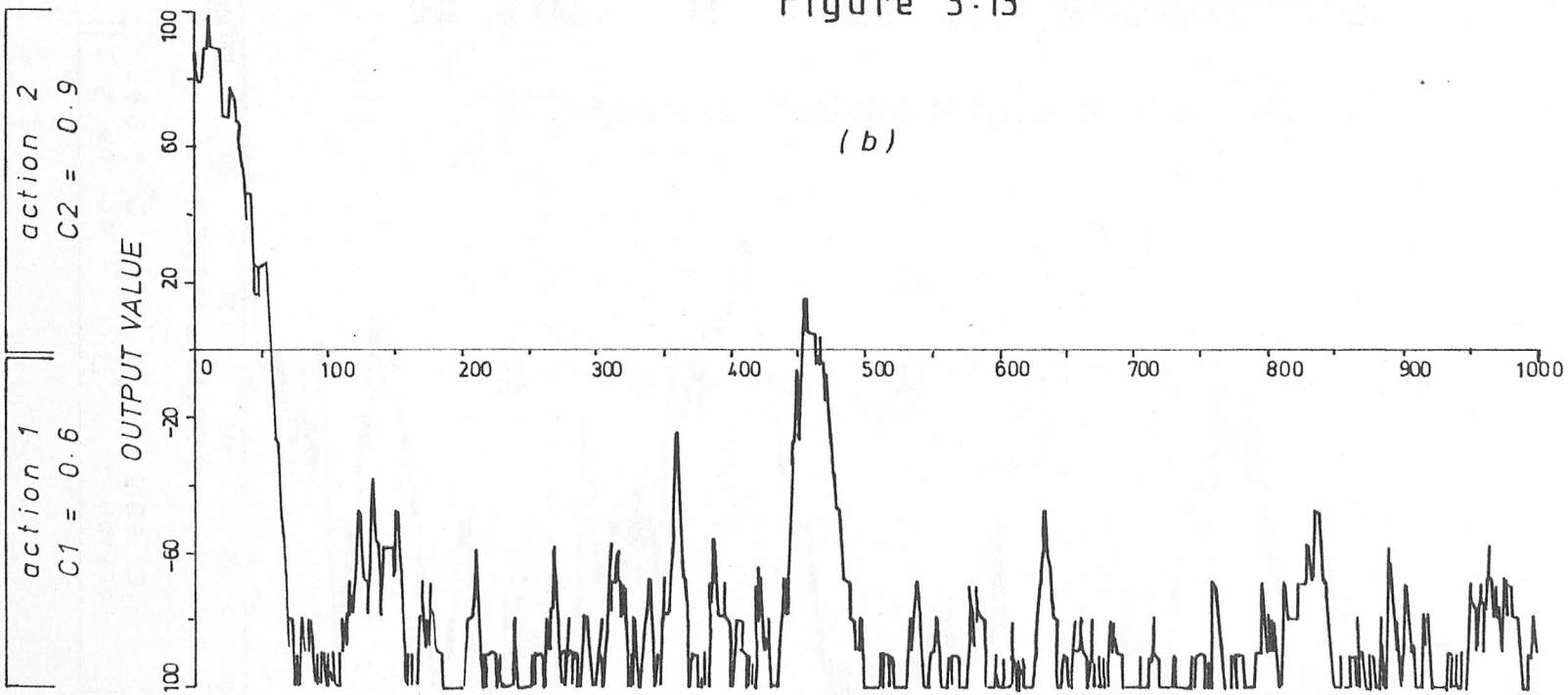


(b)



(a)

Figure 3·13



(b)

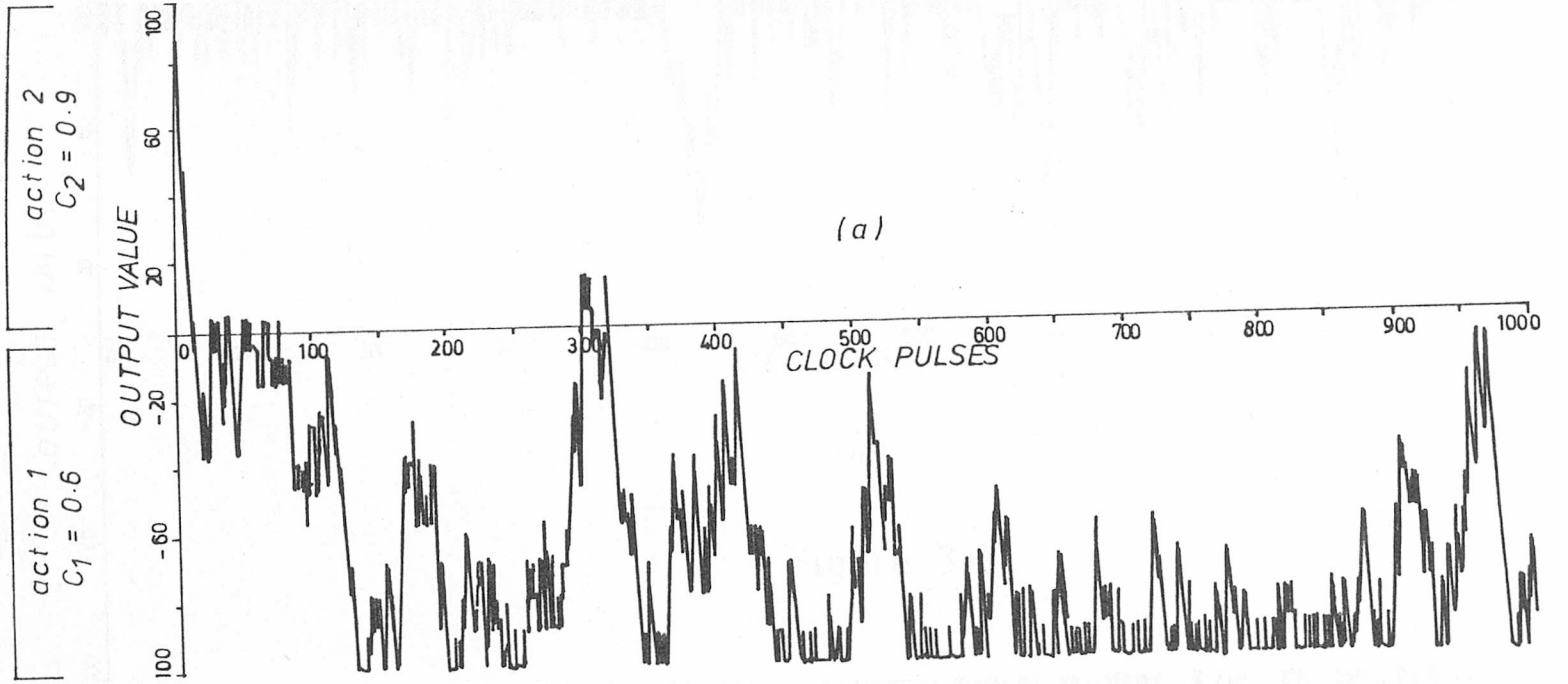
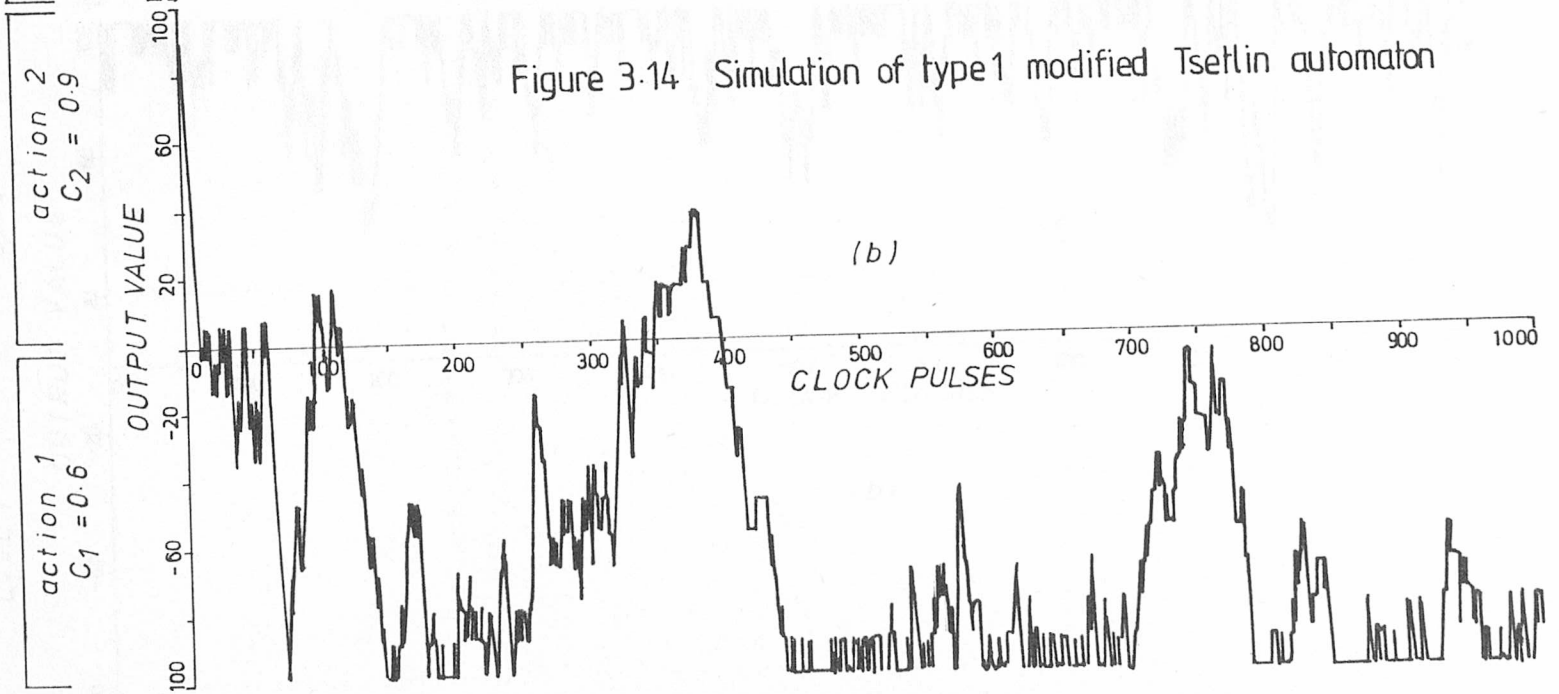
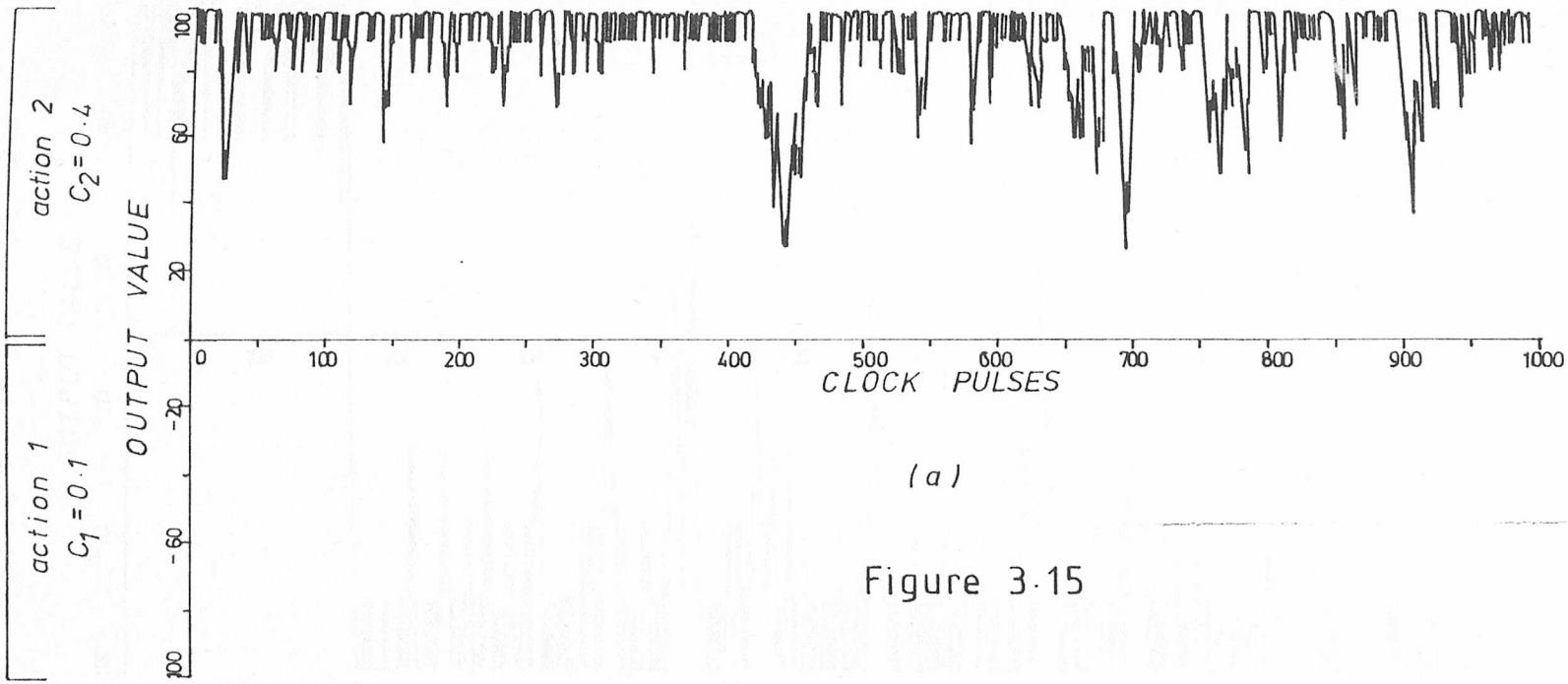


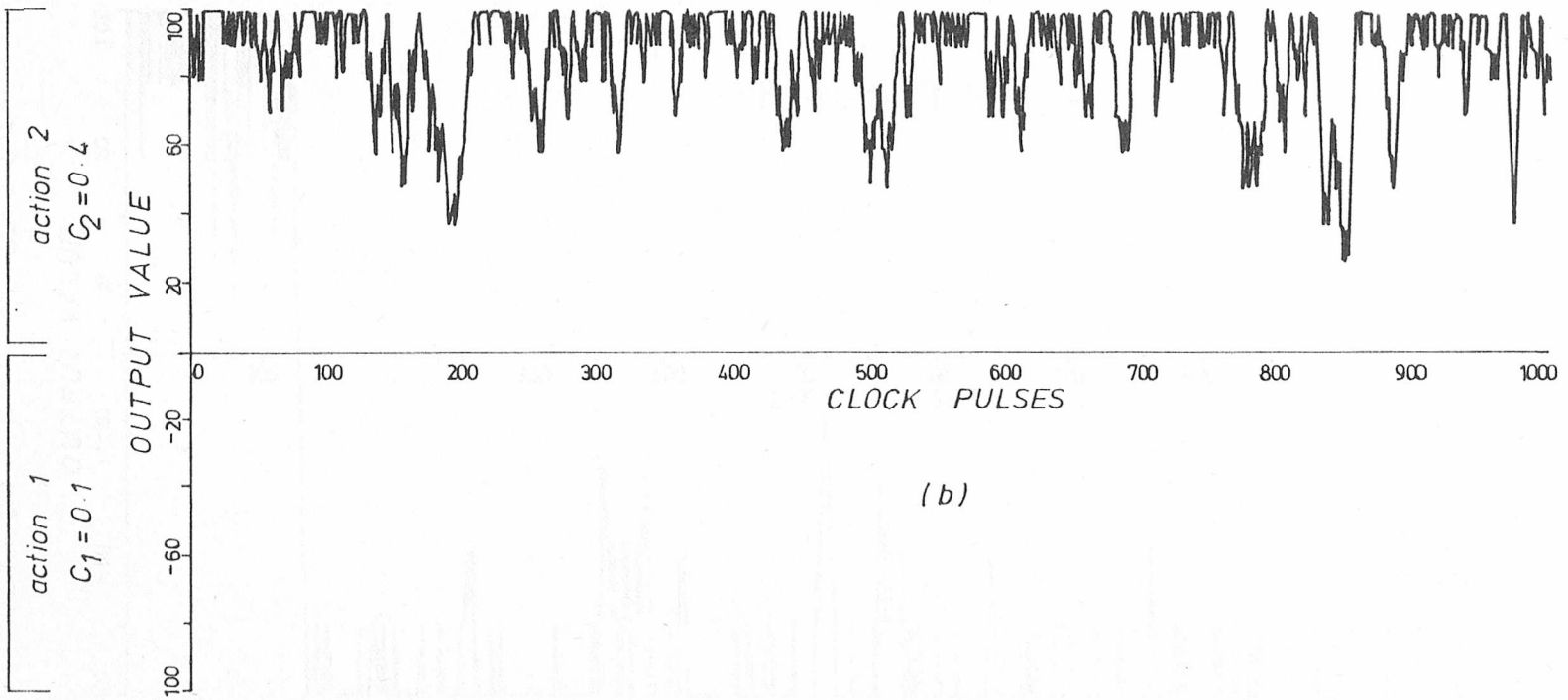
Figure 3.14 Simulation of type1 modified Tsetlin automaton





(a)

Figure 3.15



(b)

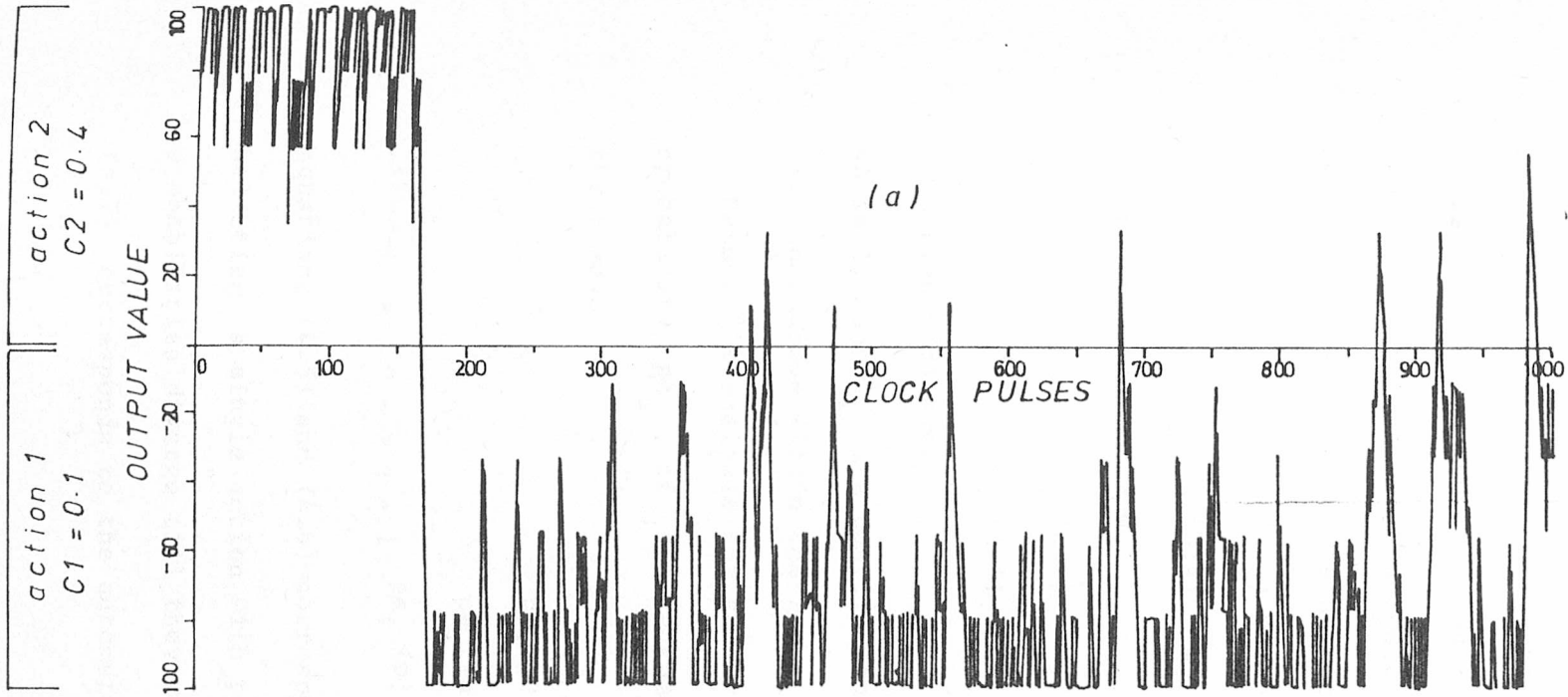
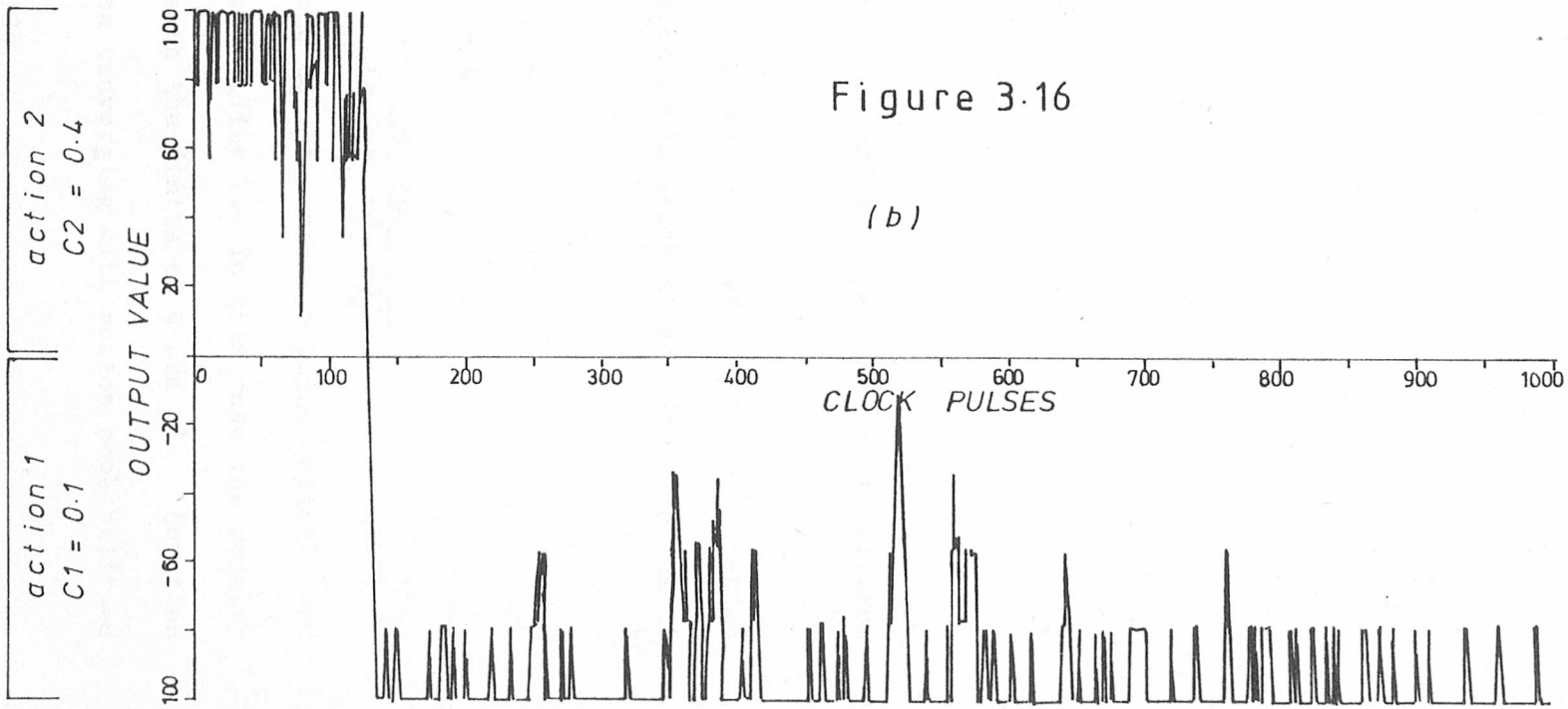

action 2
C2 = 0·4
action 1
C1 = 0·1
OUTPUT VALUE
100
60
20
-20
-60
-100
0
100
200
300
400
500
600
700
800
900
1000
CLOCK PULSES
(a)

Figure 3·16


action 2
C2 = 0·4
action 1
C1 = 0·1
OUTPUT VALUE
100
60
20
-20
-60
-100
0
100
200
300
400
500
600
700
800
900
1000
CLOCK PULSES
(b)

CHAPTER 4 NON-AUTONOMOUS ENVIRONMENTS

Linear Non-Autonomous Environments

For many of the applications of learning automata the environments are non-autonomous and have penalty probabilities which vary as the action probabilities. The first person to realise this and propose a model for non-autonomous environments was Narendra who analysed the operation of the Lri automaton in a two action non-autonomous environment [35] where the penalty probabilities were given by

$$c_1(n+1) = c_1(n) + \theta_1(n) \qquad (4.1)$$

$$c_2(n+1) = c_2(n) - \phi_2(n) \qquad (4.2)$$

if action $\alpha(n)=\alpha_1$ and

$$c_1(n+1) = c_1(n) - \phi_1(n) \qquad (4.3)$$

$$c_2(n+1) = c_2(n) + \theta_2(n) \qquad (4.4)$$

if action $\alpha(n) = \alpha_2$

where θ_i and ϕ_i are positive constants and the penalty probabilities are constrained within the range (0,1).

Narendra's analysis for the Lri automaton showed that the action probability pa_i of the automaton could reach steady state in one of three ways,

$$pa_i(n) = 0 \qquad (4.5)$$

$$pa_i(n) = 1 \qquad (4.6)$$

$$1/n \sum_{i=0 \longrightarrow n-1} pa_i(n) = \frac{(\theta_2 + \phi_1)}{(\theta_1 + \theta_2 + \phi_1 + \phi_2)} \qquad (4.7)$$

Equations (4.5) and (4.6) correspond to the automaton going optimal and selecting a single action with probability 1. In this case the penalty probabilities diverge till they reach the limits of 0 and 1. Equation (4.7) corresponds to the automaton converging with action probabilities

other than 0 and 1. The automaton selects both actions in a particular ratio. In this case the penalty probabilities converge to a value of 0 or 1 depending on whether $\theta_1 * \theta_2 - \phi_1 * \phi_2$ is negative or positive.

For any automaton operating in the linear non-autonomous environment, if the probability of action α_i is pa_i, the steady state value of the penalty probability c_1 will be zero if

$$(1-pa_i)\phi_i > pa_i\ \theta_i \tag{4.8}$$

and will be one if

$$pa_i\ \theta_i > (1-pa_i)\phi_i \tag{4.9}$$

thus c_i will converge to zero if

$$pa_i < \phi_i /(\theta_i + \phi_i) \tag{4.10}$$

and will converge to one if

$$pa_i > \phi_i /(\theta_i + \phi_i) \tag{4.11}$$

In general in a two action environment the penalty probabilities will not change from converging to zero to converging to one at the same probability. This is illustrated in Figure 4.1 which shows the steady state penalty probabilities c_1 and c_2 for a linear non-autonomous environment plotted against the probability of action 1. It can be seen that there is a band of action probabilities which produces penalty probabilities which are equal and so a band of action probabilities which will produce the minimum average penalty rather than a unique optimal action probability.

A different non-autonomous environment has been proposed by Kumar [36] where the penalty probability is a function of the action probabilities. Chrystall [37] has used this and defined the action probabilities more positively as

$$c_i(n) = k_i\ pa_i(n) \tag{4.12}$$

and has done simulations using this model. It is an improvement on

Narendra's model in that there is a unique action probability and the penalty probabilities have steady state values other than 0 or 1. However the model requires the use of the action probabilities of the automaton which in practice would not be available to the environment. This also limits the model to use with automata where the action probabilities are easily available. Automata with deterministic output functions do not have action probabilities as part of their operation and could not be used with this model. Also by being directly connected to the action probabilities the model does not have the same variance and time lags which were a realistic feature of Narendra's scheme.

Linear Non-Autonomous Environment-Simulation Results

Narendra's linear non-autonomous environment scheme was added to the stochastic simulation program described in Chapter 3 in order to investigate the operation of this environment, confirm the band structure and determine the operation of various automata in it.

Figure 4.2 shows a typical result showing a Lri automaton operating in an environment with $\theta_1 = 0.01$ $\phi_1 = 0.03$ $\theta_2 = 0.003$ and $\phi_2 = 0.009$. This produces an environment with a band structure in which the steady state penalty probabilities will change at action probabilities of 0.25 and 0.75 as indicated in Figure 4.2(a). Initially the penalty probabilities have a value of 0.5 and the action probabilities are outside the band which will produce converging c_i's. At first the penalty probabilities diverge but as the automaton responds and changes its action probability pa_1 to below the value of 0.75 both penalty probabilities converge to the value of 0. When the penalty probabilities have converged the automaton receives the same penalty probability whatever action it takes so its action probability becomes free to vary randomly. Only when this random wandering takes the action

probability above the value of 0.75 and the penalty probabilities start to diverge does the automaton receive feedback to keep the action probability in the range (0.25,0.75).

The simulation results confirmed the band theory for the linear non-autonomous environment and showed that the environment was not a very typical representation of a non-autonomous environment. Nor was it very useful for examining the operation of automata since it has bands where the automaton receives no useful feedback.

The Non-Linear Non-Autonomous Environment

In order to produce a non-autonomous environment in which the penalty probabilities converge to a value other than 0 or 1 a non-autonomous environment was proposed where the penalty probabilities were given by

$$c_i(n+1) = c_i(n) + \theta_i (1-c_i(n)) \qquad (4.13)$$

if action $\alpha(n) = \alpha_i$ and

$$c_i(n+1) = c_i(n) - \phi_i c_i(n) \qquad (4.14)$$

otherwise

where θ_i and ϕ_i are positive constants. In a physical sense the factor $\theta_i (1-c_i(n))$ can be related to the decreased availability as its use increases while $\phi_i c_i(n)$ corresponds to the increasing availability of a resource as its use decreases. When the penalty probabilities have reached their steady state values

$$E(\text{amount of increase in } c_i) = E(\text{amount of decrease in } c_i) \qquad (4.15)$$

With the environment as defined by equations (4.13) and (4.14) equation (4.15) can be expressed as

$$pa_i \theta_i (1-c_i) = (1-pa_i) \phi_i c_i \qquad (4.16)$$

so

$$pa_i = (\phi_i c_i)/(\theta_i - \theta_i c_i + \phi_i c_i) \qquad (4.17)$$

or

$$c_i = (\theta_i \; pa_i)/(\phi_i - \phi_i \; pa_i + \theta_i \; pa_i) \qquad (4.18)$$

Figure 4.3 shows how the penalty probabilities vary with action probability in this non-linear non-autonomous environment.

The average penalty received by an automaton is

$$M = pa_1 \; c_1 + pa_2 \; c_2 \qquad (4.19)$$

Using $pa_2 = 1-pa_1$

and substituting equation (4.18) into equation (4.19) the average penalty in a non-linear non-autonomous environment is

$$M = \theta_1 \; pa_1^2 \; /(\phi_1 - \phi_1 \; pa_1 + \theta_1 \; pa_1) + \left(\frac{\theta_2 \; -2\theta_2 \; pa_1 + \theta_2 \; pa_1^2}{\phi_2 \; pa_1 + \theta_2 \; -\theta_2 \; pa_1}\right) \qquad (4.20)$$

This expression for the average penalty can be differentiated with respect to pa_1 to produce a quartic equation. When this is equated to zero and the roots found, the real result in the interval (0,1) gives the action probability pa_1 which corresponds to minimum average penalty.

Steady State Conditions of the Lrp and Lri Automata

To appreciate the operation of a Lri automaton in a non-linear non-autonomous environment consider Figure 4.3 with the action probability set initially to 0.5. Will the action probability tend to increase or decrease? The probabilities of selecting either action are equal as is the change in action probabilities due to the reinforcement algorithm. Thus the only difference between the actions is their penalty probability and the action probability will change to favour the action corresponding to the most rewards, in this case action α_1. The action probability of α_1 will tend to increase till the automaton reaches steady state.

This occurs when

$$pa_1 (1-c_1)(pa_2 - \alpha pa_2) = pa_2 (1-c_2)(pa_1 - \alpha pa_1) \qquad (4.21)$$

which reduces to

$$c_1 = c_2 \qquad (4.22)$$

Thus the Lri automaton moves to make the penalty probabilities from both actions equal.

Substituting equation (4.18) into equation (4.22)

$$\phi_1 \; pa_1 / (\phi_1 - \phi_1 \; pa_1 + \theta_1 \; pa_1) =$$
$$\phi_2 \; pa_2 / (\phi_2 - \phi_2 \; pa_2 + \theta_2 \; pa_2) \qquad (4.23)$$

Using $pa_2 = 1-pa_1$ gives the steady state action probability of the Lri automaton in the non-linear non-autonomous environment as

$$pa_1 = \frac{(-\phi_1 \theta_2 \; +/- \; \theta_1 \theta_2 \phi_1 \phi_2)}{(\theta_1 \phi_2 - \phi_1 \theta_2)} \qquad (4.24)$$

Only one of the solutions for pa_1 is in the range (0,1).

To find the steady state value of c_i use

$$pa_1 = 1-pa_2 \qquad (4.25)$$

Substituting equation (4.17) gives

$$\phi_1 \; c_1 / (\theta_1 - \theta_1 \; c_1 + \phi_1 \; c_1)$$
$$= 1-(\phi_2 \; c_2 (\theta_2 - \theta_2 \; c_2 + \phi_2 \; c_2))$$

Substituting $c_1 = c_2$ gives the steady state penalty probabilities of the Lri automaton in a non-linear non-autonomous environment as

$$c_1 = c_2 = \frac{(\theta_1 \theta_2 \; +/- \; \theta_1 \theta_2 \phi_1 \phi_2)}{(\theta_1 \theta_2 - \phi_1 \phi_2)} \qquad (4.26)$$

which is also the steady state average penalty received by the Lri automaton.

For the Lrp automaton the calculations are more complex. Steady state occurs when

increase in pa_1 due to action 1 being rewarded

+ increase in pa_1 due to action 2 being penalised

=decrease in pa 1 due to action 2 being rewarded

+ decrease in pa_1 due to action 1 being penalised

$$\Rightarrow pa_1 (1-c_1)(pa\ 2-\alpha pa_2)+pa_2 (c_2)(pa_2 - \beta pa_2)$$

$$= pa_2 (1-c_2)(pa_1 - \alpha pa_1)+pa_1 (c_1)(pa_1 - \beta pa_1) \qquad (4.27)$$

if α is set equal to β equation (4.27) reduces to

$$pa_1 = c_2 /(c_1 + c_2) \qquad (4.28)$$

since $pa_1 = 1-pa_2$ equation (4.28) can be expressed as

$$pa_1\ c_1 = pa_2\ c_2 \qquad (4.29)$$

That is the Lrp automaton with $\alpha = \beta$ moves so that it receives the same penalty rate from each action.

Using the equations (4.22) and (4.29) the steady state conditions of the Lrp automaton with $\alpha = \beta$ and the Lri automaton in the non-linear non-autonomous environment can be calculated and as an example this has been done for the environment shown in Figure 4.3. The steady state conditions of the automata are a long way from the optimum action probability showing that the automata in satisfying their own steady state conditions do not converge to the optimum action probability.

Non-Linear Non-Autonomous Environment-Simulation Results

Like the linear non-autonomous environment the non-linear non-autonomous environment was added to a learning automaton simulation program in order to investigate the operation of the environment and confirm the equations derived above.

Figures 4.4, 4.5 and 4.6 show typical results of a Lri automaton operating in various non-linear non-autonomous environments. In all three examples it can be seen that the steady state values of action and penalty probabilities are near the values given by equations (4.24) and (4.26). The action and penalty probabilities do not converge to the values given by equations (4.24) and (4.26) since as the penalty probabilities converge to become equal the automaton receives a similar

response from the environment whichever action it choses and so its action probability is free to vary randomly. This in turn causes the penalty probabilities to diverge a little before the automaton detects this and causes the penalty probabilities to converge again.

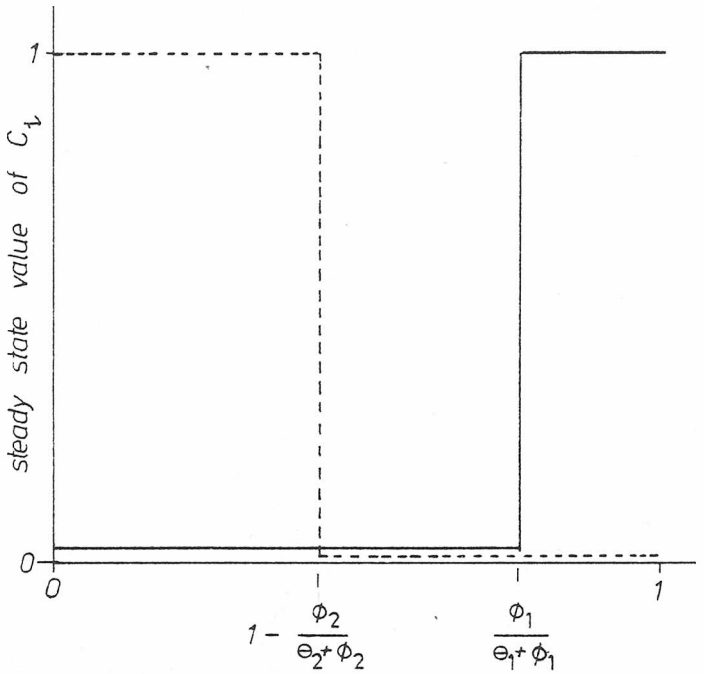
Simulations were done using the Tsetlin, Krylov, modified Tsetlin and modified estimating automata in the non-linear non-autonomous environment but the performance of these automata was poor. All these automata have deterministic output functions and so when they are in steady state a single action is output. In a non-autonomous environment the best performance is gained by selecting all actions with a particular ratio. Automata with deterministic output functions can never achieve this and reach a steady state condition. The automata tested attempted to switch between actions but caused the penalty probabilities to oscillate and never achieved the smooth performance achieved by the Lri and Lrp automata with stochastic output functions.

Another factor which was found to be important from the results of simulations was the convergence rate of the automaton relative to the rate of change of the penalty probabilities. If the convergence rate of the automaton is fast compared to the rate of change of the penalty probabilities an oscillation can occur. This is illustrated in Figures 4.7 and 4.8 with the action and penalty probabilities initially set to simulate a disturbance from the steady state. If the automaton reaches the steady state action probability before the penalty probabilities reach their steady state there will be a difference between the penalty probabilities which will cause the automaton to overshoot the steady state. This will in turn cause the penalty probabilities to overshoot. In most cases the resulting oscillation dies out but in some cases as in Figure 4.7, where the action of the automaton has gone optimal, the

oscillations can grow. Such oscillations can be prevented by avoiding the use of an automaton which has a convergence rate which is fast relative to the rate of change of the penalty probabilities while the output of one action continuously can be prevented by the use of an automaton which is merely expedient rather than optimal.

Conclusions

Theoretical consideration of the Lrp and Lri automata has resulted in simple formula describing the steady state behaviour of these automata. Theoretical investigations into various non-autonomous environments have resulted in greater understanding of their operation and formula for the optimal action probabilities. It has also been shown that the Lri and Lrp do not converge to these optimal action probabilities. Results from simulation has shown that only automata with stochastic output functions are suitable for use in non-autonomous environments and a number of other practical considerations have been highlighted.



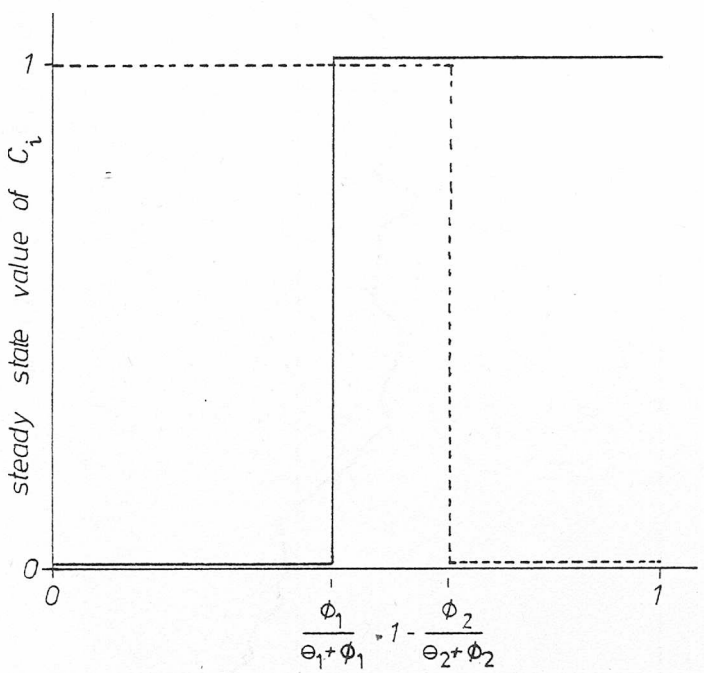


Figure 4.1 Band structure of Narendras non-autonomous environment

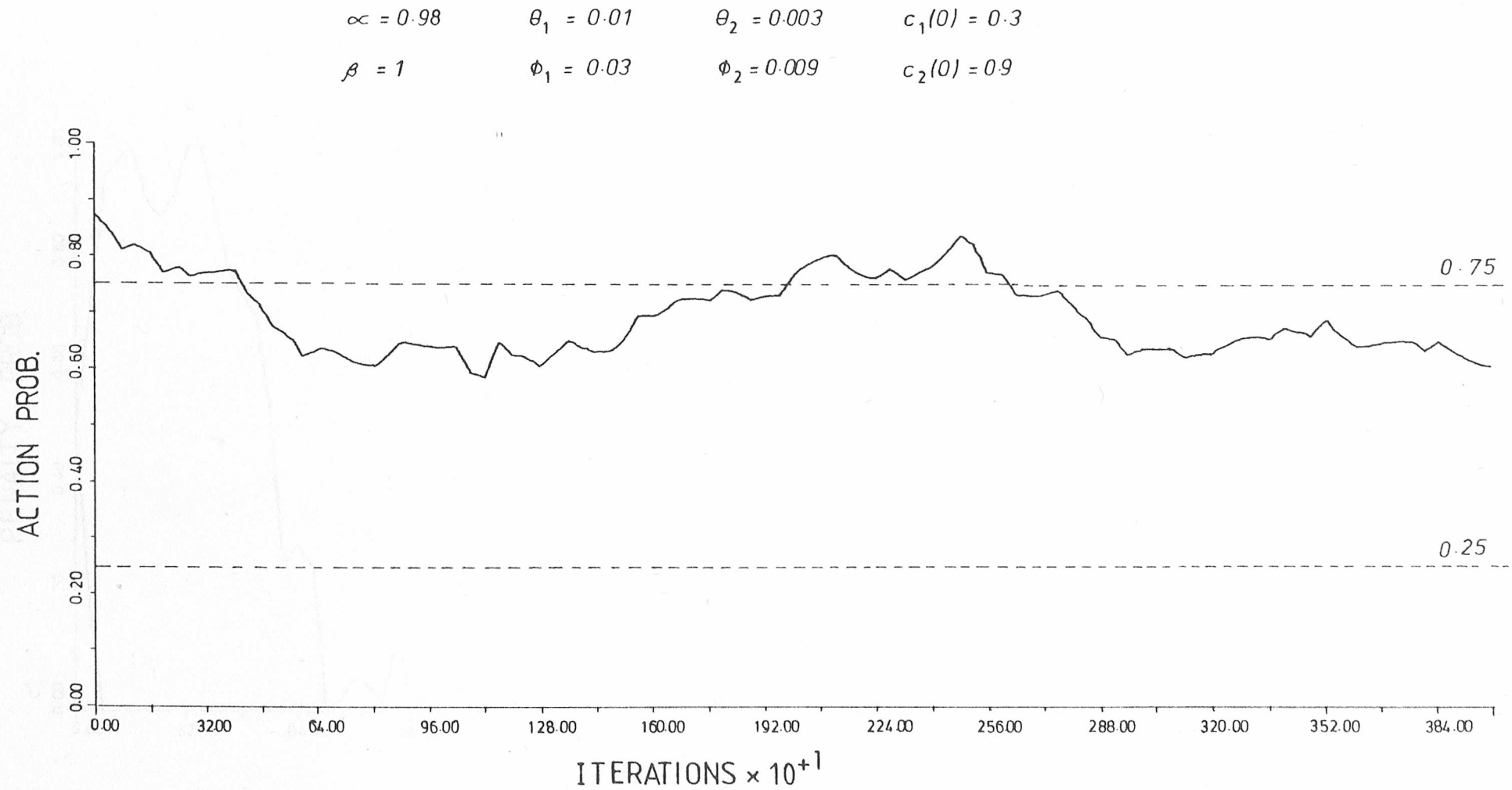


Figure 4.2(a) Simulation of L_{RI} automaton in Narendra's non-autonomous environment

$\alpha = 0.98$ $\theta_1 = 0.01$ $\theta_2 = 0.003$ $c_1(0) = 0.3$

$\beta = 1$ $\phi_1 = 0.03$ $\phi_2 = 0.009$ $c_2(0) = 0.9$

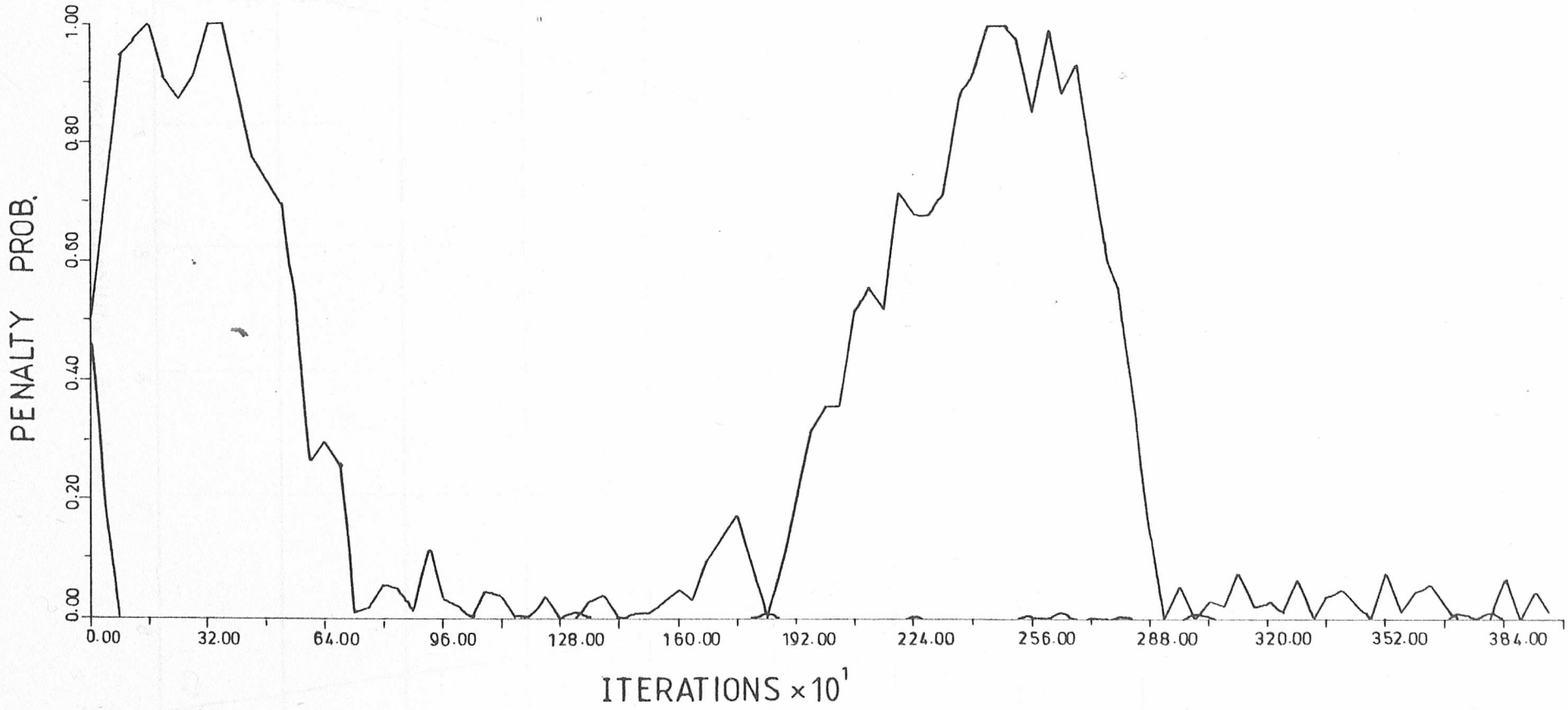


Figure 4.2 (b)

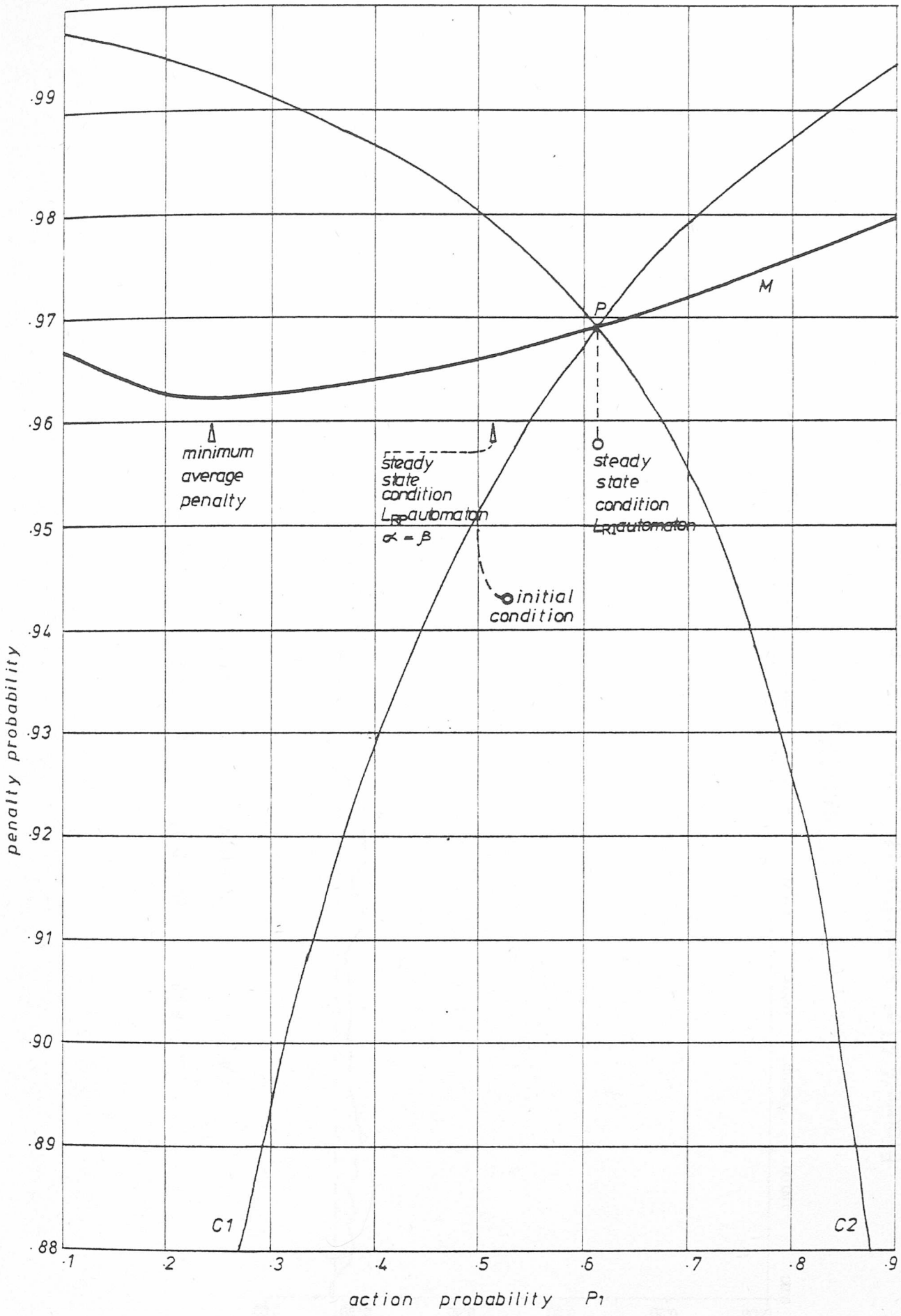


Figure 4.3 Example of automata in a non-autonomous environment

$\theta_1 = 0.5 \quad \phi_1 = 0.025 \quad \theta_2 = 0.1 \quad \phi_2 = 0.002$

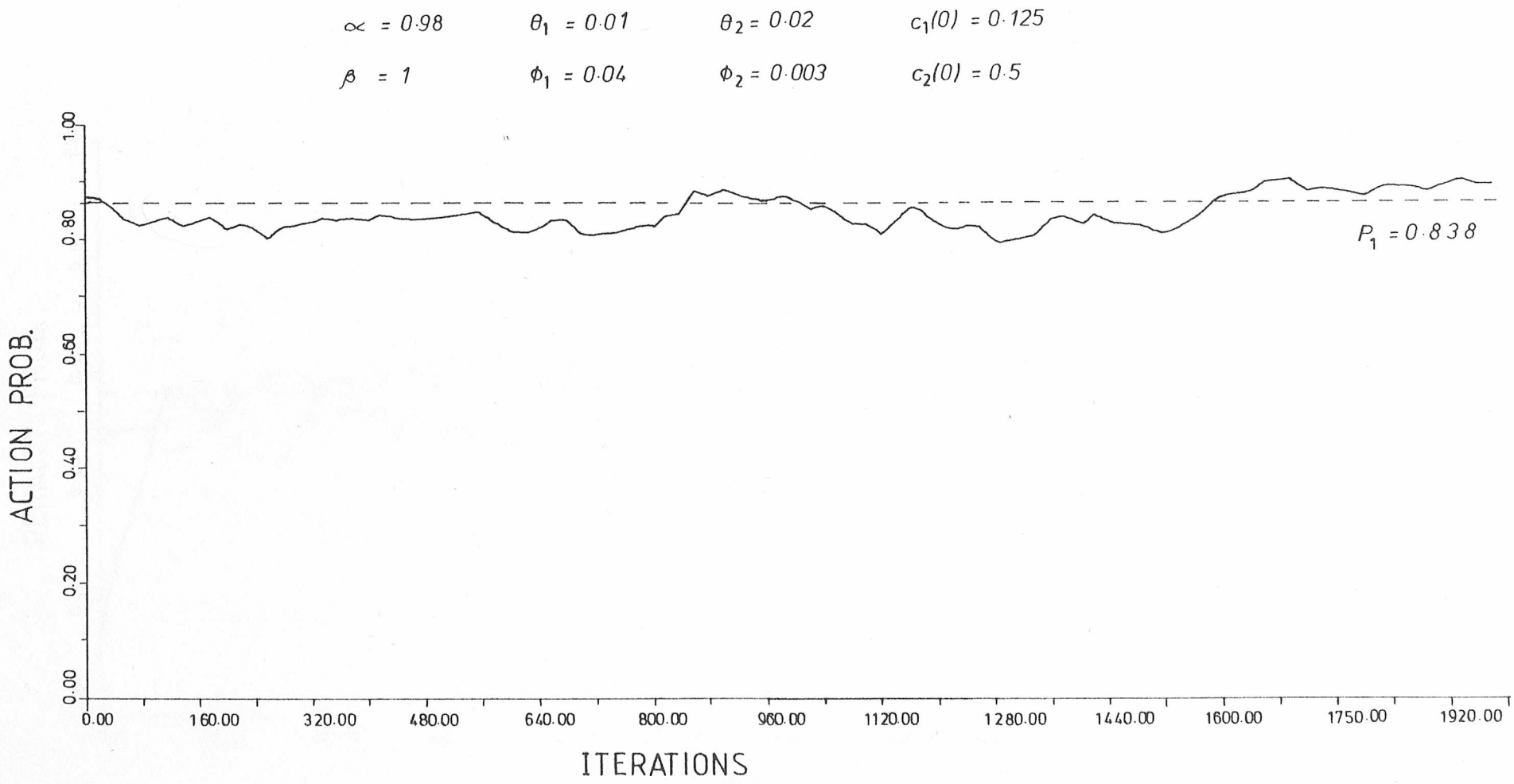


Figure 4·4 (a) Simulation of L_{RI} automaton in non-linear non-autonomous environment

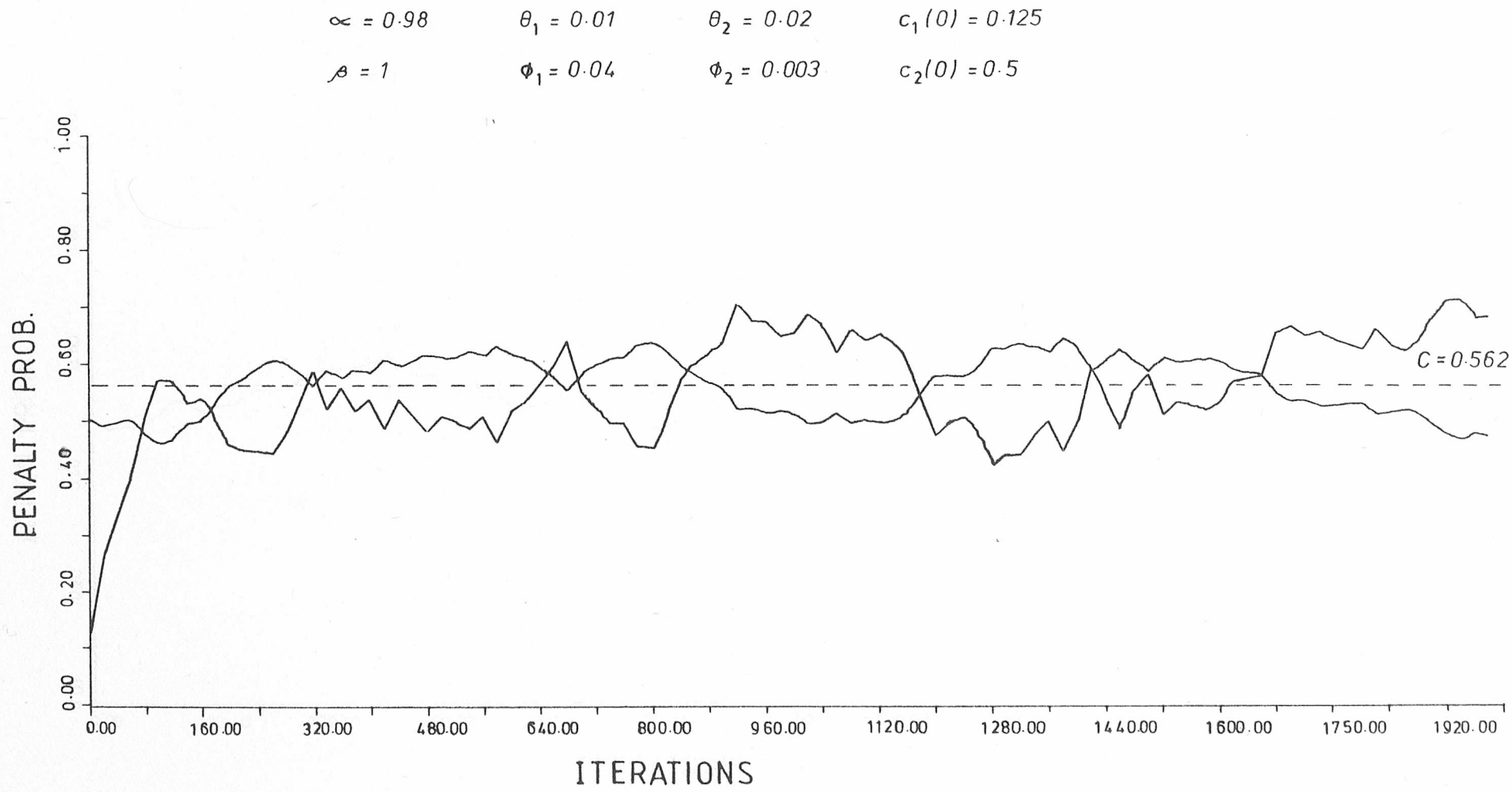

$\alpha = 0.98$
$\theta_1 = 0.01$
$\theta_2 = 0.02$
$c_1(0) = 0.125$
$\beta = 1$
$\phi_1 = 0.04$
$\phi_2 = 0.003$
$c_2(0) = 0.5$
PENALTY PROB.
1.00
0.80
0.60
0.40
0.20
0.00
C = 0.562
0.00
160.00
320.00
480.00
640.00
800.00
960.00
1120.00
1280.00
1440.00
1600.00
1750.00
1920.00
ITERATIONS

Figure 4.4(b)

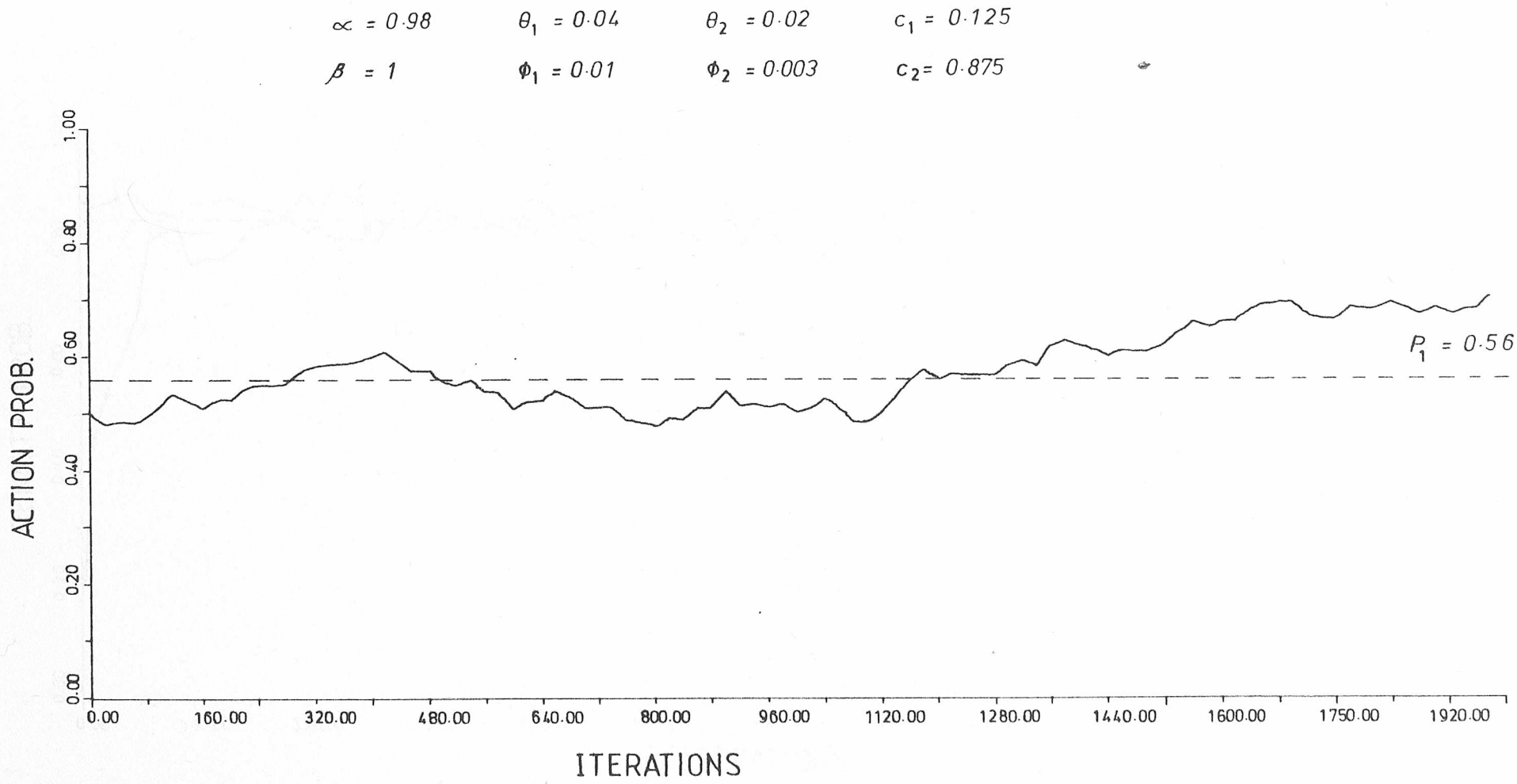

$\alpha = 0.98$
$\theta_1 = 0.04$
$\theta_2 = 0.02$
$c_1 = 0.125$
$\beta = 1$
$\phi_1 = 0.01$
$\phi_2 = 0.003$
$c_2 = 0.875$
ACTION PROB.
1.00
0.80
0.60
0.40
0.20
0.00
$P_1 = 0.56$
0.00
160.00
320.00
480.00
640.00
800.00
960.00
1120.00
1280.00
1440.00
1600.00
1750.00
1920.00
ITERATIONS

Figure 4.5(a)

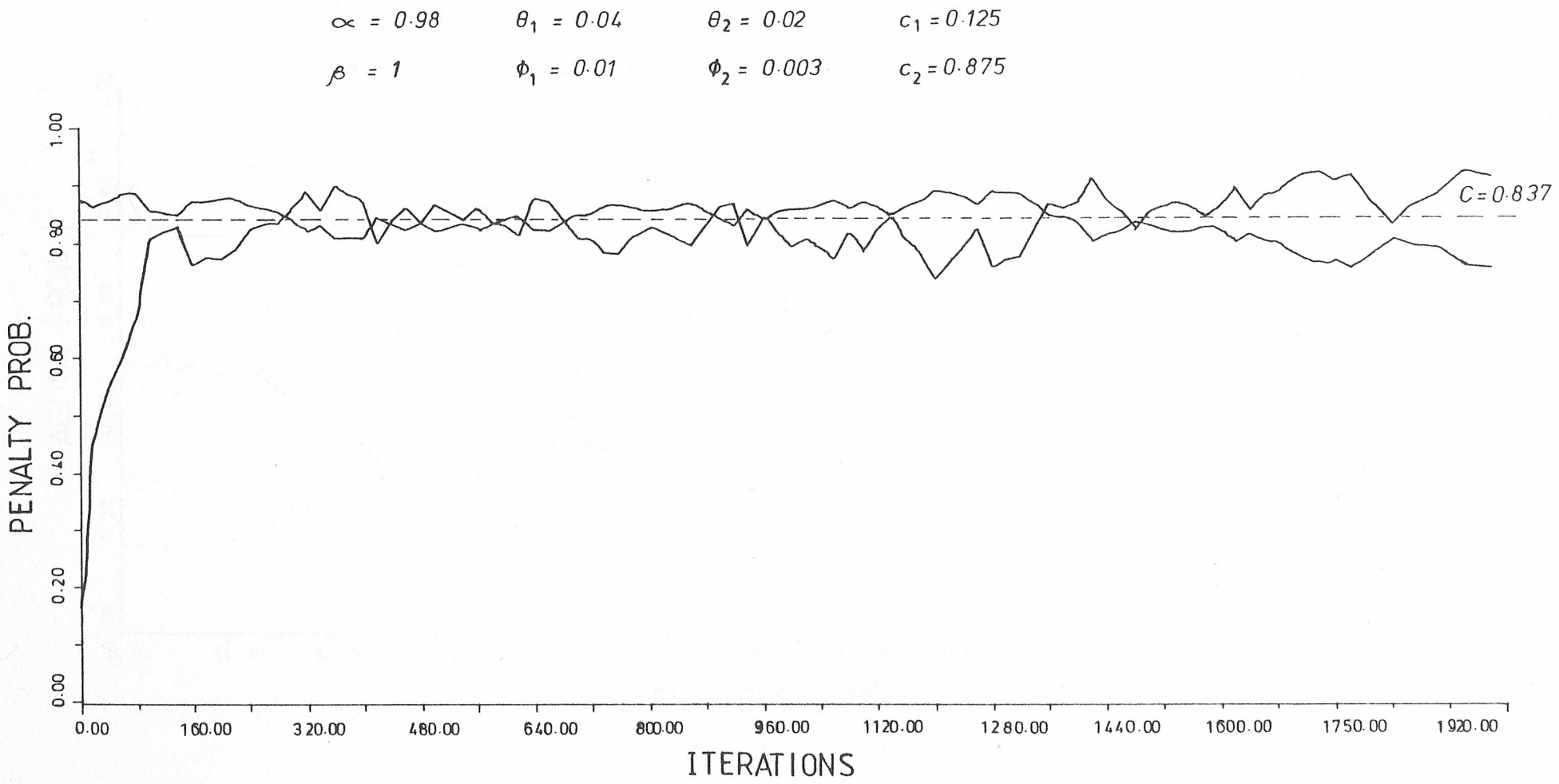

$\alpha = 0.98$
$\theta_1 = 0.04$
$\theta_2 = 0.02$
$c_1 = 0.125$
$\beta = 1$
$\phi_1 = 0.01$
$\phi_2 = 0.003$
$c_2 = 0.875$
C = 0.837
PENALTY PROB.
0.00
0.20
0.40
0.60
0.80
1.00
0.00
160.00
320.00
480.00
640.00
800.00
960.00
1120.00
1280.00
1440.00
1600.00
1750.00
1920.00
ITERATIONS

Figure 4.5 (b)

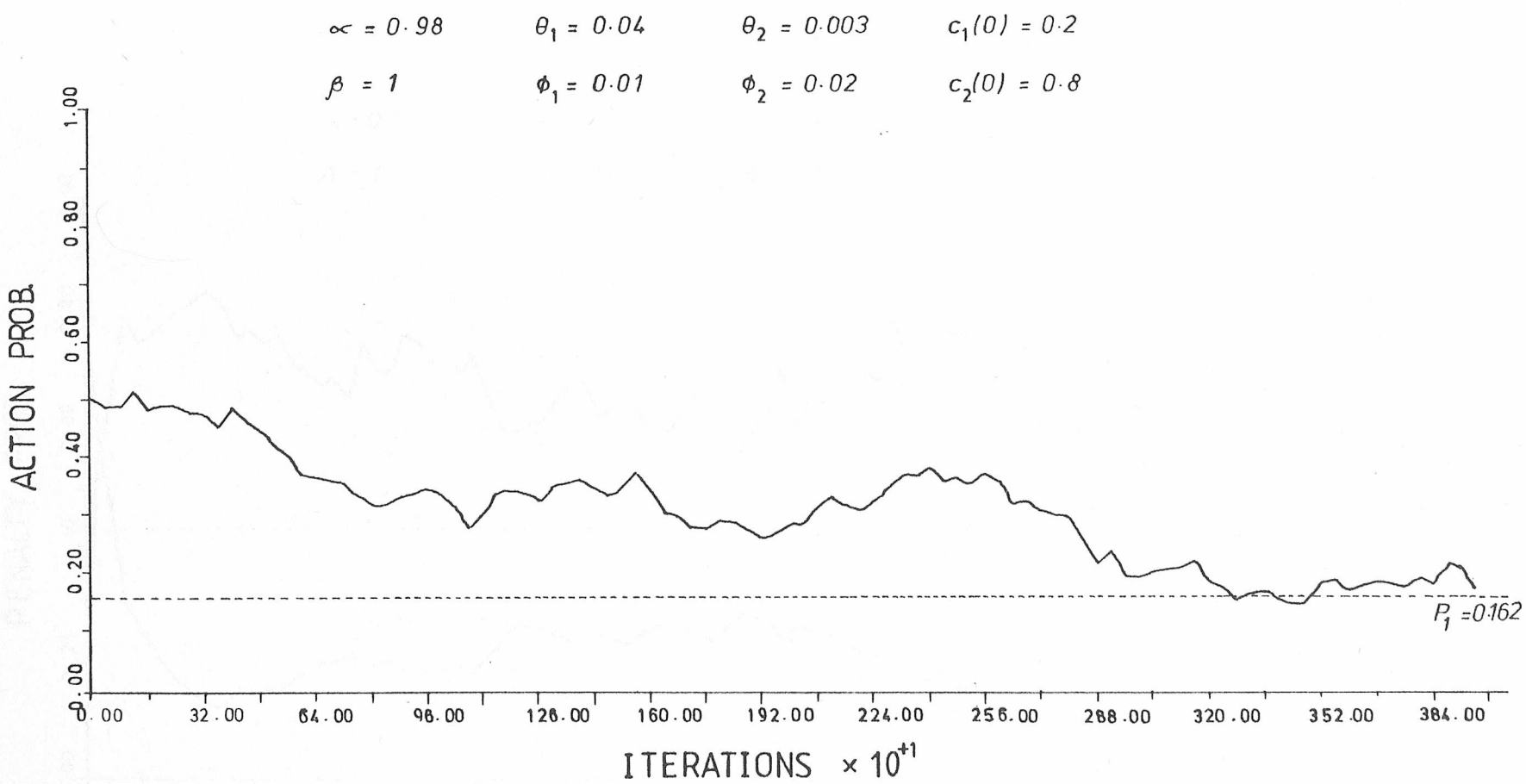

$\alpha = 0.98$
$\theta_1 = 0.04$
$\theta_2 = 0.003$
$c_1(0) = 0.2$
$\beta = 1$
$\phi_1 = 0.01$
$\phi_2 = 0.02$
$c_2(0) = 0.8$
ACTION PROB.
1.00
0.80
0.60
0.40
0.20
0.00
$P_1 = 0.162$
0.00
32.00
64.00
96.00
128.00
160.00
192.00
224.00
256.00
288.00
320.00
352.00
384.00
ITERATIONS $\times 10^{+1}$

Figure 4.6(a)

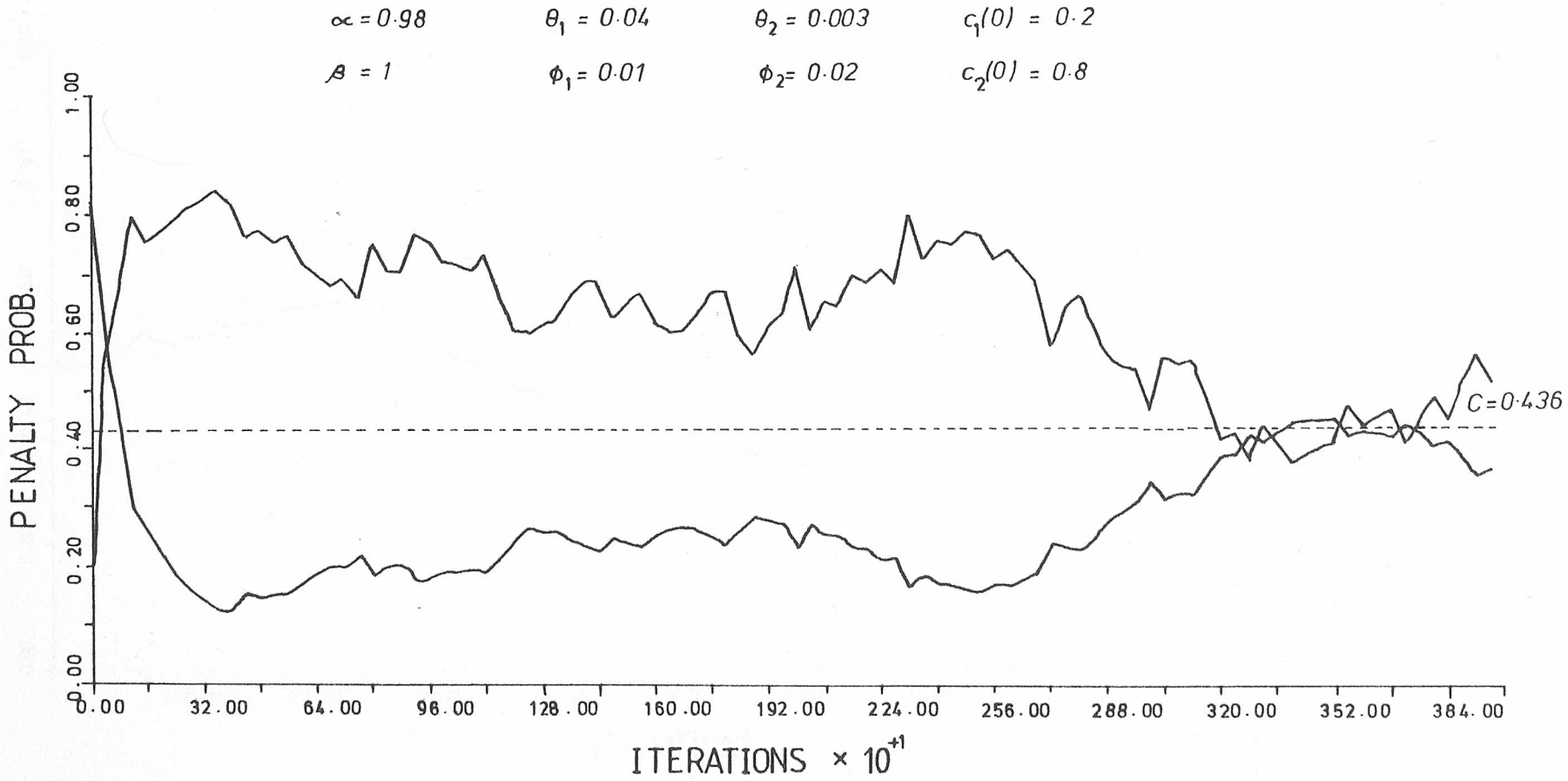

$\alpha = 0.98$
$\theta_1 = 0.04$
$\theta_2 = 0.003$
$c_1(0) = 0.2$
$\beta = 1$
$\phi_1 = 0.01$
$\phi_2 = 0.02$
$c_2(0) = 0.8$
PENALTY PROB.
1.00
0.80
0.60
0.40
0.20
0.00
C = 0.436
0.00
32.00
64.00
96.00
128.00
160.00
192.00
224.00
256.00
288.00
320.00
352.00
384.00
ITERATIONS × 10^{+1}

Figure 4.6 (b)

$\alpha = 0.98$ $\theta_1 = 0.004$ $\theta_2 = 0.001$ $c_1(0) = 0.25$

$\beta = 1$ $\phi_1 = 0.002$ $\phi_2 = 0.003$ $c_2(0) = 0.75$

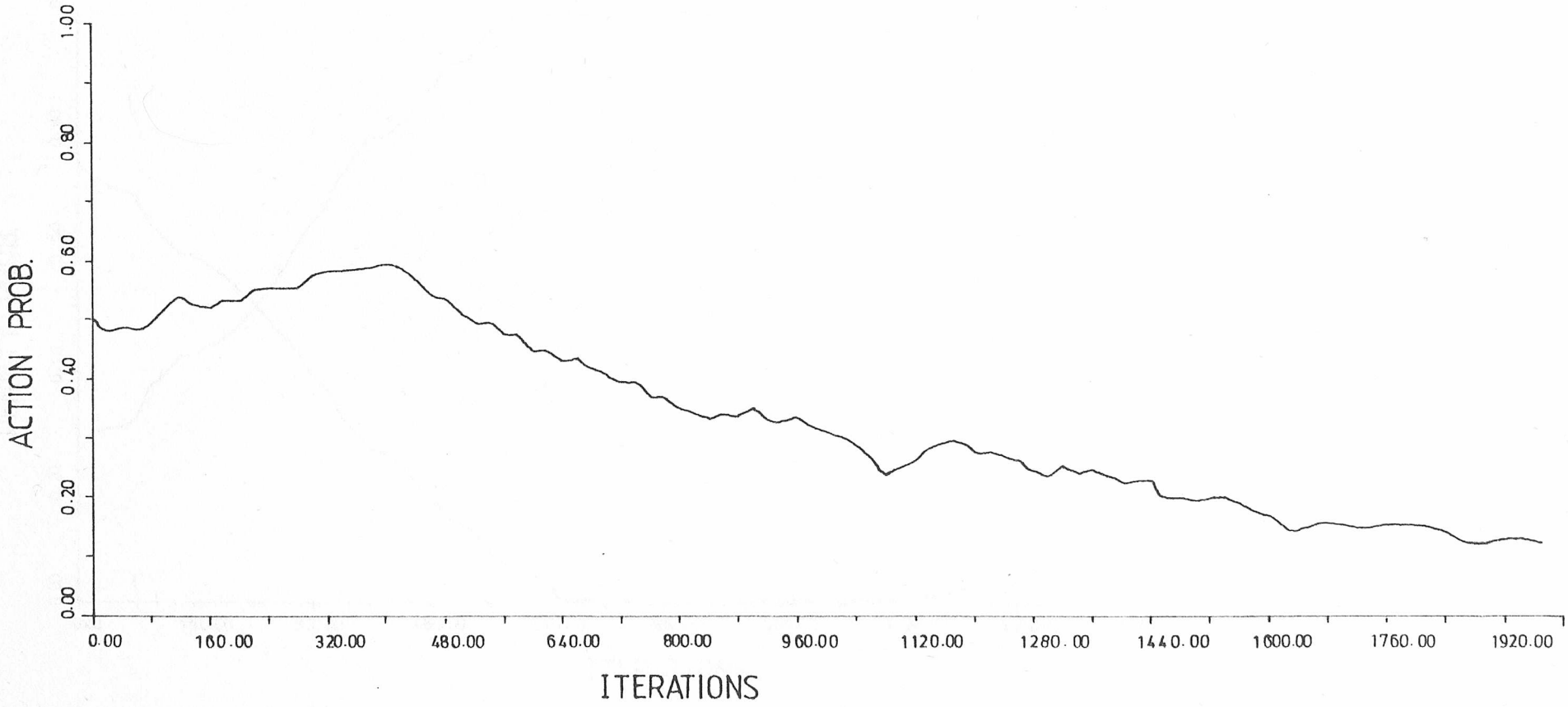


Figure 4.7(a)

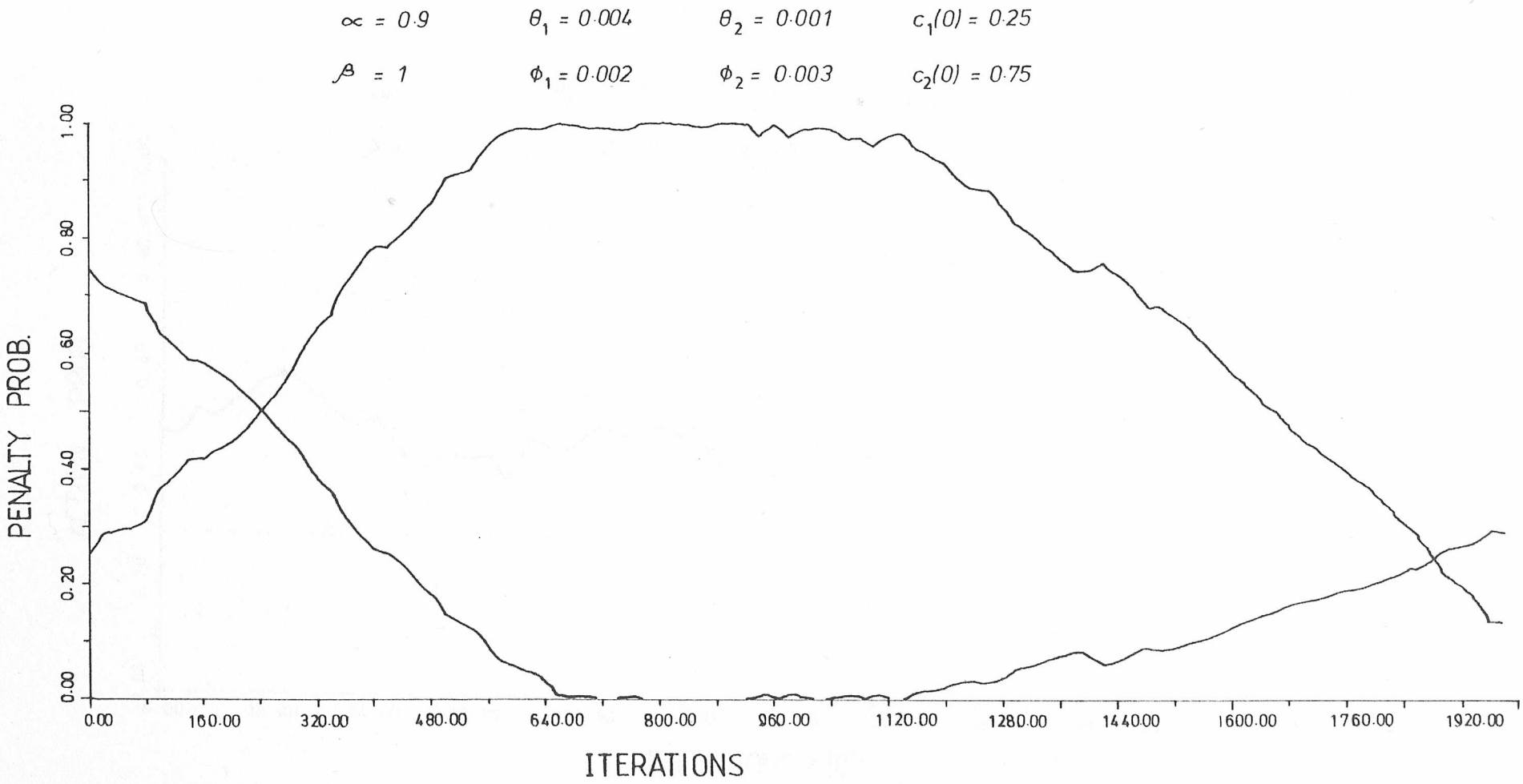

α = 0.9
θ_1 = 0.004
θ_2 = 0.001
$c_1(0)$ = 0.25
β = 1
ϕ_1 = 0.002
ϕ_2 = 0.003
$c_2(0)$ = 0.75
PENALTY PROB.
0.00
0.20
0.40
0.60
0.80
1.00
0.00
160.00
320.00
480.00
640.00
800.00
960.00
1120.00
1280.00
1440.00
1600.00
1760.00
1920.00
ITERATIONS

Figure 4.7(b)

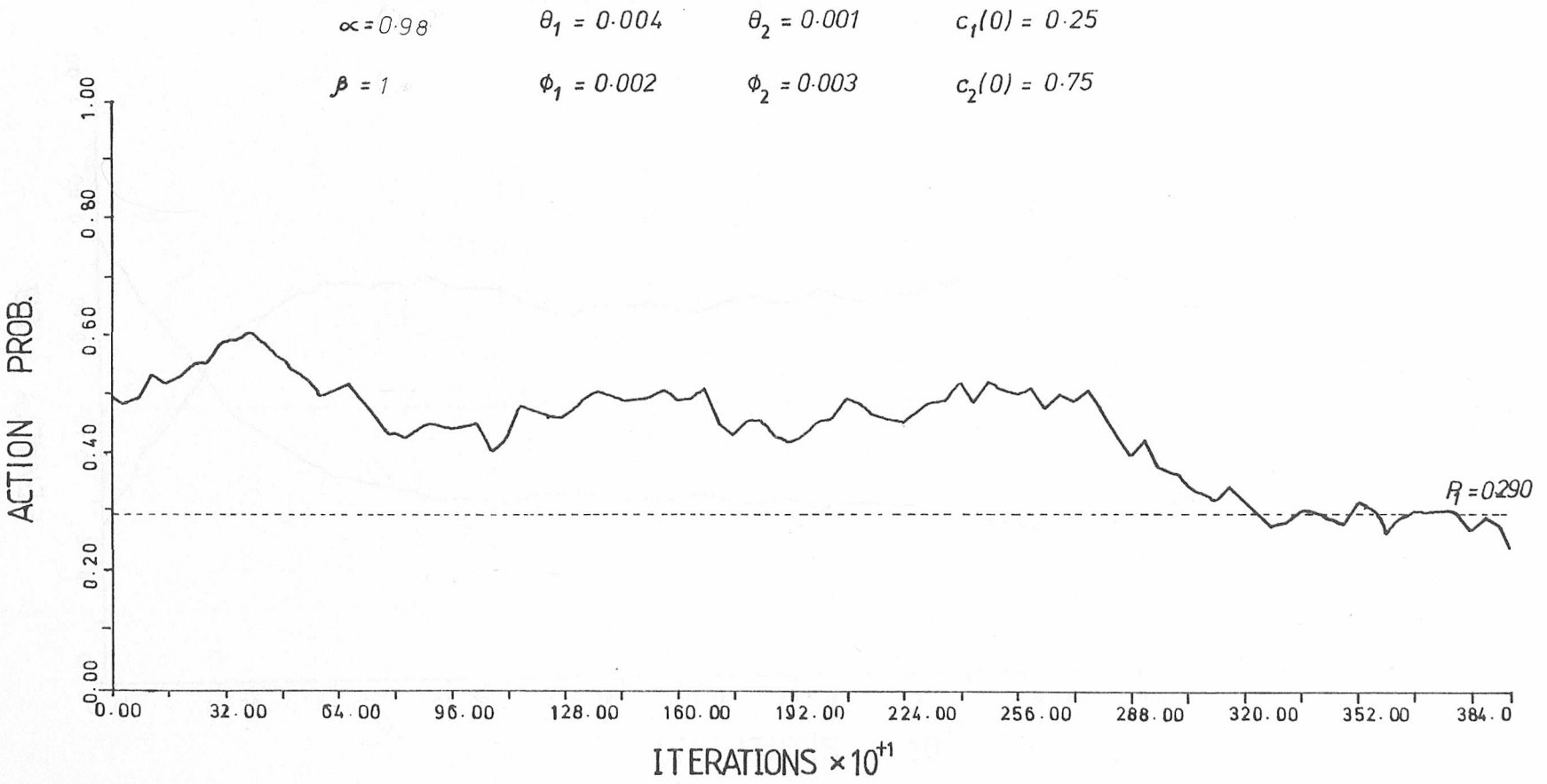

$\alpha = 0.98$
$\theta_1 = 0.004$
$\theta_2 = 0.001$
$c_1(0) = 0.25$
$\beta = 1$
$\phi_1 = 0.002$
$\phi_2 = 0.003$
$c_2(0) = 0.75$
ACTION PROB.
1.00
0.80
0.60
0.40
0.20
0.00
$R_1 = 0.290$
0.00
32.00
64.00
96.00
128.00
160.00
192.00
224.00
256.00
288.00
320.00
352.00
384.0
ITERATIONS $\times 10^{+1}$

Figure 4.8 (a)

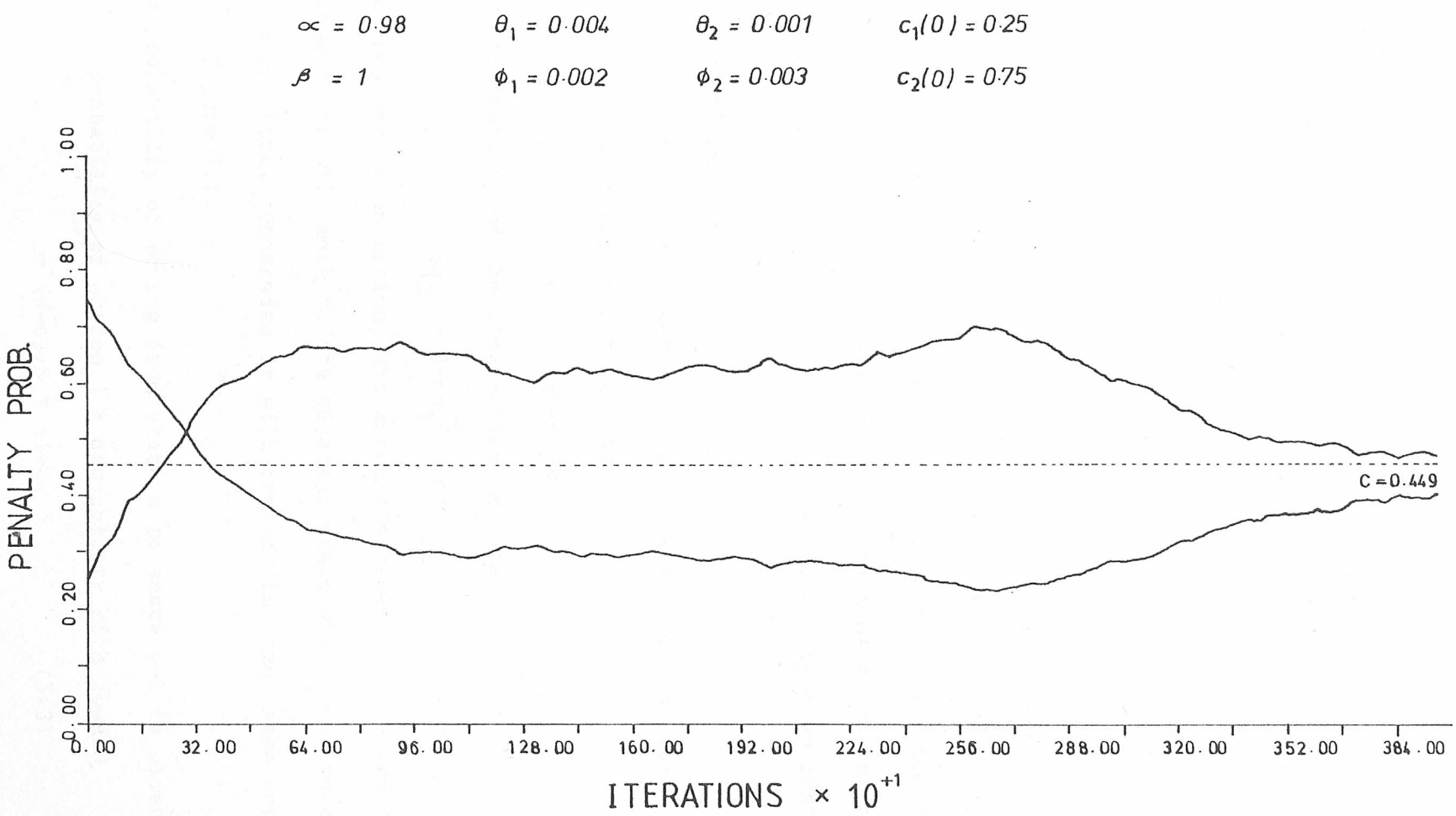

$\alpha = 0.98$
$\theta_1 = 0.004$
$\theta_2 = 0.001$
$c_1(0) = 0.25$
$\beta = 1$
$\phi_1 = 0.002$
$\phi_2 = 0.003$
$c_2(0) = 0.75$
PENALTY PROB.
1.00
0.80
0.60
0.40
0.20
0.00
C = 0.449
0.00
32.00
64.00
96.00
128.00
160.00
192.00
224.00
256.00
288.00
320.00
352.00
384.00
ITERATIONS × 10^{+1}

Figure 4·8 (b)

CHAPTER 5 PROBABILISTIC TSETLIN AUTOMATA

Introduction

The Tsetlin automaton was of interest for use in autonomous environments because of its simple structure and good performance under certain conditions. In non-autonomous environments the performance of the Tsetlin was found less satisfactory because of its deterministic output function. Three automata based on the Tsetlin structure, deterministic in operation but with stochastic output functions were proposed. These were investigated in the hope that they would be suitable for use in non-autonomous environments. The automata were named Tri, Tip and Trp using the naming convention used for the Lrp automata with T representing a Tsetlin type automaton.

The Tri Automaton

The Tri automaton has a series of N-1 states. In any state n the probability of choosing action α_1 is

$$pa_1 = (N-n)/N \qquad (5.1)$$

and the probability of choosing action α_2 is

$$pa_2 = 1-pa_1 = n/N \qquad (5.2)$$

if, in response to an action, the automaton receives a reward from the environment it will move to the adjacent state which will select that action more often. Otherwise it will remain in the same state as shown in Figure 5.1.

The probability of moving from state n to state n-1 is given by

probability of action 1 * probability of a reward

$$= (N-n)/N * (1-c_1) \qquad (5.3)$$

The probability of moving from state n to state n+1 is given by

probability of action 2 * probability of a reward

$$= n/N * (1-c_2) \qquad (5.4)$$

The probability of remaining in the same state is given by

probability of action 1 * probability of a penalty

+ probability of action 2 * probability of a penalty

$$= (N-n)/n * c_1 + (n/N)\, c_2 \qquad (5.5)$$

From the above equations the Markov transition matrix can be found and is given in Appendix 3.

The Tip Automaton

The Tip automaton has a series of N-1 states like the Tri automaton. If, in response to an action, the automaton receives a penalty it will move to the adjacent state which will select that action less often. Otherwise it will remain in the same state as shown in Figure 5.2.

The probability of moving from state n to state n+1 is given by

probability of action 1 * probability of a penalty

$$= (N-n)/N * c_1 \qquad (5.6)$$

The probability of moving from state n to state n-1 is given by

probability of action 2 * probability of a penalty

$$= (n/N)c_2 \qquad (5.7)$$

The probability of remaining in the same state is given by

probability of action 1 * probability of a reward

+ probability of action 2 * probability of a reward

$$=((N-n)/N)(1-c_1) + (n/N)(1-c_2) \qquad (5.8)$$

From the above equations the Markov transition matrix can be found and is given in Appendix 3.

The Trp Automaton

The Trp automaton has a series of N-1 states like the Tri and Tip automata. If in response to an action the automaton receives a reward it will move to the adjacent state which will select that action more often. Otherwise in response to an action the automaton will receive a penalty and will move to the adjacent state which will select that action less often as shown in Figure 5.3.

The probability of moving from state n to state n+1 is given by

probability of action 1 * probability of a penalty

+ probability of action 2 * probability of a reward

$$=((N-n)/N)c_1 + (n/N)(1-c_2) \qquad (5.9)$$

The probability of moving from state n to state n-1 is given by

probability of action 1 * probability of a reward

+ probability of action 2 * probability of a penalty

$$=((N-n)/N)(1-c_1) + (n/N)c_2 \qquad (5.10)$$

From the above equations the Markov transition matrix can be found and is given in Appendix 3.

The Lrp Automaton

In this and previous chapters the Lrp automaton has been used as a reference. Theoretical results for the Tsetlin, Krylov, modified Tsetlin and probabilistic Tsetlin automata are compared to corresponding results for the Lrp automaton.

Using the normal description of the Lrp automaton as a variable structure automaton, with a variable transition matrix operating on the action probability vector Pa, the Lrp automaton cannot be analysed like the Tsetlin, Krylov and probabilistic Tsetlin automata have been, as this requires a fixed structure to be able to construct a Markov transition matrix. However by imposing a set of states on the Lrp

automaton and limiting the action probabilities from a continuous range to a discrete range corresponding to the states, a fixed structure is produced and a Markov transition matrix can be found. Unlike the Tsetlin automata, movement is not only to adjacent states but can be to any state in the automaton. The probability of movement to states corresponding to the result of the updating algorithm given in equations 1.7-1.10 is high while the probability of movement to states far from the updating algorithm result is low. In the specification of the probabilistic Tsetlin automata, absorbing end states were excluded since they were unwanted. To be able to compare like with like the absorbing states were excluded from the analysis of the Lrp automaton. Thus the Lri automaton as presented here could not go optimal as the Lri automaton normally would. This gives the Lri automaton a better performance than the Lrp and it is the Lri that is used for comparison later in the chapter.

The Lrp automaton can be considered as a set of N-1 states. In state n the probability of choosing action 1 is

$$pa_1 = (N-n)/n \qquad (5.11)$$

and the probability of choosing action 2 is

$$pa_2 = 1-pa_1 = n/N \qquad (5.12)$$

The response to an action is applied to an algorithm which determines the new action probabilities. The new action probability for action α_1 is from equations (1.7)-(1.10)

$$\begin{aligned} pa_1(n+1) &= pa_1(n)(1-c_1)(1-\alpha * pa_2(n)) \\ &+ pa_1(n)c_1(\beta * pa_1(n)) \\ &+ pa_2(n)(1-c_2)(\alpha * pa_1(n)) \\ &+ pa_2(n)c_2(1-\beta * pa_2(n)) \end{aligned} \qquad (5.13)$$

The automaton can move to any other state or remain in the same state

depending on the new action probability. The probability of moving to state n is [29]

$$[p1^{(n-1)}][(1-p1)^{(N-n-1)}][(N-2)!/((n-1)!(N-1-n))] \qquad (5.14)$$

The element pij, an element in the Markov transition matrix is given by the probability of moving from state i to j.

$$\begin{aligned} pij=&[((N-i)/N)(1-c_1)(1-\alpha(i/n))+((N-i)/N)\, c_1\, \beta(N-i)/N \\ &+(i/N)(1-c_2)\alpha((N-i)/N)+(i/N)c_2\,(1-\beta(i/N))]^{(j-1)} \\ *&[1-(((N-i)/N)(1-c_1)(1-\alpha(i/N))+((N-i)/N)\, c_1\, \beta((N-i)/N) \\ &+(i/N)(1-c_2)\alpha(N-i)/N + (i/N)c_2\,(1-\beta(i/N))]^{(N-j-1)} \\ &*(N-2)!/[(N-1)!(N-j-1)!] \qquad (5.15) \end{aligned}$$

Tri,Tip,Trp and Lrp Automata-Theoretical Results

The theoretical degree of optimality and mean switching times of the Tri, Tip, Trp and Lrp automata were calculated for various autonomous environments and the results are shown in Figures 5.4-5.7. The environments used are the same as used in Figures 3.3-3.6. Figures 5.4 show that the Tri and Trp automata have the most optimal performance while the Tip automaton has poor optimality which does not improve as the memory size is increased. However Figures 5.5 reveal that the Tri automaton has large mean switching times compared to the other automata and because of this the Tri automaton has been omitted from Figures 5.6 and 5.7. Figures 5.5 have a constant difference between the penalty probabilities and again show that the performance of the Tip automaton is limited. At high penalty probabilities the performance of the Trp automaton is like that of the Tip automaton but as the penalty probabilities are reduced the performance improves. Figures 5.7 showing mean switching time results again shows the similarity between the Trp and Tip automata at high penalty probabilities while at low penalty probabilities the Trp automaton has

relatively high switching times like the Tri automaton. As stated earlier the memory sizes used here for the Lrp automaton are smaller than used in practice giving the effect of lower degrees of optimality than would normally be found and lower switching times.

The automaton used as the reference is a Lri automaton with α=0.6. This automaton was chosen after obtaining results for the Lrp automaton like those shown in Figures 5.8-5.10. These results show the probabilities of the individual states of the automaton in addition to the overall action probabilities and mean switching times as shown in Figures 5.4-5.7. Also included in Figure 5.10 is the average penalty received by the automaton which is used as a measure of performance. These results are for automata with 19 states, a value which was smaller than desired but which was close to the limit imposed by speed and accuracy. Figures 5.8-5.10 (a)-(b) show distributions for Lrp automata with constant δ but varying α and β while Figures 5.8-5.10 (c)-(d) have α constant but varying β and δ. The distributions confirm that increasing δ makes the automaton more optimal. What can also be seen is that increasing δ makes the distribution spread. Comparing Figures 5.8-10 (a)-(b) it can be seen that even with δ constant, lowering α and β can make the automaton more optimal and make the distribution narrower. The automaton which performs best is the Lri with α=0.6. This performance could be improved by lowering αwhich would also decrease the learning time of the automaton however it has been found in practice that automata which learn too quickly do not perform well. The value of α=0.6 was taken as a compromise between good theoretical results and practical considerations.

Figures 5.11 and 5.12 compare, for a range of penalty probabilities, the state probability distributions of the Tri, Tip and Trp automata with the best Lrp automaton selected from Figures 5.8-10. These results reveal in more detail the operation of the automata. To accurately determine the performance of the automata the average penalties received by the automata were calculated and included in the figures. In Figure 5.11(a) the Trp automaton performs best but in Figures 5.11(b)-(d) it is the Tri automaton which has the best performance. The probability distributions show that as the penalty probabilities increase the operation of the Trp automaton becomes more like that of the Tip automaton. Throughout this series of results the Trp automaton performs better than the Lrp.

Tri-Operation

The Tri automaton when at the centre of its range will select both actions equally and move to select the action which gives most rewards. However, at either end of its range the automaton will select one action far more often than the other. Because the automaton changes state only in response to a reward it will tend to move in response to rewards from the action it is selecting most often. Thus the automaton tends to move to the extremes of its range. Over most of its range it will tend to move toward selecting the action corresponding to the smaller penalty probability but over part of its range it will tend to move toward selecting the action corresponding to the larger penalty probability. The division between these two stable ranges occurs when

number of rewards from action 1 = number of rewards from action 2

$$\Rightarrow \quad pa_1 (1-c_1) = (1-pa_1)(1-c_2)$$

$$\Rightarrow \quad pa_1 = (1-c_2)/(2-c_1-c_2) \qquad (5.16)$$

Tip-Operation

At the centre of its range, the Tip automaton will select both actions equally and will move away from the action which gives the most penalties. It will continue to move until the number of penalties tending to make it move in opposite directions becomes equal. The automaton has two unstable regions with a steady state between the two when

$$\text{number of penalties from action 1} = \text{number of penalties from action 2} \qquad (5.17)$$

$$\Rightarrow \quad pa_1 \, c_1 = (1-pa_1)c_2$$

$$\Rightarrow \quad pa_1 = c_2/(c_1+c_2) \qquad (5.18)$$

Trp-Operation

The Trp automaton can be considered as a combination of the Tri and Tip automata. Thus when the penalty probabilities are small the automaton will receive few penalties and the automaton will show behaviour like the Tri automaton. When the penalty probabilities are high there will be few rewards and the automaton will behave like the Tip automaton. The automaton will tend not to change state when

$$\text{penalties from action 1} + \text{rewards from action 2} = \text{penalties from action 2} + \text{rewards from action 1}$$

$$\Rightarrow \quad pa_1 \, c_1 + (1-pa_1)(1-c_2) = (1-pa_1)c_2 + pa_1 (1-c_1)$$

$$\Rightarrow \quad pa_1 = (1-2c_2)/(2-2c_1-2c_2) \qquad (5.19)$$

When this equation has a value outwith the range (0,1) there is nowhere, other than at one of the end states, where the automaton will reach steady state.

The operation of the Trp automaton is illustrated well in Figures 5.12(a)-(d). In Figure 5.12(a) with small penalty probabilities the distribution of the Trp automaton is similar to that of the Tri automaton. In Figure 5.12(b) with large penalty probabilities the distribution of the Trp automaton is similar to that of the Tip automaton. Figures 5.12(c)-(d) show the Trp with penalty probabilities equal and about the value of a half. When the penalty probabilities are about the value of a half equation (5.18) does not have a solution in the range (0,1) and the Trp automaton exhibits its best performance being like neither the Tri nor the Tip. Figure 5.12(d) illustrates this by showing the distribution with penalty probabilities of 0.51 and 0.49.

Non-Autonomous Environments-Results

In addition to their performance in autonomous environments the performance of the probabilistic Tsetlin automata in non-autonomous environments is also of interest. Figures 5.13-5.15 give the state probability distributions, the average penalty received by the automaton, which is the equivalent of the degree of optimality but for non-autonomous environments, and mean switching times, of the probabilistic Tsetlin automata in a range of non-autonomous environments with the Lrp automaton included as a reference. The non-autonomous environments used in these figures were chosen to provide a range of optimal action probabilities and penalty probabilities.

In Figures 5.13(a)-(f) it is the Tip automaton which has the lowest average penalty while the Lrp and Trp automata having lesser but on the whole similar performances while the Tri automaton has a poor performance. These results are reinforced by Figures 5.14(a)-(f). In

some of these figures the results for the Tri automaton have been omitted because the large mean switching times were unsuitable for inclusion in Figures 5.15(a)-(f). The form of Figures 5.15 requires some explanation. The mean switching time is defined by equation A2.1 where the state the automaton is switching to is the new optimal action probability after the environment has switched. Because the number of states in the automaton limits the number of possible action probabilities at each memory size the new optimal state was defined as the state corresponding to the action probability closest to the optimal action probability. Since the optimal action probability lies between two states, as the memory size changes the optimal state will change from corresponding to an action probability lower than the optimal action probability to higher and back. This accounts for the stepped appearance of some of the Figures 5.15. These results show that though the Lrp automaton has the best overall mean switching time results the performance of the Trp and Tip automata is still good by comparison and when combined with the average penalty results the Tip automaton could be expected to give the best performance in a non-autonomous environment.

Conclusions

The characteristics of the probabilistic Tsetlin automata in autonomous environments have parallels with the Tsetlin and Krylov automata. The Trp automaton operates at its best when the penalty probabilities are about the value of a half like the Tsetlin. The Tri automaton is stable in both actions like the Krylov automaton. Though the average penalty results for the Tri automaton are good, because the automaton is stable in both actions the response to a non-stationary environment is slow so the Trp automaton is prefered.

The probabilistic Tsetlin automata were proposed as automata which could operate well in non-autonomous environments because of their stochastic output functions. This has been shown to be true and when compared to the Lrp automaton it is the Tip automaton which has shown the best performance.

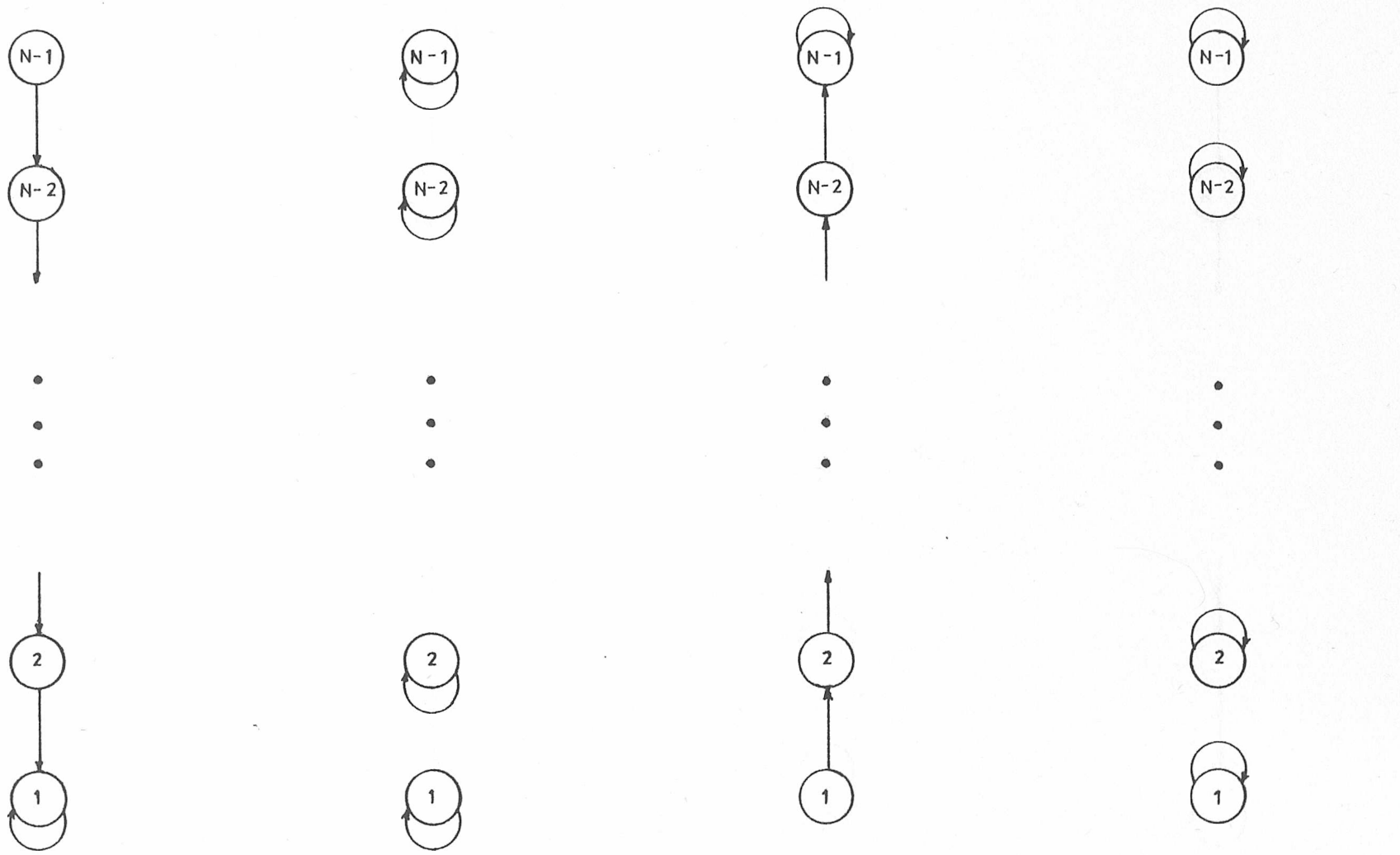
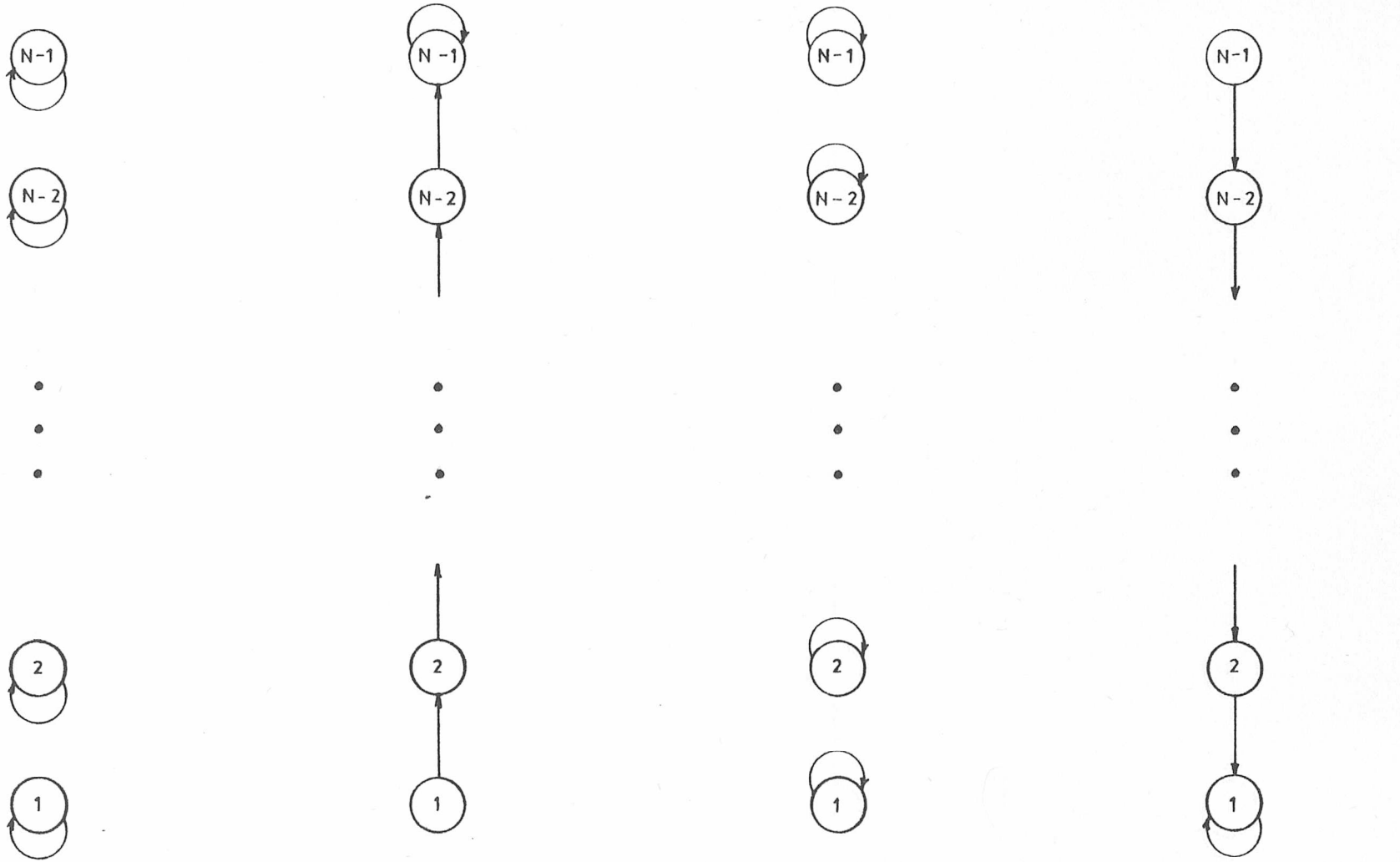
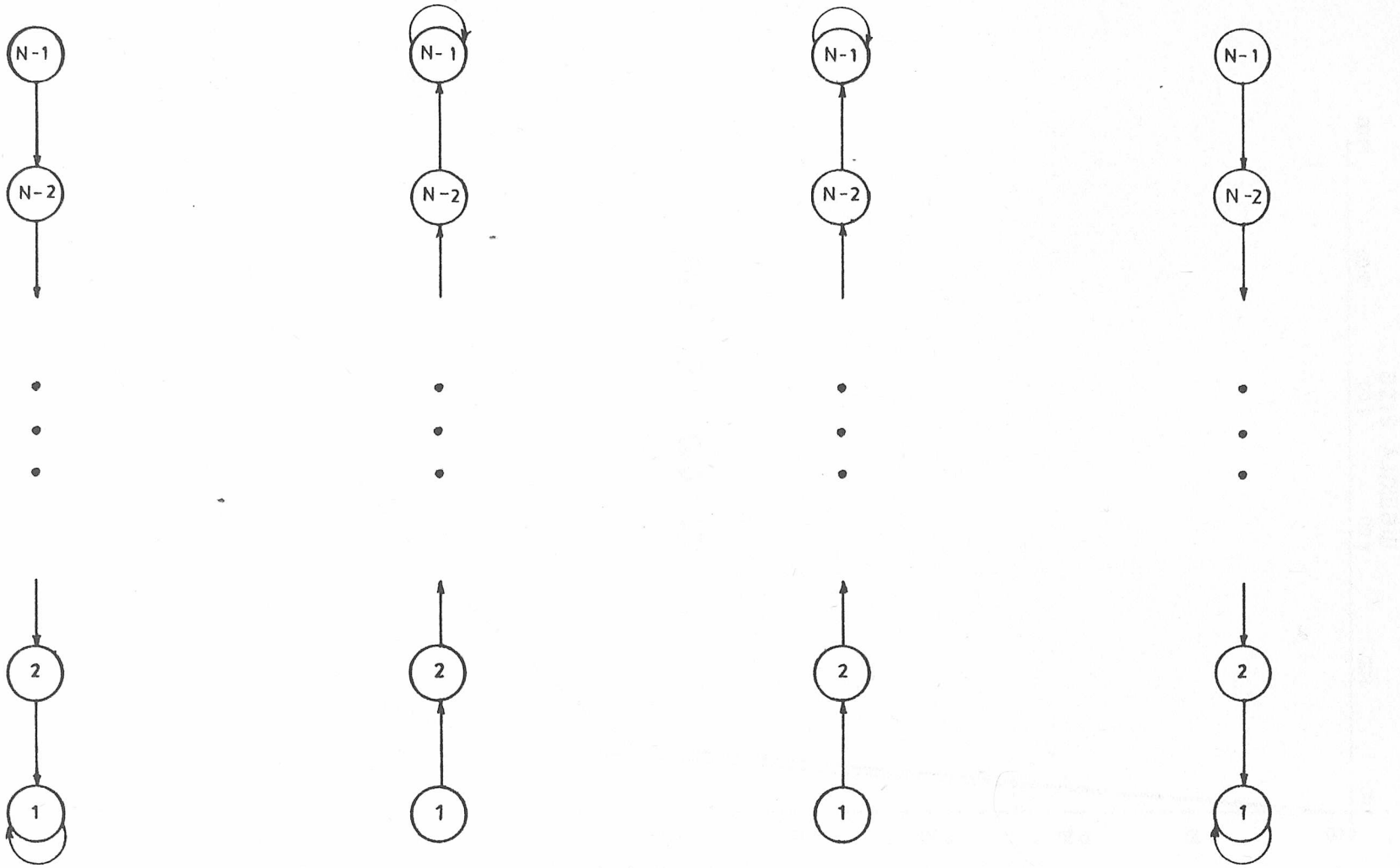


Figure 5.1 State diagram of T_{RI} automaton



action α_1 gives reward action α_1 gives penalty action α_2 gives reward action α_2 gives penalty

Figure 5.2 State diagram of T_{IP} automaton



action α_1 gives reward

action α_1 gives penalty

action α_2 gives reward

action α_2 gives penalty

Figure 5.3 State diagram of T_{RP} automaton

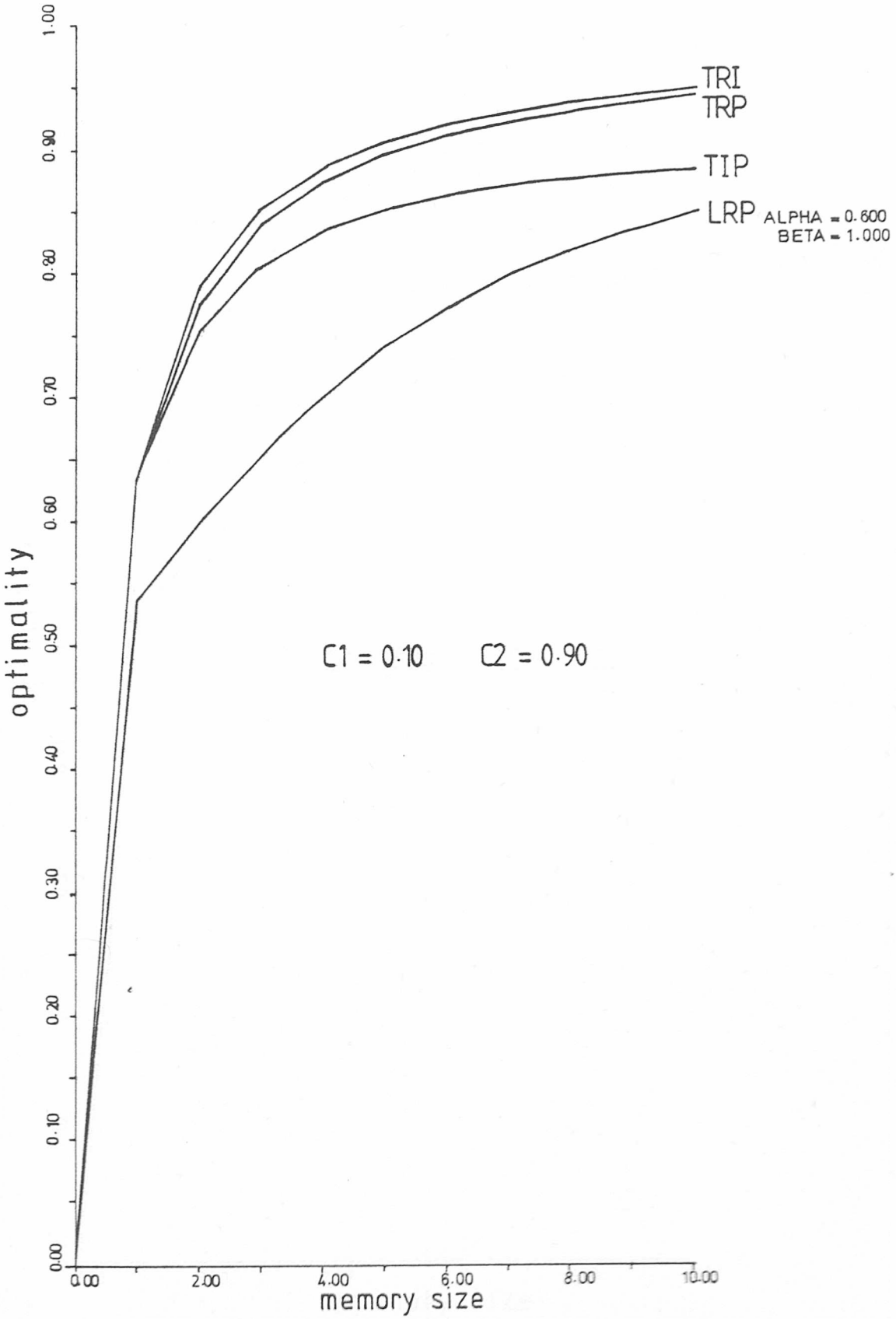


Figure 5.4(a) Theoretical steady state action probabilities

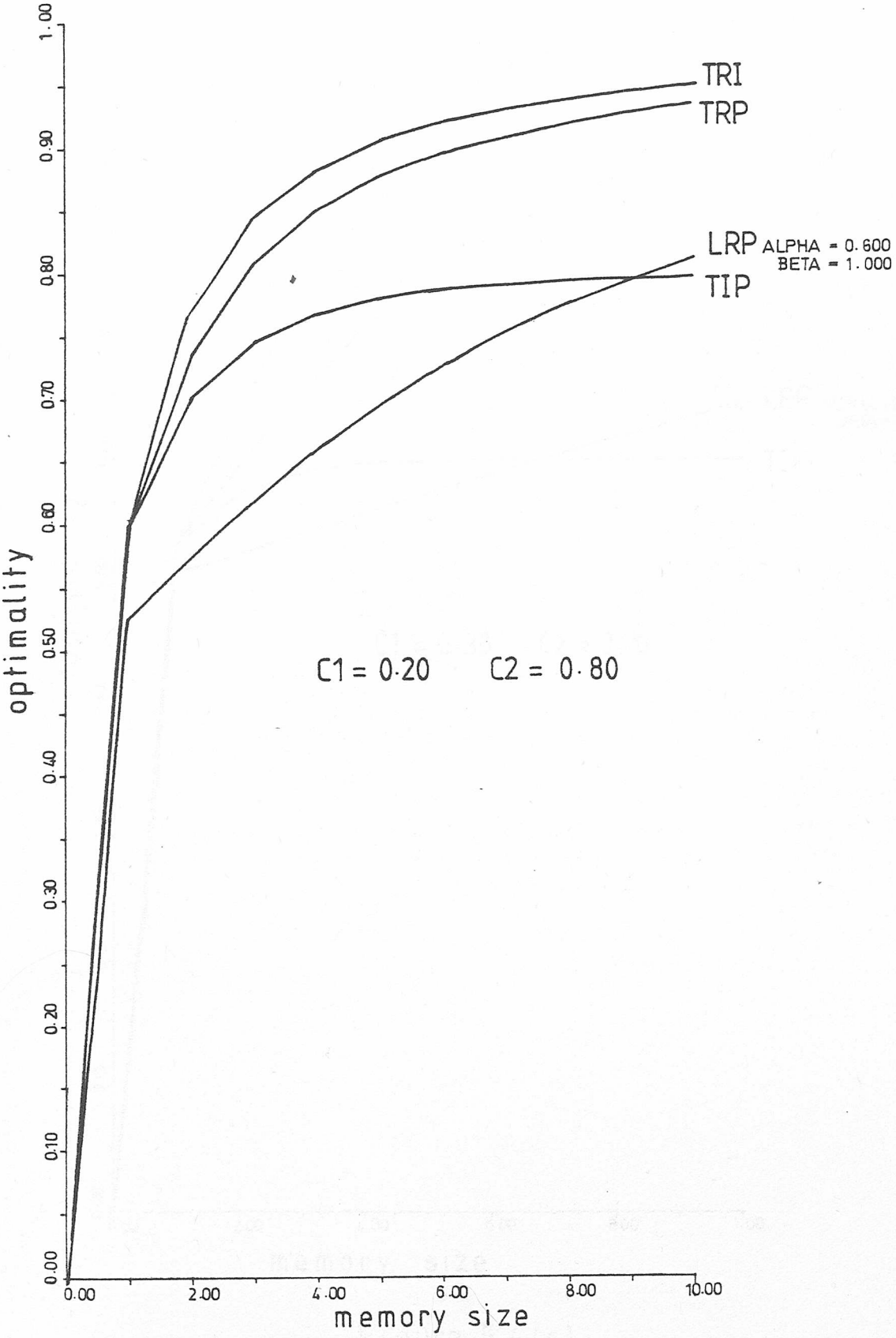

TRI
TRP
LRP ALPHA = 0.600
BETA = 1.000
TIP
C1 = 0.20 C2 = 0.80
optimality
memory size
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5.4 (b)

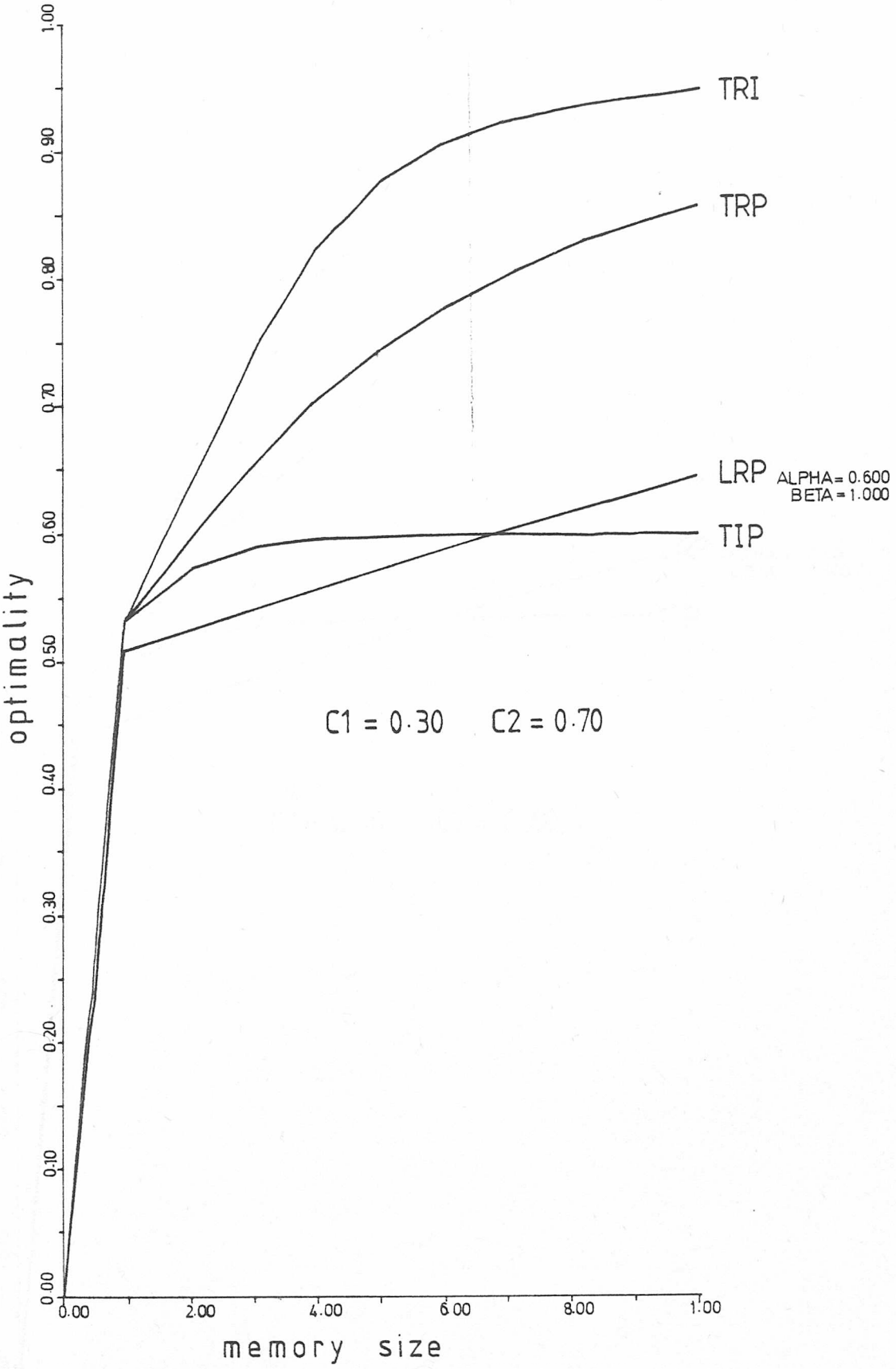

TRI
TRP
LRP
ALPHA = 0.600
BETA = 1.000
TIP
C1 = 0·30 C2 = 0·70
optimality
memory size
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
1.00

Figure 5.4(c)

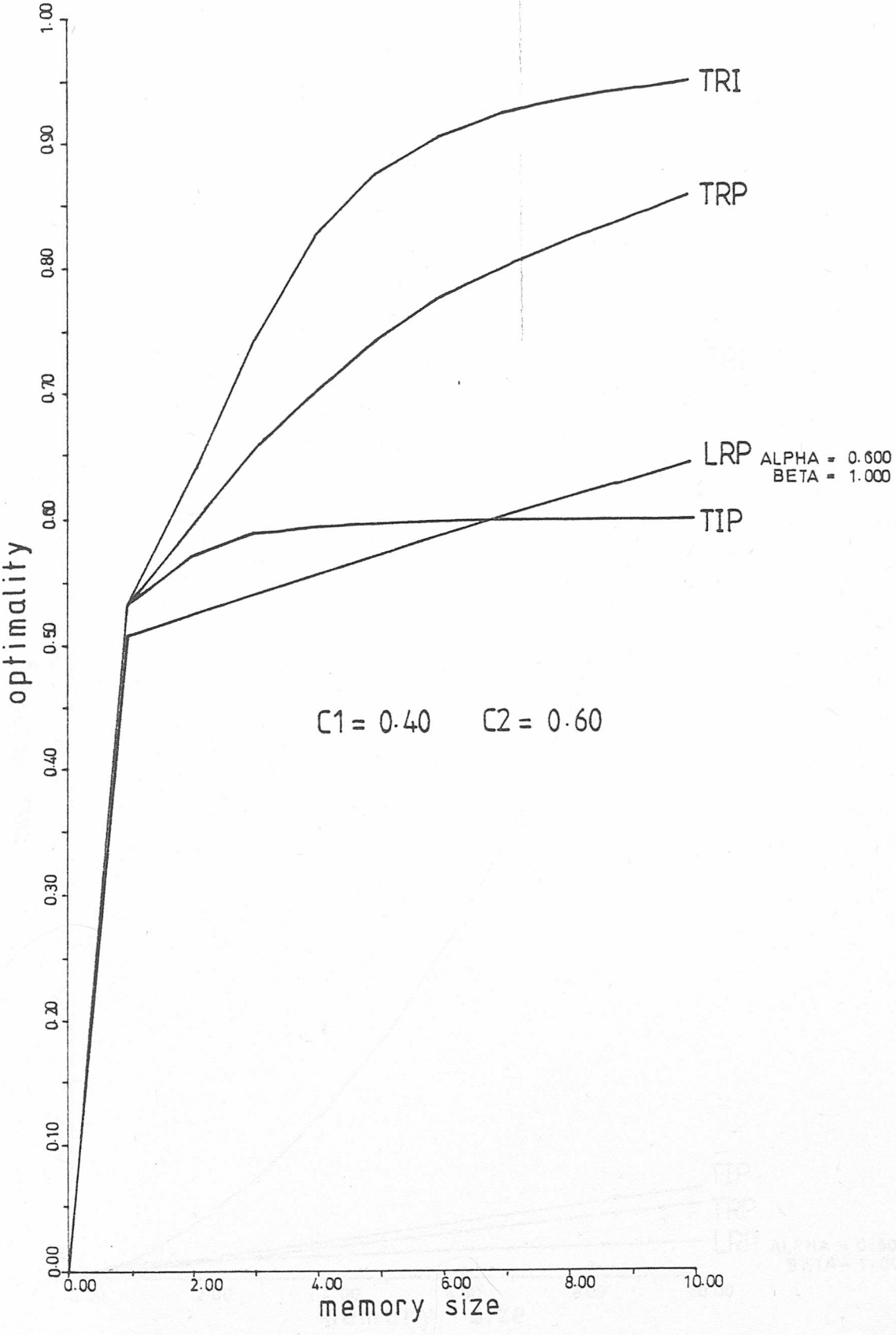

optimality
memory size
TRI
TRP
LRP
TIP
ALPHA = 0.600
BETA = 1.000
C1 = 0.40
C2 = 0.60
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5.4(d)

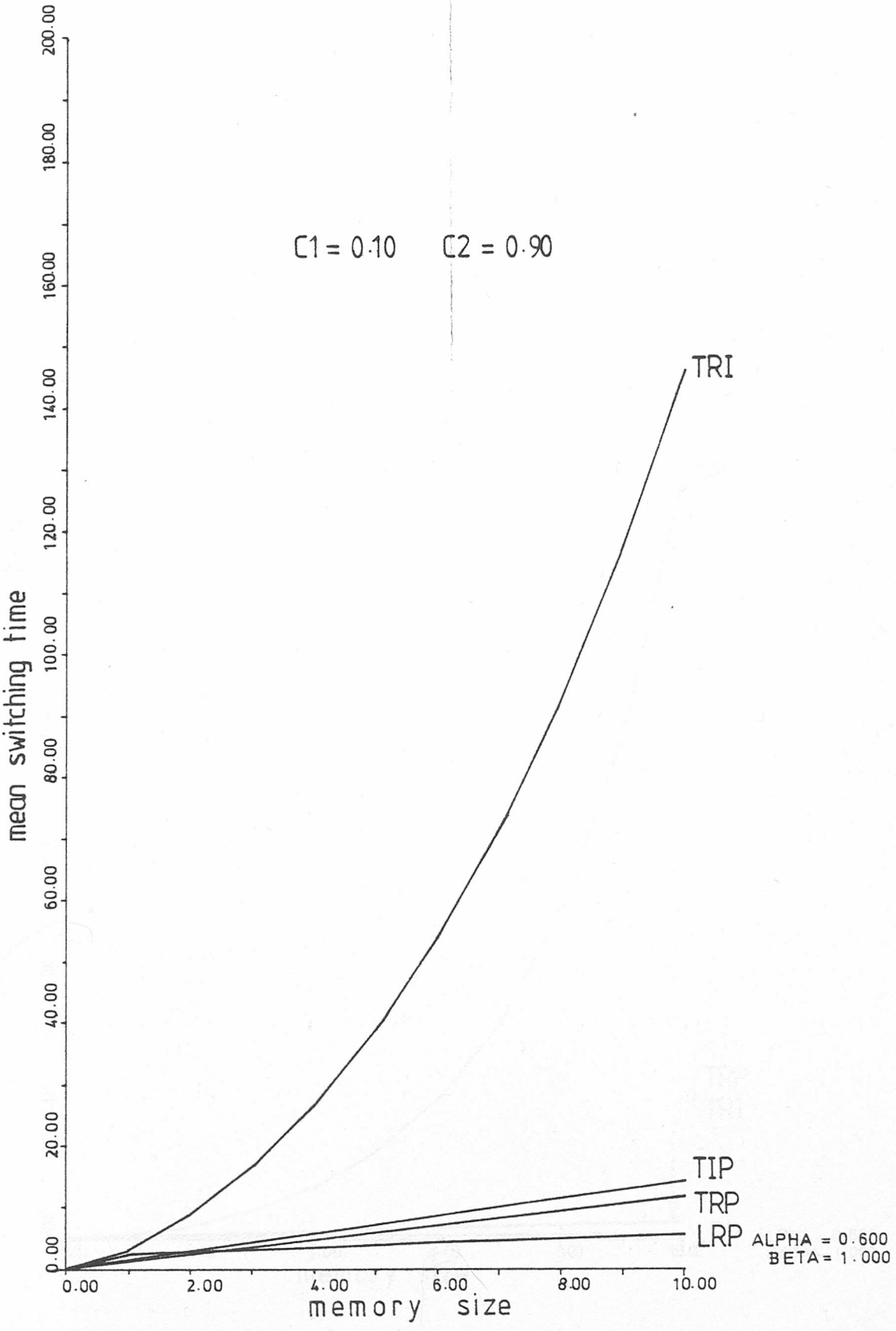


Figure 5.5(a) Theoretical mean switching times

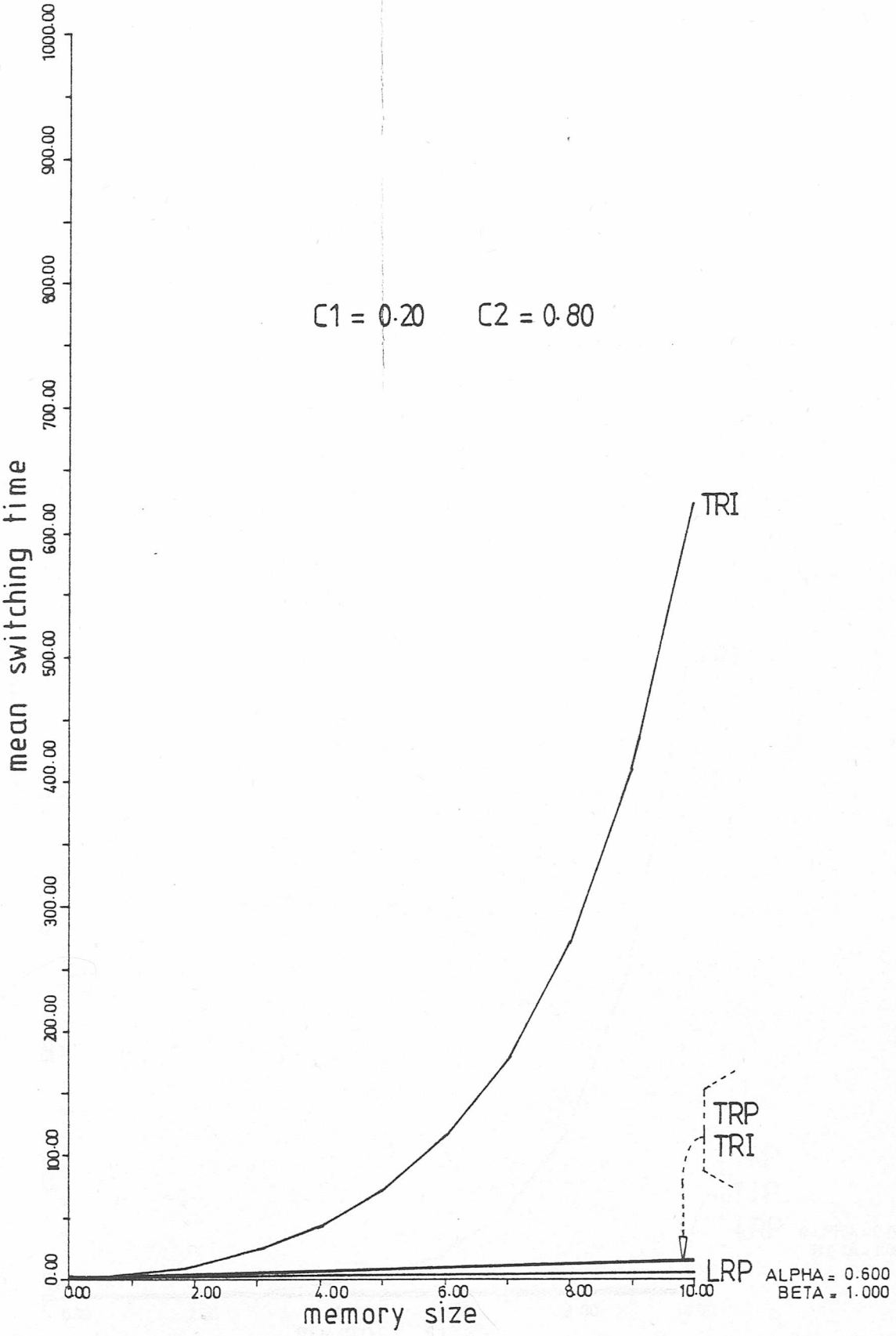

C1 = 0·20 C2 = 0·80
mean switching time
1000.00
900.00
800.00
700.00
600.00
500.00
400.00
300.00
200.00
100.00
0.00
TRI
TRP
TRI
LRP
ALPHA = 0.600
BETA = 1.000
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 5·5(b)

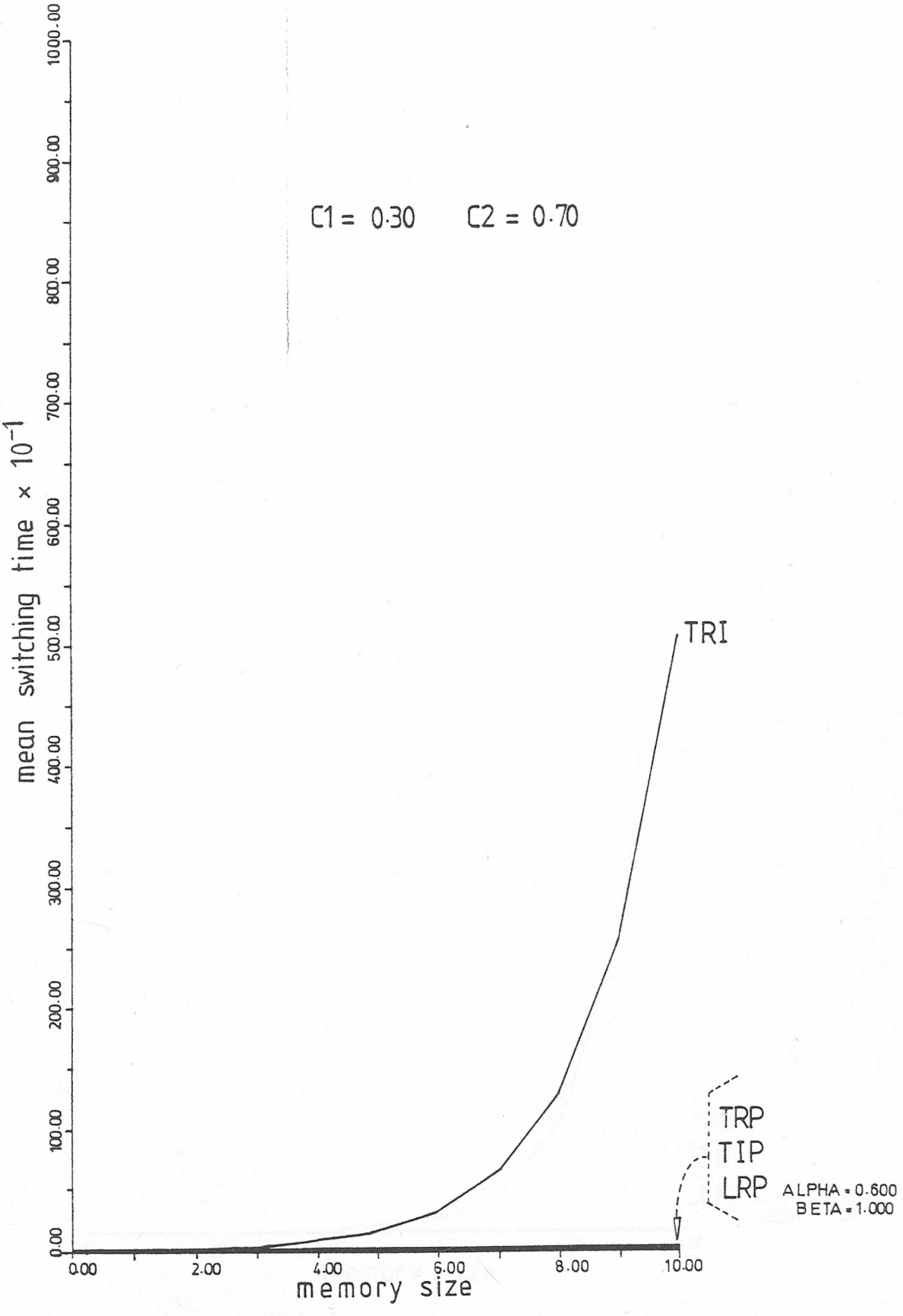

C1 = 0·30 C2 = 0·70
mean switching time × 10^{-1}
1000.00
900.00
800.00
700.00
600.00
500.00
400.00
300.00
200.00
100.00
0.00
TRI
TRP
TIP
LRP
ALPHA = 0.600
BETA = 1.000
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 5·5 (c)

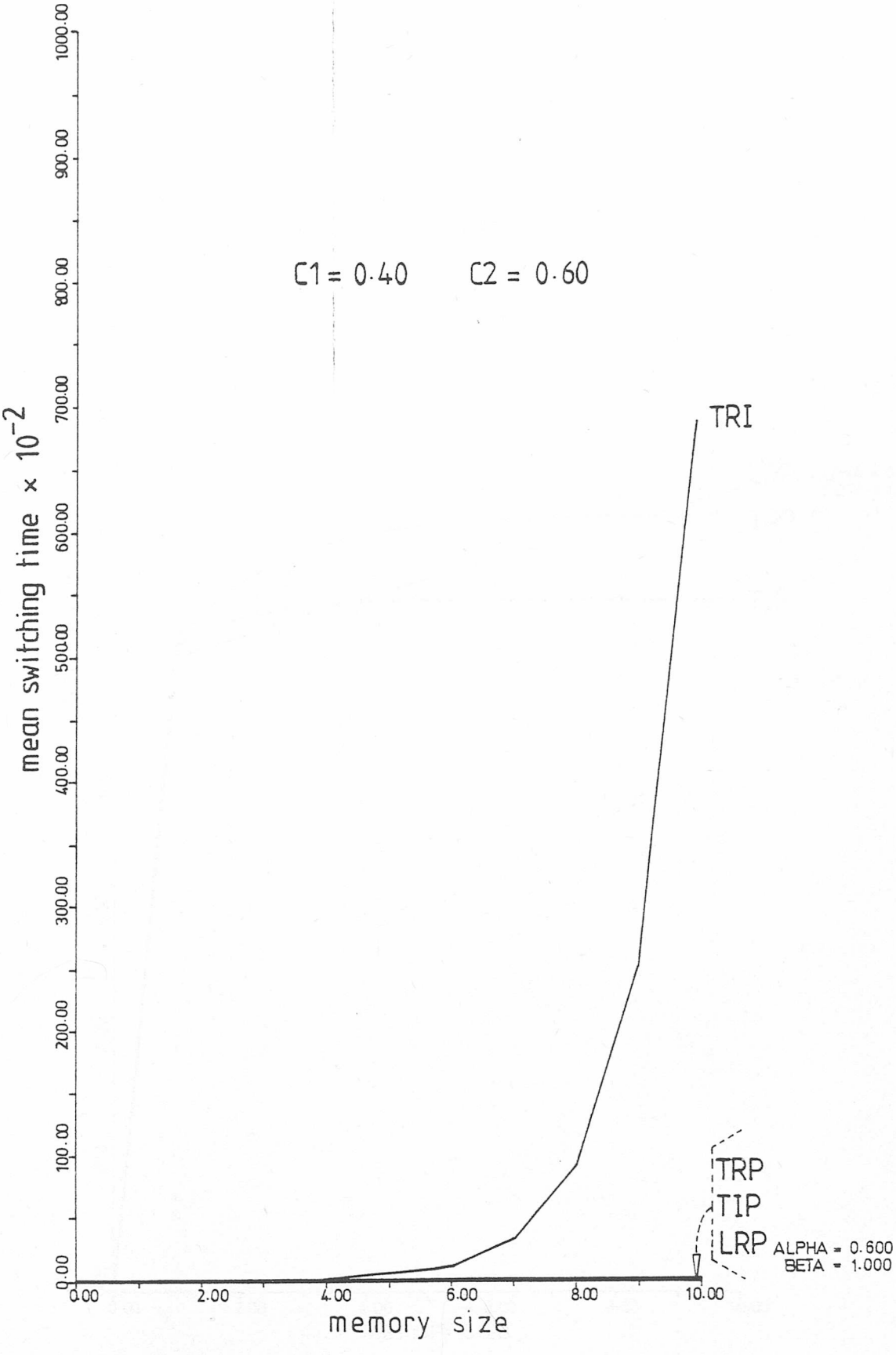

C1 = 0·40
C2 = 0·60
mean switching time × 10^{-2}
1000.00
900.00
800.00
700.00
600.00
500.00
400.00
300.00
200.00
100.00
0.00
TRI
TRP
TIP
LRP
ALPHA = 0.600
BETA = 1.000
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 5·5(d)

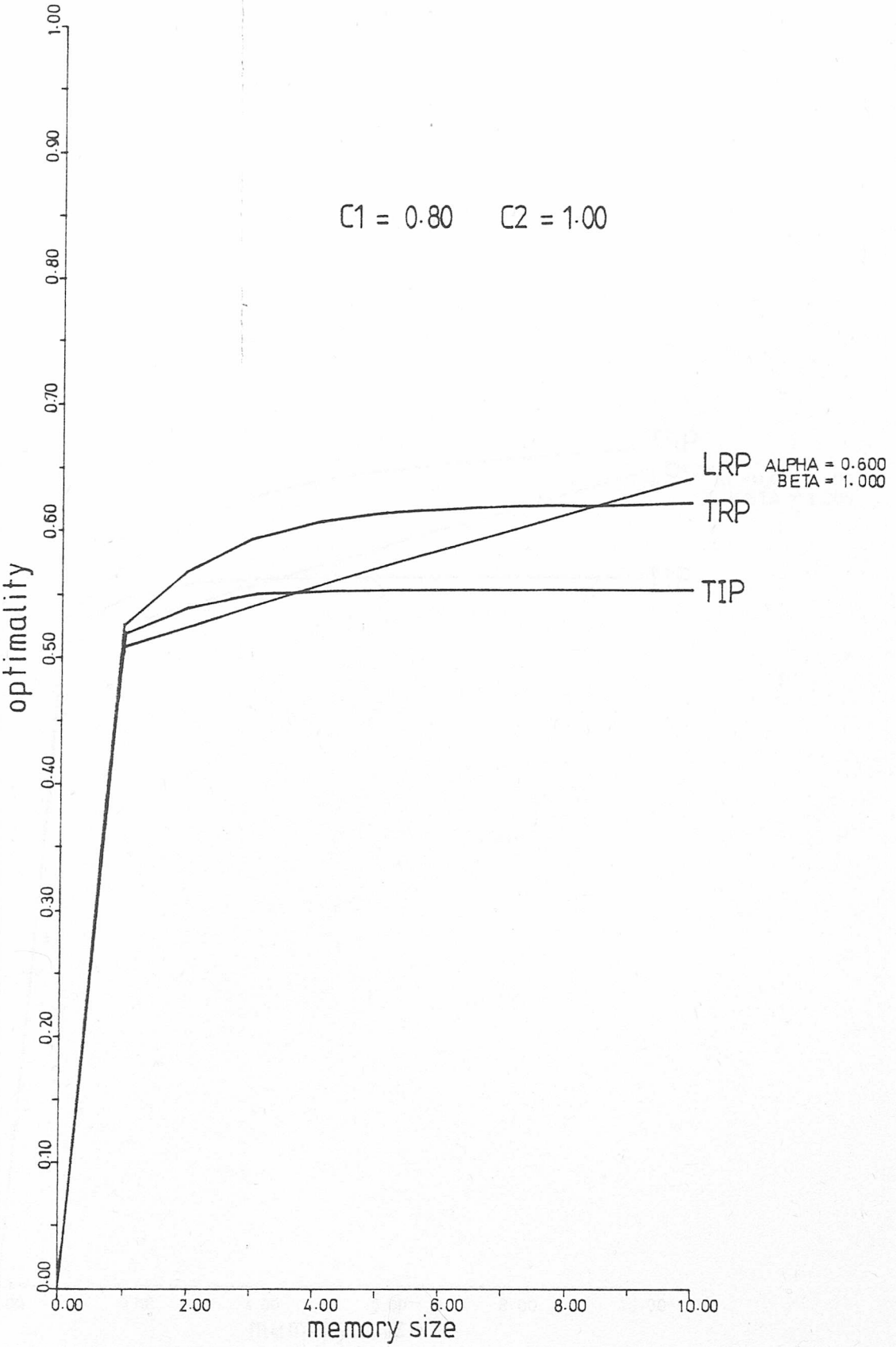


Figure 5·6 (a) Theoretical steady state action probabilities

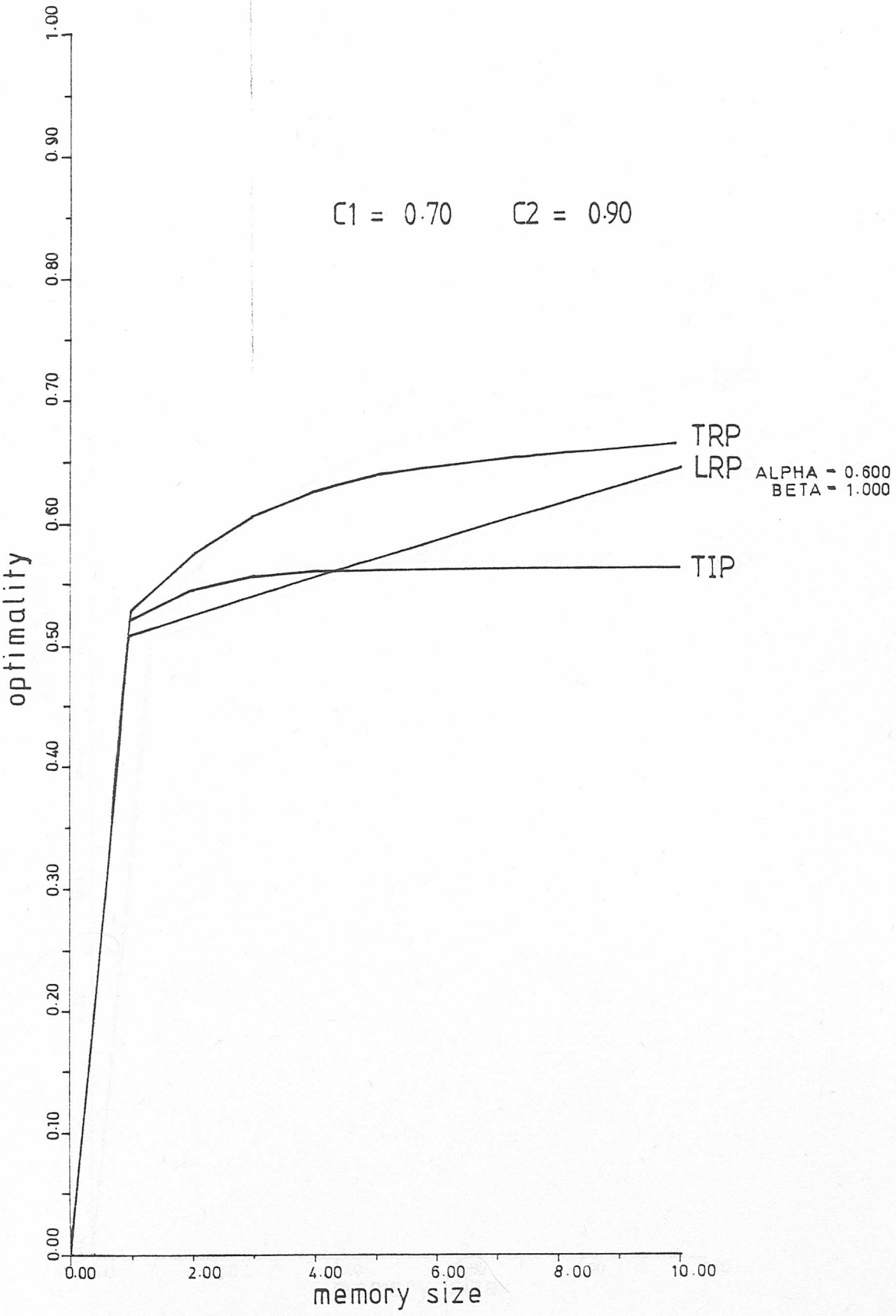

C1 = 0.70
C2 = 0.90
TRP
LRP
ALPHA = 0.600
BETA = 1.000
TIP
optimality
memory size
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5.6(b)

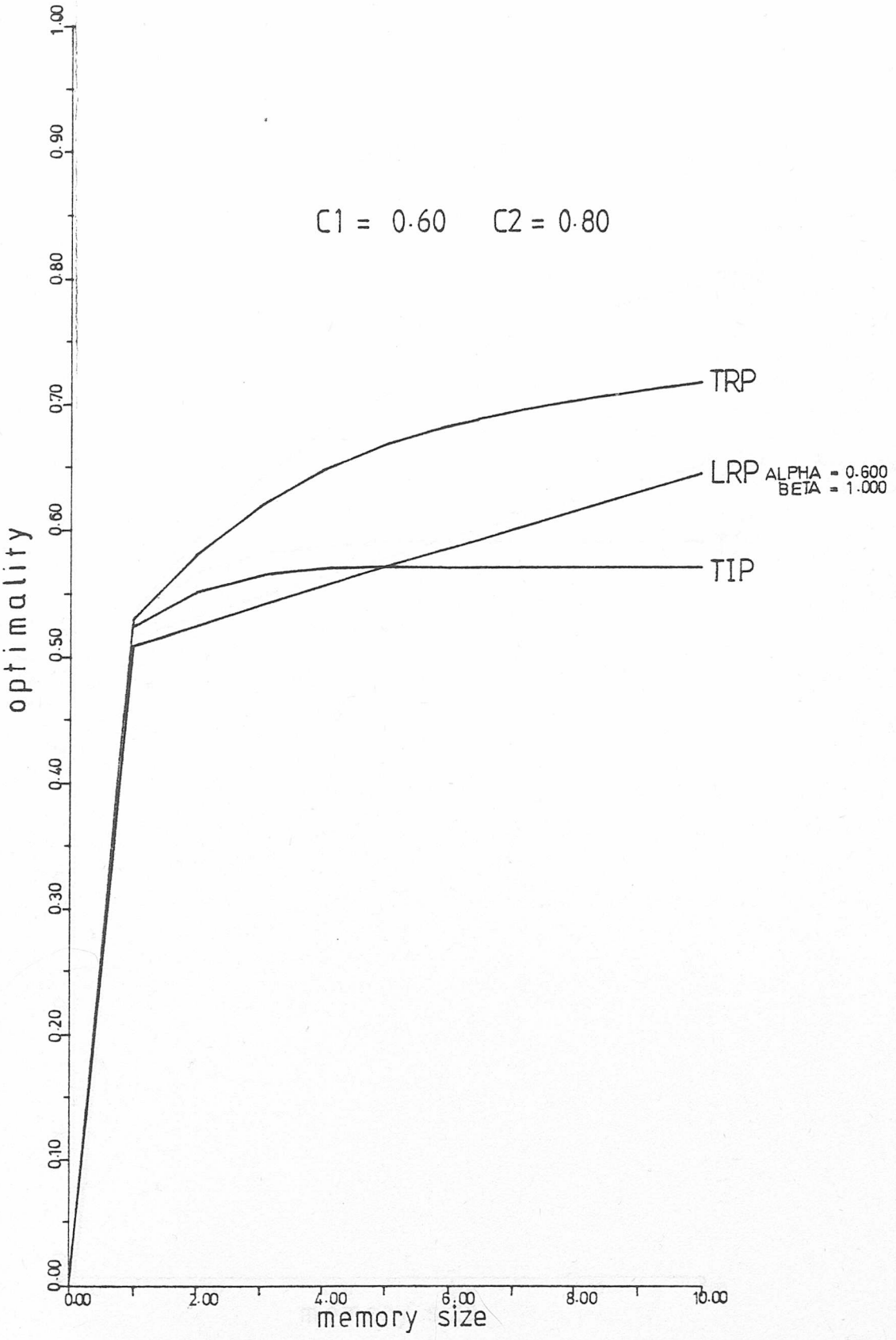

C1 = 0·60 C2 = 0·80
TRP
LRP ALPHA = 0·600
BETA = 1·000
TIP
optimality
memory size
0·00
0·10
0·20
0·30
0·40
0·50
0·60
0·70
0·80
0·90
1·00
0·00
2·00
4·00
6·00
8·00
10·00

Figure 5·6(c)

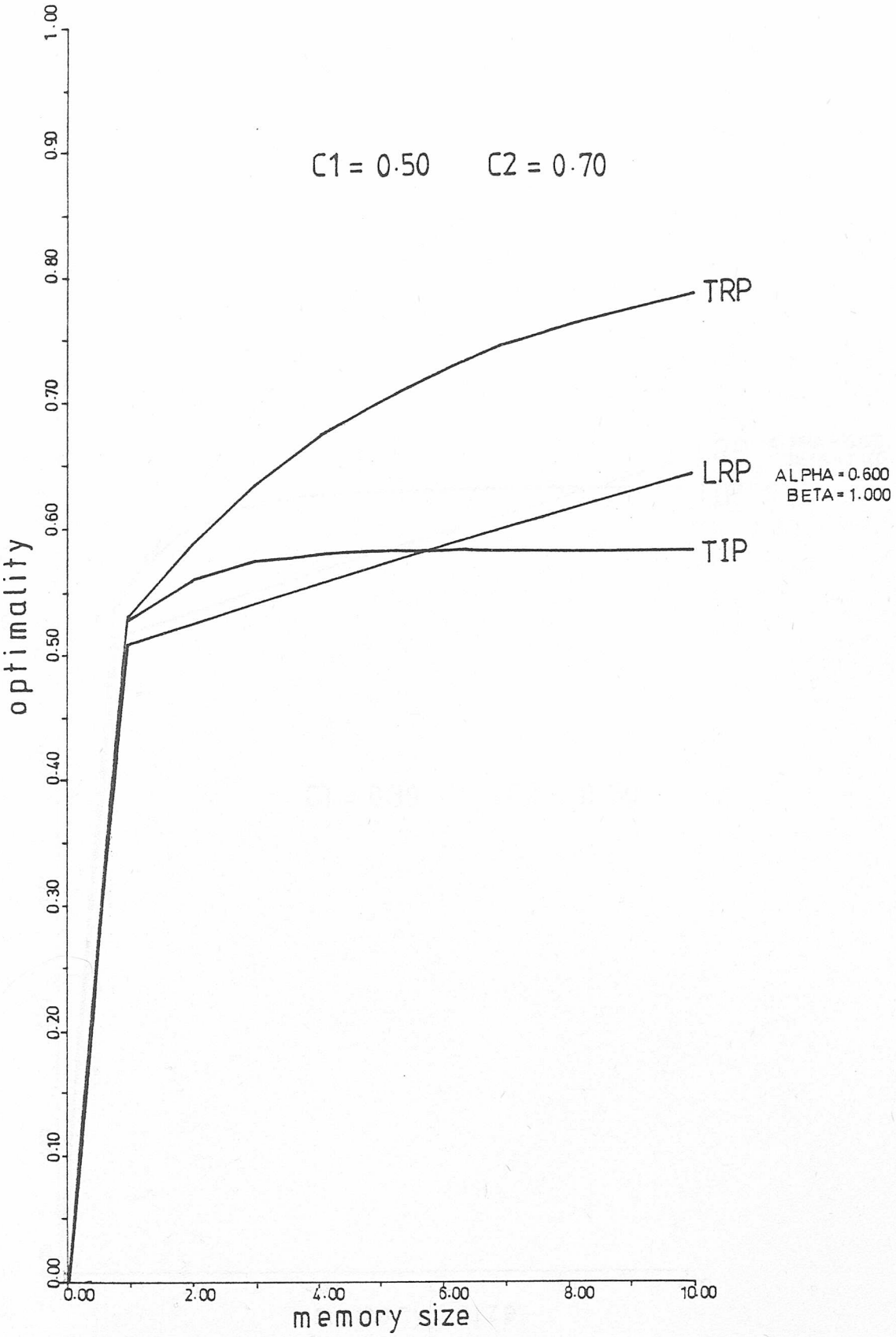

C1 = 0·50 C2 = 0·70
TRP
LRP
TIP
ALPHA = 0·600
BETA = 1·000
optimality
memory size
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5·6(d)

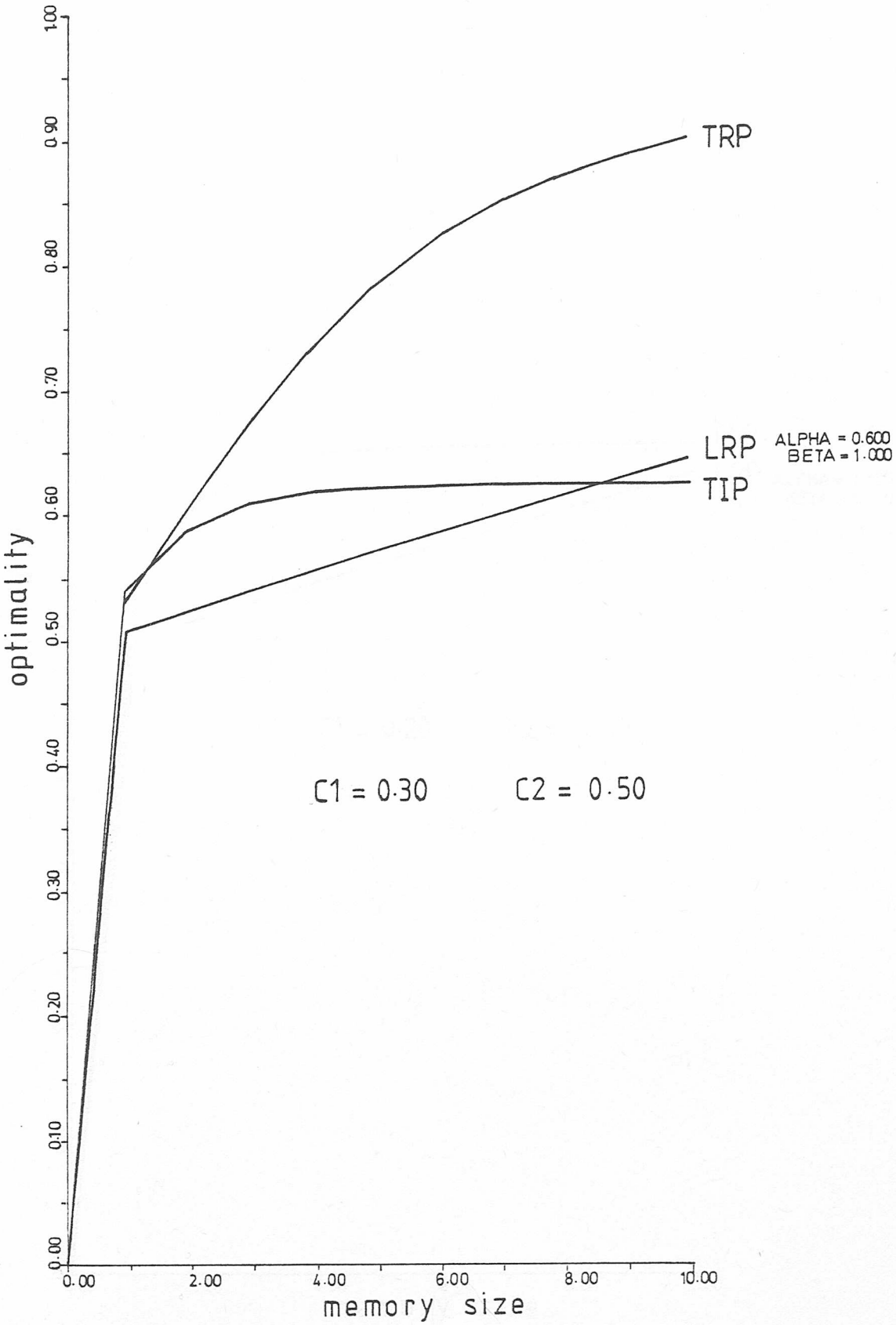

optimality
memory size
TRP
LRP
TIP
ALPHA = 0.600
BETA = 1.000
C1 = 0.30
C2 = 0.50
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5.6 (e)

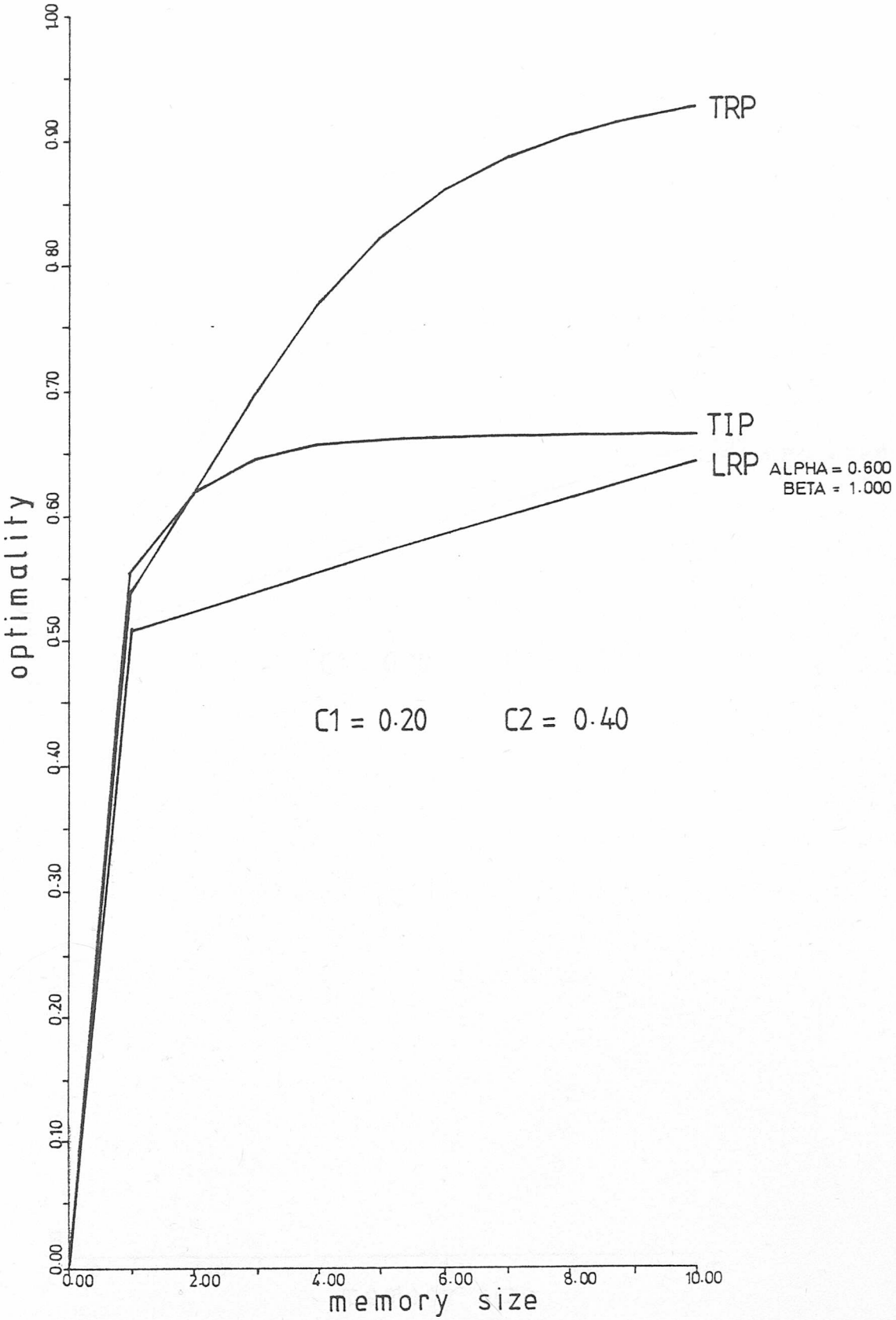

optimality
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size
TRP
TIP
LRP
ALPHA = 0.600
BETA = 1.000
C1 = 0.20
C2 = 0.40

Figure 5.6(f)

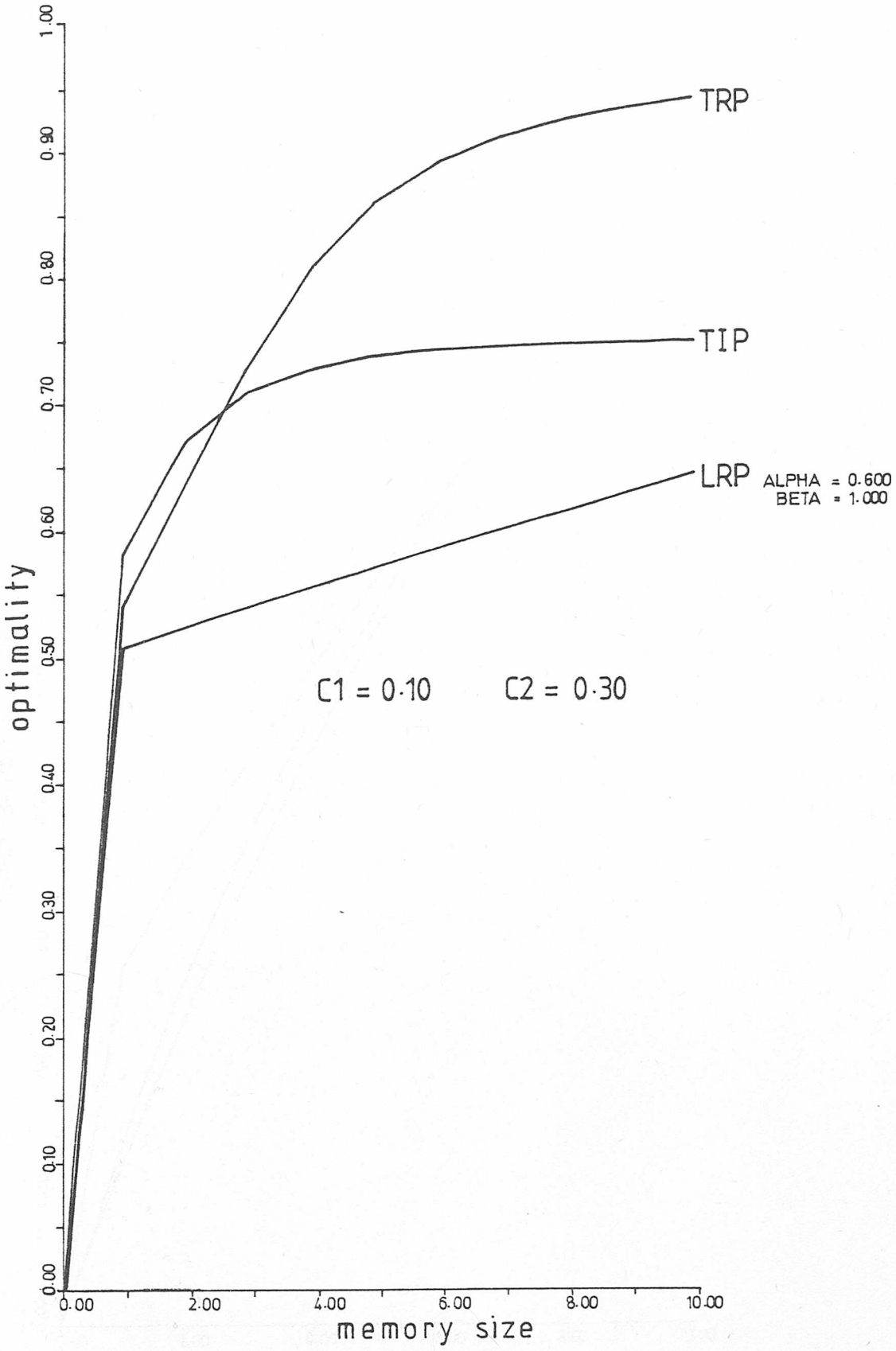

optimality
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size
TRP
TIP
LRP
ALPHA = 0.600
BETA = 1.000
C1 = 0.10
C2 = 0.30

Figure 5.6 (g)

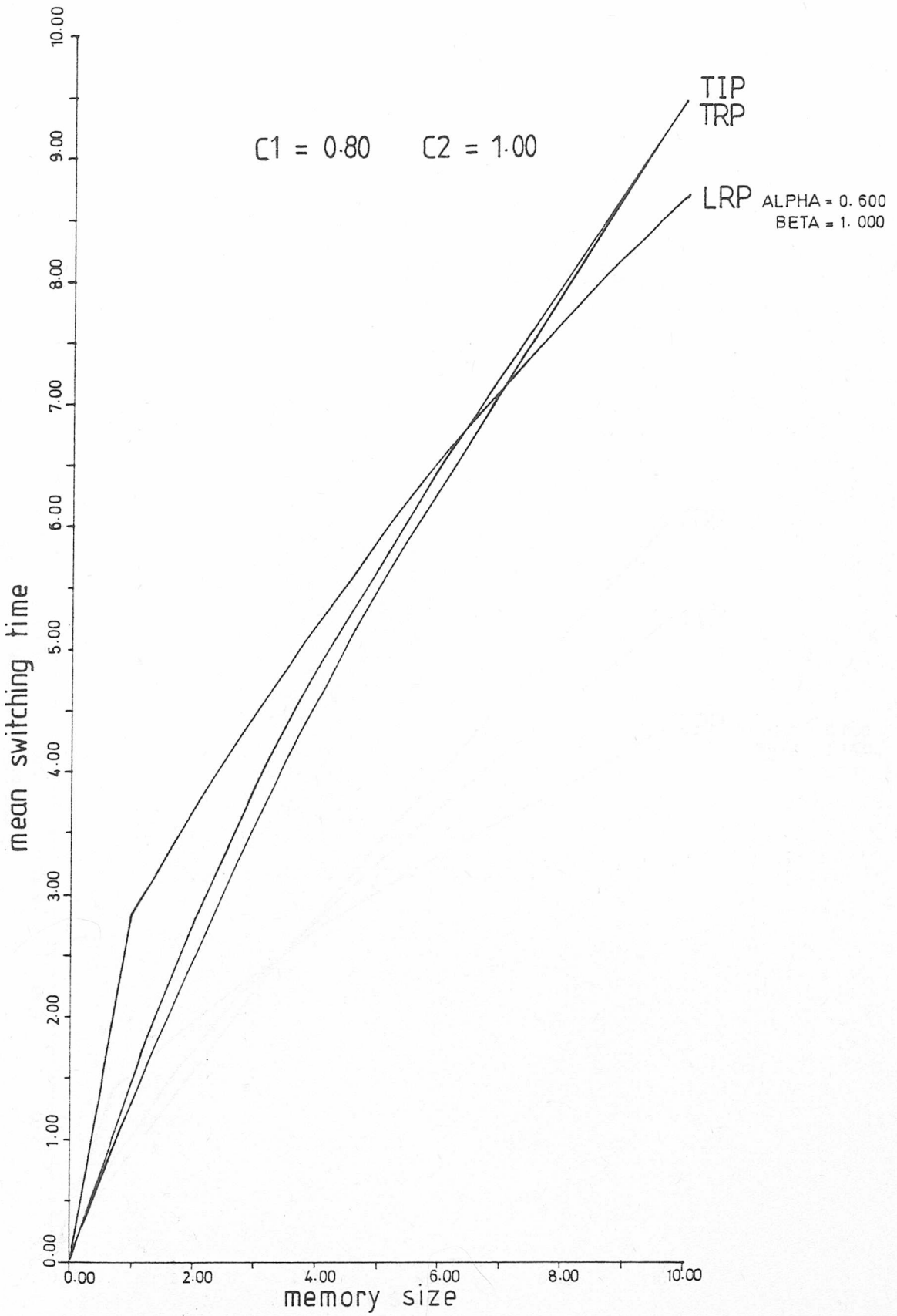


Figure 5·7(a) Theoretical mean switching times

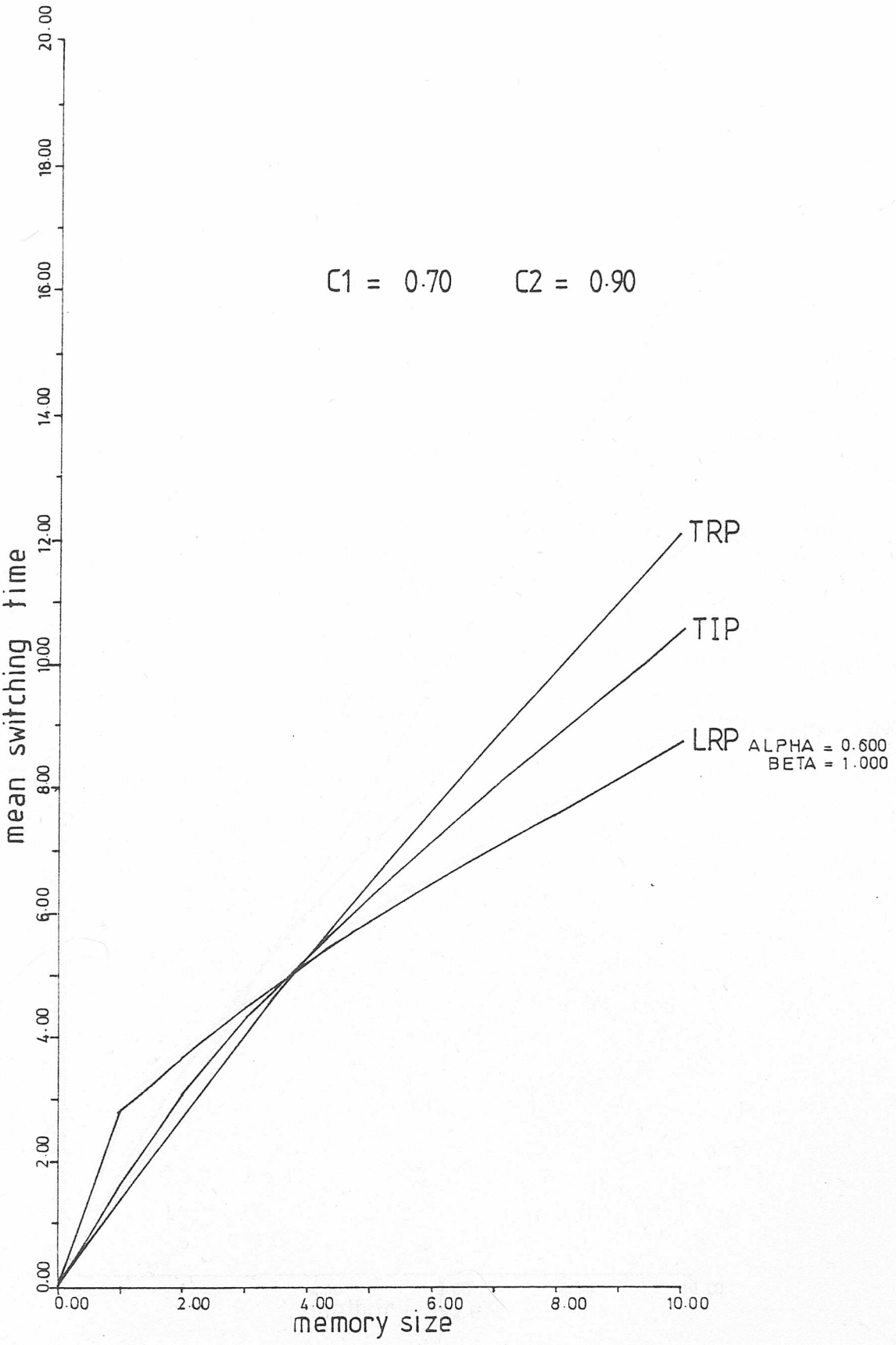

C1 = 0.70
C2 = 0.90
TRP
TIP
LRP
ALPHA = 0.600
BETA = 1.000
mean switching time
memory size
0.00
2.00
4.00
6.00
8.00
10.00
12.00
14.00
16.00
18.00
20.00

Figure 5.7(b)

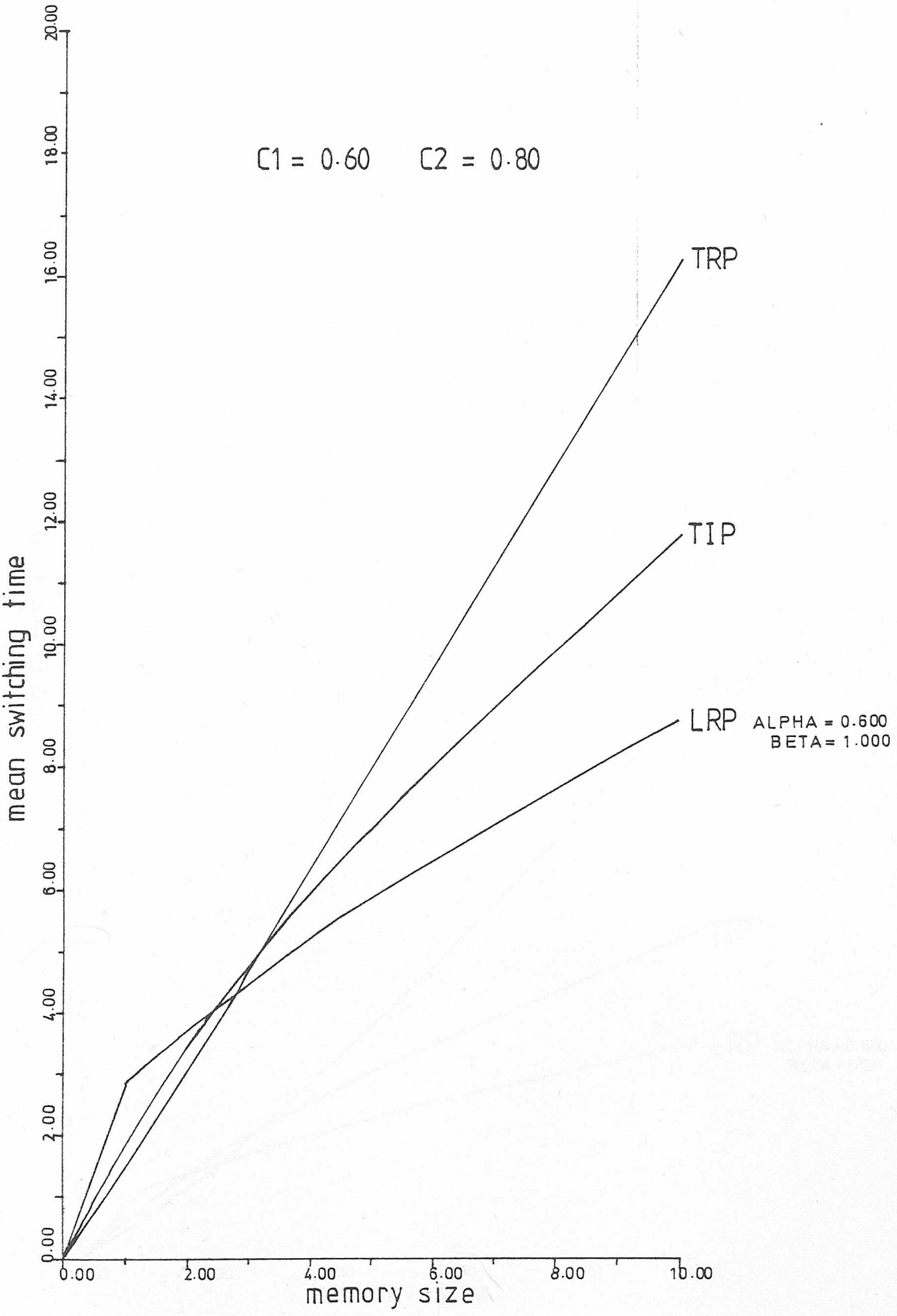

C1 = 0·60 C2 = 0·80
TRP
TIP
LRP
ALPHA = 0.600
BETA = 1.000
mean switching time
memory size
0.00
2.00
4.00
6.00
8.00
10.00
12.00
14.00
16.00
18.00
20.00

Figure 5·7(c)

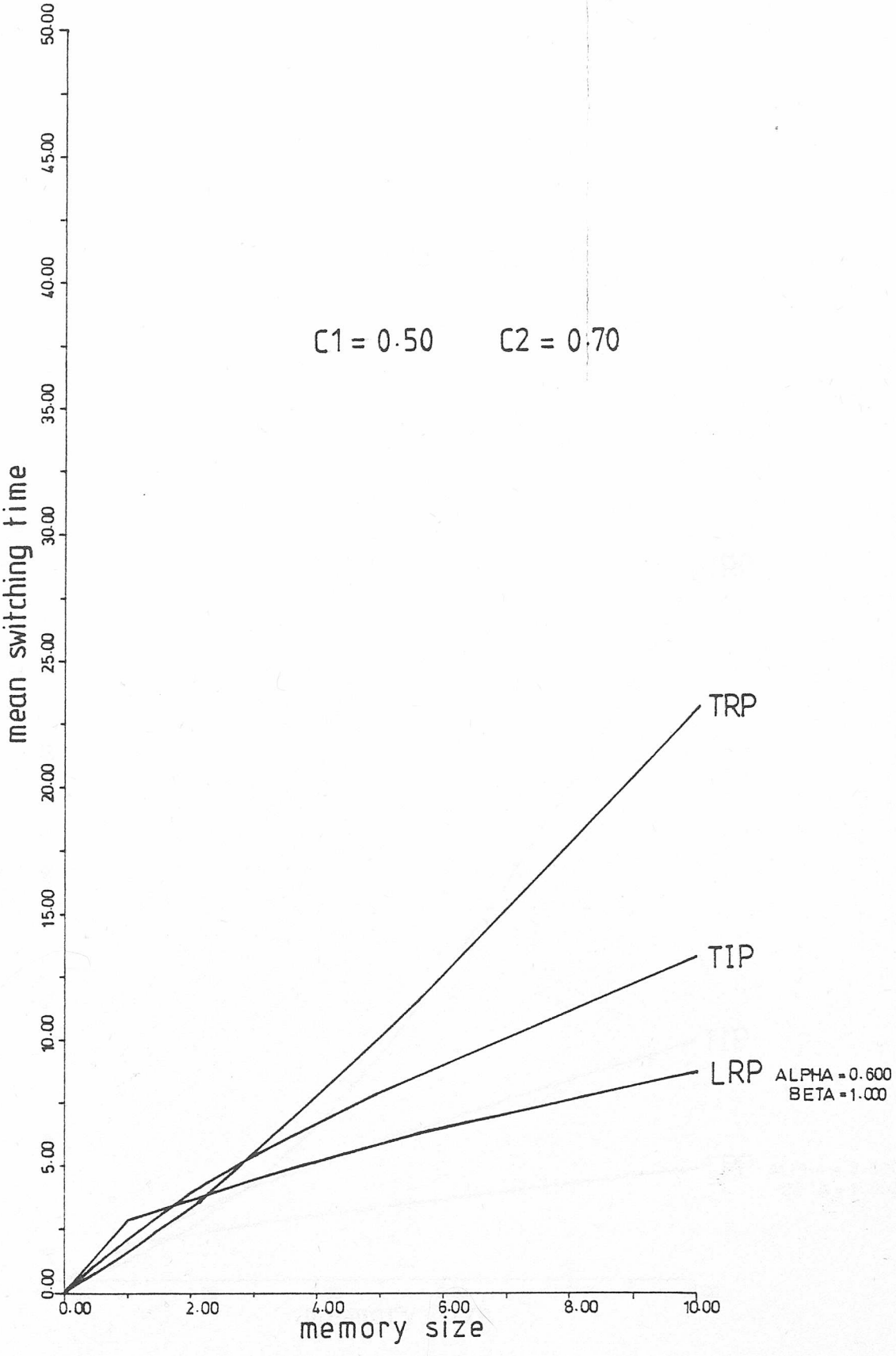

C1 = 0·50
C2 = 0·70
mean switching time
0.00
5.00
10.00
15.00
20.00
25.00
30.00
35.00
40.00
45.00
50.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size
TRP
TIP
LRP
ALPHA = 0.600
BETA = 1.000

Figure 5·7(d)

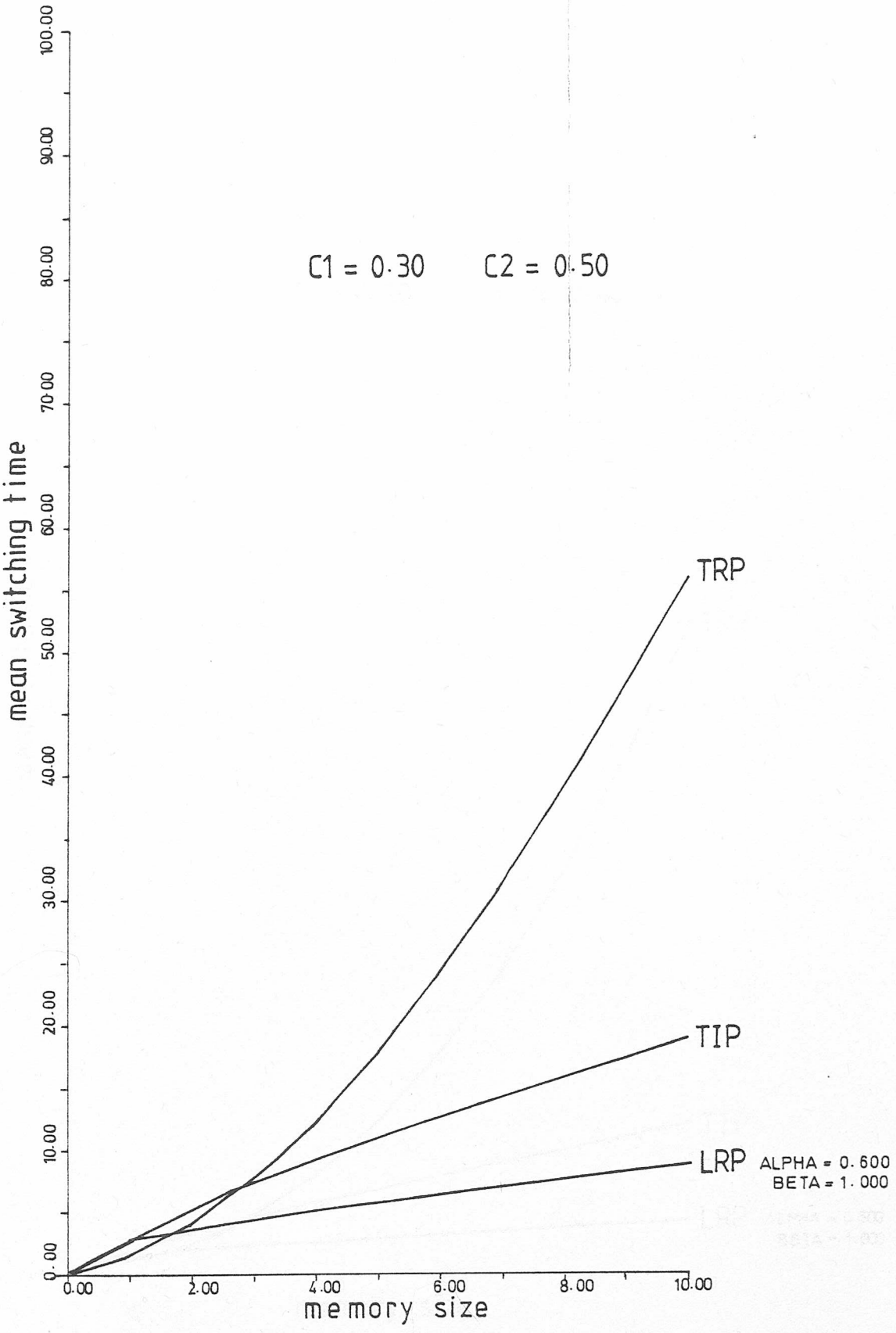

C1 = 0·30 C2 = 0·50
mean switching time
0.00
10.00
20.00
30.00
40.00
50.00
60.00
70.00
80.00
90.00
100.00
TRP
TIP
LRP
ALPHA = 0.600
BETA = 1.000
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 5·7 (e)

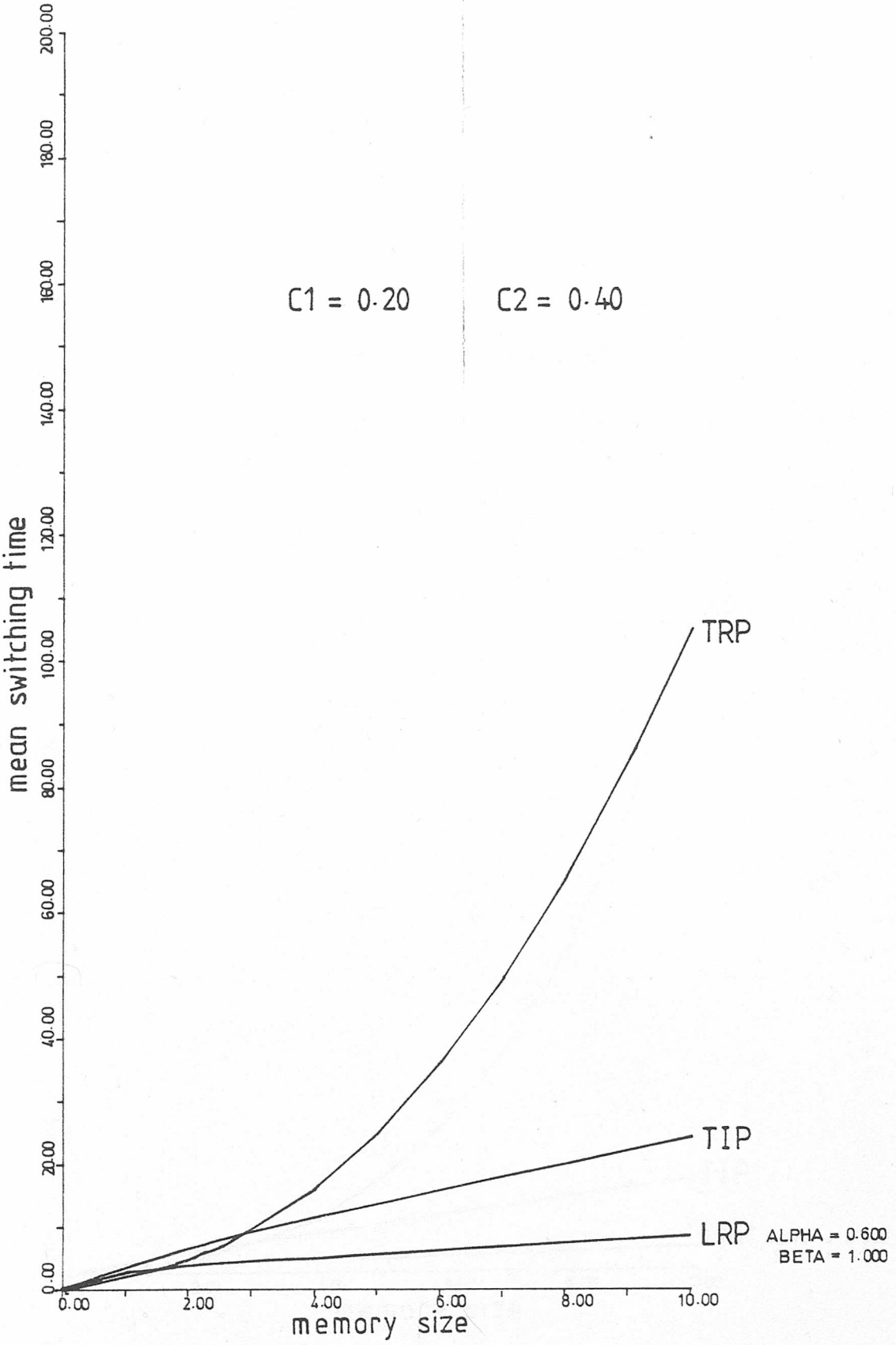

C1 = 0·20
C2 = 0·40
mean switching time
memory size
TRP
TIP
LRP
ALPHA = 0·600
BETA = 1·000
0.00
20.00
40.00
60.00
80.00
100.00
120.00
140.00
160.00
180.00
200.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5·7(f)

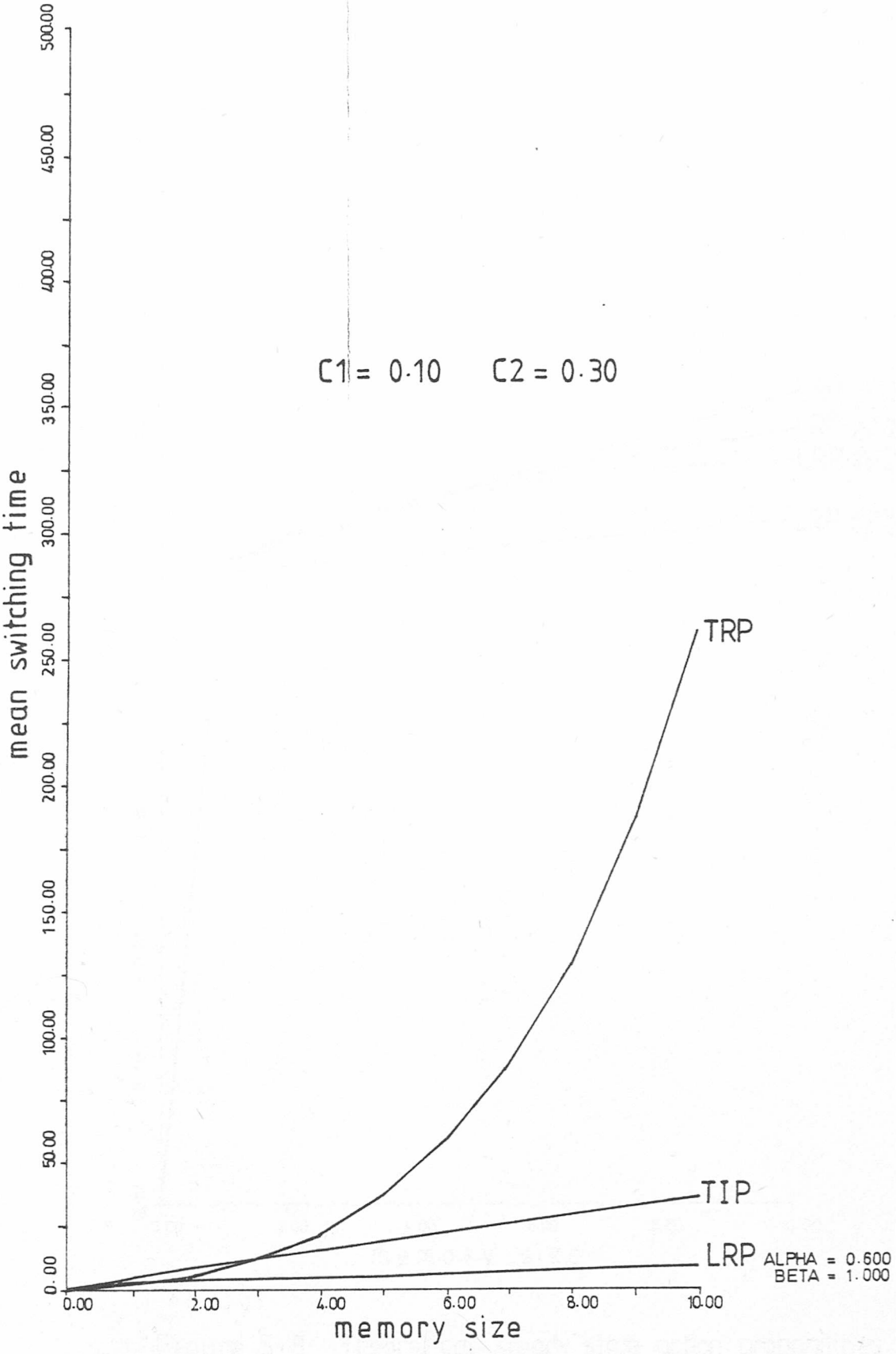

C1 = 0·10 C2 = 0·30
mean switching time
0.00
50.00
100.00
150.00
200.00
250.00
300.00
350.00
400.00
450.00
500.00
TRP
TIP
LRP
ALPHA = 0·600
BETA = 1·000
0.00
2.00
4.00
6.00
8.00
10.00
memory size

Figure 5·7(g)

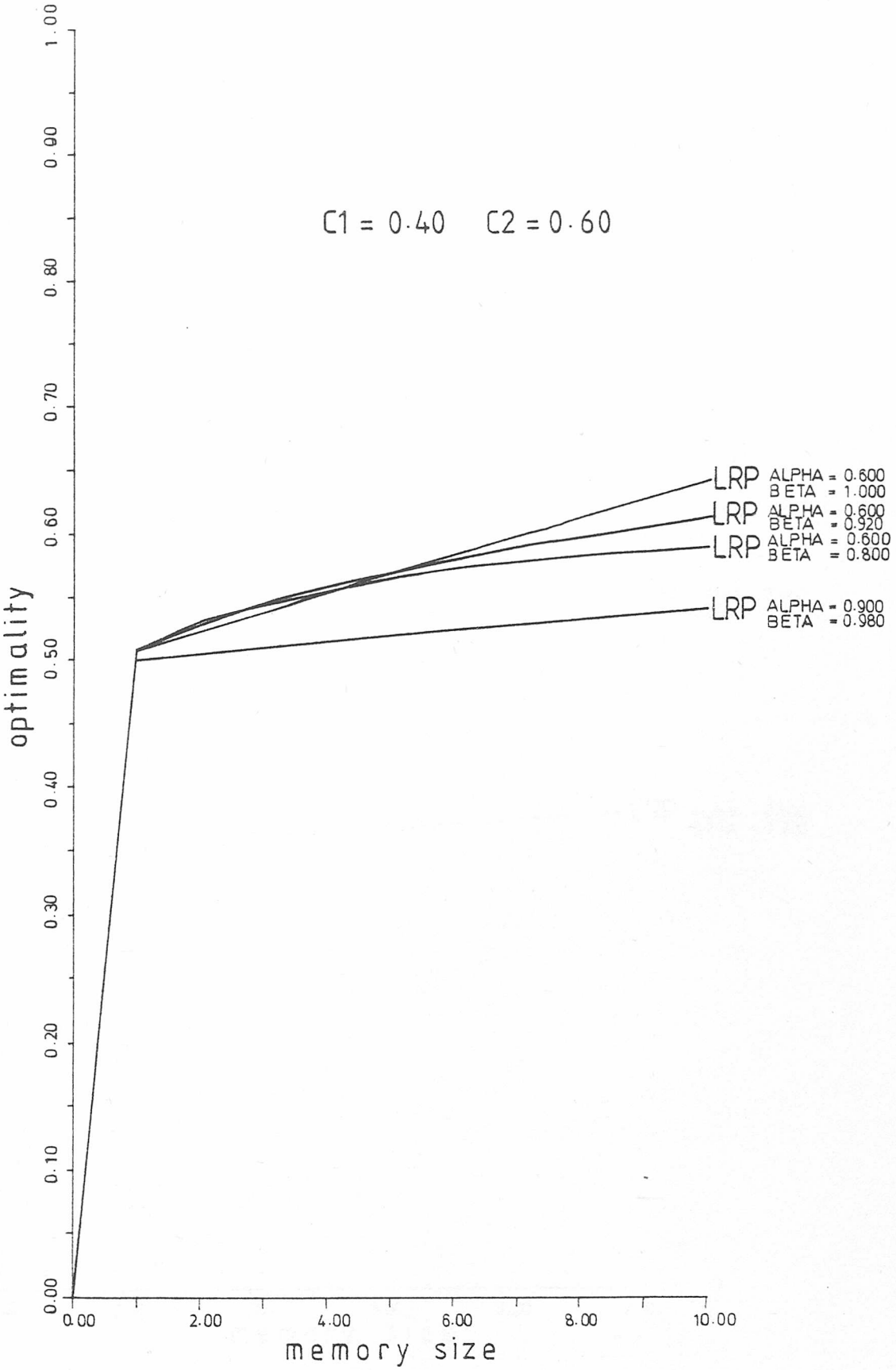


Figure 5.8 Theoretical steady state action probabilities

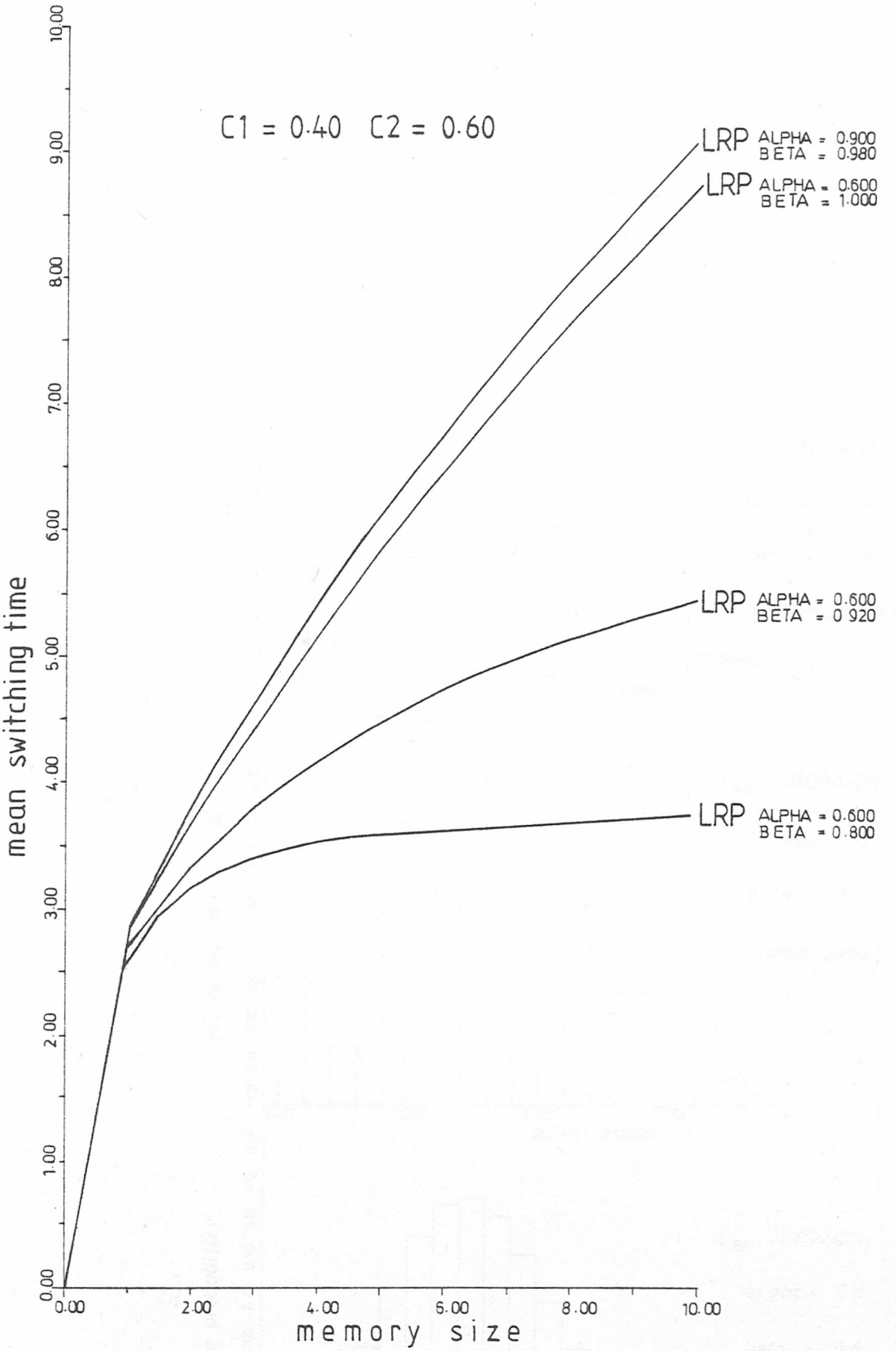


Figure 5.9 Theoretical mean switching time

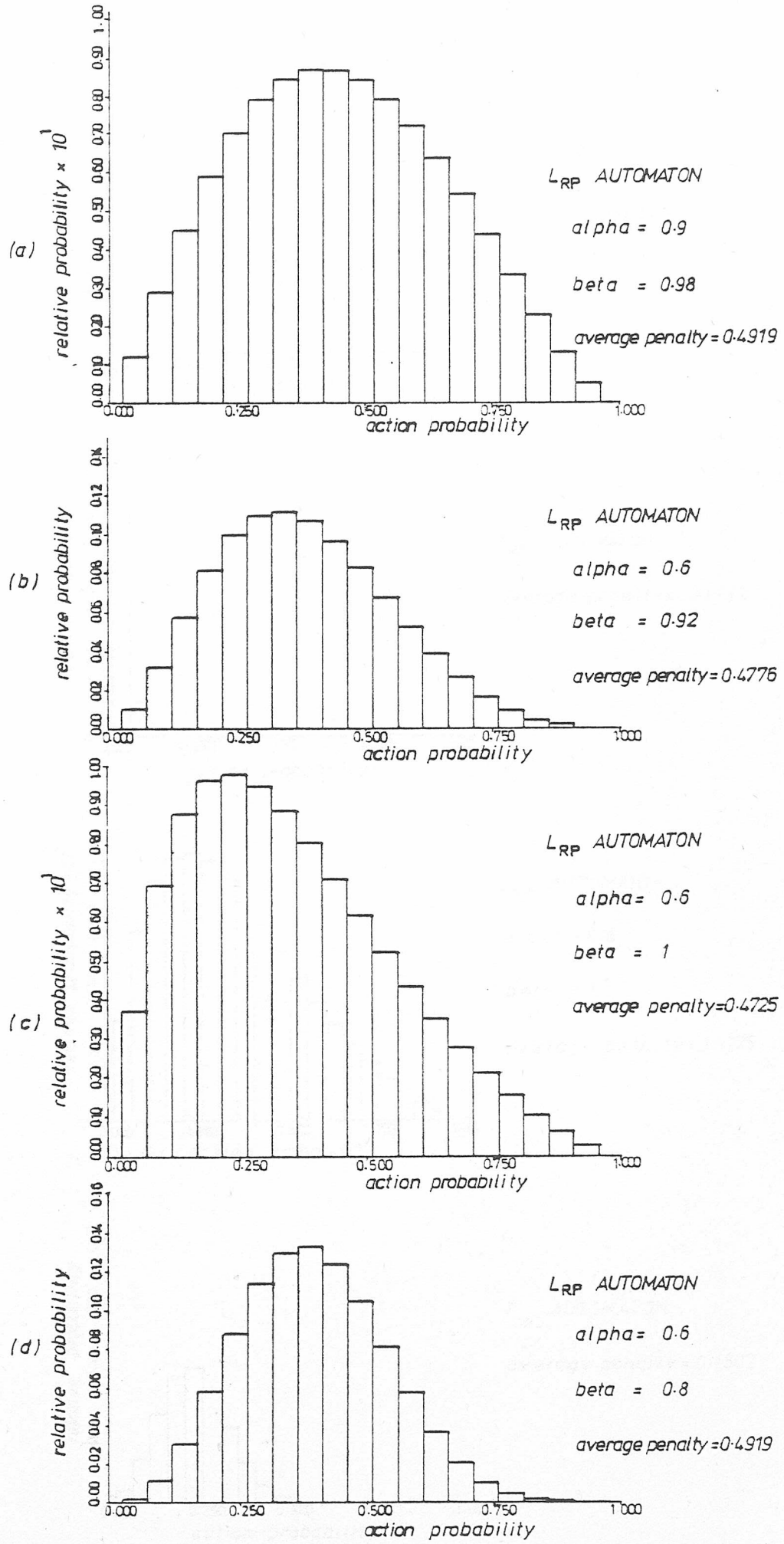
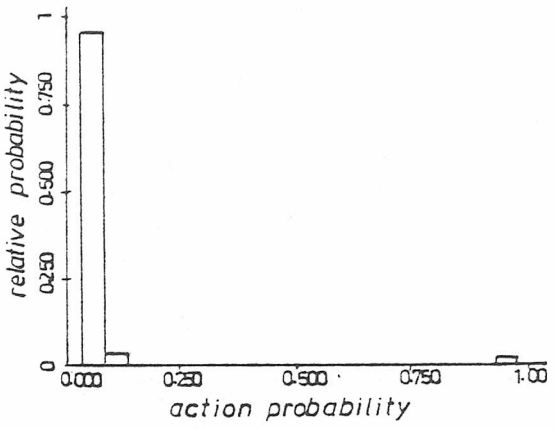
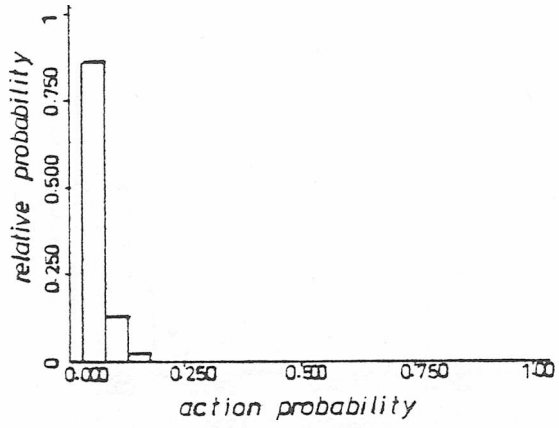


Figure 5·10 Distribution of States



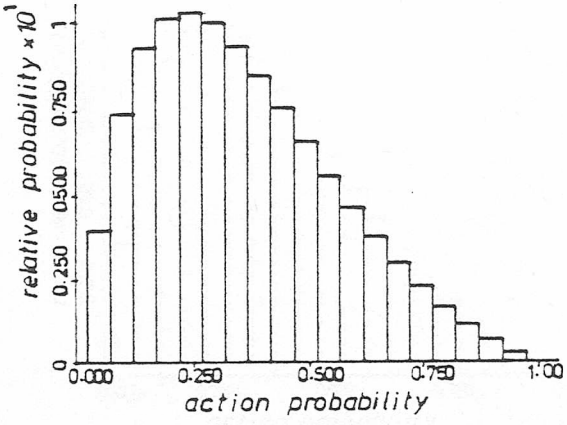
T_{R1} AUTOMATON

average penalty = 0·1125



T_{RP} AUTOMATON

average penalty = 0·1122

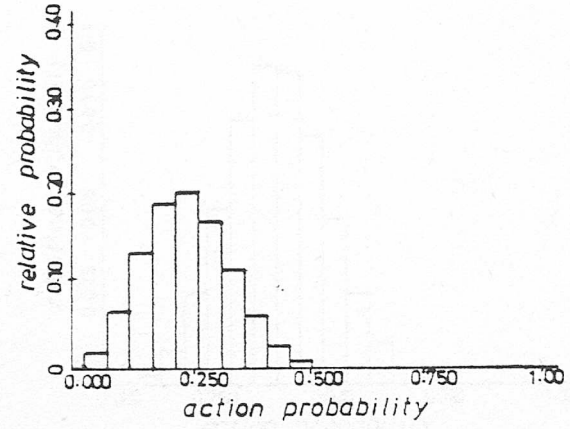


L_{RP} AUTOMATON

alpha = 0·6

beta = 1

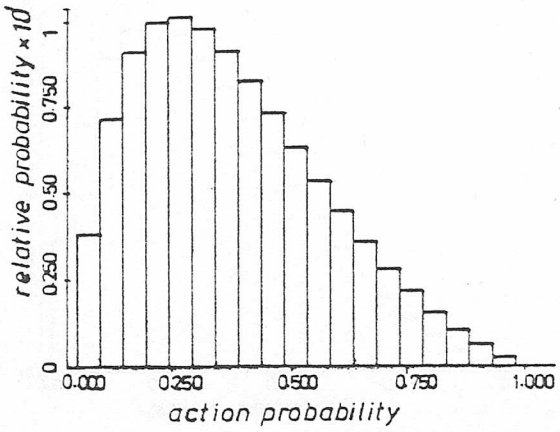
average penalty = 0·1275



T_{IP} AUTOMATON

average penalty = 0·1502

Figure 5·11(a) Distribution of States

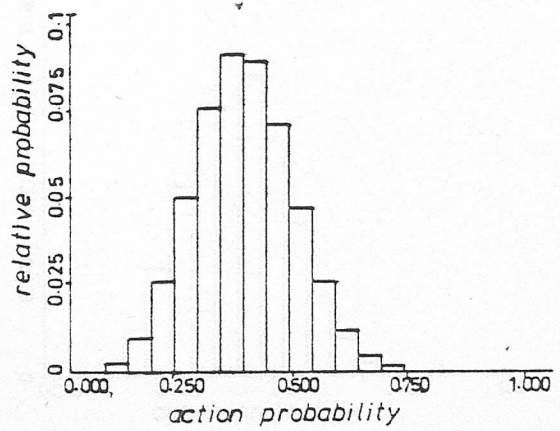


L_{RP} AUTOMATON

alpha = 0·6

beta = 1

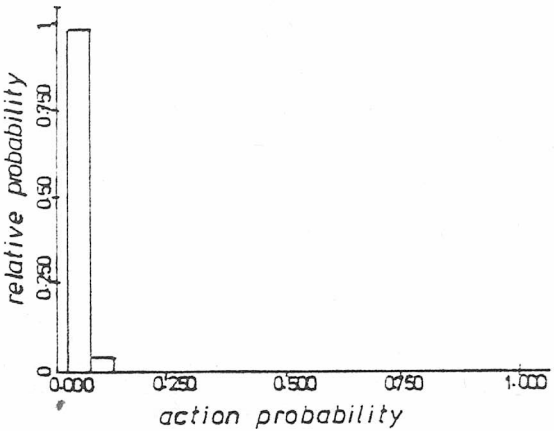
average penalty=03725



T_{IP} AUTOMATON

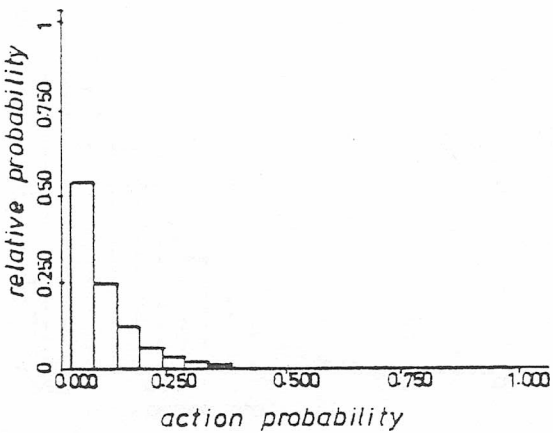
average penalty =03750

Figure 5·11 (b) Distribution of States



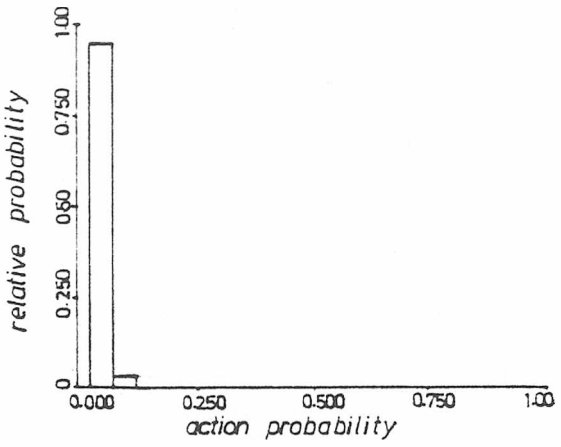
T_{R1} AUTOMATON

average penalty = 0.3109



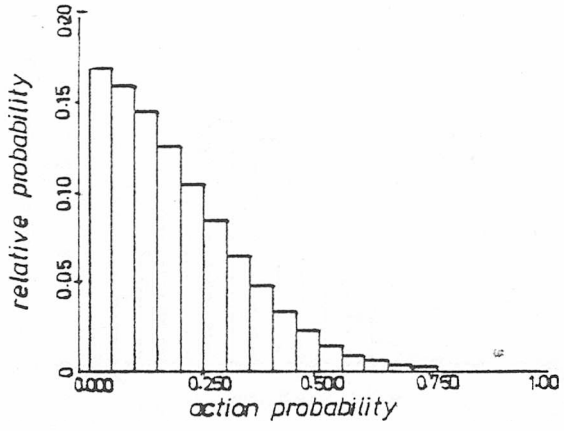
T_{RP} AUTOMATON

average penalty = 0.3205



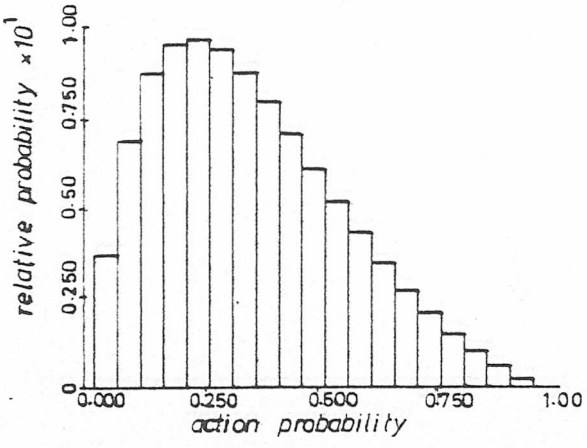
T_{R1} AUTOMATON

average penalty = 0.5104



T_{RP} AUTOMATON

average penalty = 0.5432

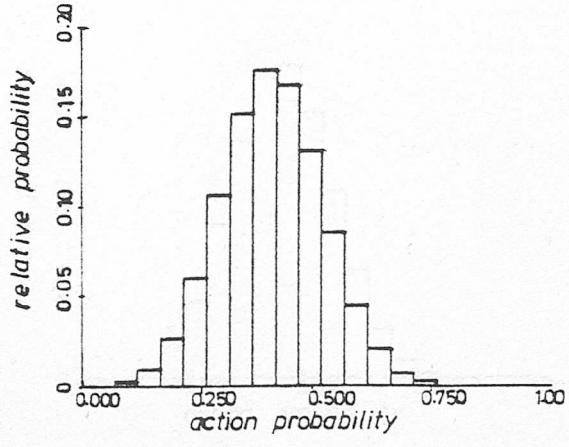


L_{RP} AUTOMATON

alpha = 0.6

beta = 1

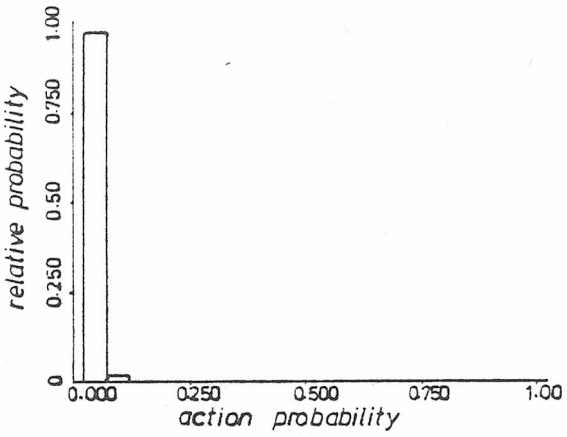
average penalty = 0.5725



T_{IP} AUTOMATON

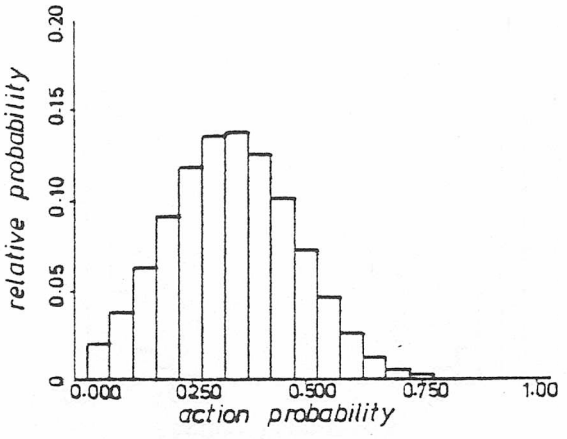
average penalty = 0.5833

Figure 5.11(c) Distribution of States



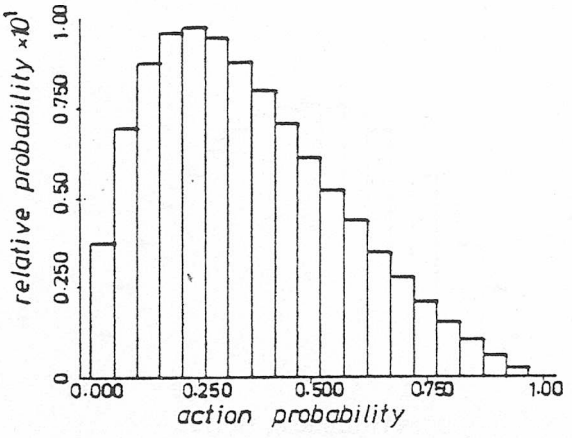
T_{RI} AUTOMATON

average penalty = 0.7102



T_{RP} AUTOMATON

average penalty = 0.7678

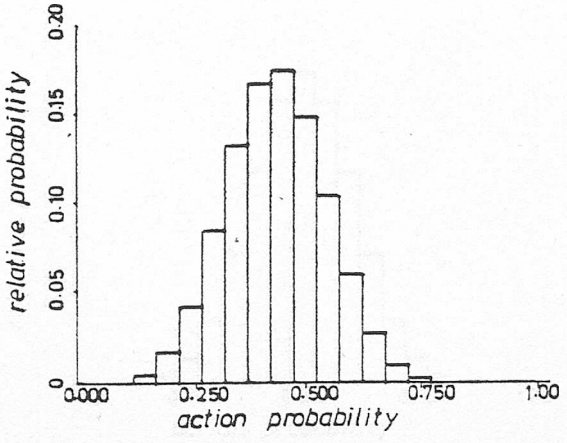


L_{RP} AUTOMATON

alpha = 0.6

beta = 1

average penalty = 0.7725



T_{IP} AUTOMATON

average penalty = 0.7875

Figure 5.11(d) Distribution of States

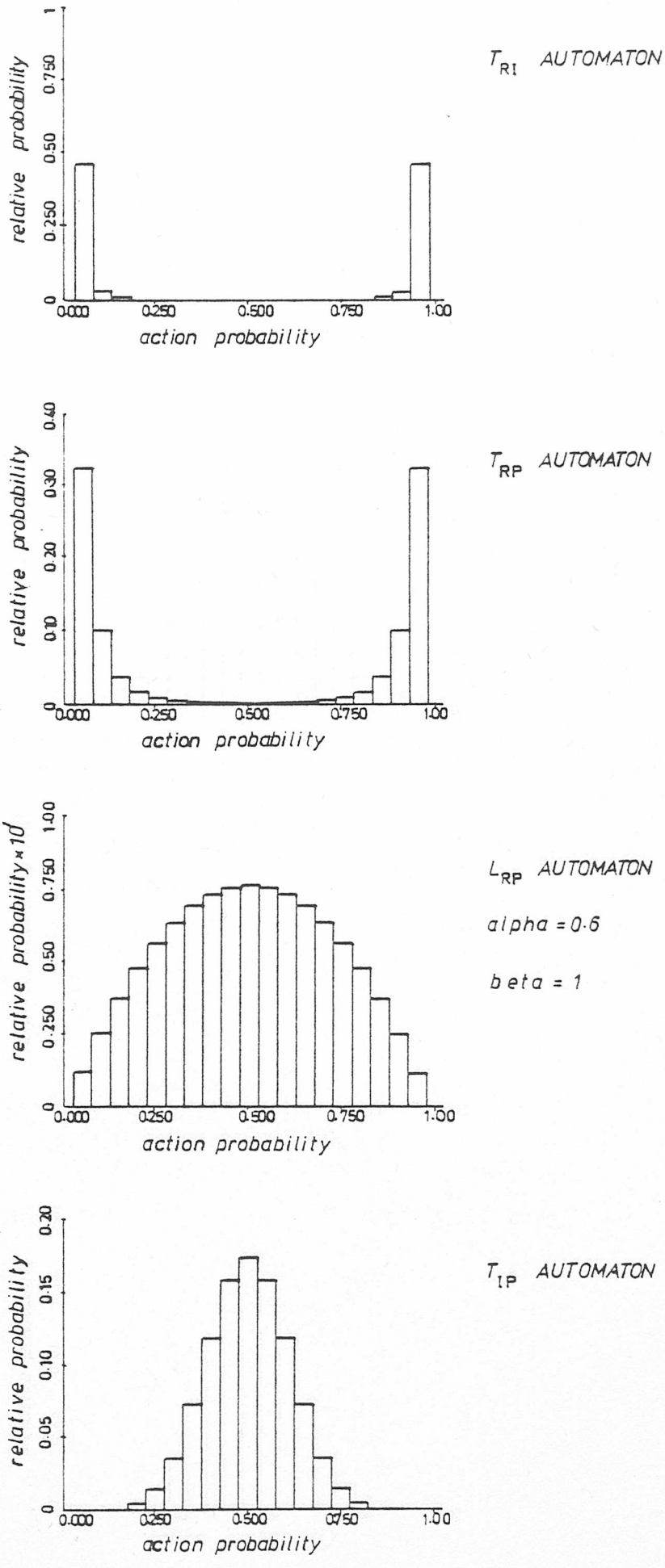
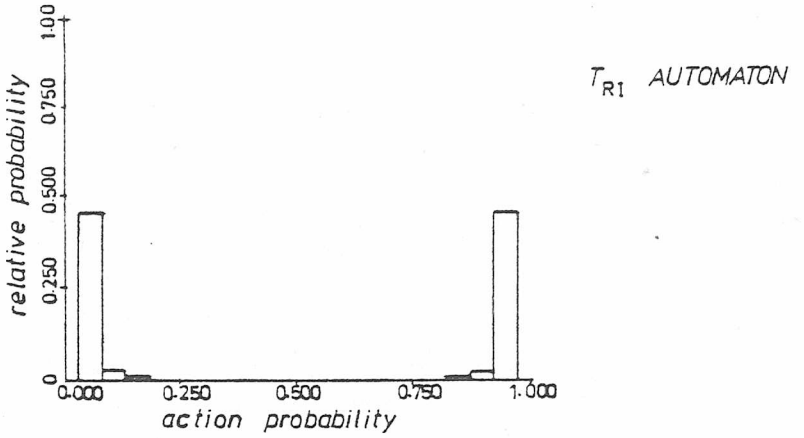
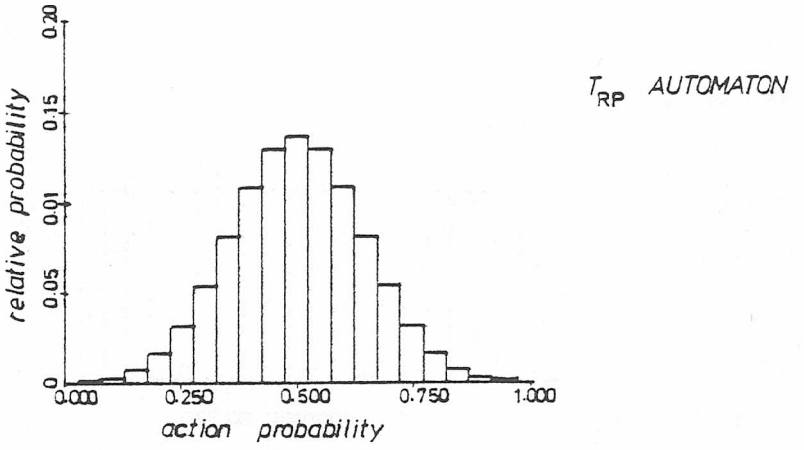
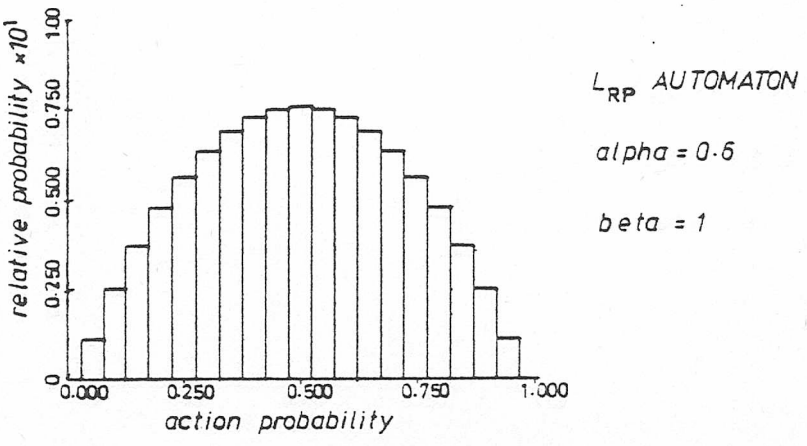


Figure 5·12 (a) Distribution of States







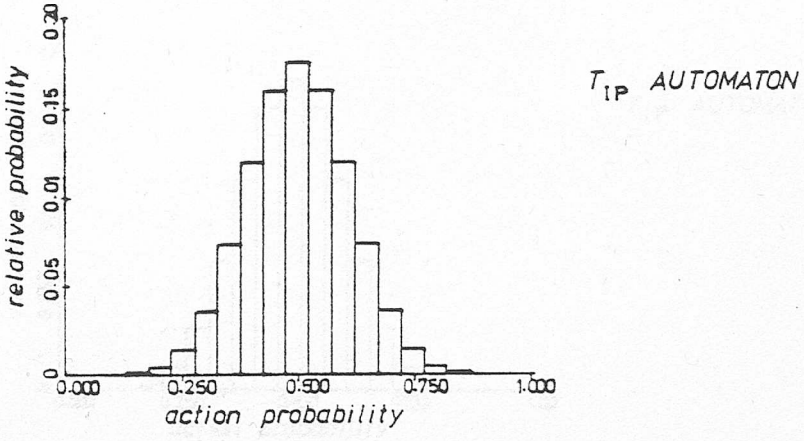
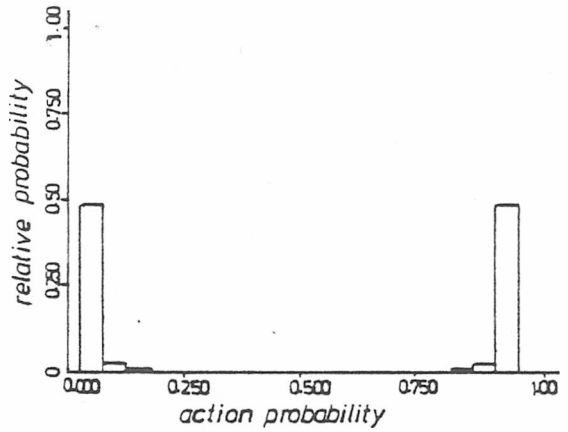
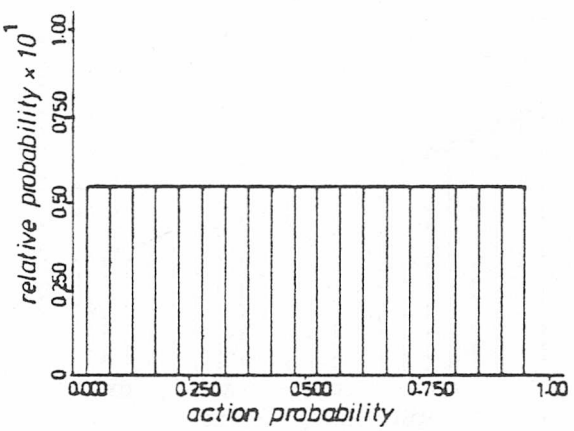


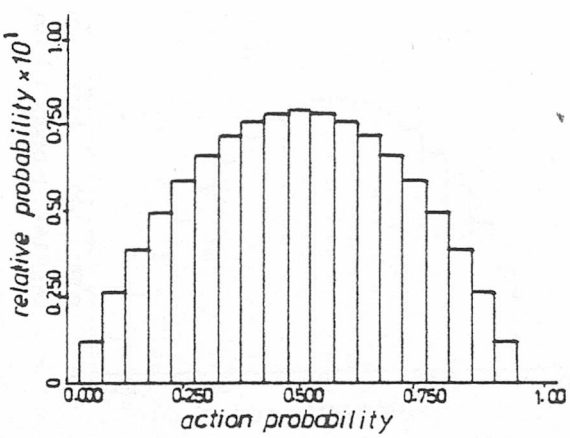
Figure 5·12(b) Distribution of States



T_{RI} AUTOMATON



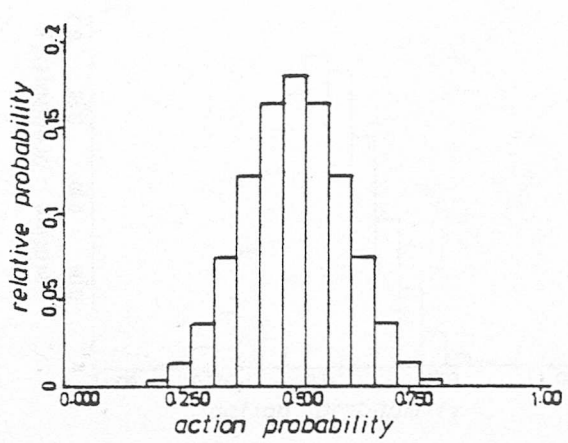
T_{RP} AUTOMATON



L_{RP} AUTOMATON

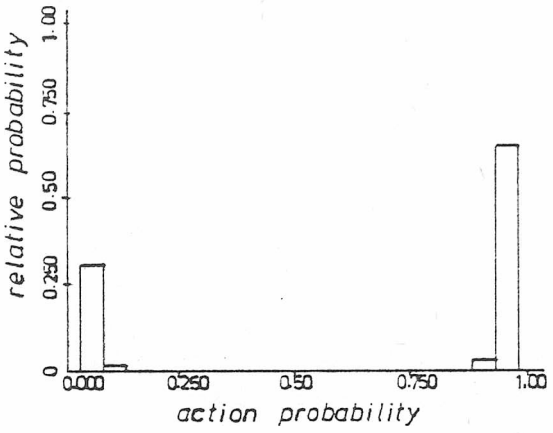
alpha = 0.6

beta = 1



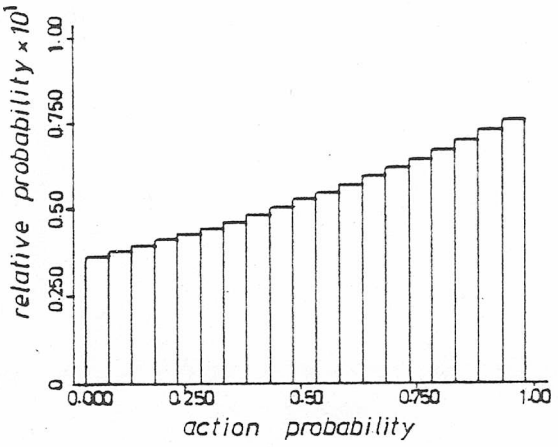
T_{IP} AUTOMATON

Figure 5·12 (c) Distribution of States



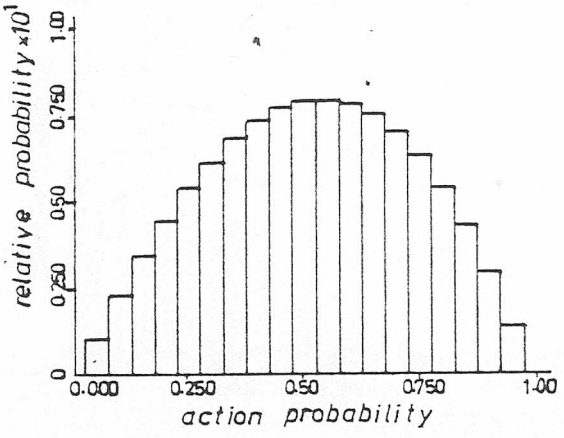
T_{RI} AUTOMATON

average penalty = 0.4969



T_{RP} AUTOMATON

average penalty = 0.4988

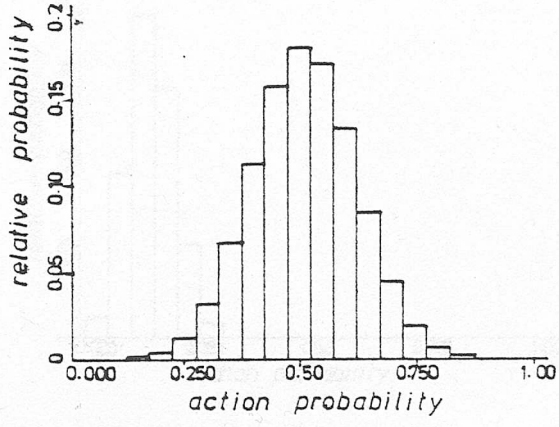


L_{RP} AUTOMATON

alpha = 0.6

beta = 1

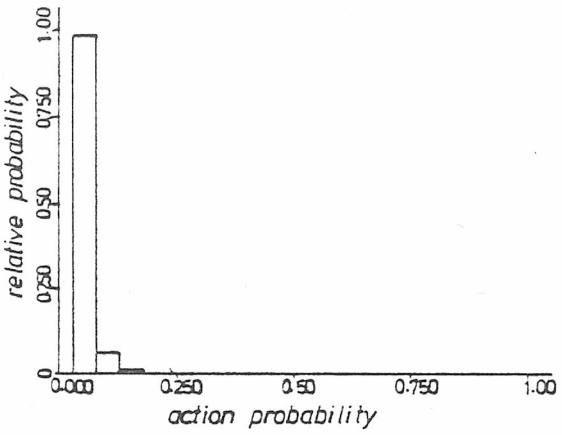
average penalty = 0.4997



T_{IP} AUTOMATON

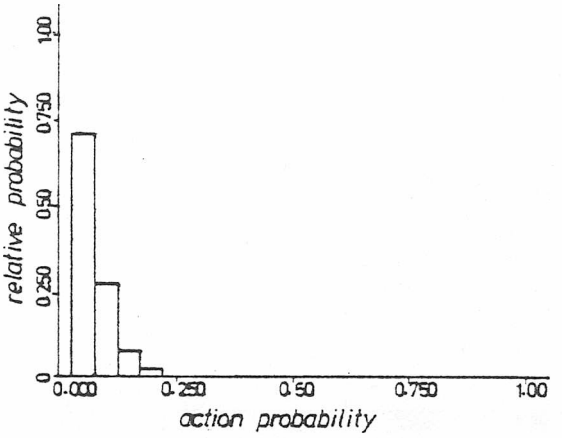
average penalty = 0.4998

Figure 5.12 (d) Distribution of States



T_{R1} AUTOMATON

average penalty = 0·2666



T_{RP} AUTOMATON

average penalty = 0·2385

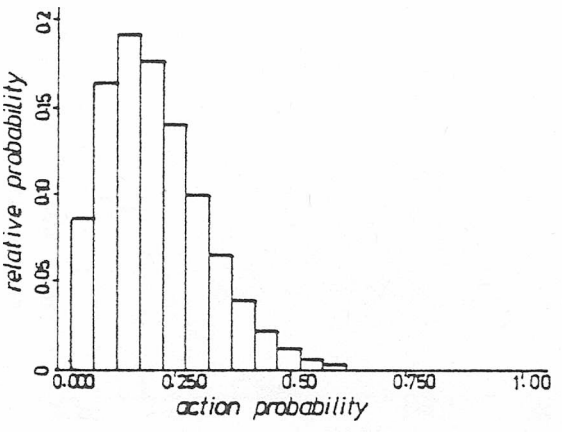
Distribution of States

$\theta_1 = 0 \cdot 001$

$\phi_1 = 0 \cdot 05$

$\theta_2 = 0 \cdot 1$

$\phi_2 = 0 \cdot 02$

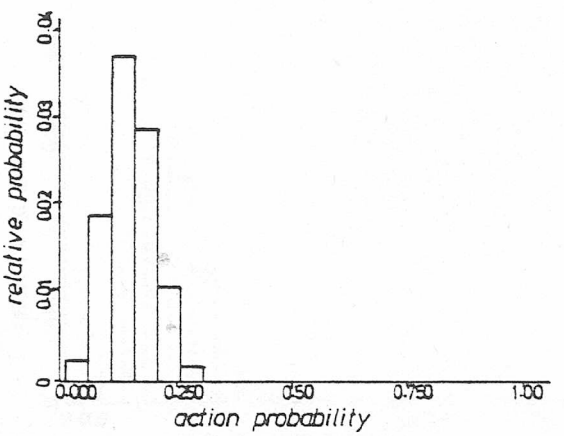


L_{RP} AUTOMATON

alpha = 0·6

beta = 1

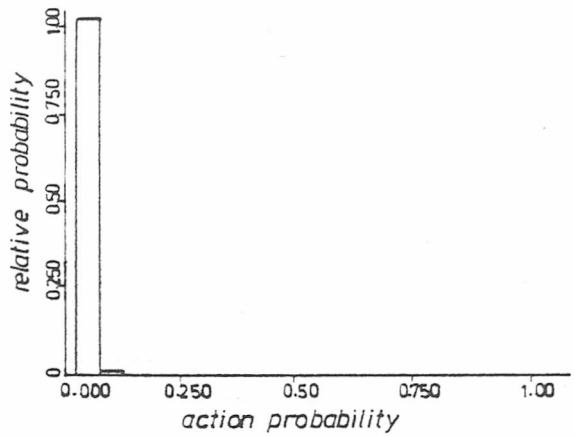
average penalty = 0·2136



T_{IP} AUTOMATON

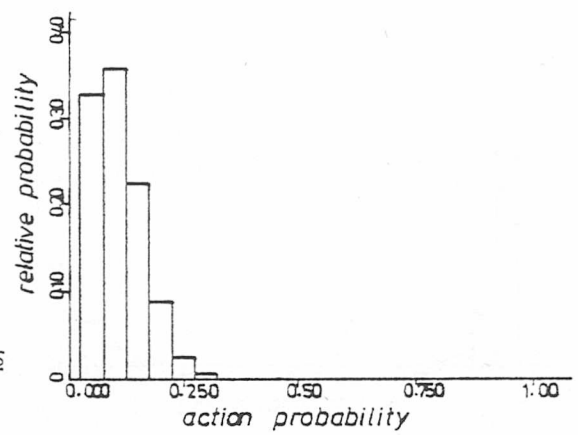
average penalty = 0·1734

Figure 5·13 (a)



T_{R1} AUTOMATON

average penalty = 0·6685



T_{RP} AUTOMATON

average penalty = 0·5433

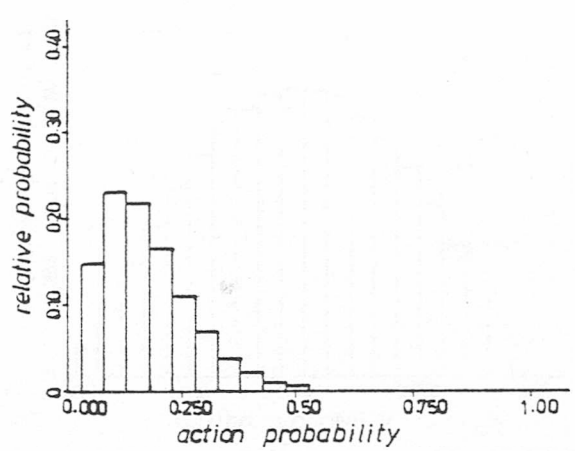
Distribution of States

θ_1 = 0·005

ϕ_1 = 0·05

θ_2 = 0·3

ϕ_2 = 0·001

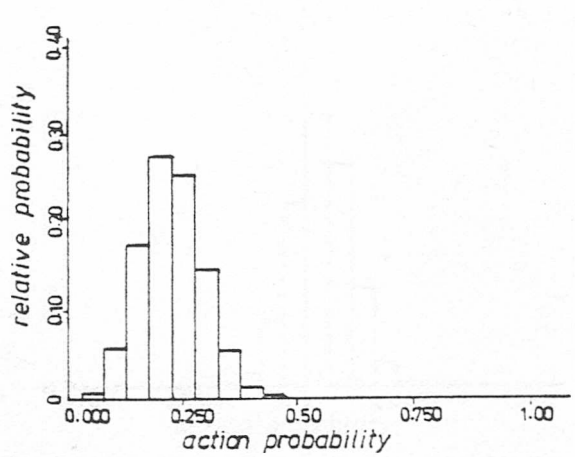


L_{RP} AUTOMATON

alpha = 0·6

beta = 1

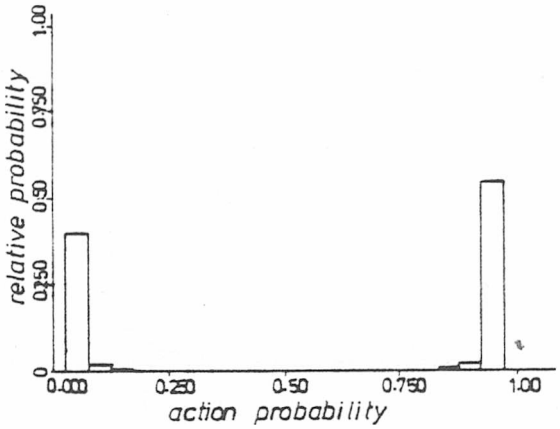
average penalty = 0·4930



T_{IP} AUTOMATON

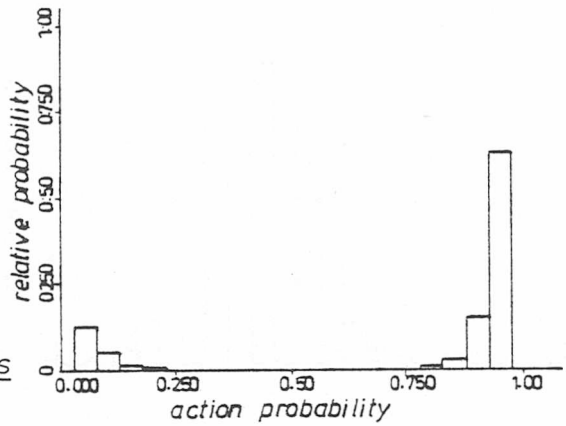
average penalty = 0·4409

Figure 5·13(b)



T_{R1} AUTOMATON

average penalty = 0·1913



T_{RP} AUTOMATON

average penalty = 0·1465

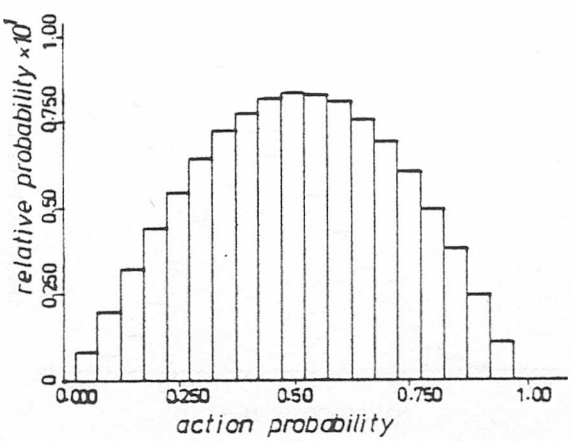
Distribution of States

$\theta_1 = 0 \cdot 01$

$\phi_1 = 0 \cdot 5$

$\theta_2 = 0 \cdot 001$

$\phi = 0 \cdot 1$

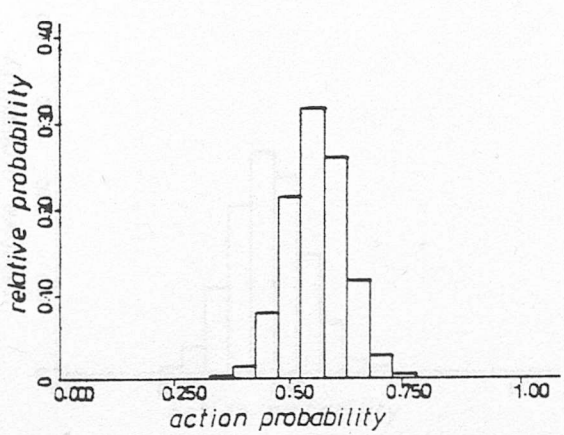


L_{RP} AUTOMATON

alpha = 0·6

beta = 1

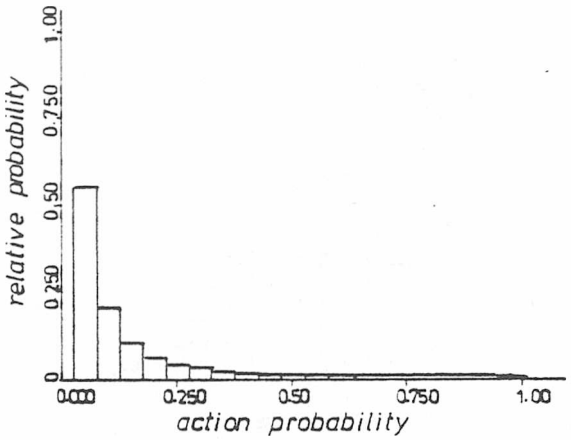
average penalty = 0·0329



T_{IP} AUTOMATON

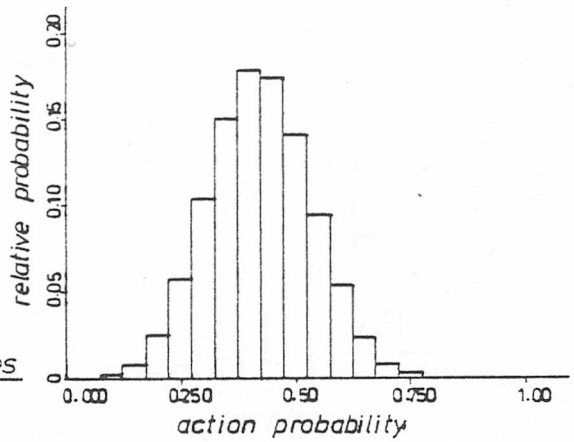
average penalty = 0·0148

Figure 5·13 (c)



T_{RI} AUTOMATON

average penalty = 0.7943



T_{RP} AUTOMATON

average penalty = 0.5049

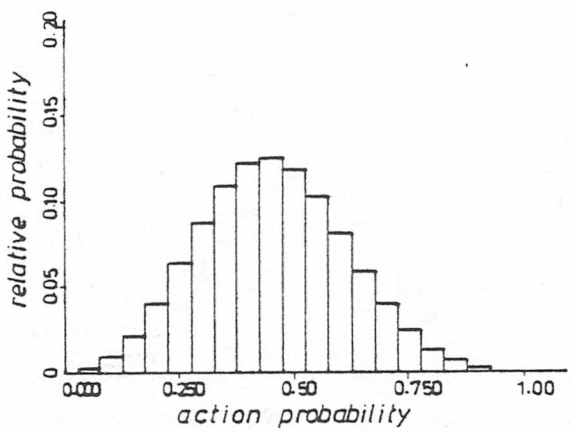
Distribution of States

$\theta_1 = 0.01$

$\phi_1 = 0.015$

$\theta_2 = 0.025$

$\phi_2 = 0.02$

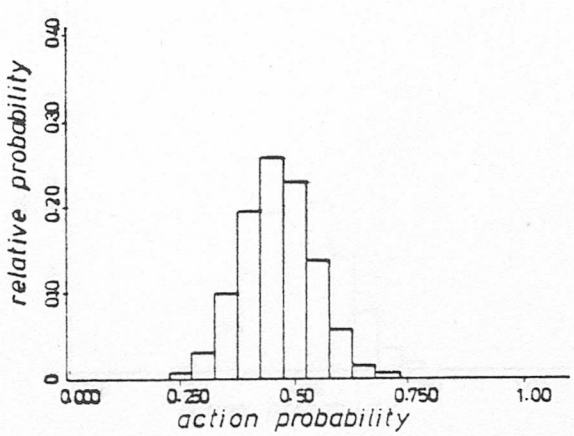


L_{RP} AUTOMATON

alpha = 0.6

beta = 1

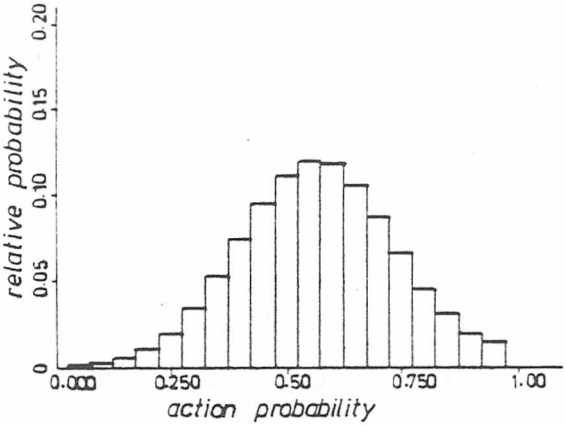
average penalty = 0.5281



T_{IP} AUTOMATON

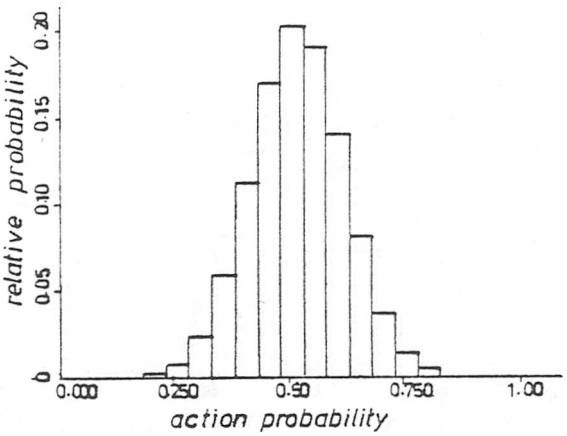
average penalty = 0.4875

Figure 5.13(d)



T_{RI} AUTOMATON

average penalty = 0.7889



T_{RP} AUTOMATON

average penalty = 0.7712

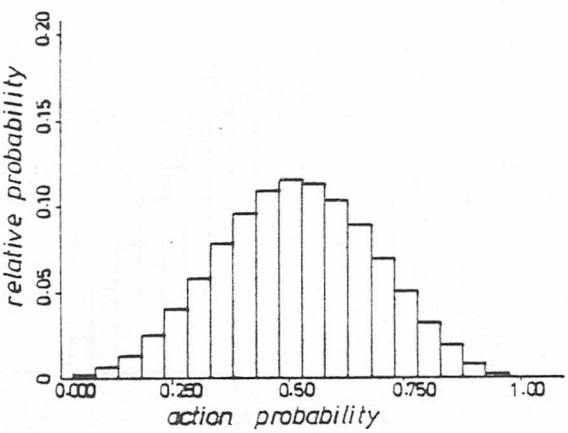
Distribution of States

$\theta_1 = 0.07$

$\phi = 0.02$

$\theta_2 = 0.07$

$\phi = 0.03$

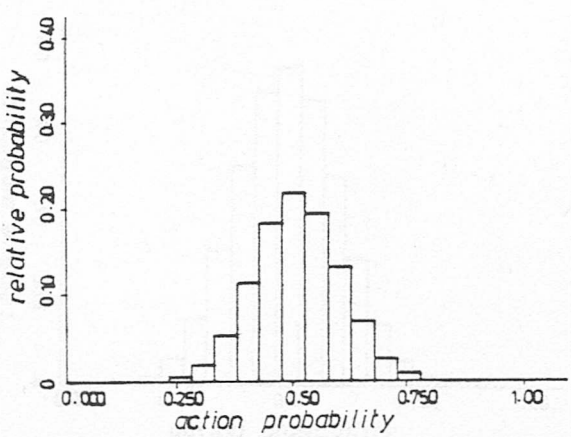


L_{RP} AUTOMATON

alpha = 0.6

beta = 1

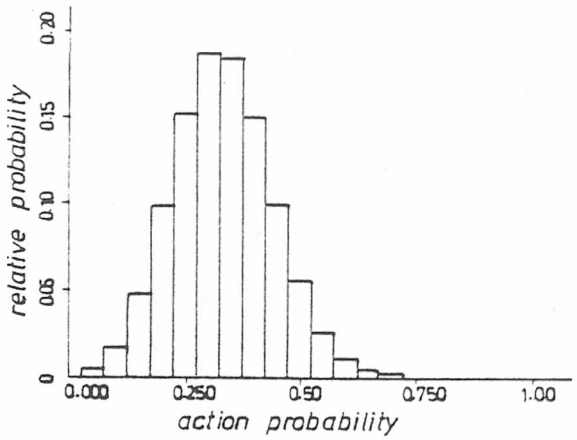
average penalty = 0.7853



T_{IP} AUTOMATON

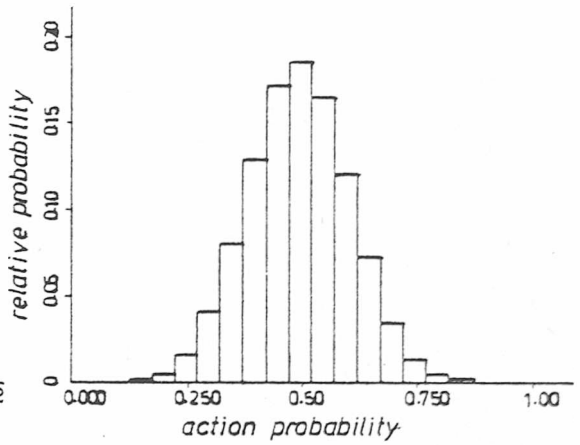
average penalty = 0.7700

Figure 5.13(e)



T_{R1} AUTOMATON

average penalty = 0·9870



T_{RP} AUTOMATON

average penalty = 0·9853

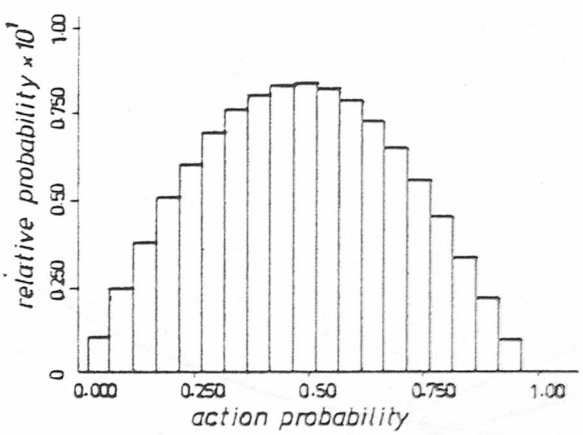
Distribution of States

$\theta_1 = 0 \cdot 5$

$\phi_1 = 0 \cdot 01$

$\theta_2 = 0 \cdot 1$

$\phi_2 = 0 \cdot 001$

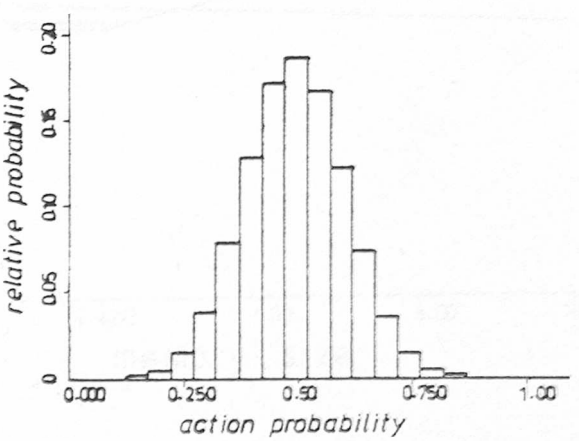


L_{RP} AUTOMATON

alpha = 0·6

beta = 1

average penalty = 0·9856



T_{IP} AUTOMATON

average penalty = 0·9853

Figure 5·13 (f)

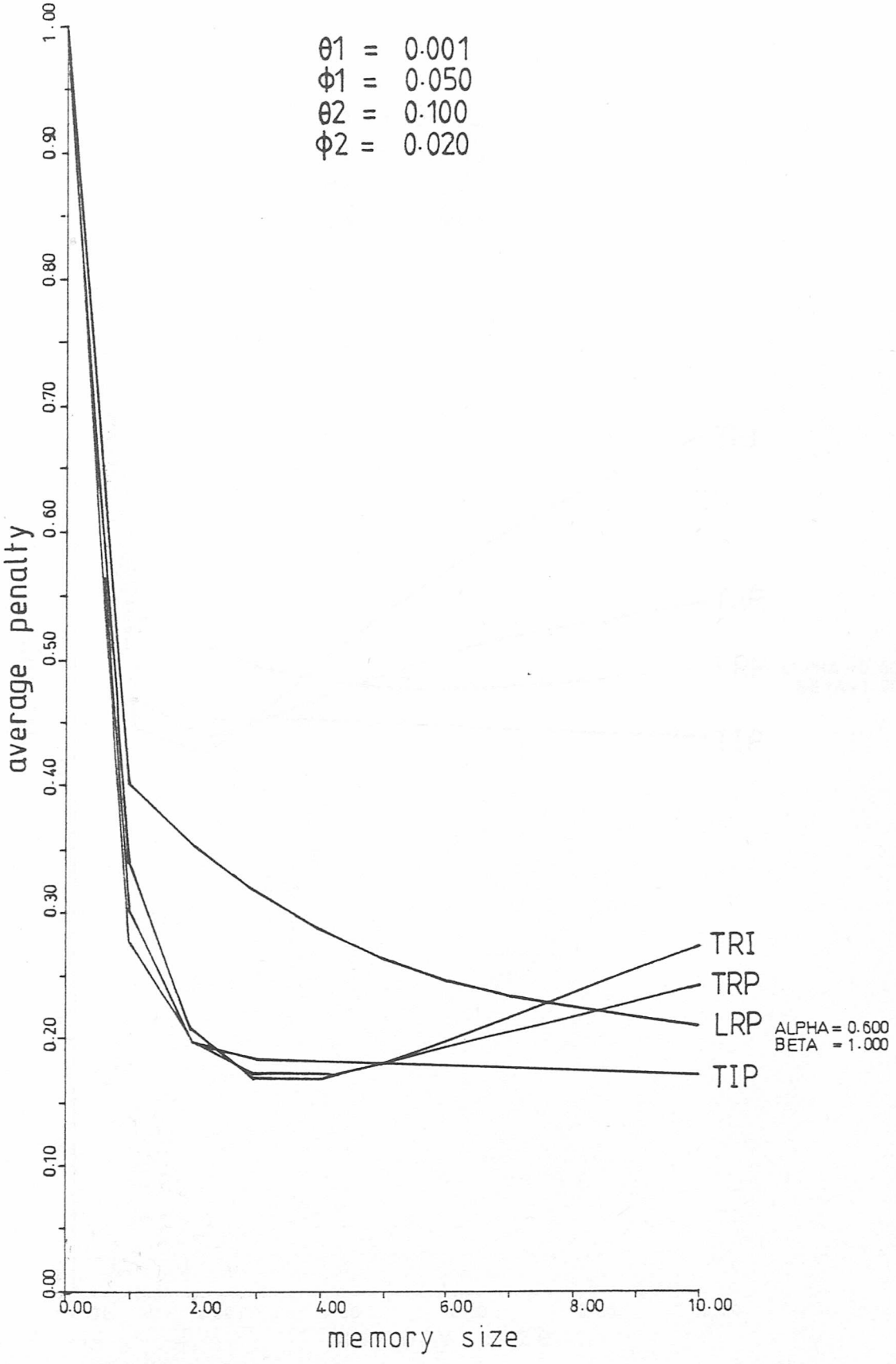


Figure 5·14(a) Theoretical average penalties

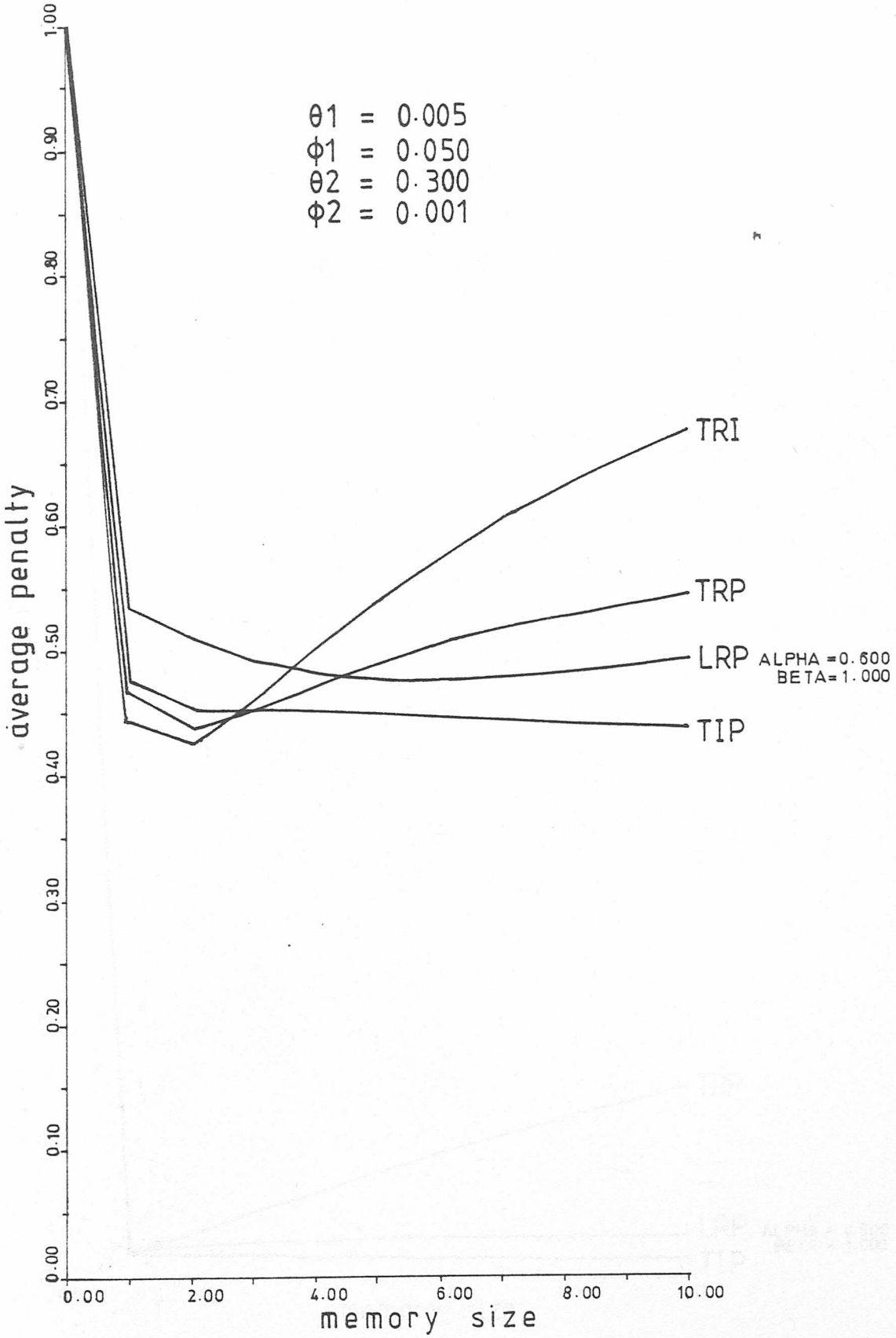

θ1 = 0.005
φ1 = 0.050
θ2 = 0.300
φ2 = 0.001
TRI
TRP
LRP
ALPHA =0.600
BETA=1.000
TIP
average penalty
memory size
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5.14 (b)

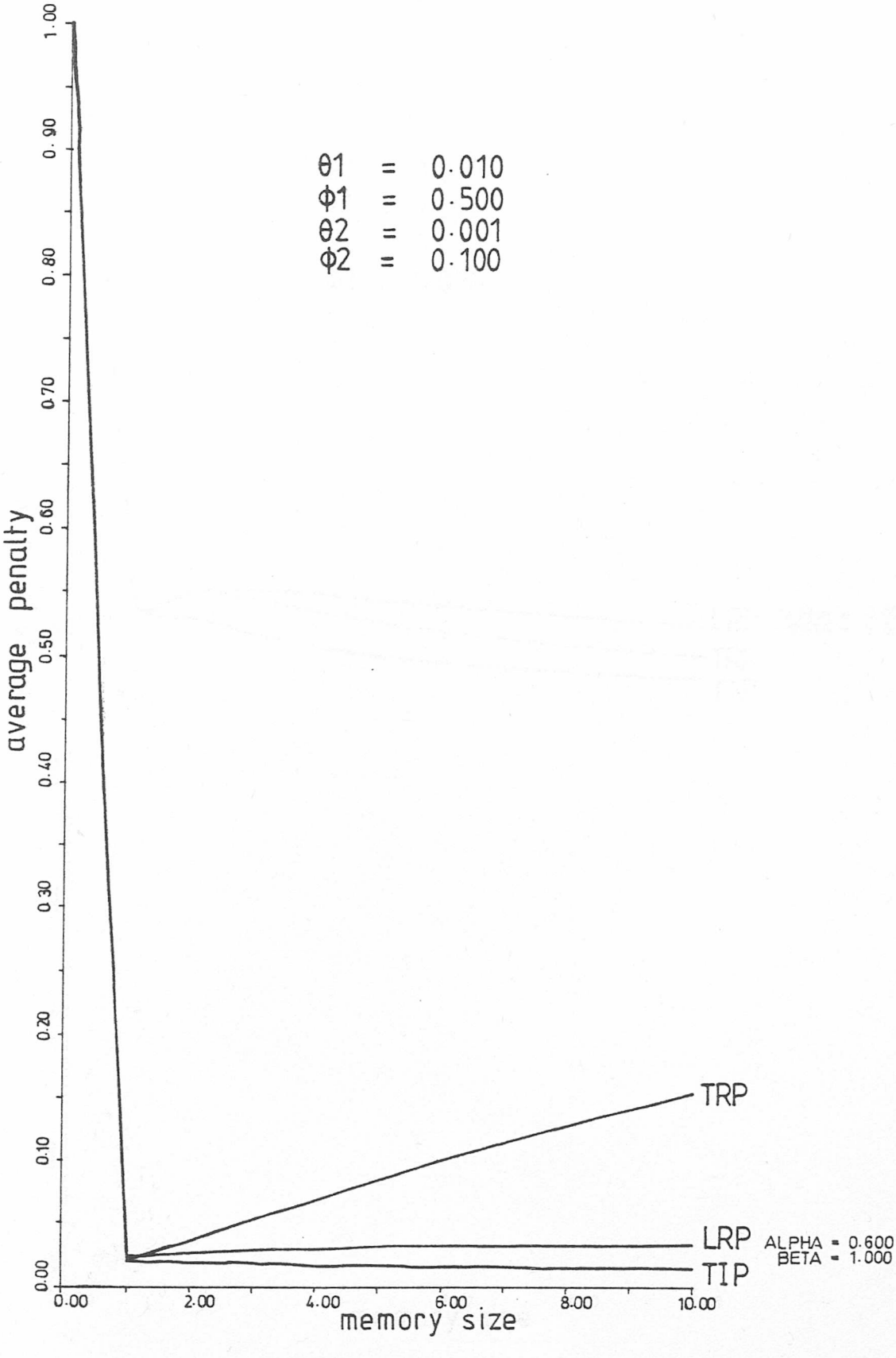

θ1 = 0·010
φ1 = 0·500
θ2 = 0·001
φ2 = 0·100
average penalty
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size
TRP
LRP
TIP
ALPHA = 0.600
BETA = 1.000

Figure 5.14(c)

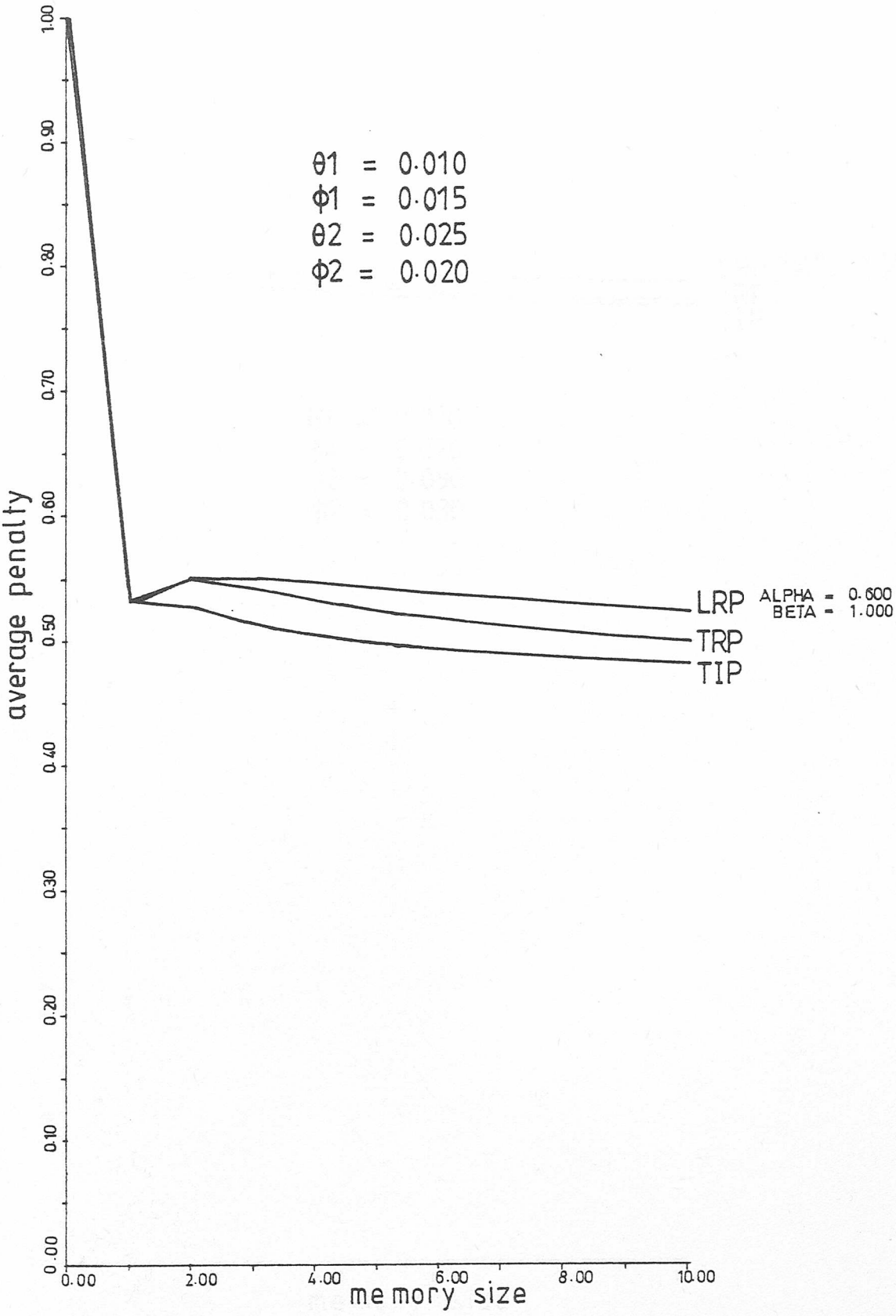

θ1 = 0·010
φ1 = 0·015
θ2 = 0·025
φ2 = 0·020
LRP
TRP
TIP
ALPHA = 0·600
BETA = 1·000
average penalty
memory size
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5·14(d)

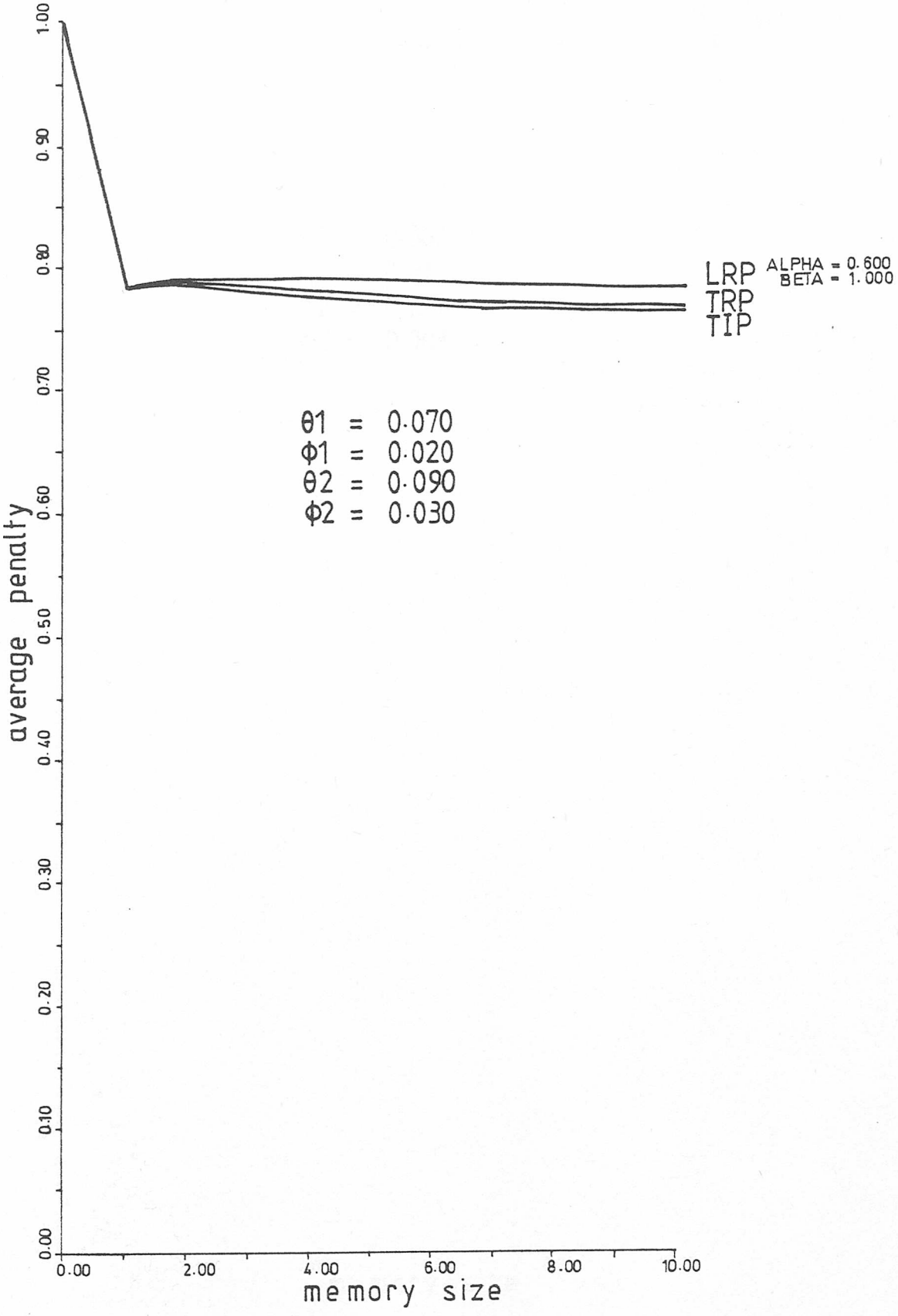

LRP
TRP
TIP
ALPHA = 0.600
BETA = 1.000
θ1 = 0.070
φ1 = 0.020
θ2 = 0.090
φ2 = 0.030
average penalty
memory size
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5.14 (e)

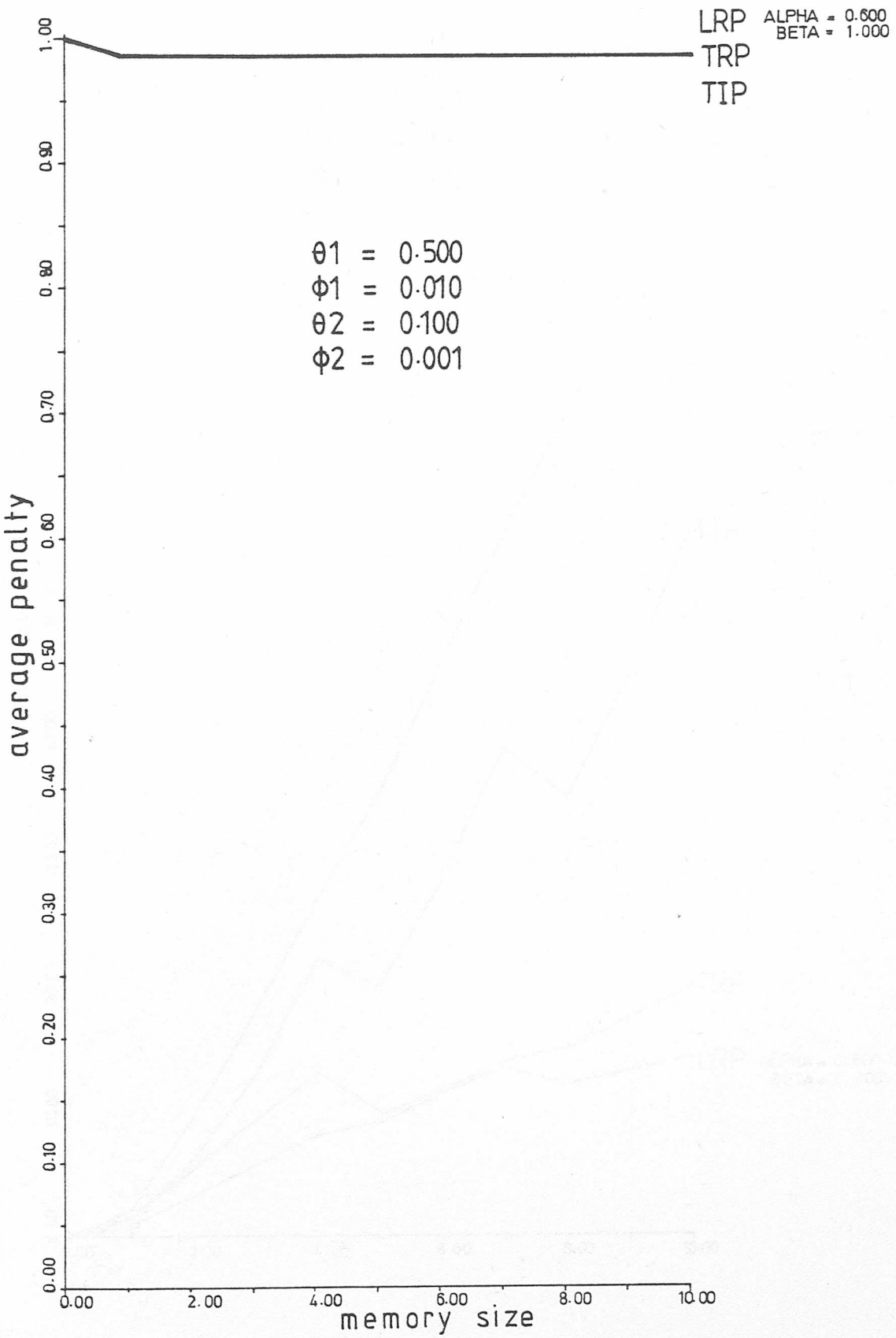

LRP
TRP
TIP
ALPHA = 0.600
BETA = 1.000
θ1 = 0·500
φ1 = 0·010
θ2 = 0·100
φ2 = 0·001
average penalty
memory size
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5·14(f)

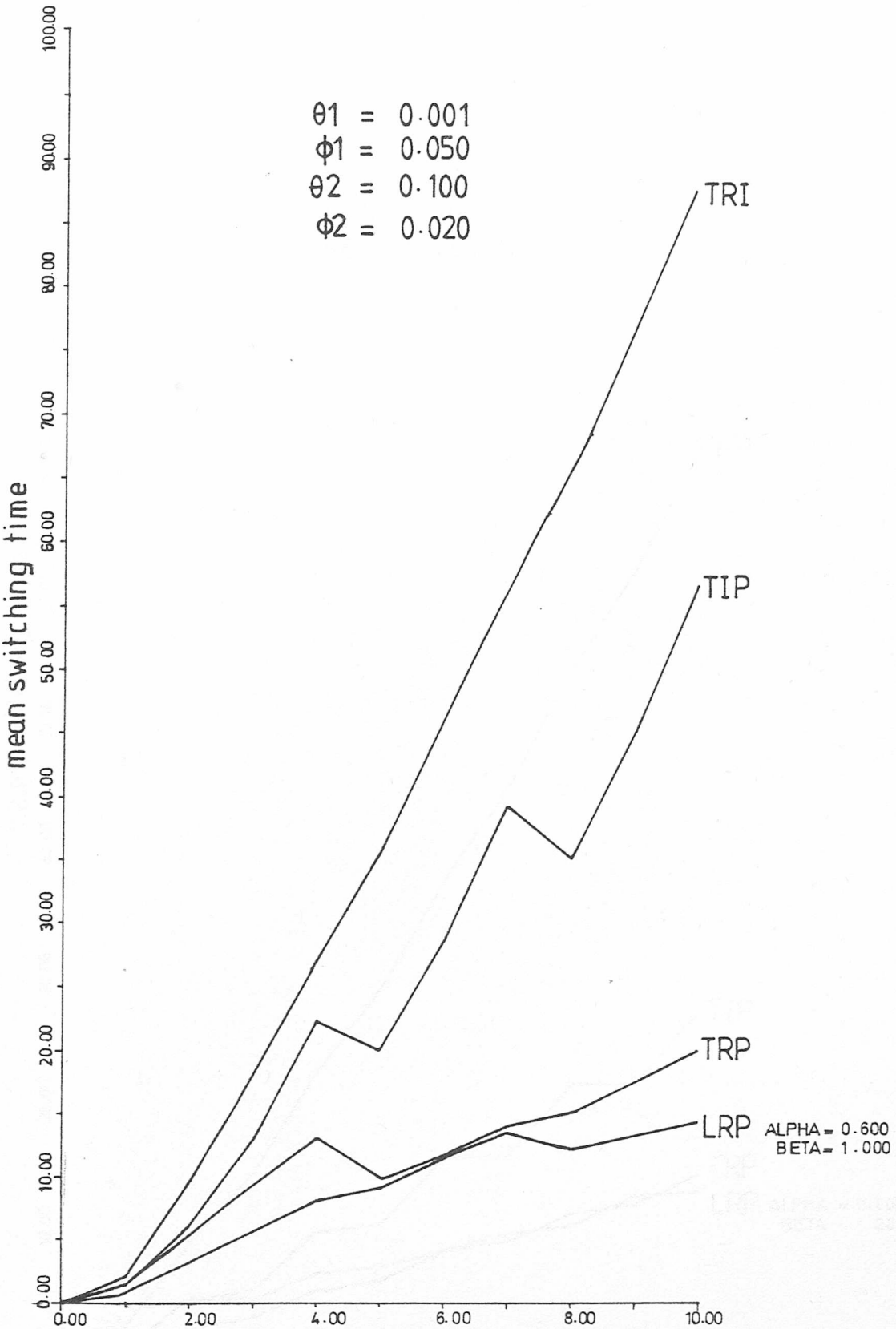


Figure 5.15(a) Theoretical mean switching times

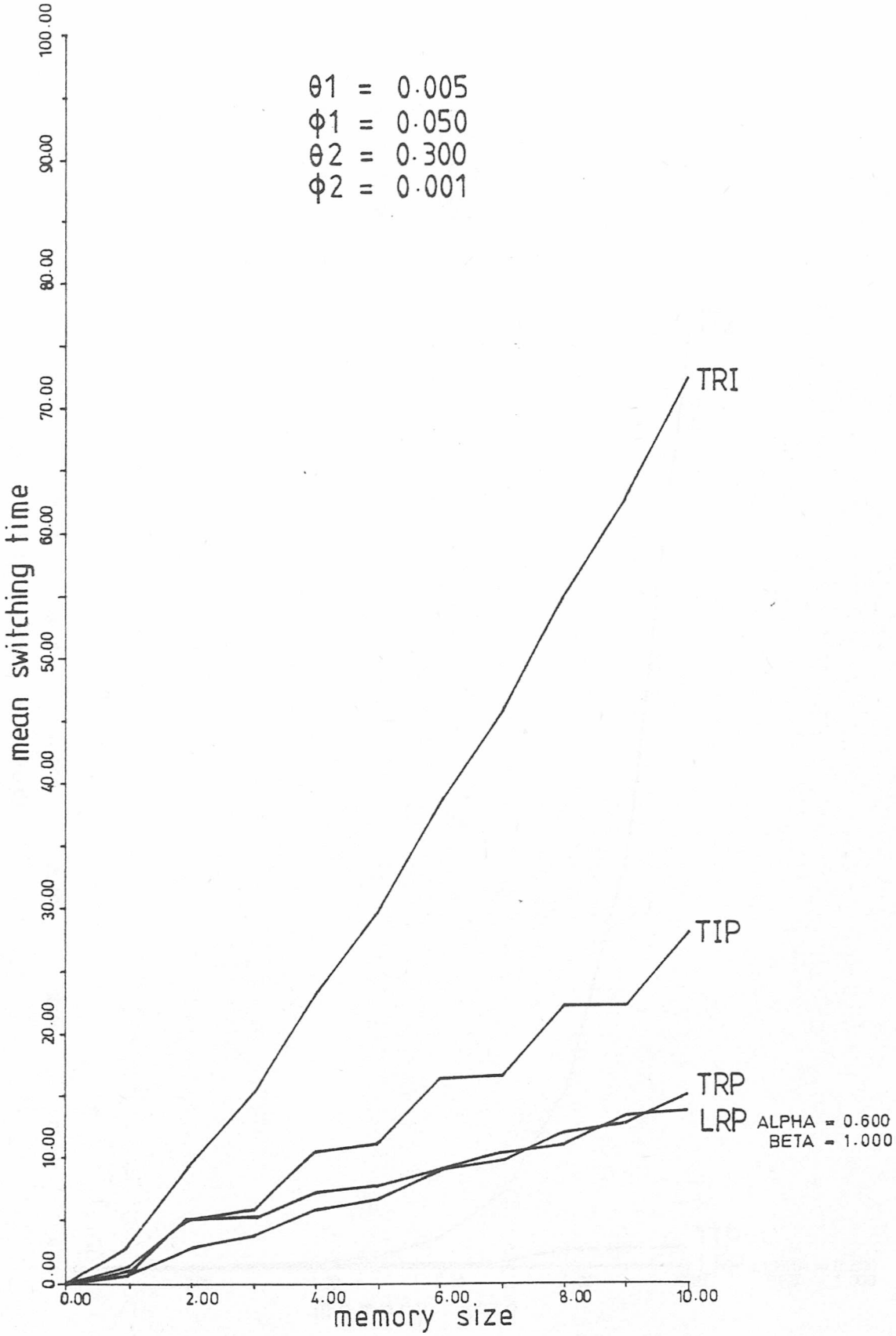

θ1 = 0·005
φ1 = 0·050
θ2 = 0·300
φ2 = 0·001
mean switching time
memory size
0.00
10.00
20.00
30.00
40.00
50.00
60.00
70.00
80.00
90.00
100.00
0.00
2.00
4.00
6.00
8.00
10.00
TRI
TIP
TRP
LRP
ALPHA = 0.600
BETA = 1.000

Figure 5·15 (b)

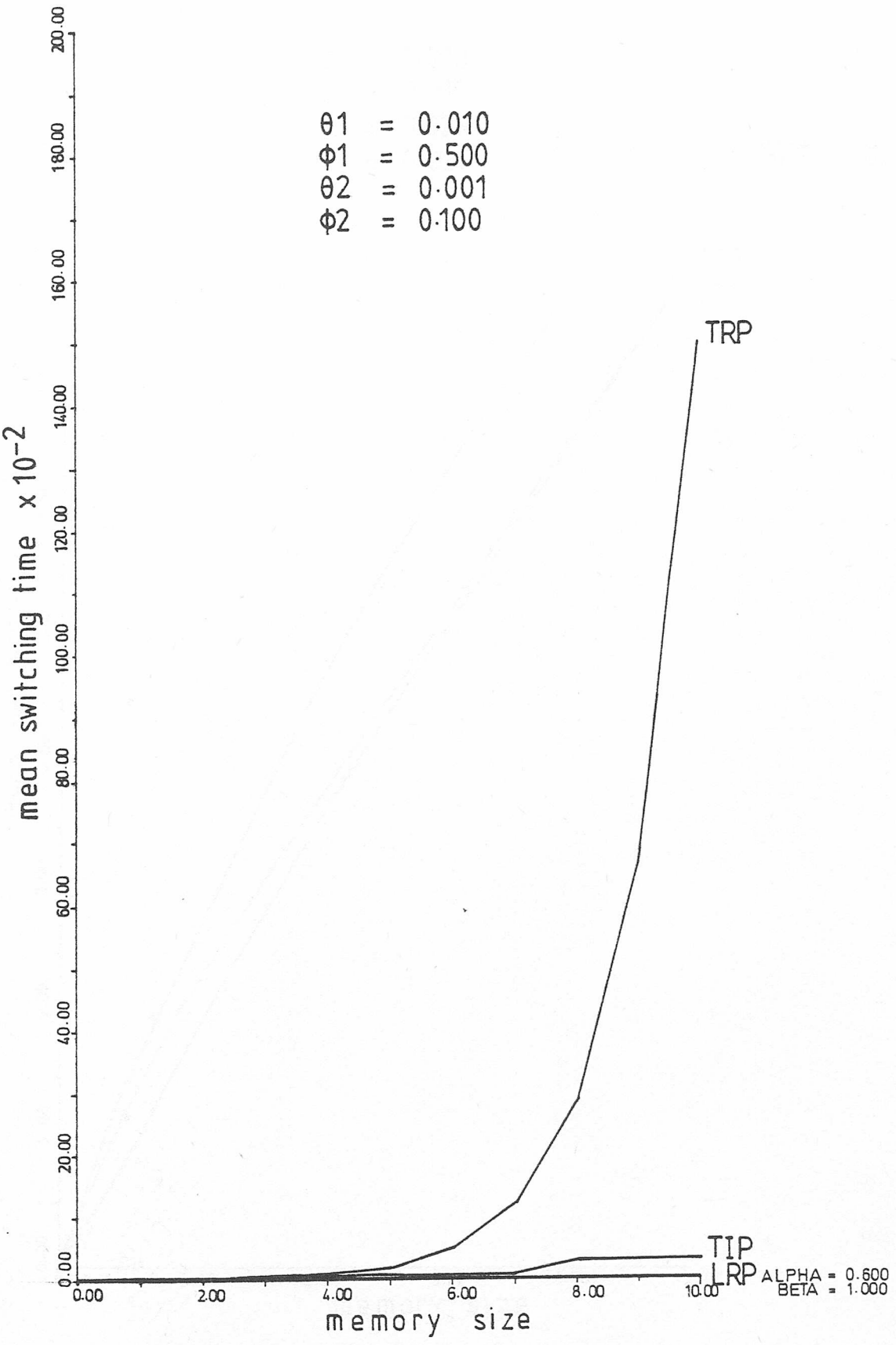

θ1 = 0·010
φ1 = 0·500
θ2 = 0·001
φ2 = 0·100
TRP
TIP
LRP
ALPHA = 0·600
BETA = 1·000
mean switching time x 10⁻²
memory size
0.00
20.00
40.00
60.00
80.00
100.00
120.00
140.00
160.00
180.00
200.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5·15(c)

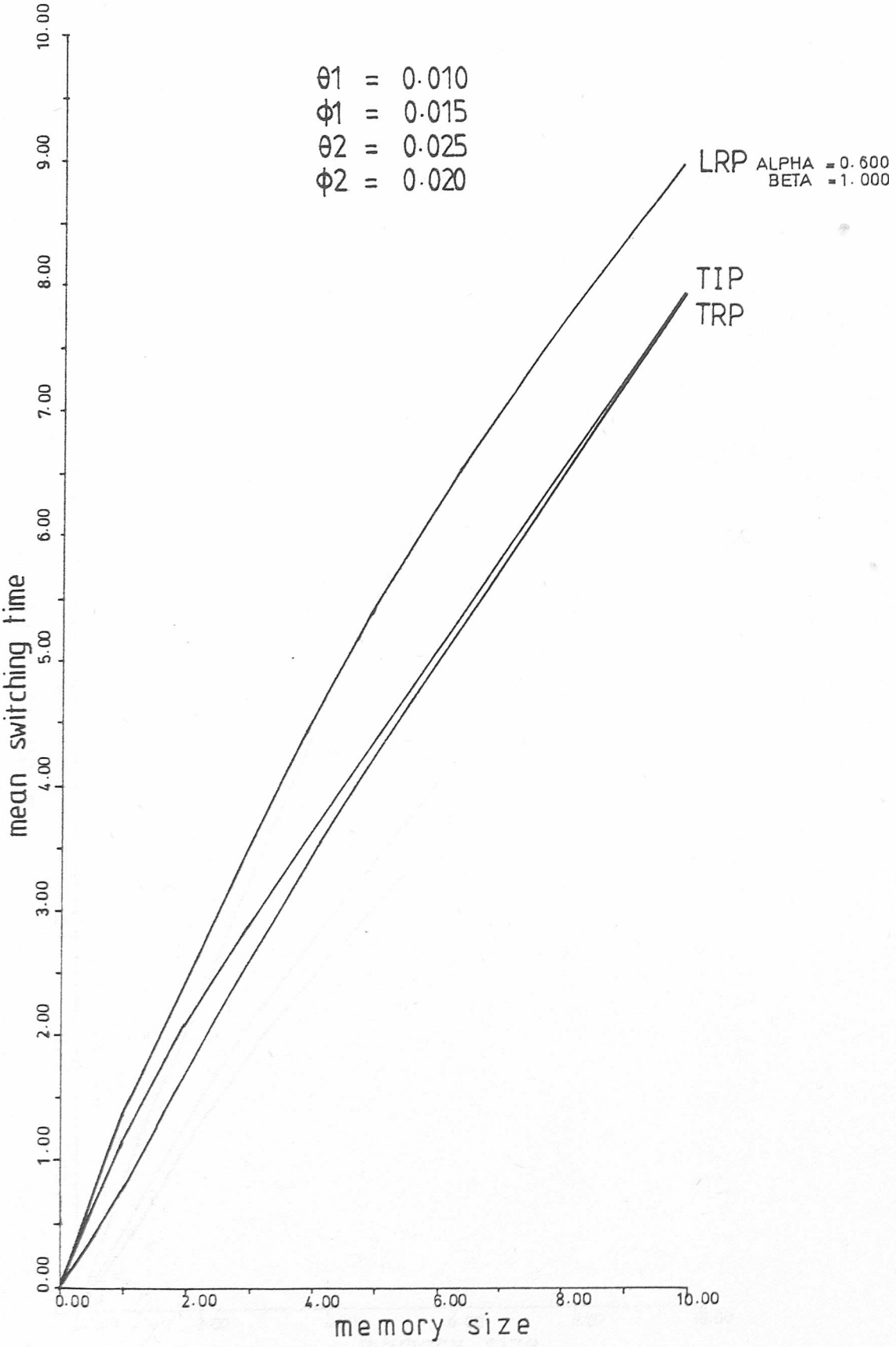

θ1 = 0·010
φ1 = 0·015
θ2 = 0·025
φ2 = 0·020
LRP ALPHA = 0.600
BETA = 1.000
TIP
TRP
mean switching time
memory size
0.00
1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00
9.00
10.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5·15(d)

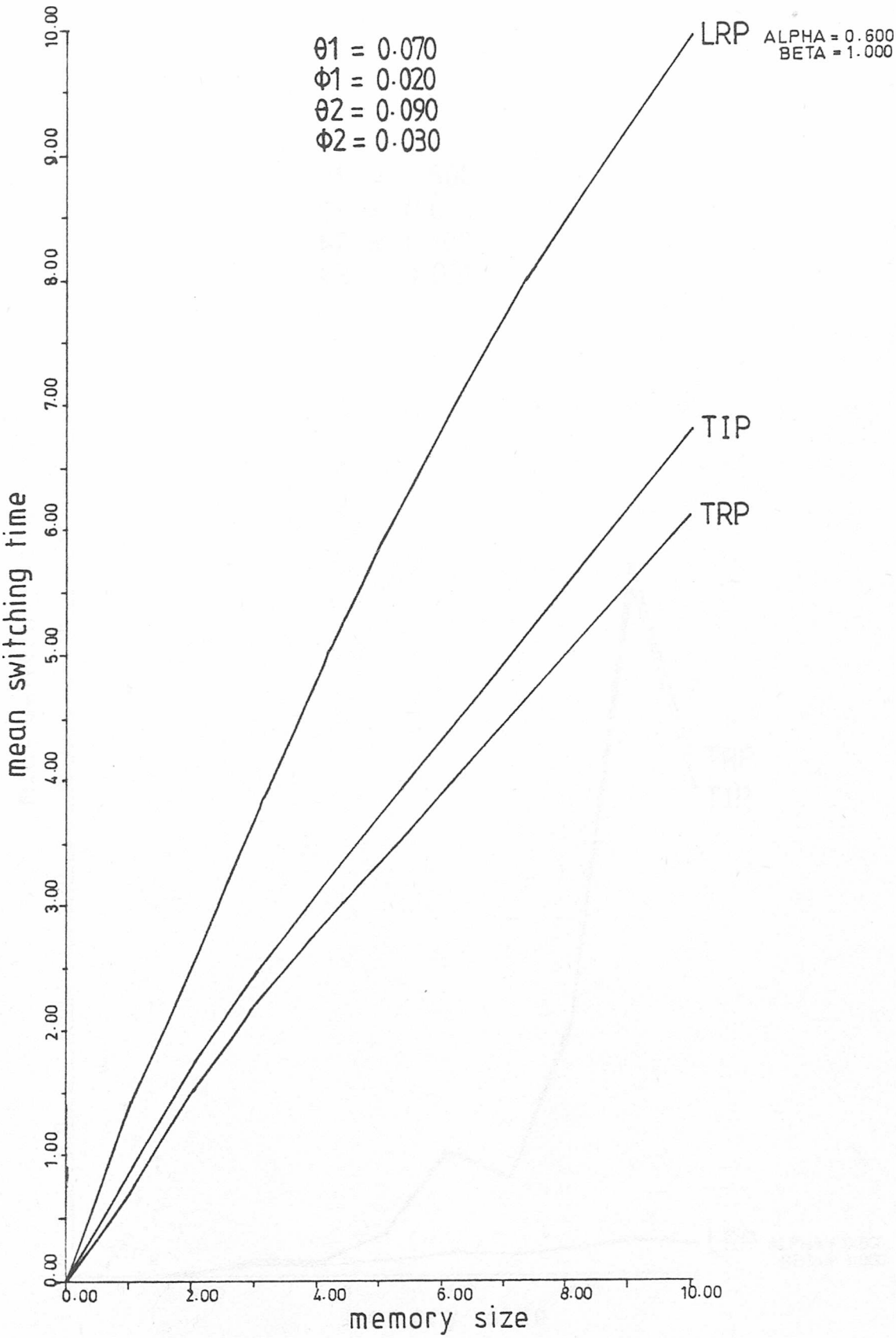

θ1 = 0·070
Φ1 = 0·020
θ2 = 0·090
Φ2 = 0·030
LRP
ALPHA = 0.600
BETA = 1.000
TIP
TRP
mean switching time
memory size
0.00
1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00
9.00
10.00
0.00
2.00
4.00
6.00
8.00
10.00

Figure 5·15 (e)

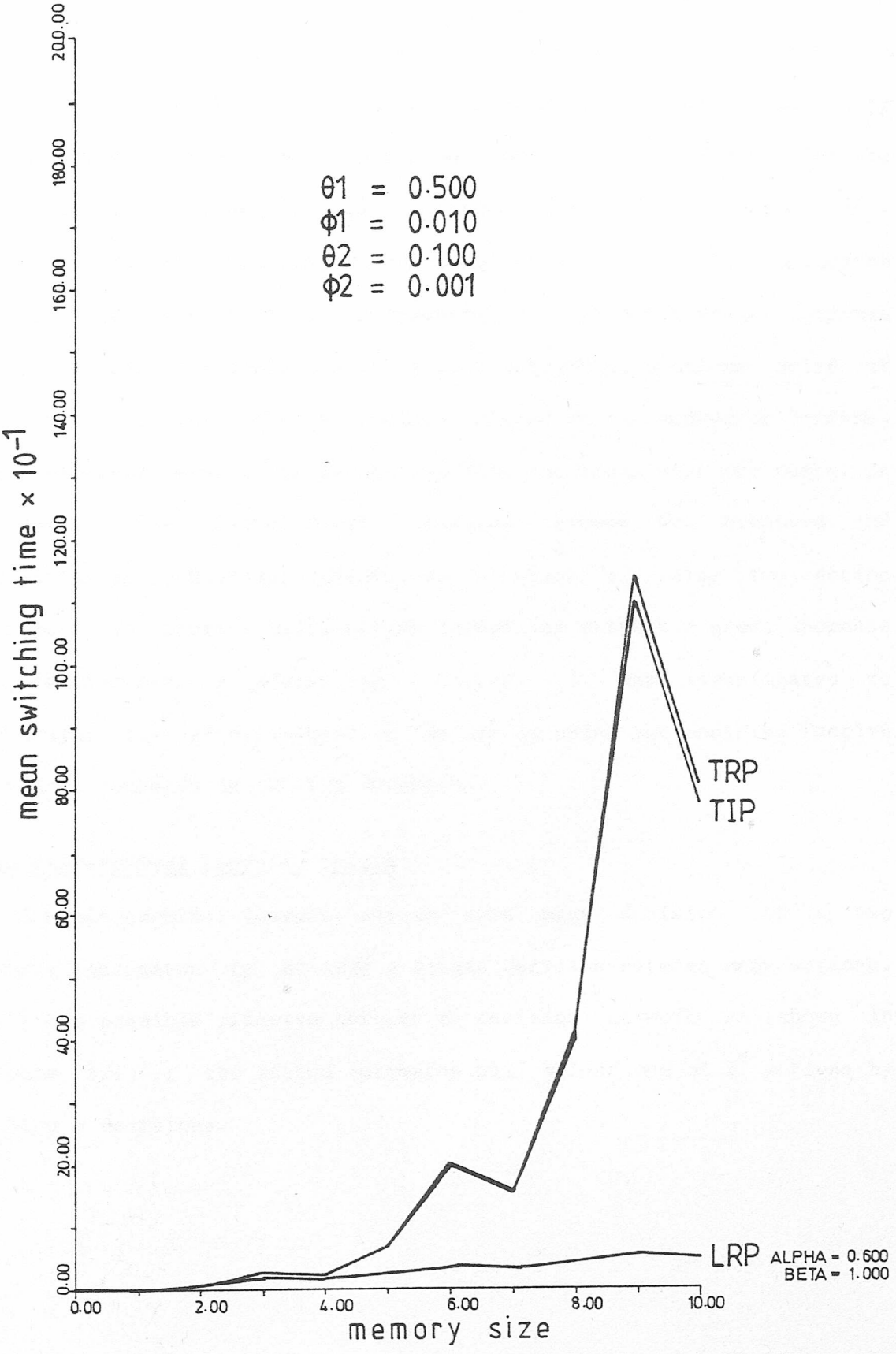

θ1 = 0·500
φ1 = 0·010
θ2 = 0·100
φ2 = 0·001
mean switching time × 10⁻¹
0.00
20.00
40.00
60.00
80.00
100.00
120.00
140.00
160.00
180.00
200.00
0.00
2.00
4.00
6.00
8.00
10.00
memory size
TRP
TIP
LRP
ALPHA = 0.600
BETA = 1.000

Figure 5·15(f)

CHAPTER 6 MULTI-ACTION AUTOMATA AND AUTOMATA GAMES

Introduction

The learning automata considered in earlier chapters were two action automata. In practice an environment was unlikely to have only two actions, multi-action automata were required. The formula for the Lrp automaton, equations (1.7)-(1.10) is for an automaton of r actions. Tsetlin also gives a multi-action scheme for his automaton [15]. When the practical implementation of multi-action automata is considered, for example a 100 action automaton, problems arise if the hardware involved is linearly related to the number of actions. In a software version the processing time increases with the number of actions. The hierarchical learning scheme was proposed and investigated by Neville [38,39] as a means of using two action automata to provide multi-action capability without a great increase in the hardware or processing required. It was investigated to determine the effectiveness of the system using the modified Tsetlin automata compared to the Lrp automaton.

The Hierarchical Learning System

The hierarchical learning system uses many decisions of a two action automaton to achieve a single decision between many actions. With the possible pathways through a decision network as shown in Figure 6.1, a two action automaton will select one of 2^n actions by taking n decisions.

In operation a single two-action automaton is used to make a decision at each node on the path through the decision network. The data necessary for a decision at every node is stored in a memory and supplied to the automaton as required. Once an action has been taken and a response obtained from the environment the path taken through the decision structure is retraced and the decision data for each node encountered is updated by the automaton.

The hierarchical learning scheme was simulated using a computer. The automata included in the simulation were the two action Lrp, as defined by equations (1.7)-(1.10) and the modified Tsetlin automata as described in Chapter 3.

The Hierarchical Learning System-Results

The results show the average number of times a particular action was selected by the automaton over 100 runs of the simulation. Each graph shows the situation after the automaton has selected I actions. Also included with each graph is the average penalty received by the automaton and the probability of selecting the optimal action up to that time. The initial conditions at the start of each simulation were such that each path through the decision network was of equal probability. The values of α and β chosen for the Lrp automaton and the ADDIE and memory sizes chosen for the type 1 modified Tsetlin automaton represented the best from a variety of values tested which corresponded with consistent behaviour.

Figures 6.2(a)-(c) show the operation of a type 1 modified Tsetlin automaton with memory size of 10 and ADDIE bit size 6 in a 16 state environment. The penalty probabilities of the environment were chosen at random with the result that action 14 corresponded to the minimum penalty probability. Figure 6.2(a) shows that after the first 100

iterations the automaton is already selecting the optimal action most often but that during this period the automaton has been selecting the other actions to a considerable extent. Figure 6.2(b) shows that by 500 iterations action 14 has been selected more often than the other actions put together. Figure 6.2(c) shows that in the long term the total average penalty received by the automaton continues to decrease and over 10,000 iterations the probability of selecting the optimal action is approximately 0.88.

Figures 6.3(a)-(c) show results for an Lrp automaton with $\alpha = 0.96$ and $\beta = 0.99$ operating under the same conditions as Figure 6.2. Over the first 100 iterations the performance of the Lrp automaton is better than that of the modified Tsetlin automaton. Over 1000 iterations Figures 6.3(b) and 6.2(b) show that the average penalties received by the two automata are approximately equal after 750 iterations while the probabilities of selecting the optimal action are equal after 1000 iterations. In the long term the performance of the Lrp automaton reached steady state at values which were poorer than the corresponding values for the modified Tsetlin automaton.

Figures 6.4 and 6.5 show results for the automata in a different environment. In this case the penalty probabilities were chosen in order to make it difficult for the automaton to select the optimal action. Action 13 was selected as the optimal action while the actions closest to it in the decision network, actions 14, 15 and 16 were given high penalty probabilities.

The aim of this was to discourage the use of the decision network paths leading to actions 14, 15 and 16 during the initial learning period. Since the path leading to action 13 was for most of its length common to the paths leading to actions 14, 15 and 16, giving

these actions high penalty probabilities would have the effect of discouraging the selection of action 13. For similar reasons the second most optimal action, action 4 was surrounded by actions with relatively low penalty probabilities in order to encourage the automaton to select action 4 rather than the optimal action.

The results show that neither automaton converges to the wrong action in the difficult environment though using automata with different parameters it was found that convergence to the non-optimal action 4 was more likely in the difficult environment.

The Hierarchical Learning System-Conclusions

The results show that though the Lrp automaton has a shorter learning time in a hierarchical learning system the type 1 modified Tsetlin automaton has a better steady state performance. Attempts to improve the steady state response of the Lrp automaton by adjusting the values of α and β resulted in occasional convergences to non-optimum actions. It was felt that the hierarchical learning system may have accentuated this tendency. The modified Tsetlin automaton was less prone to this as it selected the actions more evenly during the learning period.

Automata Games-Introduction

A game exists between two automata when each automaton can affect the penalty probability received by the other automaton. There are two types of automaton game, cooperative games, where the automata receive the same penalty probability and so can cooperate to receive the lowest average penalty, and zero sum or competitive games, where if one automaton receives a reward the other receives a penalty. Automata games are of interest as a way of comparing and testing

differing types of automata to determine the desirable qualities for automata. In the practical use of learning automata where many automata operate on different parts of a single environment, games exist. The use of automata in telephone traffic routing [45,46,47,48,49] as well as the system investigated in Chapter 8 are examples of practical situations where games occur. In such cases the understanding of simple games could be an advantage.

Cooperative and competitive games between the Lrp automaton and some of the Type 2 automata, including the Tsetlin automaton, have been investigated [40]. Since the probabilistic Tsetlin automata have advantages over the Tsetlin automaton these automata were tested against the Lrp automaton in games.

Cooperative Games

A simulation program was written to provide games between the Lrp, Lri, Trp, Tip and Tri automata. The program provided for up to three players in the game, though in practice only two were used, and for up to ten actions available to each player. The penalty probabilities were input to the program in the form of a matrix. The action of the first player specified a row of the penalty matrix while the action of the second player specified a column of the matrix. The combination of these defined an element of the penalty matrix from which the feedback for the learning automata could be obtained. Facilities were provided to allow the automata to receive the same feedback giving a cooperative game or to receive the opposite feedback giving a competitive game.

The first results were obtained for cooperative games with a variety of matrices and for different combinations of automata with varying parameters. The operation of the automata was as expected from previous work and in most cases the automata converged towards selecting the smallest penalty element most frequently. However when operating with an environment like

$$\begin{matrix} 0.4 & 0.5 & 0.6 \\ 0.3 & 0.9 & 0.2 \\ 0.1 & 0.05 & 0.8 \end{matrix}$$

convergence was not always to the element corresponding to the minimum penalty. In most cases the automata converged to the third row and first column, element 3,1 most frequently. Convergence to the correct element depended on the operation of player 1 which controlled which row of the penalty matrix was selected. If the player 1 automaton had a low degree of optimality there was a significant probability that rows 1 and 2 would be selected so it was an advantage to player 2 to select column 1 rather than select column 2 and also be forced to receive the high penalty probabilities associated with rows 1 and 2 in that column. If the player 1 automaton had a high degree of optimality convergence would be to row three. In this case player 2 was free to select column 2. However while player 1 was converging, player 2 was forced to select column 1 and if the rates of convergence of the players were similar or if player 2 converged faster than player 1 , player 2 would not be able to change actions after player 1 had converged. The automata only converge to select the optimal probability element if both the automata have a high degree of optimality and player 2 has a rate of convergence slower than player 1.

Competitive Games

A variety of experiments were also carried out with competitive games. In this situation the players are operating with different penalty probability matrices. For example player 1 operates with the matrix shown below on the left while player 2 operates with the matrix shown on the right.

0.7	0.6	0.4	0.3	0.4	0.6
0.6	0.5	0.1	0.4	0.5	0.9
0.5	0.45	0.3	0.5	0.55	0.7

In the experiments, if player 1 selected the rows and player 2 selected the columns there was overall convergence to the penalty element corresponding to a minimum of a column in the player 1 matrix and the minimum of a row in the player 2 matrix. Considering the player 2 matrix only, the overall convergence was to the penalty element which was the minimum of a row and the maximum of a column. For the matrix above this corresponds to element 3,1.

In the matrix given below there are no elements which satisfy the conditions given above for convergence.

0.4	0.5	0.6	0.6	0.5	0.4
0.3	0.9	0.2	0.7	0.1	0.8
0.1	0.05	0.8	0.9	0.95	0.2

Experiments with this matrix have shown that the automata select penalty elements 2,2 3,2 3,3 2,3 cyclically. The automata are constantly changing their most frequent action and never converge. Figure 6.6 shows the action probabilities of two Lri automata in a competitive game using this matrix plotted against time. It was felt that a penalty matrix in which their were no penalty elements which satisified the convergence criteria given above would be the best in which to test automata against each other.

The results given in Table 6.1 were taken for automata games using the penalty probability matrix given below.

0.1	0.3	0.7	0.9	0.7	0.3
0.1	0.7	0.3	0.9	0.3	0.7
0.5	0.9	0.9	0.5	0.1	0.1

Player 1 selecting the rows sees high penalty probability elements in row 3 and lower penalty elements in rows 1 and 2. Player 2 selecting the columns sees high penalty probability elements in column 1 and lower penalty elements in columns 2 and 3. The automata will select penalty elements 1,2 2,2 2,3 1,3 but there should not be convergence to any of these elements as the convergence conditions are not satisified. For automata of equal performance the average penalty received by each automaton should be 0.5. The results presented in Table 6.1 are the average of two simulation runs, each automaton having the player 1 and player 2 position with the same random number sequence being used in both runs.

The first results taken were for automata of the same type. For fast Lri automata with relatively low α parameters it was difficult to get results as the automata went optimal but results were obtained for more slowly acting automata. Table 6.1(a) shows that the faster Lri automata with the smaller α parameter have the better performance. Obviously if a Lri automaton is made too fast it will go optimal and the slower automaton will have the better performance. Results (b),(c) and (d) are for Lrp automata. Results (b) with $\delta=1$ are inconclusive with the faster automata not showing a particular advantage. Results (c) with constant δ are again inconclusive. The results (d) compare automata with varying δ and show that the higher the degree of optimality the better the performance.

Results (e) for Trp automata show inconclusive results till large step sizes are used when the smaller step sizes have the better performance. In these cases the advantage in a small step size is lower variance and this factor becomes of greater importance than speed with large step sizes. Results (f) for the Tip automata are similar to the Trp with speed being an advantage for small step sizes but with low variance becoming more important at large step sizes. For the Tri automaton it was difficult to get results. The Tri automaton has a high tendency to go optimal and once a player has gone optimal the opponent is free to select the best penalty element. The results for the Tri automata measured which automaton went optimal last. Results (g) are runs in which the automata did not go optimal but this happened only for large step sizes and in any case the results are inconclusive.

Next the probabilistic Tsetlin automata were tested against the Lrp and Lri automata. The results in Table 6.1 (h) show the performance of various Trp automata against an Lrp automaton with $\delta = 1$. In all cases the Trp has the better performance. Increasing δ in results (i) increases the performance of the Lrp automata. In results (j) and (k) against Lri automata, the Lri automata have the better performance. For the Tip automaton results (l) shows that it has a poorer performance than the Lrp with $\delta = 1$. For results (m) and (n) it was again difficult to get results in which the Tri automaton did not become optimal but for the results given the performance was worse than that of the Lrp and Lri automata.

Finally the probabilistic Tsetlin automata were tested against each other. Results (o) shows the Trp superior to the Tip while (p) shows it generally superior to the Tri.

Automata Games-Conclusions

The Lrp, Lri and probabilistic Tsetlin automata have been investigated operating in a variety of games situations. For cooperative games the operation of the automata was as expected and in general convergence was to the minimum penalty element. A case where convergence was not to the minimum penalty element was identified and the conditions causing it found. Conditions for convergence in competitive games have been established as well as the possibility of convergence to either of two penalty elements or to none of the penalty elements. Using a matrix where there should be no convergence the Lrp, Lri and probabilistic Tsetlin automata have been tested against each other with the Lri automaton showing the best performance.

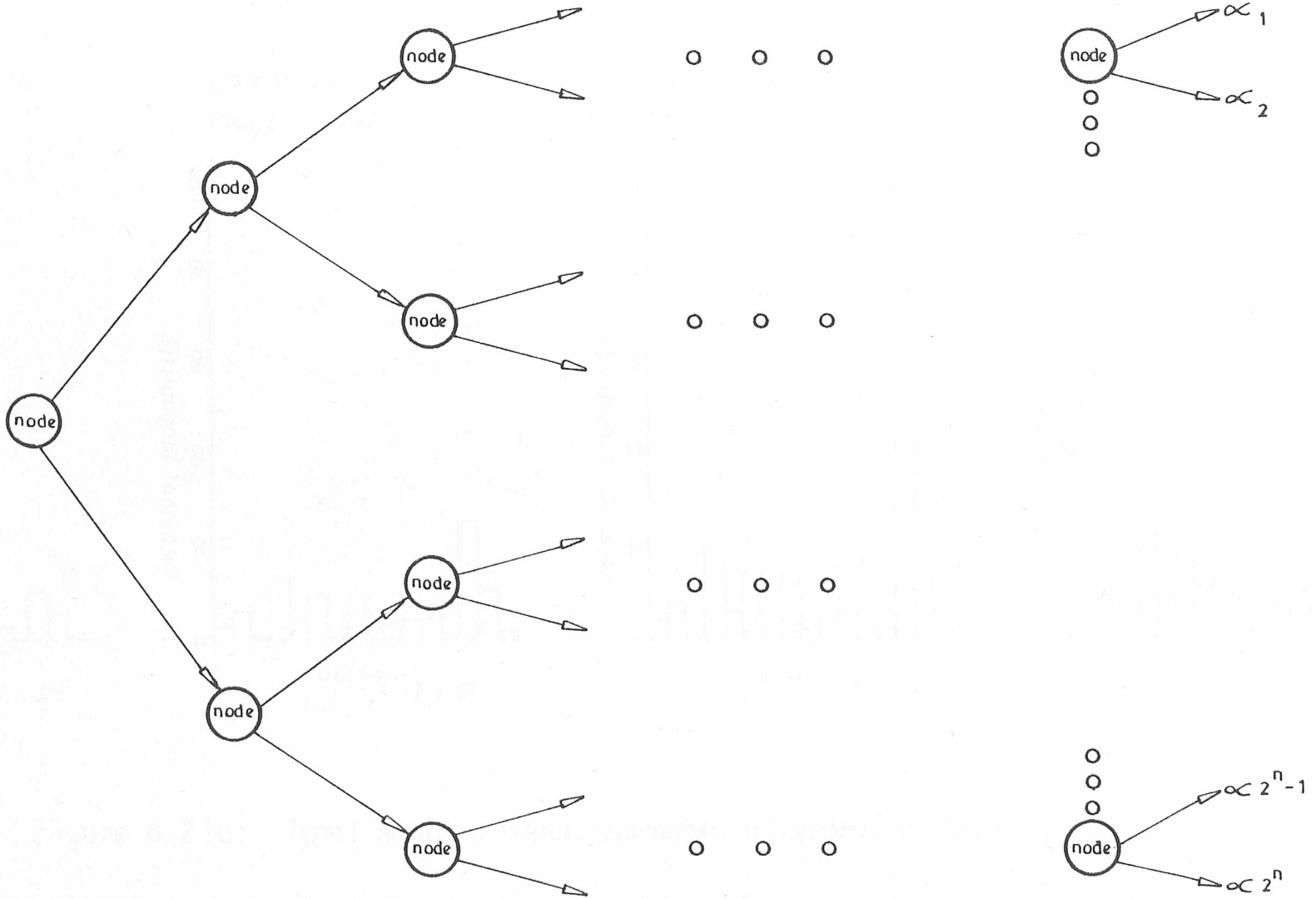


Figure 6.1 Decision network of hierarchical learning system

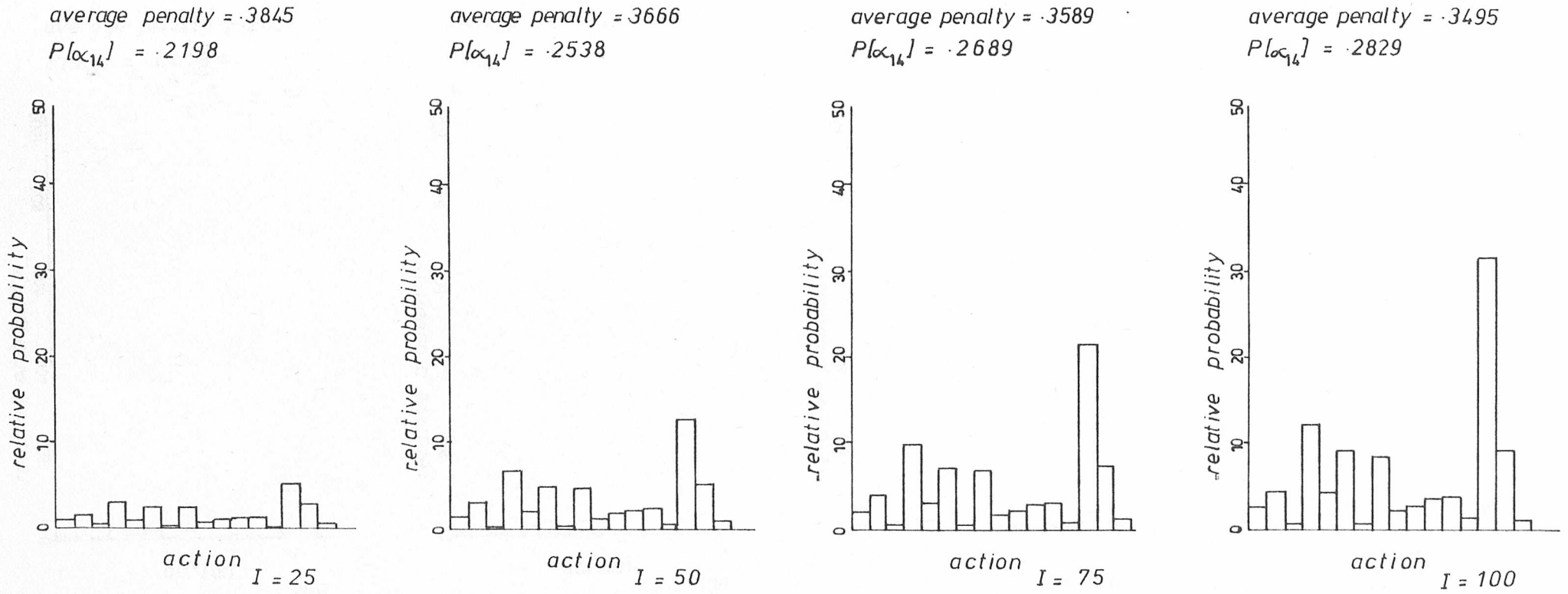


Figure 6·2 (a) Type1 modified Tsetlin automaton in hierarchical learning system

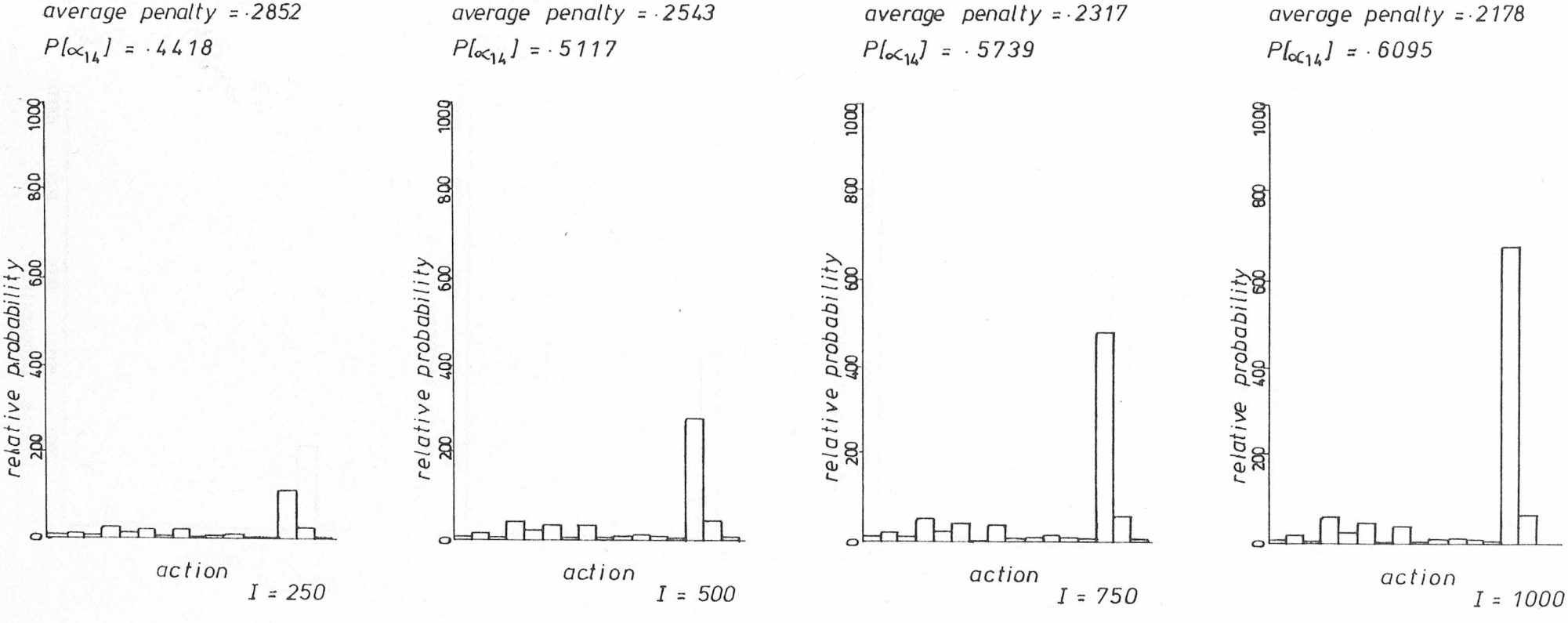

average penalty = ·2852
P[∝14] = ·4418
average penalty = ·2543
P[∝14] = ·5117
average penalty = ·2317
P[∝14] = ·5739
average penalty = ·2178
P[∝14] = ·6095
relative probability
0
200
400
600
800
1000
action
I = 250
I = 500
I = 750
I = 1000

Figure 6·2(b)

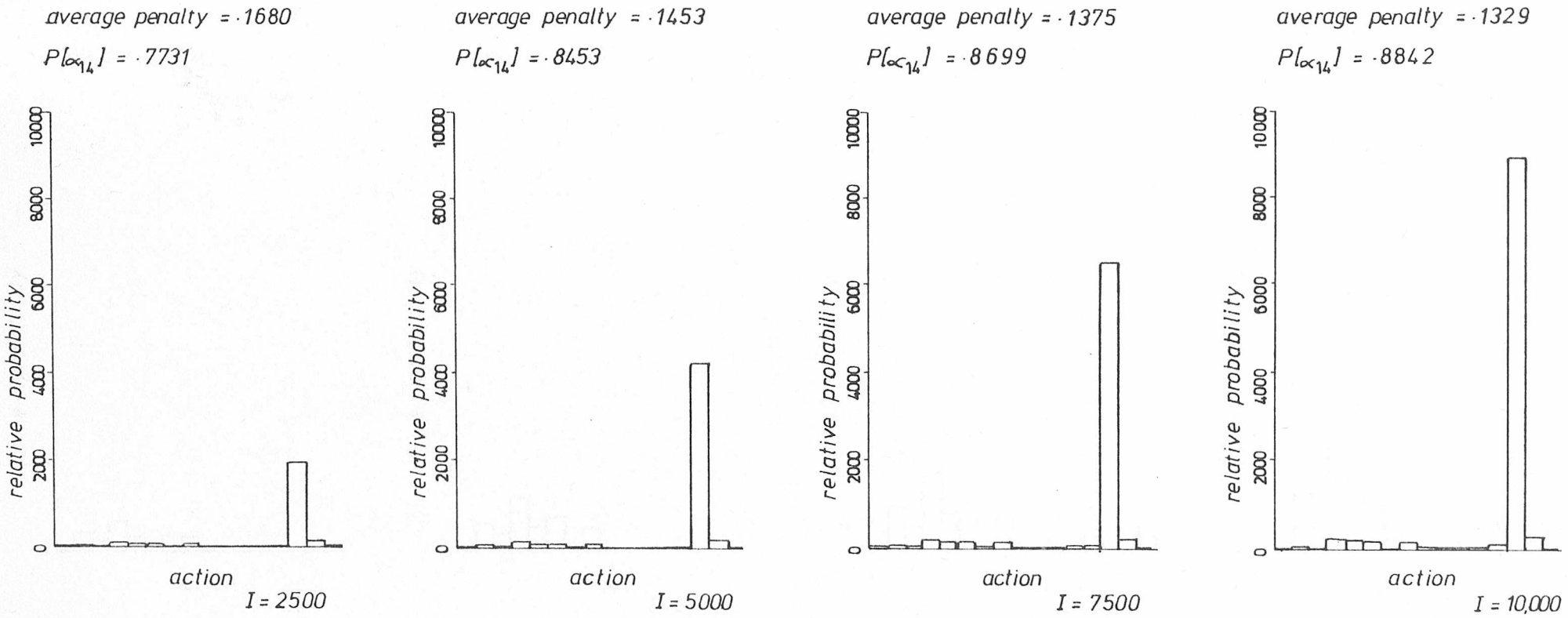

average penalty = ·1680
P[α14] = ·7731
relative probability
0
2000
4000
6000
8000
10000
action
I = 2500
average penalty = ·1453
P[α14] = ·8453
relative probability
0
2000
4000
6000
8000
10000
action
I = 5000
average penalty = ·1375
P[α14] = ·8699
relative probability
0
2000
4000
6000
8000
10000
action
I = 7500
average penalty = ·1329
P[α14] = ·8842
relative probability
0
2000
4000
6000
8000
10000
action
I = 10,000

Figure 6·2 (c)

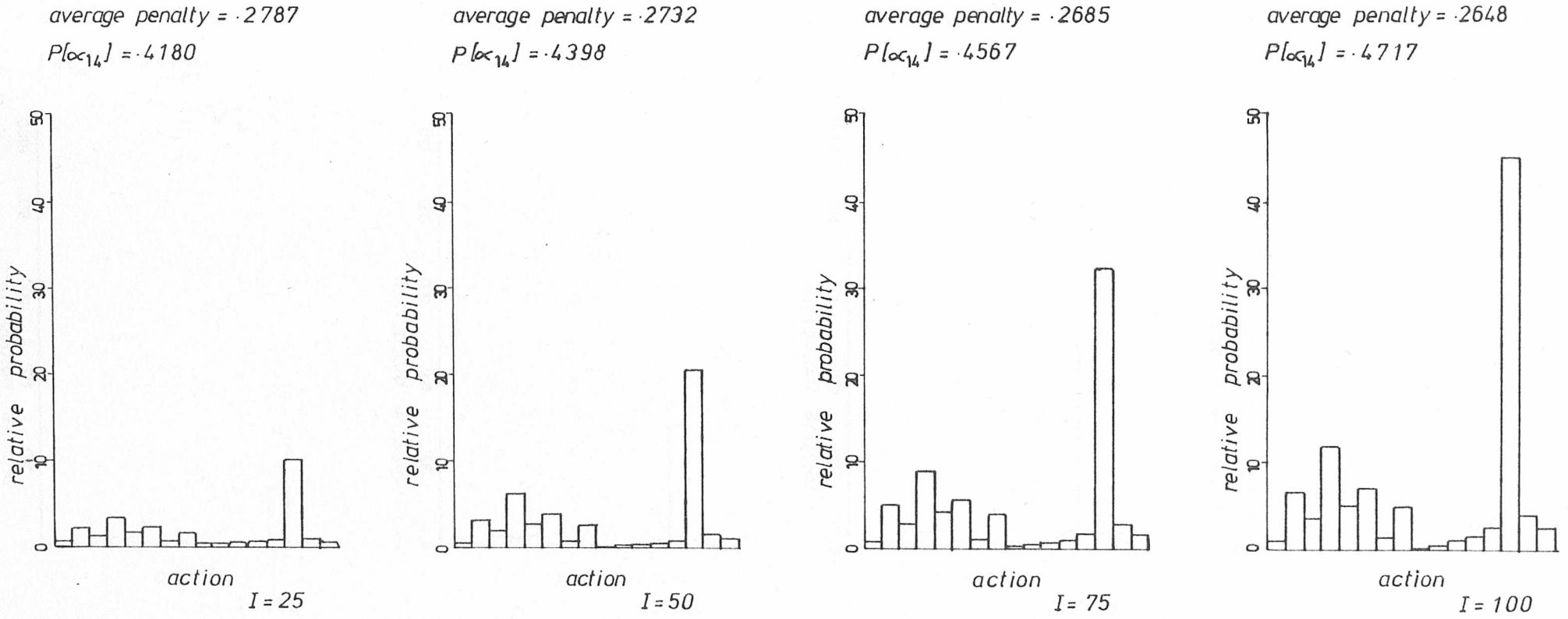


Figure 6·3 (a) L_{RP} automaton in hierarchical learning system

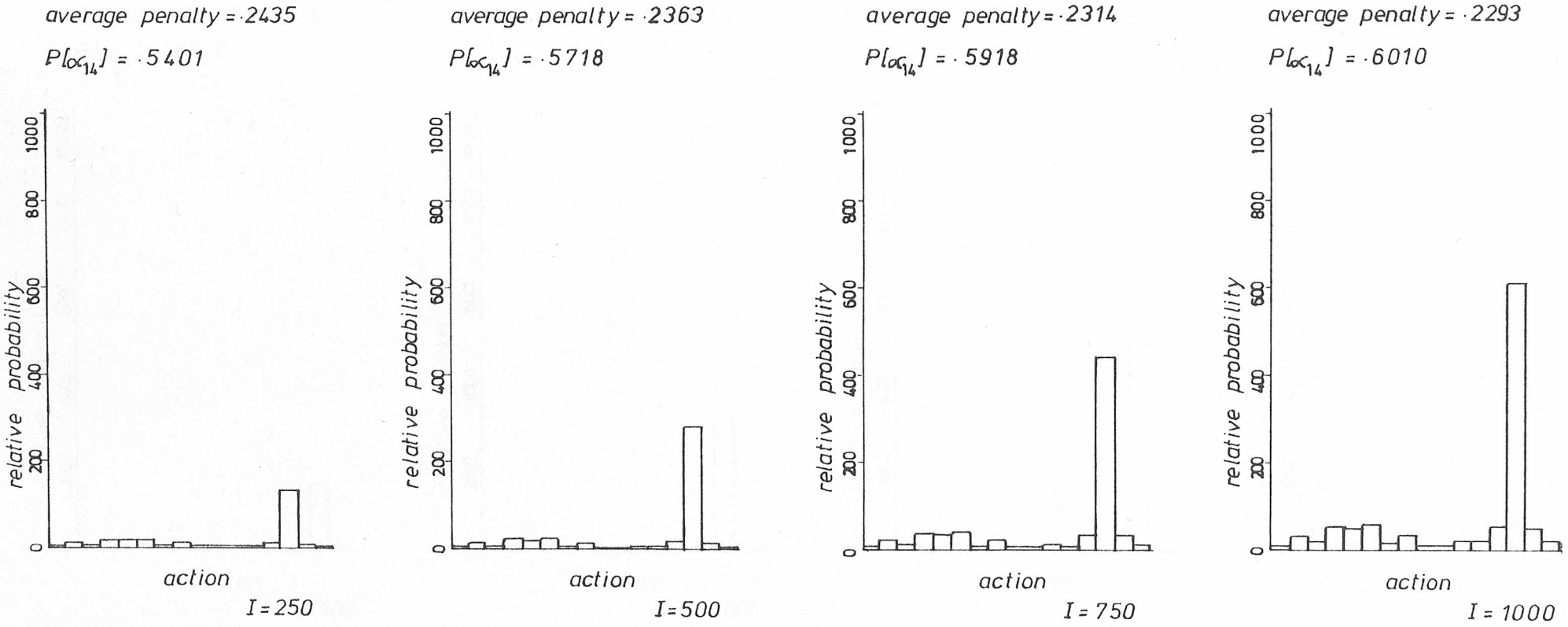

average penalty = ·2435
$P[\alpha_{14}]$ = ·5401
relative probability
0
200
400
600
800
1000
action
I = 250
average penalty = ·2363
$P[\alpha_{14}]$ = ·5718
relative probability
0
200
400
600
800
1000
action
I = 500
average penalty = ·2314
$P[\alpha_{14}]$ = ·5918
relative probability
0
200
400
600
800
1000
action
I = 750
average penalty = ·2293
$P[\alpha_{14}]$ = ·6010
relative probability
0
200
400
600
800
1000
action
I = 1000

Figure 6·3 (b)

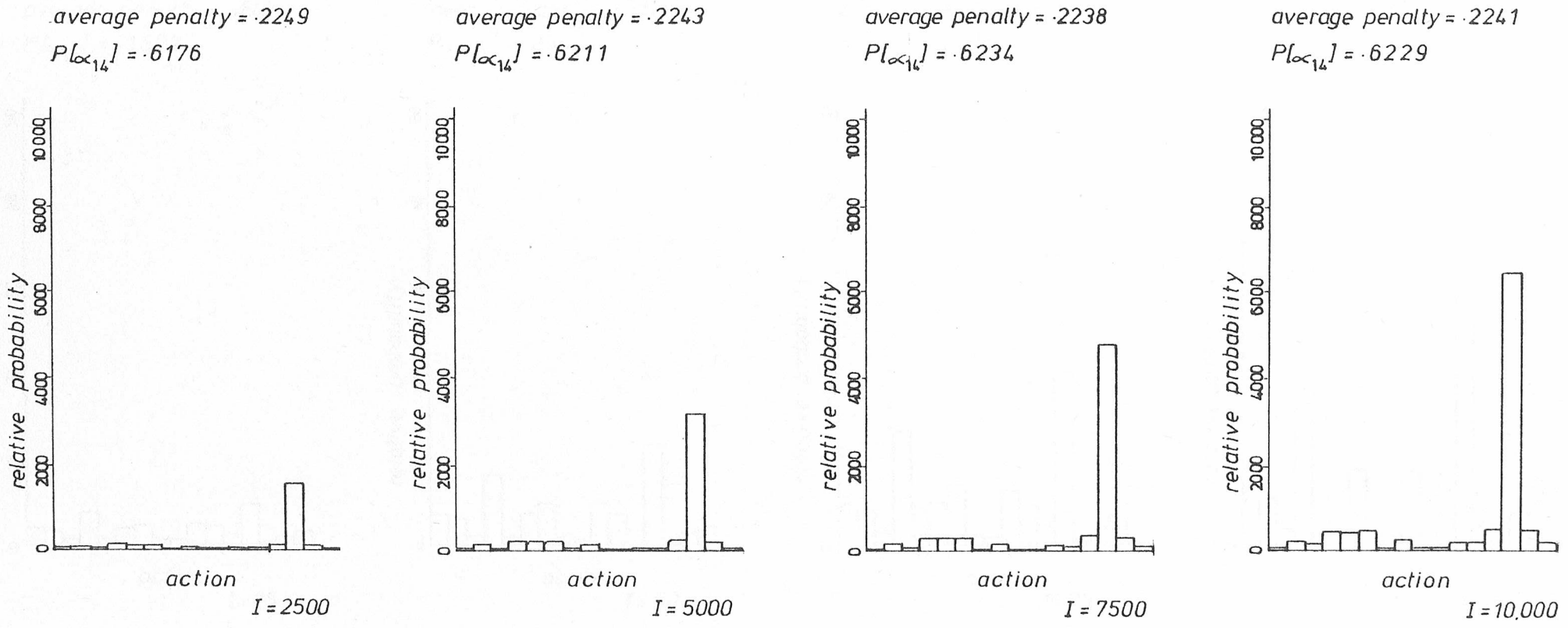

average penalty = ·2249
P[∝14] = ·6176
average penalty = ·2243
P[∝14] = ·6211
average penalty = ·2238
P[∝14] = ·6234
average penalty = ·2241
P[∝14] = ·6229
relative probability
0
2000
4000
6000
8000
10000
action
I = 2500
I = 5000
I = 7500
I = 10,000

Figure 6·3(c)

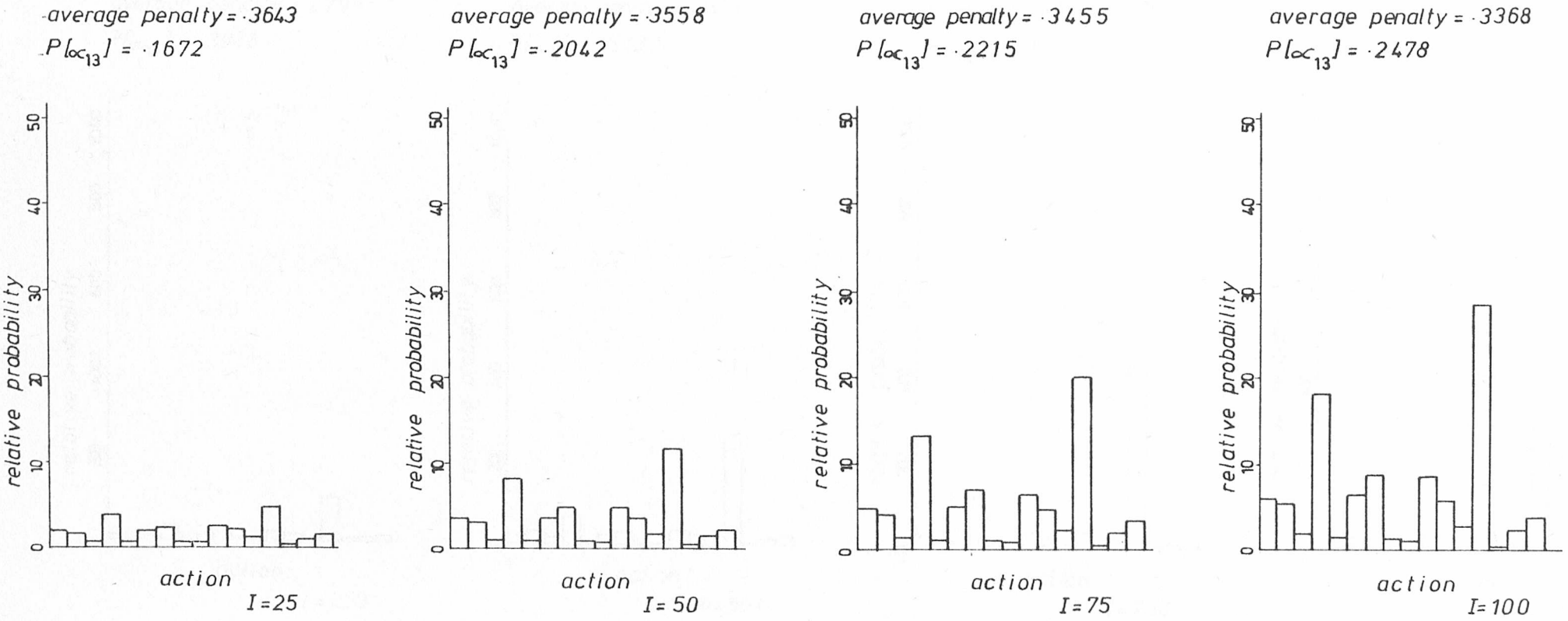


Figure 6·4(a) Type1 modified Tsetlin automaton in hierarchical learning system

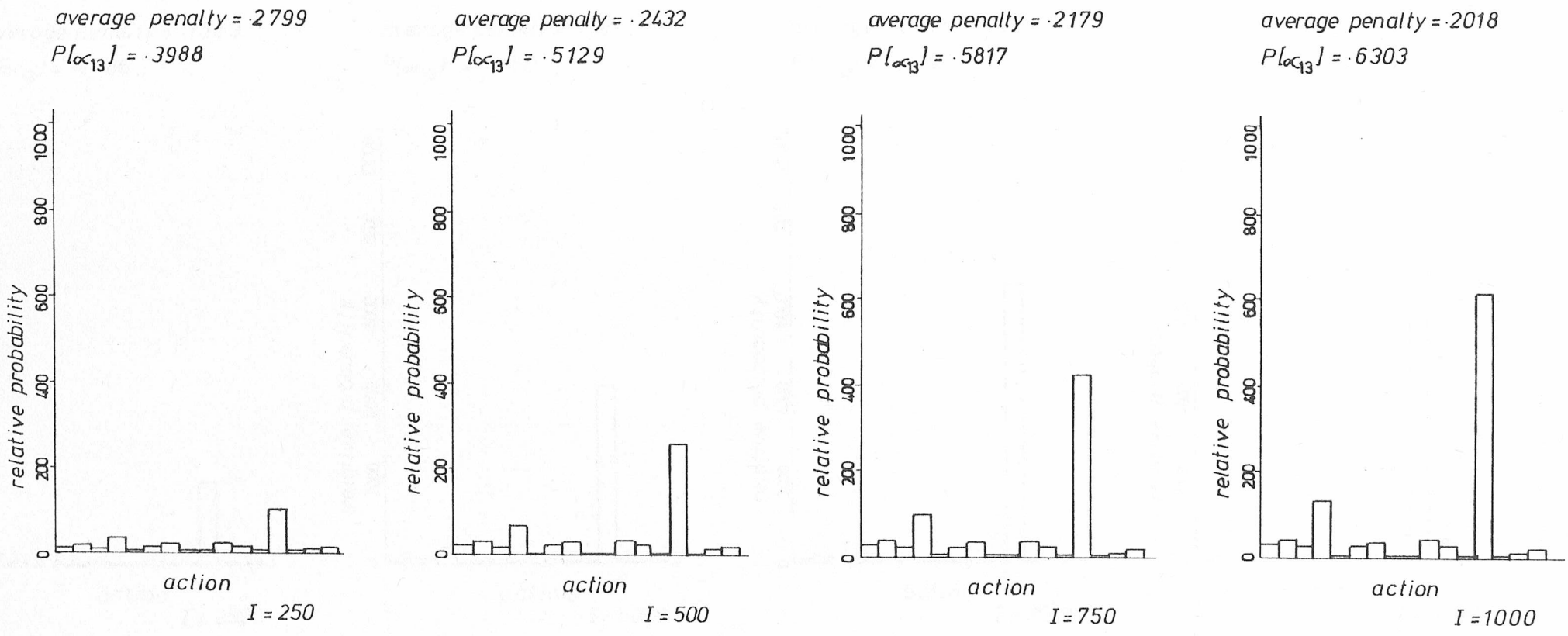

average penalty = ·2799
P[∝13] = ·3988
average penalty = ·2432
P[∝13] = ·5129
average penalty = ·2179
P[∝13] = ·5817
average penalty = ·2018
P[∝13] = ·6303
relative probability
0
200
400
600
800
1000
action
I = 250
I = 500
I = 750
I = 1000

Figure 6·4 (b)

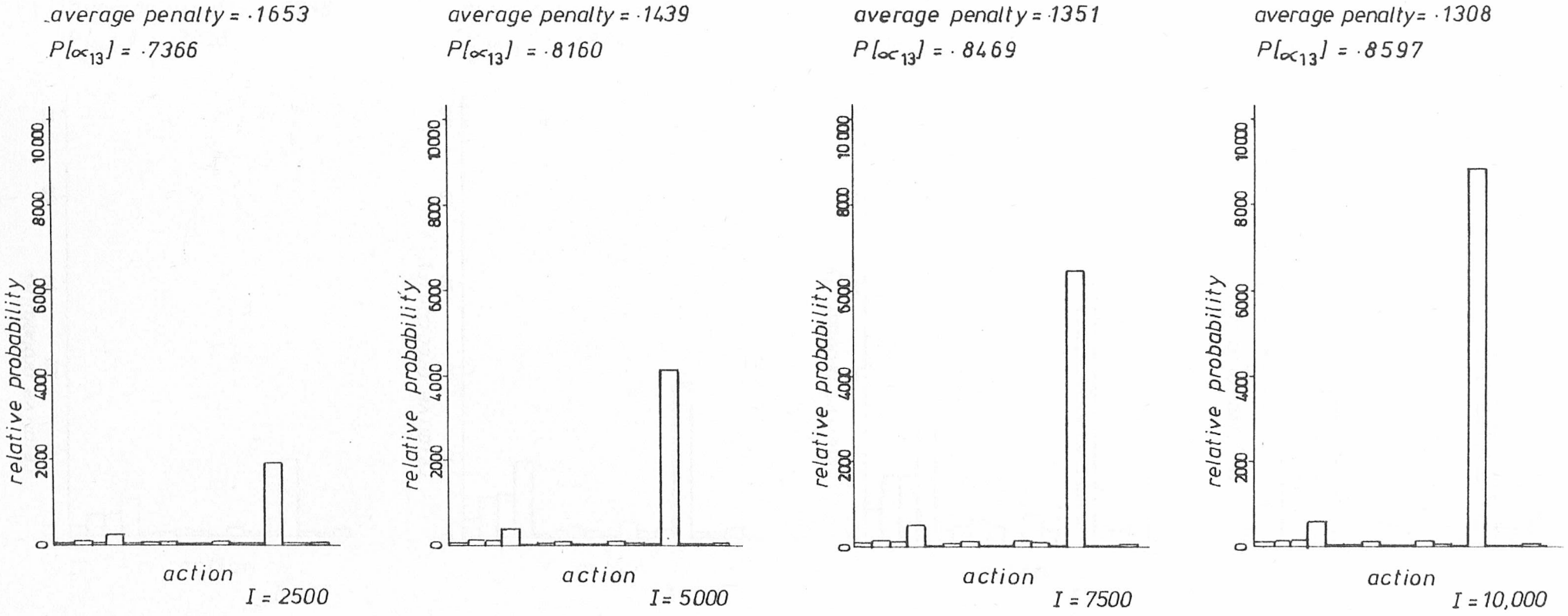

average penalty = ·1653
P[∝13] = ·7366
average penalty = ·1439
P[∝13] = ·8160
average penalty = ·1351
P[∝13] = ·8469
average penalty = ·1308
P[∝13] = ·8597
relative probability
0
2000
4000
6000
8000
10000
action
I = 2500
I = 5000
I = 7500
I = 10,000

Figure 6.4 (c)

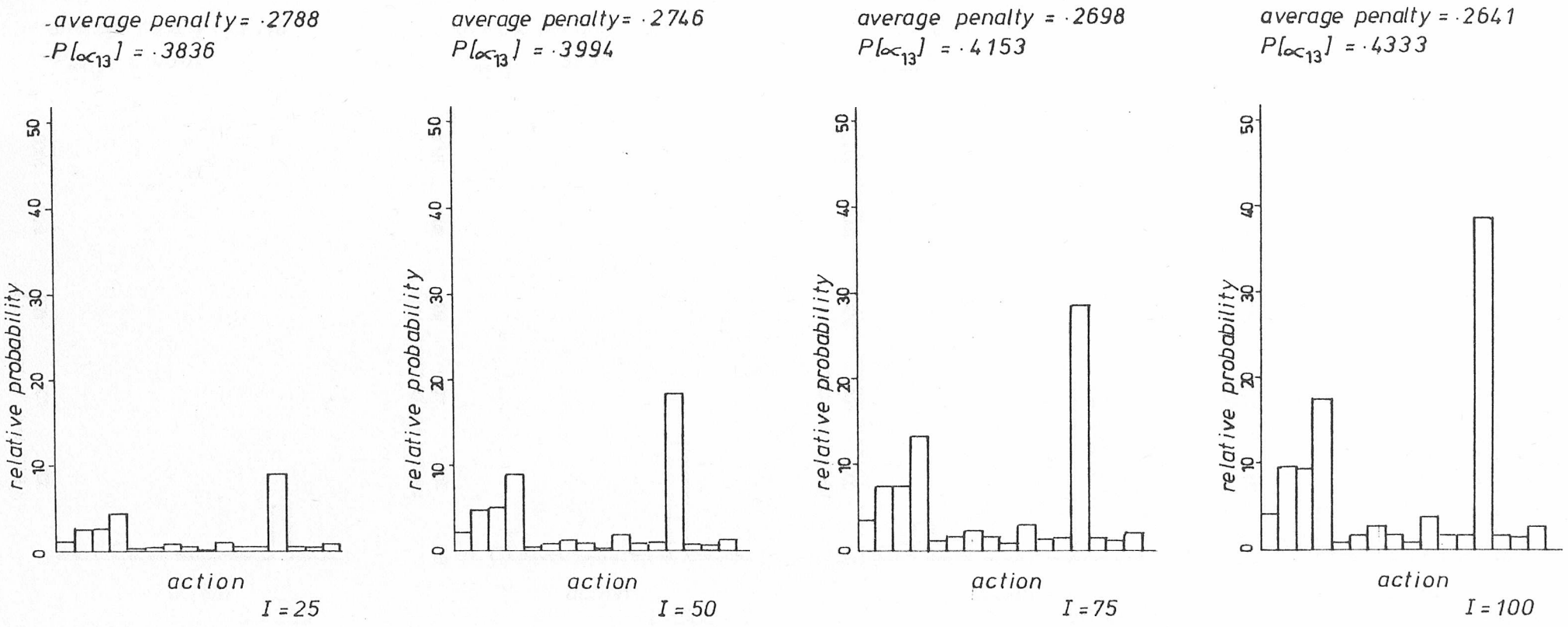


Figure 6.5 (a) L_{RP} automaton in hierarchical learning system

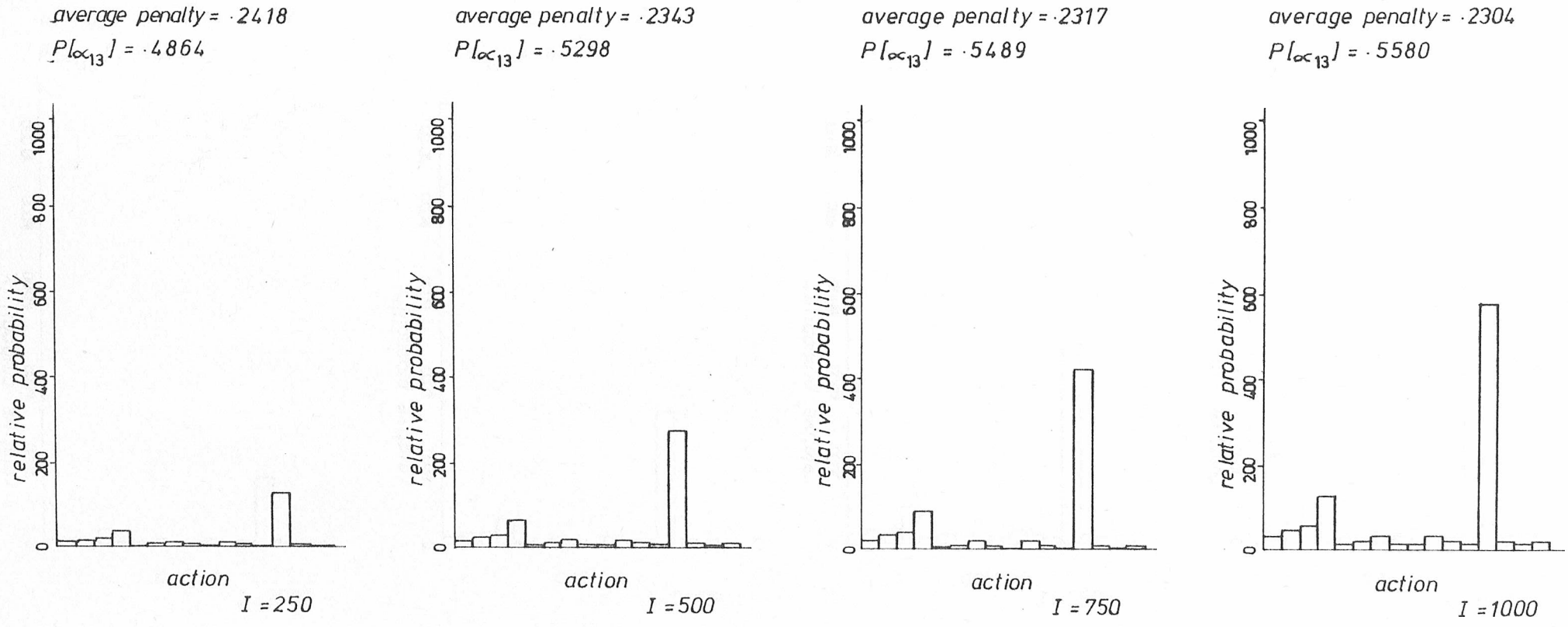

average penalty = ·2418
$P[\alpha_{13}] = \cdot 4864$
average penalty = ·2343
$P[\alpha_{13}] = \cdot 5298$
average penalty = ·2317
$P[\alpha_{13}] = \cdot 5489$
average penalty = ·2304
$P[\alpha_{13}] = \cdot 5580$
relative probability
0
200
400
600
800
1000
action
I = 250
I = 500
I = 750
I = 1000

Figure 6·5(b)

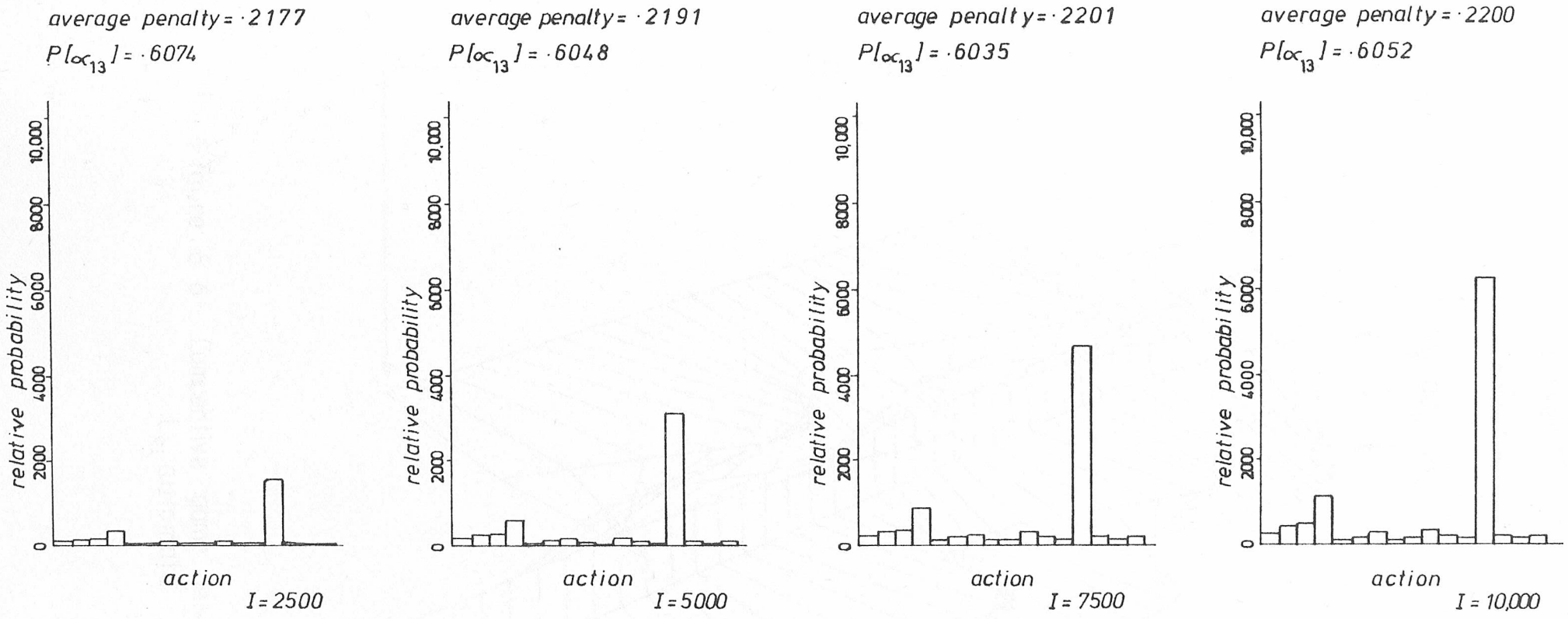

average penalty = ·2177
$P[\alpha_{13}] = \cdot 6074$
relative probability
0
2000
4000
6000
8000
10,000
action
I = 2500
average penalty = ·2191
$P[\alpha_{13}] = \cdot 6048$
relative probability
0
2000
4000
6000
8000
10,000
action
I = 5000
average penalty = ·2201
$P[\alpha_{13}] = \cdot 6035$
relative probability
0
2000
4000
6000
8000
10,000
action
I = 7500
average penalty = ·2200
$P[\alpha_{13}] = \cdot 6052$
relative probability
0
2000
4000
6000
8000
10,000
action
I = 10,000

Figure 6·5 (c)

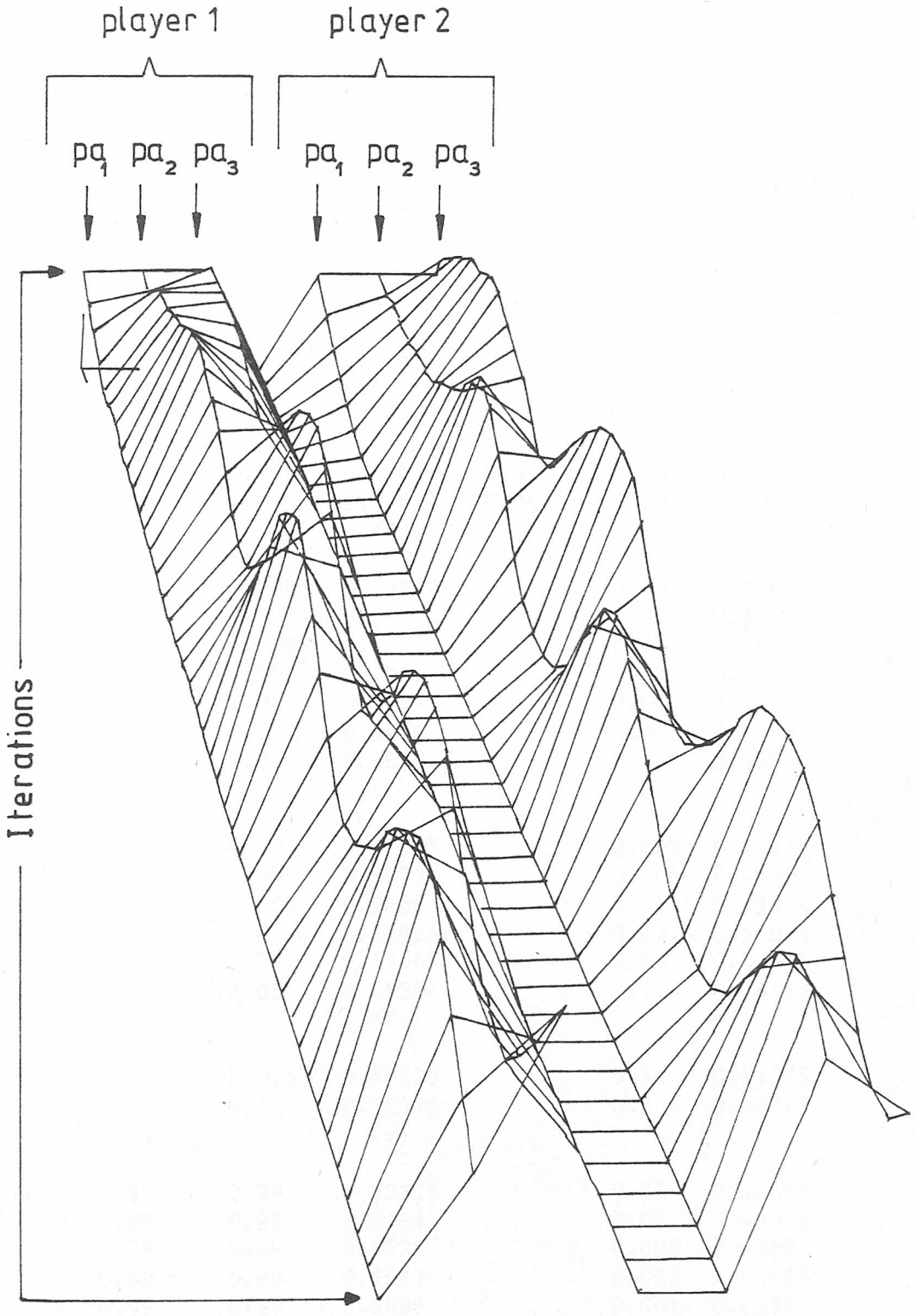


Figure 6.6 Competitive game between two L_{RI} automata

Players	Parameters player 1		Average penalty 1	Parameters player 2		Average penalty 2
LrivLri	0.999	1	0.50265	0.995	1	0.49735
	0.9995	1	0.504	0.999	1	0.496
			(a)			
LrpvLrp	0.99	0.99	0.4995	0.95	0.95	0.50055
	0.995	0.995	0.50005	0.99	0.99	0.49995
	0.999	0.999	0.5004	0.995	0.995	0.4996
	0.9995	0.9995	0.50085	0.999	0.999	0.49915
			(b)			
	0.999	0.9998	0.50105	0.998	0.9996	0.49895
	0.995	0.999	0.4996	0.99	0.998	0.5004
	0.99	0.998	0.49995	0.98	0.996	0.50005
	0.98	0.996	0.5	0.96	0.992	0.5
			(c)			
	0.995	0.999	0.4754	0.99	0.995	0.5246
	0.995	1.0	0.47825	0.995	0.999	0.52175
			(d)			
TrpvTrp		0.0005	0.4991		0.001	0.5009
		0.001	0.50055		0.002	0.49945
		0.002	0.49975		0.005	0.50015
		0.005	0.4961		0.01	0.5039
		0.01	0.501		0.02	0.499
		0.02	0.4956		0.05	0.5044
		0.05	0.4698		0.1	0.5302
			(e)			
TipvTip		0.001	0.50025		0.002	0.49975
		0.002	0.5003		0.005	0.4997
		0.005	0.50085		0.01	0.49915
		0.01	0.49925		0.02	0.50075
		0.02	0.47565		0.05	0.52435
		0.05	0.45584		0.1	0.54415
			(f)			
TrivTri		0.075	0.48525		0.15	0.51475
		0.1	0.50285		0.2	0.49715
			(g)			
LrpvTrp	0.99	0.99	0.52305		0.05	0.47695
	0.99	0.99	0.53645		0.01	0.46355
	0.99	0.99	0.5532		0.005	0.4468
	0.99	0.99	0.56335		0.002	0.43665
	0.99	0.99	0.5668		0.001	0.4332
			(h)			

Table 6.1
Results of competitive automata games
in 2 runs of 100000 iterations

	0.995	0.9975	0.5233	0.01	0.4767
	0.995	0.999	0.5021	0.01	0.4959
	0.995	0.9995	0.4924	0.01	0.5076
			(i)		
LrivTrp	0.995	1.0	0.48055	0.01	0.51945
	0.995	1.0	0.48795	0.005	0.51205
	0.995	1.0	0.4923	0.002	0.5077
	0.995	1.0	0.4936	0.001	0.5064
			(j)		
	0.999	1.0	0.4975	0.001	0.5025
	0.99	1.0	0.49455	0.001	0.50545
			(k)		
LrpvTip	0.99	0.99	0.48115	0.001	0.51885
	0.99	0.99	0.4816	0.002	0.5184
	0.99	0.99	0.48175	0.005	0.51825
	0.99	0.99	0.4819	0.01	0.5181
	0.99	0.99	0.48135	0.02	0.51865
	0.99	0.99	0.47825	0.05	0.52175
	0.99	0.99	0.4515	0.1	0.5485
			(l)		
LrpvTri	0.99	0.99	0.49825	0.1	0.50175
	0.995	0.999	0.4627	0.1	0.5373
			(m)		
LrivTri	0.995	1.0	0.4466	0.1	0.5534
			(n)		
TrpvTip		0.001	0.42935	0.001	0.57065
		0.01	0.44935	0.01	0.55065
		0.1	0.48025	0.1	0.51975
			(o)		
TrpvTri		0.1	0.5025	0.1	0.4975
		0.01	0.4649	0.1	0.5351
		0.01	0.3342	0.05	0.6658
			(p)		

Table 6.1
Results of competitive automata games
in two runs of 100000 iterations

CHAPTER 7 THE TSETLIN ALLOCATION SCHEME

Introduction

Tsetlin [41] has considered the operation of a queueing system and the effect of different priority systems. Tsetlin examined the case of subscribers requiring the use of a telephone channel. By using a system which gave priority to subscribers who made short calls, Tsetlin aimed to reduce the mean queue length and reduce the mean waiting time for the system. The system used learning automata to assign priorities to subscribers and was of interest as a practical application of learning automata. It required no a priori knowledge of the characteristics of the subscribers and was adaptive. The system was investigated using a computer simulation and was compared to a simulation of a first come, first served (f.c.f.s.) system which was used as a reference.

The Tsetlin Channel Allocation Scheme

The explanation of the Tsetlin allocation scheme which follows is presented in conjunction with Figure 7.1.

Subscribers in a system are the source of requests for the use of a channel. Before a subscriber is allowed the use of a channel the subscriber must have an automaton. As the subscribers make their requests they can either have an automaton assigned to them, in which case they are described as dominant or reserve subscribers, depending on the type of automaton they have, or have no automaton. There are two automata for every channel in the system and these are called the dominant and reserve automata. Each dominant automaton contains the identification of the subscriber it is assigned to, a queue for the

subscriber to wait in and the credit of the subscriber. Each reserve automaton is similar but without the queue for the subscriber.

A subscriber requiring a channel enters the system. If the subscriber is dominant on a channel, the subscriber is put onto that channel if it is free or is put into the queue in the dominant automaton until the channel becomes free. A subscriber who is not dominant is put onto the main queue if there are no free channels. If there are channels free, these are searched to see if the subscriber has a reserve automaton on any of them. If the subscriber has reserve automata on free channels, the subscriber uses the channel which corresponds to the automaton with the highest credit. If the subscriber has no reserve automata, the subscriber is assigned a reserve automaton on the free channel which has the least credit in its reserve automaton.

When a channel becomes free a dominant subscriber waiting in the dominant subscriber queue has first priority. If the dominant subscriber is not waiting the second priority goes to the reserve subscriber who may be waiting in the main queue. If the reserve subscriber is not waiting a subscriber is taken from the main queue on a first come first served basis, the subscriber is allocated the reserve automaton on that channel and the credit is set to zero.

When any subscriber starts to use a channel the automaton associated with the subscriber on that channel is given a constant amount of credit. When a subscriber ends the use of a channel the credit is reduced by an amount dependant on the length of time the channel has been used. In the results this is expressed as a credit gain/loss per second the channel is used less/longer than a threshold value. A subscribers credit is limited by the automaton to a maximum

amount and cannot fall below zero.

A dominant subscriber can have only one dominant automaton and no reserve automata but a reserve subscriber can have more than one reserve automaton. A reserve subscriber can become a dominant subscriber by being allocated a channel on which the subscriber has a reserve automaton. The reserve and dominant automata compete and the automaton with the largest credit becomes the dominant automaton and the subscriber becomes the dominant subscriber.

The Tsetlin Channel Allocation Scheme-Results

The allocation scheme described above was used in a computer simulation [43] with the facility for up to five channels and thirty subscribers. A number of results were taken over a range of subscriber and automata parameters. The time between the end of a call and the start of the next call and the duration of call for the subscribers were exponentially distributed. The results given below were taken over a long simulation time so that the results would be well averaged. Where results are compared directly the same random seed was used for the simulations so that the simulations were operating with the same inputs.

Table 7.1 gives results for individual subscribers for simulations over 200,000 time intervals or approximately 450,000 calls in systems with 5 subscribers using 2 channels. In simulation (a) the mean time between calls for all subscribers was made equal so that the effect of call length could be observed. Subscribers with short call lengths have the highest probability of being dominant and have low probabilities of being reserve automata while the reverse is true of subscribers with long call lengths. In simulation (b) all the subscribers have the same mean length of call so they would each tend

to gain the same amount of credit from their calls. The mean time between calls differs so that the subscribers have differing frequency of calls. The results show that subscribers who make calls frequently have a greater probability of being dominant. To become dominant a subscriber must first build up credit in a reserve automaton and then return to the reserve automaton on which the subscriber has credit. A subscriber making calls frequently will have more reserve automata, will be more likely to return to an automaton before it is assigned to another subscriber and so will build up credit. A subscriber making calls frequently is more likely to be assigned to new reserve automata and so destroy the credit of other subscribers.

Table 7.2 gives results for simulations over 10,000 time intervals corresponding to approximately 160,000 calls from 15 subscribers using 2 channels. The subscriber parameters in this simulation have a constant ratio between the mean call length and the mean time between calls. The subscribers with the shortest call lengths become dominant whilst amongst the other subscribers those with the shortest call lengths and greatest frequency are most likely to have reserve automata. The most important result in Tables 7.1 and 7.2 can be seen when the mean waiting times are compared to those for the f.c.f.s scheme. This shows that subscribers which are dominant have mean waiting times longer than the reference while it is the performance of the reserve automata which increases.

Table 7.3 gives overall results for 7 simulations. For a system performing well the number of events in the simulation will be high, the mean number in the system will be low and the mean waiting time will be low. For identical inputs the most efficient system will have more channels free but in this case because the number of calls and

their distribution amongst the subscribers varies it is difficult to equate this with system efficiency. The seven sets of results have differing loads moving from the most heavily loaded (a) to the least loaded (g). These results show that it is in the most heavily loaded systems that the Tsetlin allocation scheme gives an improved performance. As the loading on the system falls so does the performance of the Tsetlin scheme with respect to the reference until the load becomes about 90% of the total capacity when the performance of the f.c.f.s. scheme becomes best.

The aim of the Tsetlin allocation system was to reduce the mean waiting time of a system by introducing a system of priorities which would favor subscribers with short mean call lengths. The Tsetlin allocation scheme has been shown to do this only in heavily loaded systems. In the other cases the performance of subscribers with priority is decreased. This is because dominant subscribers are limited to use the channel on which they are dominant. If a reserve subscriber is using the channel the dominant subscriber must wait. A subscriber without a dominant automaton is not constrained to use a particular channel and is free to use channels as they become available. It is only in highly loaded systems that a dominant subscriber with priority on a particular channel is at an advantage over the other subscribers with no priority but free to use any channel.

The Modified Tsetlin Allocation Scheme

The Tsetlin allocation scheme has a poor performance because dominant subscribers are limited to a particular channel. The modified Tsetlin scheme allows dominant subscribers to use any channel with priority over reserve and other subscribers. When more than one

dominant subscriber requires a channel the one with the greatest credit takes priority. When a reserve subscriber competes for a dominant automaton the competition is with the dominant subscriber with the least credit. A dominant subscriber may be using a channel when the competition occurs but completes the call as normal. The number of dominant subscribers allowed is equal to the number of channels in the system.

The Modified Tsetlin Allocation Scheme-Results

Table 7.4 gives results for the Tsetlin scheme, the f.c.f.s. scheme and the modified Tsetlin scheme for six different simulations producing lightly and highly loaded systems. In all cases the modified Tsetlin scheme allows a greater number of calls to be made, has fewer calls waiting in the system and has the lowest mean waiting time. The subscribers with the lowest call length become dominant as in the Tsetlin scheme but unlike the Tsetlin scheme the performance of dominant subscribers improves whatever the loading of the system. In the modified Tsetlin scheme the number of short calls from the dominant subscribers increases while the number of long calls is reduced. The increase in the number of short calls increases the number of events in the simulation. Because a subscriber is now more likely to be held up by a short call than a long call, the overall waiting time is reduced. In addition, the replacement of a long call by a number of short ones of equivalent length makes the system more easy to run efficiently. However in some cases the f.c.f.s scheme has fewer free channels indicating that this scheme is allowing more of the channel capacity to be used by having more calls from subscribers who produce long calls.

Further Improvements to Tsetlin's Allocation Scheme

The priority system of the modified Tsetlin allocation scheme divides the subscribers into three classes, the dominant subscribers, the reserve subscribers and the others. Dominant subscribers have top priority on all channels and the dominant subscribers are themselves graded, giving greater priority to subscribers with most credit. Reserve subscribers have priority over subscribers with no automaton but only on the channel which corresponds to their reserve automaton otherwise they are treated like subscribers with no automaton. Further improvements in performance could be gained by extending the priority system. The reserve subscribers could be given priority on all channels and graded like the dominant subscribers. A further step would be to extend the priority scheme to all subscribers by grading them all. This would involve giving all subscribers an automaton which would measure the subscribers credit. The distinction between reserve and dominant automata would be removed and the priority would simply depend on the credit in the subscribers automaton.

Tsetlin's credit scheme is not a very effective method for determining the priority of subscribers on the basis of call length. Subscribers with mean call lengths greater than the threshold value will tend to lose credit while the rest will tend to gain credit. Subscribers who tend to lose credit will all tend to have credits of zero while subscribers who gain credit will all tend to have maximum credit. Thus the Tsetlin scheme tends to split the subscribers into two groups and is not suitable for giving each subscriber an individual priority. The Tsetlin scheme also requires the use of a threshold value, the value of which affects the operation of the scheme and so it is not a true a priori system. It would be more

effective to measure the call lengths of subscribers and base a priority system on this using the methods of the modified estimating automaton [24,25]. However there are automata better than the modified estimating which could be used in an allocation scheme.

In this way it was decided that the next step in the allocation scheme would not be based on Tsetlin's scheme. It would give individual priorities for each subscriber provided by a learning automaton based on the call lengths of the individual subscribers. Of the automata which had been investigated the Lrp, Trp and Tip had the best performance and so these were included in the new scheme.

Automaton Allocation Scheme-Operation

The automaton allocation scheme was simulated in the same way as the Tsetlin allocation scheme. If when a subscriber enters the system there is a channel free the automaton is not involved and the channel is allocated to the subscriber. If there are no channels free the subscriber waits in a queue. If when a channel becomes free there are two or more subscribers waiting in the main queue the automaton selects a subscriber from the queue who will be allocated the channel. The action probabilities of the automaton represent priorities for the subscribers. Though the sum of the action probabilities is unity, the automaton cannot be allowed to select from the full range of its actions since not all subscribers will be waiting in the queue. Thus only the action probabilities of the subscribers waiting in the queue are taken and modified to sum to one so that the automaton will only select one of the subscribers waiting in the queue.

When a call ends the length of the call is fed back to the automaton. A penalty/reward signal was required by the automaton with long call lengths corresponding to a high penalty probability. The equation

$$c_i = 1-1/((\text{scale}*\text{call length})^{1/N}) \qquad (7.1)$$

was used to convert call lengths into probabilities. The scale factor was chosen so that few calls would be shorter than 1/scale and if this did occur the penalty probability was set to 0. The root factor N was included to separate long call times. Figure 7.2 shows the characteristics of equation (7.1) in converting call lengths into penalty probabilities using the values used in the simulation compared to the characteristics with N=1. The root factor has the effect of producing a less steeply rising characteristic as well as moving the penalty probabilities nearer to the centre of their range.

Automaton Allocation Scheme-Results

Simulations were made using the automaton allocation scheme with the same subscriber and channel parameters as used previously for the modified Tsetlin scheme.

During the simulations the Tip automaton was found to be performing poorly. Table 7.5 (c) gives results for a simulation using the Tip automaton. Comparing results with similar results using Lrp automata as given in Table 7.5 (a) and (b) the Tip automaton has fewer events and longer waiting times. However the results for the action probabilities was of most interest as these indicated that the automaton was trying to select the subscriber with the longest mean call length most often rather than the subscriber with the shortest call length.

The analysis of the Tip automaton in a two action environment showed that the automaton will reach steady state when

penalties from action 1 = penalties from action 2 (5.17)

The action probabilities for the Tip automaton in Table 7.6 (c) can be explained as an attempt by the automaton to satisfy this condition for all subscribers. Since the mean time between calls was different for each subscriber the arrival rates were different. However since the penalty probabilities were fixed by the mean call length the only way the automaton could satisfy the condition was by trying to change the frequency of calls from the different subscribers. The action of the automaton was to slow calls from subscribers who made calls frequently and attempt to increase the frequency of calls from subscribers who make calls infrequently. It did this by having a high action probability for subscriber 5 who had a long mean time between calls but also a long mean call length. This gave subscriber 5 a good performance but this resulted in a reduced overall performance and poorer performances for the other subscribers.

The Lrp with $\alpha = \beta$ and the Tip automata satisfy the same steady state conditions and so the action probability results for the Lrp automaton could be expected to show the same effect as for the Tip automaton. Though the action probabilities in Table 7.5 (b) are higher for infrequent subscribers than the corresponding results for the Lri automaton the difference is far less than the Tip results. The results in Table 7.5 (b) and (d) do not satisfy the condition (5.17) indicating that the Lrp automaton is not operating as expected.

A conventional analysis of the Lri automaton shows that the automaton operates to equalise the penalty probabilities of the environment. If this is not possible, as in an autonomous environment, the automaton selects the action corresponding to minimum average penalty probability with a high probability. As shown in Table 7.5 (a) the Lri does not equalise the penalty probabilities or select the action corresponding to the minimum penalty probability with a high probability.

The anomalies discribed above were the result of a common cause. Normal analysis assumes that every time an automaton selects an action the selection is made between every action and that the feedback is applied to every action probability. In the channel allocation scheme the selection was made between only the subscribers waiting in the queue and if only one subscriber was waiting the automaton was not involved. However the updating was appled to every action every time a call ended. Because of this the normal analysis does not apply and the automata will not operate as expected.

Once the cause of the unusual results had been determined further results were taken with the exclusion of the Tip automaton. Table 7.6 gives results for the Lrp and Trp automata corresponding to the simulations in Table 7.4. In results (a) and (b) the modified Tsetlin scheme produces a better performance by having a distinct priority for subscribers 1 and 2 compared to the less defined priority of the automata schemes. In (c) the Trp automaton produces the best performance by giving more priority to subscriber 1 than the Lrp. In results (d),(e) and (f) the number of subscribers is increased to 15 and the range of mean time between calls and mean call lengths is much greater. In all these results the Lrp automaton has the best

performance. The performance of the Trp automaton is good for subscriber 1 but is degraded because the automaton gives a relatively high priority to subscribers 12-15 who make calls infrequently but have long call lengths. Because these subscribers are selected by the automaton relatively frequently the performance of the other subscribers falls as does the overall performance.

In Chapter 5 the operation of the Trp automaton was described as a mixture of Tri and Tip automata. However the Tip when operating in this simulation tended to choose the subscriber with the lowest frequency of calls. Since the Trp is a mixture of the Tri and Tip automata the behaviour of the Trp in the results above can be explained as the character of the Tip automaton showing through.

Conclusions

Investigation into the Tsetlin allocation scheme has shown that in most cases the system does not operate as intended and give priority to subscribers with short call lengths. Instead, except at very high loadings, the performance of dominant subscribers who should have priority is reduced.

Having discovered the shortcomings of the Tsetlin scheme the modified Tsetlin scheme was developed to operate as Tsetlin intended his scheme to operate. This modification was sucessful and provided a better performance for all loadings.

As a further development each subscriber in the system was given an individual priority using learning automata. When the results were not as expected this was found to be due to the unusual selection and updating procedures required in the system. This resulted in a distortion to the automata algorithms so changing the characteristics of the automata. Despite this the Lrp and Trp automata were able to

produce good performances. These simulations highlighted an aspect of the use of automata which had not been considered before.

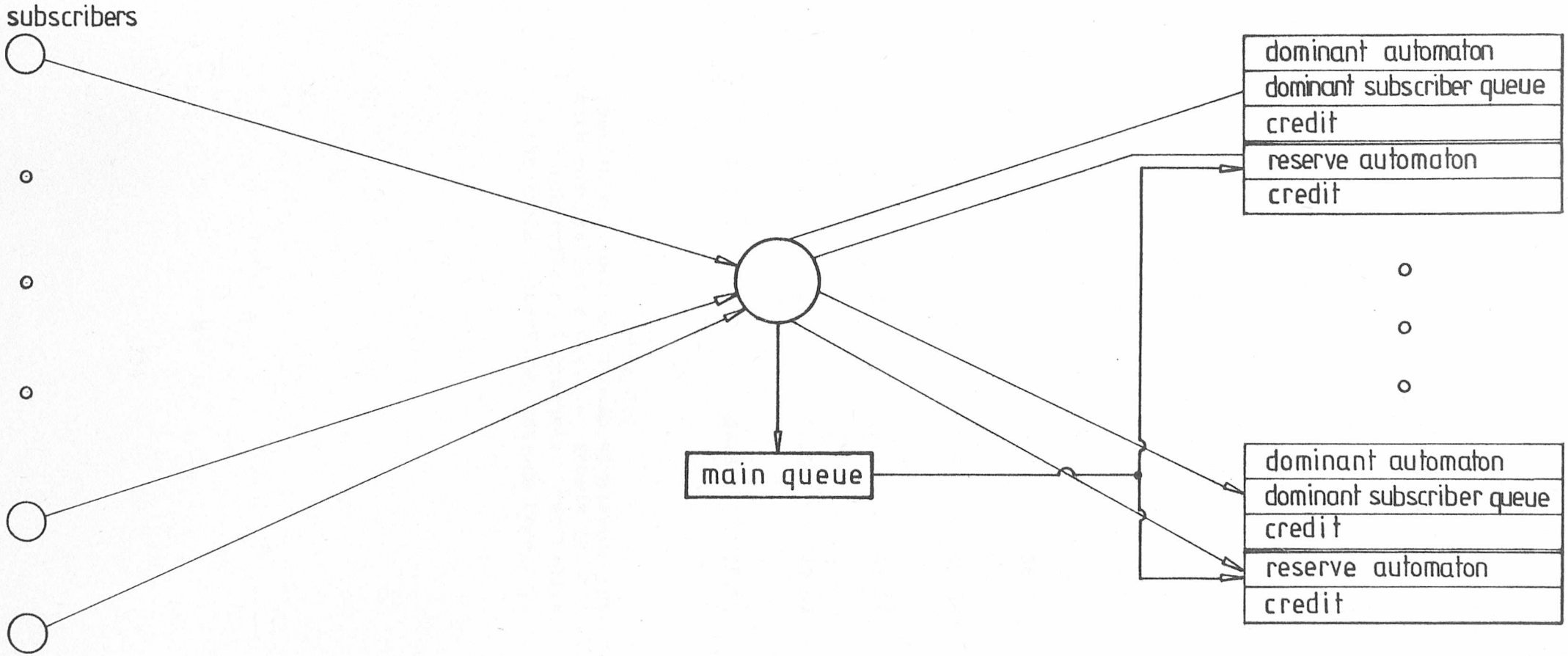


Figure 7.1 The Tsetlin allocation scheme

Subscriber	Mean length of call (seconds)	Mean time between calls (seconds)	Percentage time as dominant subscriber	Percentage time as reserve subscriber	Mean waiting time
1	0.1	2.0	99.98	0.0	0.0971 (0.0211)
2	0.2	2.0	99.23	0.53	0.0941 (0.0179)
3	0.3	2.0	0.72	66.31	0.0135 (0.0141)
4	0.4	2.0	0.05	66.34	0.0097 (0.0109)
5	0.5	2.0	0.0	66.82	0.0082 (0.0089)

(a)

1	0.3	1.0	59.90	36.36	0.0351 (0.0138)
2	0.3	1.5	47.94	40.98	0.0366 (0.0171)
3	0.3	2.0	35.99	43.81	0.0344 (0.0197)
4	0.3	2.5	31.51	40.57	0.0350 (0.0213)
5	0.3	3.0	24.66	38.28	0.0342 (0.0223)

(b)

Table 7.1
Tsetlin's allocation scheme simulation results
with results for a f.c.f.s. scheme in brackets
5 subscribers, 2 channels, credit=30/s
threshold value=0.3s, maximum credit=91

Subscriber	Mean length of call (seconds)	Mean time between calls (seconds)	Percentage time as dominant subscriber	Percentage time as reserve subscriber	Mean waiting time
1	0.01	0.1	99.93	0.02	0.0637 (0.0369)
2	0.02	0.2	99.51	0.18	0.0942 (0.0514)
3	0.05	0.5	0.10	27.39	0.0694 (0.0727)
4	0.06	0.6	0.13	24.68	0.0736 (0.0754)
5	0.07	0.7	0.07	22.23	0.0763 (0.0787)
6	0.09	0.9	0.02	18.90	0.0784 (0.0831)
7	0.1	1.0	0.04	17.81	0.0817 (0.0862)
8	0.15	1.5	0.06	14.83	0.0833 (0.0882)
9	0.2	2.0	0.00	13.22	0.0898 (0.0976)
10	0.3	3.0	0.01	11.78	0.0952 (0.0958)
11	0.4	4.0	0.00	11.04	0.0896 (0.0965)
12	0.6	6.0	0.00	10.06	0.1006 (0.0875)
13	0.8	8.0	0.00	9.90	0.0846 (0.0907)
14	1.0	10.0	0.06	9.33	0.0889 (0.0847)
15	2.0	20.0	0.07	8.63	0.0814 (0.0736)

Table 7.2
Tsetlin's allocation scheme simulation results
with results for a f.c.f.s. scheme in brackets
15 subscribers, 2 channels, credit=400/s
threshold value=0.03s, maximum credit=61

Number of subscribers	Number of channels	Number of events	Mean number in system	Mean waiting time (seconds)	Mean number of free channels
15	2	677393 (606617) (a)	11.6189 (11.9706)	1.4189 (1.6426)	0.0003 (0.0002)
15	2	630252 (609376) (b)	8.6998 (8.9062)	1.0655 (1.1360)	0.0212 (0.0214)
15	2	611733 (602417) (c)	7.6587 (7.7707)	0.9326 (0.9657)	0.0512 (0.0520)
15	2	591912 (587132) (d)	6.7098 (6.7770)	0.8112 (0.8291)	0.0966 (0.0957)
15	2	569826 (568326) (e)	5.8811 (5.9053)	0.7074 (0.7133)	0.1539 (0.1527)
15	2	544797 (546990) (f)	5.1866 (5.1496)	0.6240 (0.6148)	0.2163 (0.2167)
15	2	519514 (523832) (g)	4.5989 (4.5117)	0.5545 (0.5330)	0.2850 (0.2837)

Table 7.3
Tsetlin's allocation scheme simulation results
with results for a f.c.f.s. scheme in brackets
(a) credit=100/s, threshold value=0.006s, maximum credit=61
(b)-(g) credit=400/s, threshold value=0.003s, maximum credit=61

Number of subscribers	Number of channels	Number of events	Mean number in system	Mean waiting time (seconds)	Mean number of free channels
5	2	316840	4.2080	1.4175	0.0390
		350071	4.1253	1.2254	0.0211
		396468	4.0101	1.0243	0.0221
		(a)			
5	2	241392	2.5845	0.8586	0.4526
		271739	2.2826	0.5093	0.4103
		274985	2.2496	0.4797	0.4107
		(b)			
5	2	426831	0.7274	0.0461	1.3712
		433043	0.6655	0.0149	1.3670
		433102	0.6649	0.0146	1.3671
		(c)			
15	2	884245	12.7990	2.4402	0.0000
		762986	13.1004	2.9075	0.0000
		1020868	12.4581	2.0465	0.0000
		(d)			
15	2	519514	4.5989	0.5545	0.2850
		523832	4.5117	0.5330	0.2837
		533851	4.3138	0.4860	0.2843
		(e)			
15	2	167657	2.5190	0.0763	0.7613
		182862	2.3459	0.0590	0.7332
		188759	2.3013	0.0545	0.7276
		(f)			

Table 7.4
Comparason of results for
Tsetlin's allocation scheme
f.c.f.s. scheme
modified Tsetlin scheme

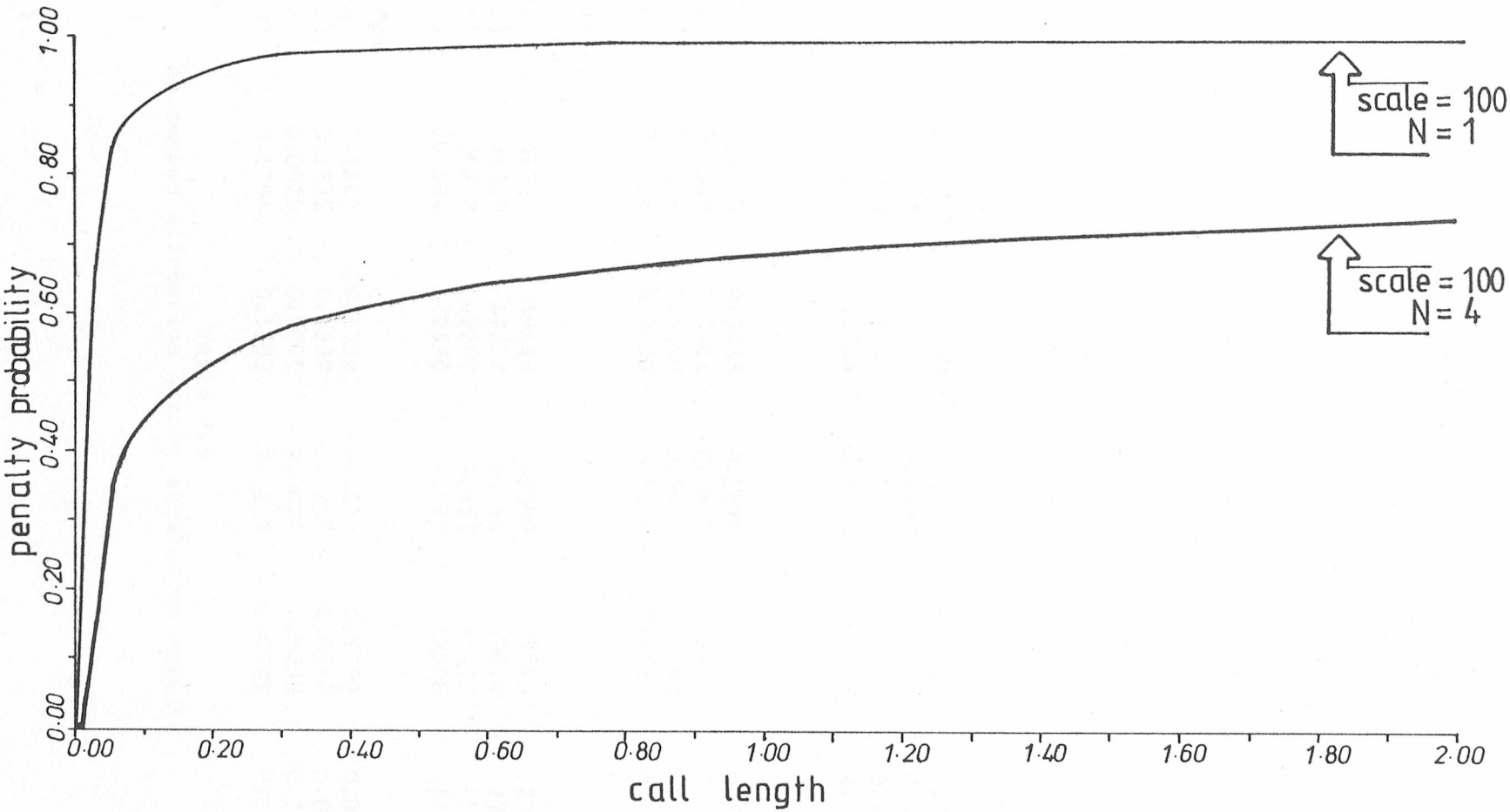


Figure 7.2 Penalty probability characterics for the automaton allocation scheme

Automaton	Automaton parameters	Number of events	Mean number in system	Mean waiting time (seconds)	Mean number of free channels
(a) Lrp	0.9 1.0	378772	4.0553	1.0960	0.0217
(b) Lrp	0.95 0.95	365111	4.0873	1.1542	0.0214
(c) Tip	0.0005	332915	4.1687	1.3141	0.0203
(d) Lrp	0.95 0.95	558211	3.1565	0.4708	0.1603

Mean call length (seconds)	Subscriber 1	Subscriber 2	Subscriber 3	Subscriber 4	Subscriber 5
(a,b,c,d)	0.2	0.5	0.6	2.0	5.0
Mean time between calls (seconds)					
(a,b,c)	0.5	0.5	0.5	0.5	0.5
(d)	0.2	0.5	0.6	2.0	0.5
Action probability					
(a)	0.3700	0.2453	0.2260	0.1120	0.0467
(b)	0.3159	0.2269	0.2150	0.1428	0.0994
(c)	0.0059	0.0046	0.0042	0.0048	0.9805
(d)	0.3913	0.2204	0.1979	0.1079	0.0825
Penalty probability					
(a)	0.4339	0.5435	0.5638	0.6777	0.7422
(b)	0.4343	0.5461	0.5618	0.6751	0.7418
(c)	0.4376	0.5459	0.5674	0.6783	0.7391
(d)	0.4356	0.5459	0.5654	0.6761	0.7391
Number of penalties					
(a)	47725	49448	49399	39890	23211
(b)	44449	47377	46966	40674	23856
(c)	36978	42207	42853	41412	25430
(d)	107524	72480	72480	29506	14052
ci*pi					
(a)	0.1605	0.1333	0.1274	0.0759	0.0347
(b)	0.1372	0.1239	0.1208	0.0964	0.0737
(c)	0.0026	0.0025	0.0024	0.0033	0.7224
(d)	0.1704	0.1203	0.1119	0.0729	0.0609

Table 7.5
Automaton allocation scheme simulation results

Number of subscribers	Number of channels	Number of events	Mean number in system	Mean waiting time (seconds)	Mean number of free channels
5	2	350071	4.1253	1.2254	0.0211
		396468	4.0101	1.0243	0.0221
		378722	4.0553	1.0960	0.0217
		389099	4.0266	1.0520	0.0219
		(a)			
5	2	271739	2.2826	0.5093	0.4103
		274985	2.2496	0.4797	0.4107
		272324	2.2725	0.5026	0.4128
		274546	2.2503	0.4819	0.4121
		(b)			
5	2	433043	0.6655	0.0149	1.367
		433102	0.6649	0.0146	1.3671
		432732	0.6629	0.0147	1.3691
		433174	0.6635	0.0146	1.3686
		(c)			
15	2	762986	13.1004	2.9075	0.0000
		1020868	12.4581	2.0465	0.0000
		1281417	11.8092	1.5288	0.0000
		1016796	12.4687	2.0568	0.0000
		(d)			
15	2	523832	4.5117	0.5330	0.2837
		533851	4.3138	0.4860	0.2843
		539747	4.1892	0.4583	0.2876
		535934	4.2614	0.4745	0.2847
		(e)			
15	2	182862	2.3459	0.0590	0.7332
		188759	2.3013	0.0545	0.7276
		188963	2.2779	0.0527	0.7189
		188369	2.3226	0.0551	0.7189
		(f)			

Table 7.6
Automaton allocation scheme results for
f.c.f.s. scheme
modified Tsetlin scheme
Lrp α =0.9 β=1
Trp step size=0.0005

CHAPTER 8 JOB ALLOCATION IN A MULTIPROCESSOR SYSTEM

Introduction

In a single processor computer system, users of the system are sources of jobs which require the use of the processor. Since the processor can only carry out the tasks associated with one job at a time, jobs must be queued if more than one job is in the system. There are a variety of queueing systems used to determine which job is allowed the use of the processor e.g. round-robin, where each job in turn is allocated a set amount of processing time, batch, where each job is allocated the processor until the job is completed and priority schemes where the processor is allocated according to a priority system based on the amount of processing time a job has already received [42]. Some queueing systems favour short jobs and ensure that they have short waiting times while others are more favourable to jobs with long processing times. In either case the amount of processor time available is limited and only the distribution of the processing capacity amongst the jobs can be changed. A single processor system is similar to the system examined in Chapter 7 in that it has a limited capacity resource being allocated in a variety of possible ways amongst a number of users.

In a computer system with multiple processors there is more flexibility in that the jobs can be allocated to different processors with the aim of obtaining the best service. In a system where the speed of each processor is known as well as the queue length, at each processor a fixed scheduling discipline can be used to calculate the processor with the least waiting time. However if the parameters of

the system are not known, the fixed scheduling discipline cannot be used and if the parameters change with time the performance of the fixed scheduling discipline can be surpassed. Colon-Osorio [44] investigated the operation of the fixed scheduling discipline in a multiprocessor system by simulation and compared the performance with an adaptive scheme using Lrp automata. A similar investigation was carried out but with the addition of the Trp, Tip and Tri automata.

Multiprocessor System Simulation

The multiprocessor system simulation, illustrated in Figure 8.1, had provision for up to 5 processors with individual processing rates and up to 30 sources of jobs with individual exponentially-distributed processing requirement and time between job arrivals. The allocation scheme could either be the fixed scheduling discipline or an automaton scheme. The automaton scheme used an automaton at each source to allocate the jobs to the processors. The automaton at any source could be any of the types Lrp, Trp, Tip or Tri. This simulation provided a system which had not been investigated before. The environment was non-autonomous but the number of automata operating in the environment was equal to the number of job sources and the automata were operating in a games situation in that the actions of one automaton could affect the other automata via the penalty probabilities of the processors. The penalty probabilities were determined using the method used by Colon-Osorio i.e. a penalty was received if the processor chosen by the automaton was busy otherwise a reward was received.

Multiprocessor System Simulation-Identification of Environment

The initial measurements obtained from the multiprocessor simulation showed how the penalty probabilities varied with respect to the action probabilities. This was done by running the simulation with a variety of fixed action probabilities. Results were obtained for a system with a single source of jobs and two processors with the system loaded to 0.357 of capacity and are shown in Figure 8.2. The second set of results were obtained for a larger system with 5 sources and 2 processors with the system loaded to 0.7 of capacity. These results are given in Figure 8.3 and with Figure 8.2 confirm that the system represents a non-autonomous environment with the penalty probabilities linearly related to the action probabilities. Also included in these figures are the average penalty and mean turnaround time results. It should be noted that these measures of performance do not have their minima at the same action probability and so an automaton achieving the minimum average penalty would not minimise the mean turnaround time which is the measure of performance for the system.

Multiprocessor System Simulation-Automaton Steady State Conditions

Included in Figures 8.2 and 8.3 are the results of a number of simulations using a variety of automata. The results indicate the average action probability during the simulation. Figure 8.2 gives results for a single processor operating over 50000 iterations and includes the average penalty received by the automaton. It can be seen that the Lrp automaton with $\theta = 1$ and the Tip automaton converge close to the point where $c_1\ pa_1 = c_2\ pa_2$ as expected from equation (4.29) and (5.17). Also as expected from equation (4.22) two Lri automata converge to the action probability where $c_1 = c_2$ and a Lrp

automaton with $\alpha \neq \beta$ converges between the Lri and Lrp automata. The most unusual results are three for Lri automata with α=0.9 which have action probabilities less than 0.1. Because α is so low these automata have large step sizes and so converge quickly. What has happened is that the automata have gone optimal and converged to selecting a single action before the system reached steady state.

Table 8.1 gives results for a variety of automata operating in the environment shown in Figure 8.3. In Table 8.1 there were 5 job sources so the results are the average over five automata. Again the Lrp automaton with θ =1 and the Tip automata converge near the same action probability, the difference in the mean turnaround times being due to the different learning times for the automata, as shown by the result for the slowest automata, the Tip with the step size of 0.001. The Lri automata achieve a better performance in terms of mean turnaround time since the point where $c_1 = c_2$ is closer to the optimum action probability for the system.

None of the results for these automata achieve a mean turnaround time as low as that achieved by the fixed scheduling discipline. Even the minimum mean turnaround time given in Figure 8.3 is far larger than the fixed scheduling discipline result. This is because the scheduling discipline is a deterministic rule while the automata implement a stochastic rule. Because of variance, an automaton may allocate a series of jobs to a single processor while a second processor may be free. The fixed scheduling discipline with its up to date information of the queue lengths would easily avoid this. Thus provided the fixed scheduling discipline has accurate information about the system its performance can never be matched by an automaton scheme.

Colon-Osorio saw the use of an automaton system being an advantage in situations where the parameters of the system changed so that the performance of the fixed scheduling discipline was decreased allowing an adaptive system scope to provide a better performance.

Table 8.2 gives results of simulations using system parameters used by Colon-Osorio where the processor speeds are switched though the systems loading remains constant at 0.7. This degrades the performance of the fixed scheduling discipline while the automata schemes should be able to adapt to achieve the same steady state performance before and after the switch.

Table 8.2 shows that the performance of the Lrp automata with $\delta = 1$ is poor. Throughout the simulation the processor queues are growing longer resulting in long turnaround times. The Lri automata have a good performance before the switch but a poor performance after. This is because the automata are slow to switch and because any automata which have gone optimal will be unable to switch. A better performance is produced by the Lrp automata in the third simulation result as these cannot go optimal and the result after the switch is only poorer because of the delay in the automata responding to the switch in the environment.

The results for the Trp automaton show that the automata produce a reasonable performance prior to the switch but afterwards the performance is poor with the processor queues growing longer. This illustrates an aspect of the operation of the Trp automaton. When the simulation is started the system is empty and the penalty probabilities are low. With low penalty probabilities the operation of the Trp automaton is like that of the Tri automaton and this

automaton produces good results as Table 8.2 shows. When the environment switches the penalty probabilities rise and the automata have to adapt. However the operation of the Trp automaton with high penalty probabilities is like that of the Tip automaton and the results show that this automaton produces a poor performance. The Trp automaton is unable to regain its previous performance after the switch because of the high penalty probabilities, but cannot reduce the penalty probabilities because of its poor performance.

The results produced by the Tri automaton were the best that were obtained, particularly the first result. This had a lower mean turnaround time after the switch than before and producing the lowest result of all the automata schemes. The second result is less good even though the final action probabilities of both runs are the same. The difference is in the speed of response to the switch with the second result being slower and allowing large queues to develop before responding to the switch in the environment.

Table 8.3 gives results of simulations in the same environment as Table 8.1 but with a switch after 2000 iterations. The switch in the environment is much less drastic than in Table 8.2 and the loading on the system is 0.7. Again the Tip automaton produces the poorest results and again the Tri automata produce the best results but marred by a slow response to the switch. Chapter 5 suggested that the Tri automaton would have a poor performance in a non-autonomous environment because the automaton tends to converge to select a single action almost exclusively. In the simulations above, a number of automata are used together so that each can converge to a single action and still as a whole produce an action probability between 0 and 1. In fact the more automata working together the better, as the

combined action probability produced by the automata will be closer to the optimal action probability. When responding to a switch in the environment the Tri automata are reluctant to change their action. However when a queue builds up at a processor and the penalty probability rises to 1 this forces a number of the automata selecting that action to change their action which changes the overall action probability to nearer the new optimal action probability.

In operation the fixed scheduling discipline calculates the expected turnaround time of a job allocated to each processor in the system knowing the processor speeds and the processor queue lengths and allocates the job to the processor with the shortest turnaround time. In a heavily loaded system the effect of the fixed scheduling discipline will be to establish processor queues, the length of which is proportional to the processor speed. When the environment switches, the fixed scheduling discipline is working with inaccurate data and will establish the longest queue for the slowest processor. However this does not have as large an effect on the performance as might be expected as the fixed scheduling discipline will stop filling the long queue in favour of the short queue which will be processed quickly by the fast processor. The fixed scheduling discipline also responds well if a processor fails completely since any jobs allocated to that processor will enter the queue and not be processed. The queue will only grow to an extent where the fixed scheduling discipline allocates all the jobs to the other processors. Since the fixed scheduling discipline still has accurate information about the other processors in the system the performance will still be optimal. Thus the fixed scheduling discipline provides a reasonable performance even when it has inaccurate information on the processor speeds.

So far the results presented have been for simulations in which the automata were of the same type and with the same parameters. In Tables 8.4 and 8.5 the automata are of the same type but with different parameters. The environment used was that of Figure 8.3 with 5 automata and 2 processors. In Table 8.4 runs (a)-(d), the number of optimal Lri automata is increased from 0 to 3. The results show that the Lri automata converge to selecting the fastest processor so gaining the shorter mean turnaround time. As more Lri automata are introduced the remaining Lrp automata are forced more and more into selecting the slower processor.

Runs (e) and (f) are for the Trp automaton, while Table 8.5 has results for the Tri and Tip automata. In these cases changing the automaton has no effect on the theoretical steady state conditions of the automata, instead the speed of convergence and variance are changed, small step sizes producing slow automata with low variance. In Table 8.4 (e) and (f) results for Trp automata show that changing the speeds of the automata results in a poorer performance. This is because the overall action probability is reduced by the slow automata and though the fast automata compensate it in not sufficient. For the Tip automata, as shown in Table 8.5(a)-(b), a variety of processor speeds increases performance. Overall, the action probability and penalty probability changed little but the mean turnaround time is affected. A variety of automata speeds can have two benefits. The first is a decrease in variance as the slow processors have reduced variance and the second is an increase in speed. Any tendency toward increased variance due to the fast automata is reduced as the response to the effects of the variance on the system is speeded up. Also any

tendency toward a decreased response time due to the slow automata is reduced provided the fast automata can compensate until the slow automata reach steady state.

Table 8.5(c)-(f) gives results for the Tri automata with a range of parameters. Tri automata are expected to go optimal but in (c) with the parameters all equal, the automata all have the same convergence rate. They all converge to select processor 1 but because of their similar speeds prevent each other from going optimal. In results (d)-(e) the fastest automata converge to selecting processor 1 almost exclusively. This forces the slowest automaton to select processor 2. Since this is the only automaton selecting processor 2 the penalty probability received is low even though the mean turnaround time is high. The overall performance is good since the mean turnaround time provided by processor 1 is low as only 4 automata are selecting it and because the variance of the system is reduced because the automata are nearly optimal. Result (f) has even more widely spaced parameters and in this case two automata converge to select processor 2 most frequently. Once all the automata have converged the fastest automata switches to select processor 2 because of its low penalty probability. However this has a detrimental effect on the overall performance.

Finally a closer look was taken at the steady state conditions of automata which have been analysed theoretically, i.e. the Lrp, Lri and Tip automata. The environment used had two automata and two processors, using the same processing speeds as in Figure 8.3 but with the mean time between jobs changed to keep the systems load at 0.7 of capacity. The results are given in Table 8.6.

The results given in (a) and (b) are for an Lri automaton operating with a Lrp automaton with $\theta = 1$. The Lri automaton will try to equalise the penalty probabilities while the Lrp will try to equalise the penalty rates according to equation (4.29). If the Lri automaton is successful in making $c_1 = c_2$ then from equation (4.29) the action probabilities for the Lrp automaton must be 0.5. This is what was observed in the simulations, the two automata combining to produce an action probability of approximately 0.7. The difference between the two results is the initial action probabilities of the automata. A similar performance (f), is produced by a Lri and Tip automaton operating together. This is as expected since the Lrp and Tip automata have the same steady state conditions.

Table 8.6 (c) is for a Lrp automaton operating with a fixed probabilistic rule. By satisfying its own steady state condition, the automaton produces a poor overall performance. Poor results are also shown in (d) and (g) which give results produced by two Lrp and two Tip automata operating together.

The fifth result shows two Lri automata operating together. Overall the result is the same as the other results with only a single Lri automaton. The addition of a second Lri instead of a Lrp automaton has no overall effect though the individual action probabilities of the automata are changed.

Conclusions

The investigations into allocation methods in a multiprocessor system has shown that the fixed scheduling discipline can provide a good performance even when it has inaccurate information. It has been shown that the steady state conditions of the Lri, Lrp and Tip automata do not necessarily correspond to the action probabilities

which would give the best system performance. The Trp automaton has a reasonable performance but the Tri automaton has been of most interest. Individually these automata have a poor performance in non-autonomous environments but when a number have been used together they have been shown capable of steady state results better than any of the other automata schemes.

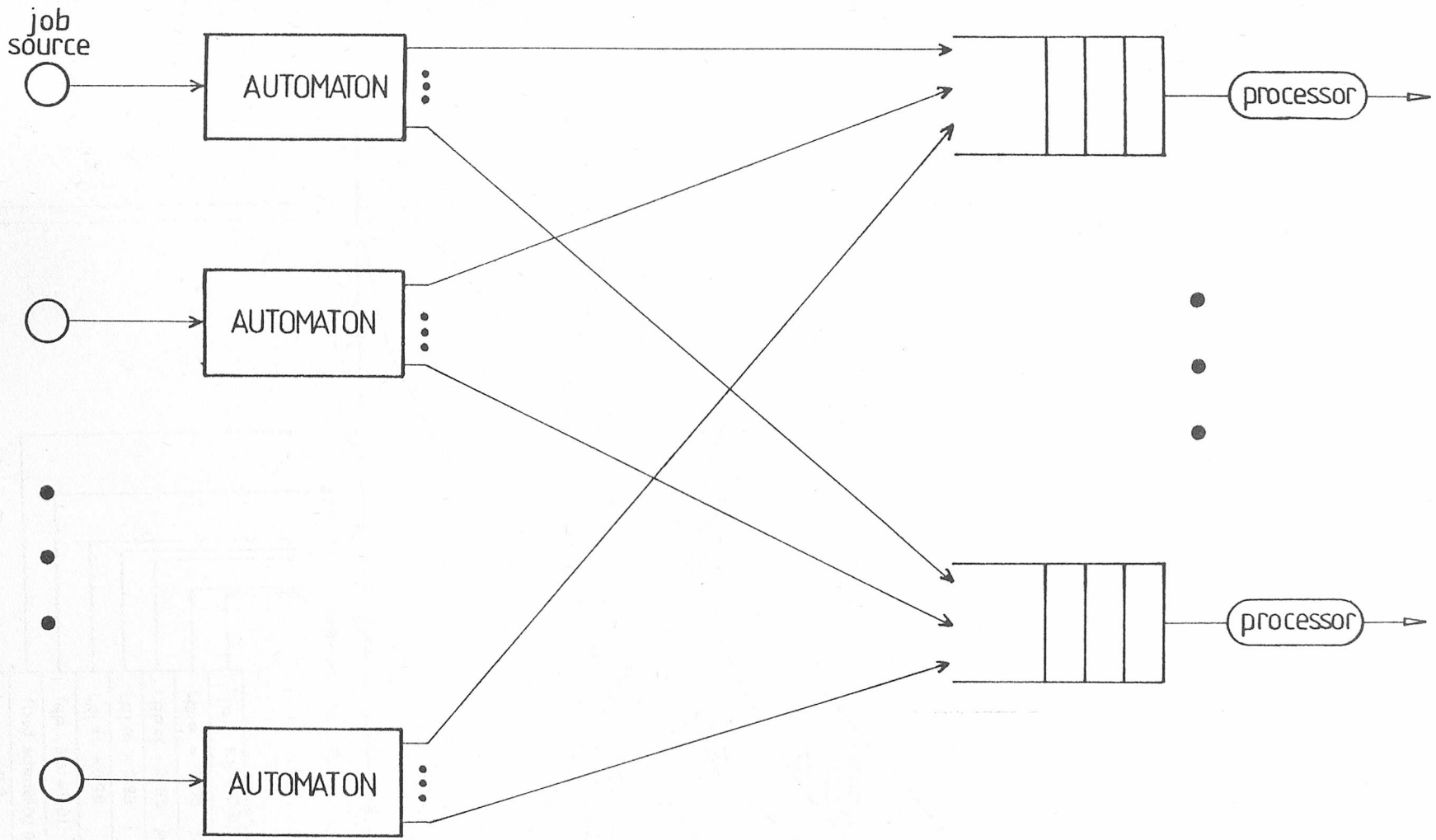


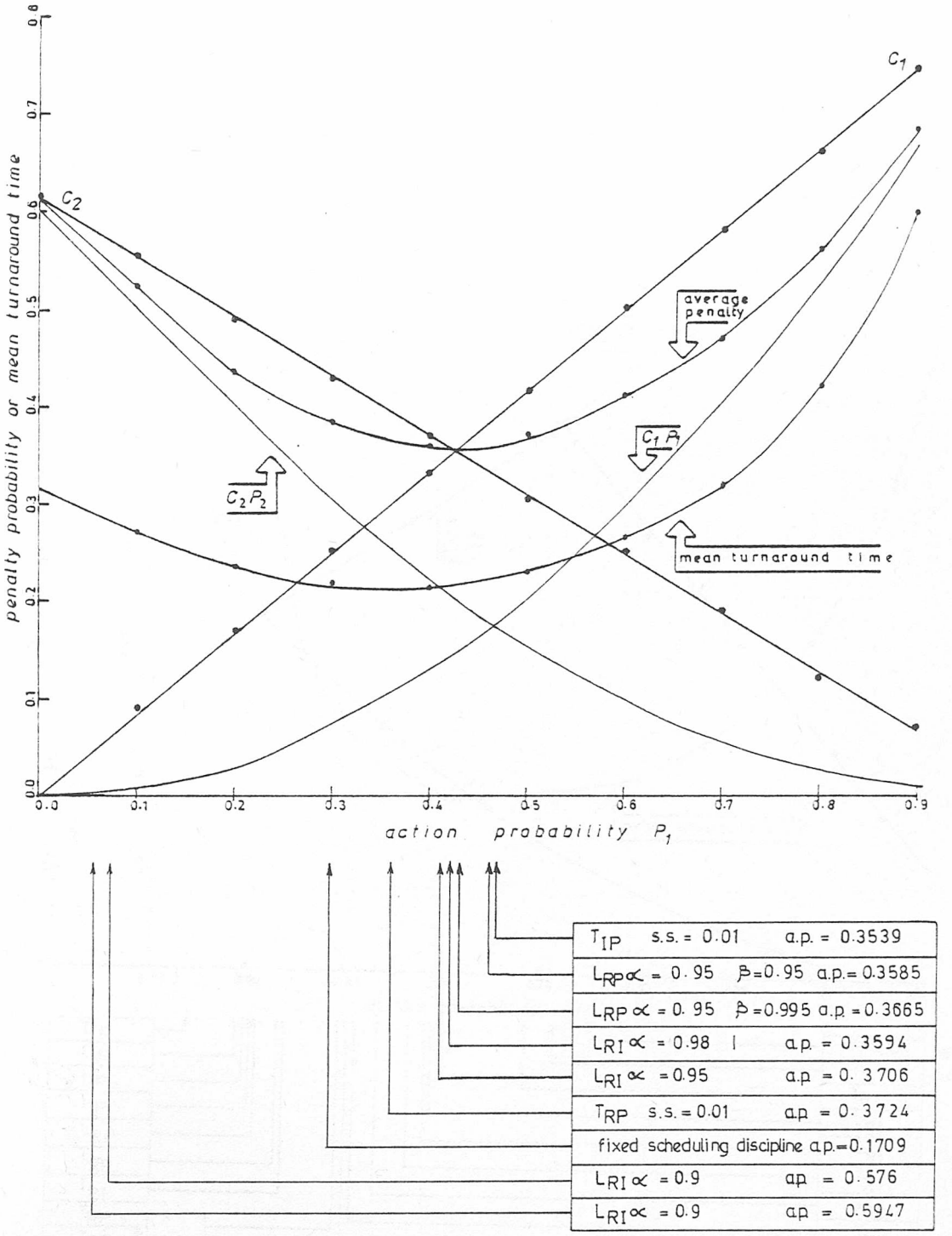
Figure 8.1 Allocation of jobs in a multi-processor system

Characteristics of multiprocessor system

1 source of jobs { *mean time between jobs = 0·2 s*; *mean processing requirement = 1c.u.*

2 processors of speeds 6c.u./s and 8c.u./s

Results taken over 50000 iterations



Automaton results show, automaton, automaton parameters, and average penalty received.

Figure 8·2

Characteristics of multiprocessor system

5 sources of jobs { mean time between jobs = 1s
mean processing requirement = 0·5 c.u.

2 processors of speeds 2·5 c.u/s and 1·0714 c.u./s

Results taken over 40000 iterations

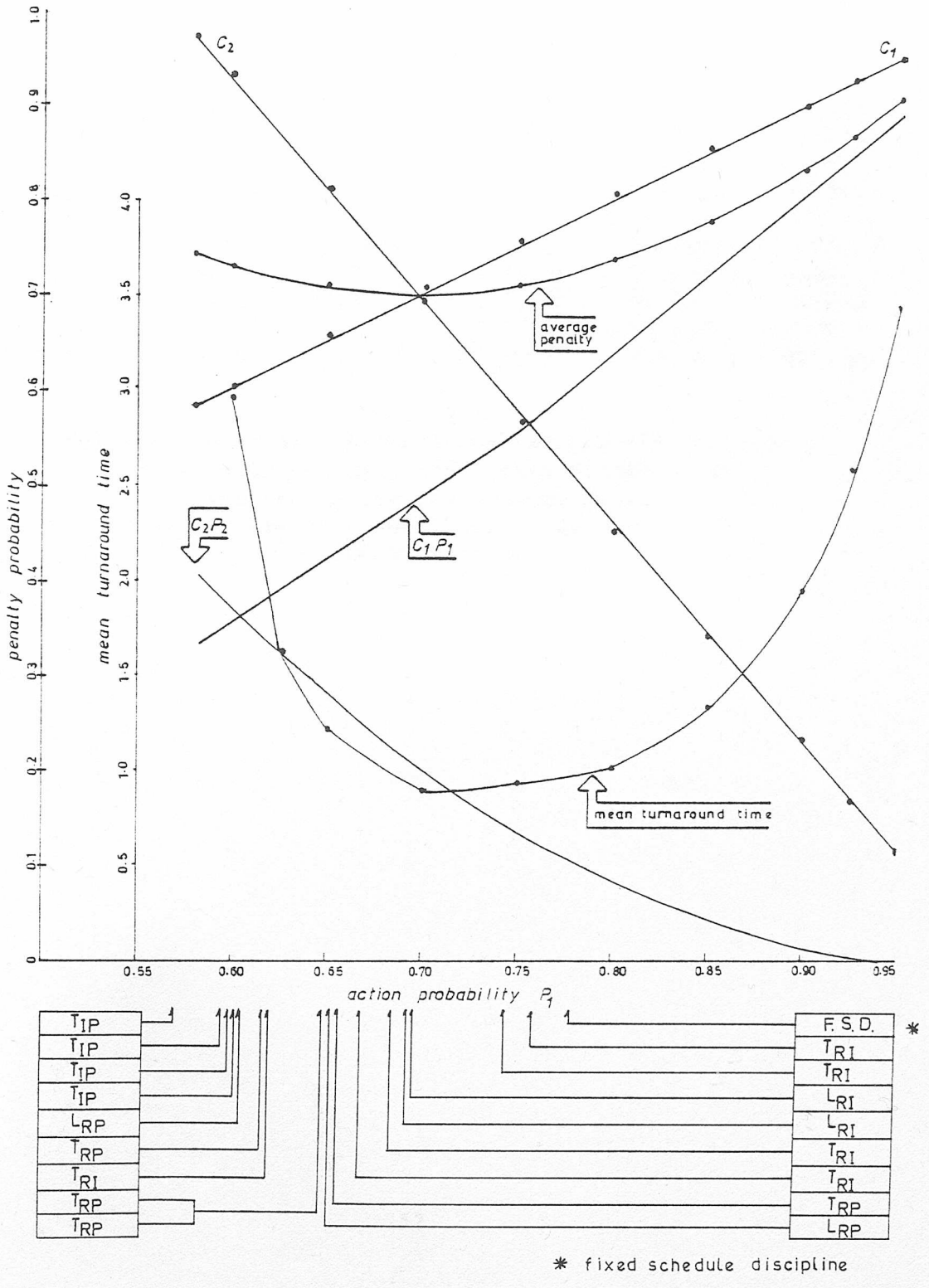


Figure 8·3

Automaton	Automaton parameters	Overall action probability processor 1	Mean turnaround time (seconds)	Percentage time processor1 busy	Percentage time processor2 busy
FSD	-	0.7774	0.5993	0.7631	0.5018
Lrp	0.95 0.95	0.6036	1.3700	0.5866	0.9138
Lrp	0.95 0.99	0.6492	1.0034	0.6308	0.8107
Lri	0.95 1.0	0.6930	0.8699	0.6726	0.7132
Lri	0.98 1.0	0.6906	0.9205	0.6672	0.7257
Trp	0.03	0.6518	1.0717	0.6298	0.8109
Trp	0.01	0.6482	1.0274	0.6312	0.8098
Trp	0.005	0.6482	1.0595	0.6299	0.8130
Trp	0.001	0.6166	1.4054	0.6005	0.8815
Tip	0.03	0.6008	1.5225	0.5822	0.9214
Tip	0.01	0.5982	1.5518	0.5815	0.9236
Tip	0.005	0.5958	1.6306	0.5785	0.9307
Tip	0.001	0.5702	4.1737	0.5553	0.9848
Tri	0.1	0.6826	1.1658	0.6642	0.7320
Tri	0.05	0.7572	0.8998	0.7403	0.5537
Tri	0.01	0.6652	1.7218	0.6470	0.7725
Tri	0.005	0.7428	0.9884	0.7219	0.5975
Tri	0.001	0.6178	1.5195	0.6038	0.8736

Table 8.1
Multiprocessor allocation scheme simulation results
5 job sources, mean time between jobs 1.0s
mean processing requirement 0.5cu
2 processors of speed 2.5cu/s and 1.0714cu/s
5000 iterations

Automaton	Automaton parameters	Overall action probability processors 1 and 2	Mean turnaround time (seconds)	Percentage time processor1 busy	Percentage time processor2 busy
FSD	-	.7285 .2485	0.8880	0.8258	0.5868
		.1333 .4250	3.7351	0.4949	0.4839
Lrp	0.95 0.95	.4775 .3235	19.0883	0.5585	0.7267
		.2014 .3276	58.2761	0.4949	0.3595
Lrp	0.98 0.98	.4707 .3265	20.0600	0.5580	0.7090
		.1994 .3274	64.2834	0.4949	0.3578
Lrp	0.95 0.995	.5370 .3408	1.8085	0.6548	0.7095
		.1138 .3284	2.0431	0.8464	0.7217
Lri	0.95 1.0	.5553 .3357	1.6011	0.6870	0.6875
		.0052 .3884	8.0939	0.2348	0.7920
Trp	0.01	.5320 .3217	3.4224	0.6261	0.7062
		.1402 .3178	11.0677	0.9822	0.7072
Trp	0.005	.5273 .3205	3.5206	0.6224	0.7076
		.1380 .3274	19.5042	1.0000	0.7173
Tip	0.03	.445 .3225	30.0657	0.5221	0.7076
		.2312 .3228	82.9226	1.0000	0.7270
Tip	0.01	.4465 .3195	31.5873	0.5169	0.7243
		.2316 .3246	87.7283	1.0000	0.7187
Tri	0.05	.5515 .3100	1.7526	0.6935	0.6735
		.0518 .3130	1.4018	0.3605	0.6985
Tri	0.01	.5480 .4110	2.1457	0.7187	0.7754
		.0194 .3118	20.0978	0.5332	0.6870

Table 8.2
Multiprocessor allocation scheme simulation results
3 job sources, mean time between jobs 0.5s, 0.75s, 1.0s.
mean processing requirement 2.0cu, 1.5cu, 1.0cu.
3 processors of speed 6cu/s, 3cu/s, 1cu/s.
switching to 1cu/s, 3cu/s and 6cu/s after 4000 iterations
first result taken over iterations 0-4000
second result taken over iterations 5000-10000

Automaton	Automaton parameters	Overall action probability processor 1	Mean turnaround time (seconds)	Percentage time processor1 busy	Percentage time processor2 busy
FSD		0.793	0.5274	0.7744	0.5001
		0.770	0.6643	0.7829	0.5539
		0.471	0.9586	0.9327	0.5062
		0.411	1.0541	0.9606	0.5879
		0.401	1.1071	0.9672	0.5633
Lrp	0.95 0.99	0.668	1.0252	0.6345	0.8256
		0.649	1.0918	0.6578	0.8398
		0.349	1.2386	0.7470	0.5880
		0.344	1.3442	0.8167	0.6502
		0.352	1.1507	0.7816	0.6483
Trp	0.01	0.651	1.0994	0.6259	0.8456
		0.639	1.0533	0.6584	0.8382
		0.373	1.4174	0.7861	0.5730
		0.346	1.4109	0.8261	0.6438
		0.339	1.0387	0.7580	0.6574
Tip	0.01	0.590	1.7851	0.5791	0.9449
		0.606	1.6716	0.6180	0.9427
		0.425	1.7915	0.8794	0.5328
		0.397	1.9348	0.9495	0.5875
		0.394	1.9422	0.8983	0.6016
Tri	0.02	0.747	0.9256	0.7281	0.6079
		0.678	1.1209	0.6821	0.7829
		0.510	13.9919	1.0000	0.4126
		0.181	4.4858	0.5927	0.8174
		0.223	1.0428	0.4668	0.7786

Table 8.3
Multiprocessor allocation scheme simulation results
5 job sources, mean time between jobs 1.0s
mean processing requirement 0.5cu
2 processors of speed 2.5cu/s and 1.0714cu/s
5 sucessive runs of 1000 iterations
with the environment switched after 2000 iterations

Automaton parameters	Run (a) Lrp	Run (b) Lrp	Run (c) Lrp	Run (d) Lrp	Run (e) Trp	Run (f) Trp
1	.95 .995	.95 .995	.95 .995	.95 1.0	0.005	0.03
2	.95 .95	.95 .95	.95 .95	.95 .95	0.005	0.01
3	.95 .9995	.95 1.0	.95 1.0	.95 1.0	0.005	0.005
4	.9 .99	.9 .99	.9 .99	.9 .99	0.005	0.002
5	.98 .9998	.98 .9998	.98 1.0	.98 1.0	0.005	0.001
Mean turnaround time						
1	0.9087	0.9461	0.9101	0.7844	0.9386	0.9420
2	0.9121	0.9199	0.8759	0.8841	0.8966	0.9497
3	0.6696	0.6163	0.6083	0.6451	0.8566	0.9264
4	1.0226	1.0457	1.0204	1.0110	0.9584	1.0420
5	0.7900	0.8139	0.7384	0.8160	1.0048	1.1080
overall	0.8608	0.8685	0.8307	0.8281	0.9305	0.9930
Average penalty received						
1	0.6606	0.6606	0.6667	0.6687	0.6747	0.6847
2	0.6905	0.6847	0.6868	0.6828	0.6692	0.6867
3	0.6660	0.6580	0.6540	0.6660	0.6740	0.6920
4	0.6957	0.7160	0.7120	0.6937	0.6815	0.6998
5	0.6728	0.6870	0.6748	0.6890	0.7012	0.7134
overall	0.6772	0.6812	0.6788	0.6800	0.6800	0.6952
Action probability processor 1						
1	0.6029	0.5567	0.5550	0.7649	0.6447	0.6702
2	0.5405	0.5386	0.5348	0.5195	0.7010	0.6958
3	0.9320	0.9977	0.9977	0.9990	0.6597	0.6709
4	0.4876	0.5037	0.4790	0.4460	0.6393	0.6265
5	0.8150	0.7907	0.8217	0.7669	0.6595	0.5817
overall	0.6716	0.6740	0.6776	0.6976	0.6588	0.6492

Table 8.4
Multiprocessor allocation scheme simulation results
5 job sources, mean time between jobs 1.0s
mean processing requirement 0.5cu
2 processors of speed 2.5cu/s and 1.0714cu/s
final 2500 of 5000 iterations

Automaton parameters	Run (a) Tip	Run (b) Tip	Run (c) Tri	Run (d) Tri	Run (e) Tri	Run (f) Tri
1	0.01	0.05	0.02	0.04	0.04	0.1
2	0.01	0.02	0.02	0.02	0.04	0.04
3	0.01	0.01	0.02	0.02	0.02	0.02
4	0.01	0.005	0.02	0.02	0.01	0.01
5	0.01	0.002	0.02	0.01	0.01	0.004
Mean turnaround time						
1	2.0077	1.8455	1.1382	0.8555	0.8575	1.3126
2	1.8977	1.7415	1.0775	0.8472	0.8513	0.6536
3	1.8124	1.7253	1.0006	0.8621	0.8647	0.6652
4	1.9404	1.8220	1.0979	0.8444	0.8473	0.6670
5	1.9828	1.8774	1.1480	0.9225	0.9202	1.2222
overall	1.9283	1.8021	1.0925	0.8662	0.8681	0.9051
Average penalty received						
1	0.7201	0.7201	0.6978	0.7711	0.7711	0.7050
2	0.7177	0.7204	0.6942	0.7623	0.7583	0.6771
3	0.7127	0.7113	0.6839	0.7629	0.7636	0.6839
4	0.7143	0.7088	0.6857	0.7517	0.7558	0.6937
5	0.7313	0.7306	0.7076	0.4990	0.4983	0.6941
overall	0.7192	0.7183	0.6939	0.7101	07101	0.6888
Action probability processor 1						
1	0.6077	0.6084	0.6405	0.9769	0.9769	0.2230
2	0.6053	0.5988	0.6644	0.9779	0.9720	0.9774
3	0.6048	0.6049	0.7000	0.9785	0.9785	0.9794
4	0.6195	0.6202	0.6562	0.9791	0.9893	0.9899
5	0.5926	0.5957	0.6600	0.0101	0.0102	0.3382
overall	0.6069	0.6077	0.6612	0.7843	0.7851	0.6945

Table 8.5
Multiprocessor allocation scheme simulation results
5 job sources, mean time between jobs 1.0s
mean processing requirement 0.5cu
2 processors of speed 2.5cu/s and 1.0714cu/s
final 7500 of 10000 iterations

Automaton parameters		Overall action probability processor 1	Automaton action probabilities		Mean turnaround time	c1 & c2 for Lri c_1 pa_1 & c_2 pa_2 for Lrp & Tip	
Lrp 0.98 0.98		0.6981	0.5024	0.4976	1.0824	0.3449	0.3471
Lri 0.98 1.0			0.8930	0.1071		0.6886	0.6912
			(a)				
Lrp 0.98 0.98		0.6930	0.5080	0.4920	1.0965	0.3480	0.3473
Lri 0.98 1.0			0.8782	0.1218		0.6900	0.6928
			(b)				
Lrp 0.98 0.98		0.5739	0.6345	0.3655	6.9390	0.3577	0.3559
			0.5	0.5			
			(c)				
Lrp 0.98 0.98		0.6056	0.5989	0.4011	2.3982	0.3601	0.3559
Lrp 0.98 0.98			0.6030	0.3970			
			(d)				
Lri 0.98 1.0		0.6968	0.6188	0.3812	1.1302	0.6894	0.6946
Lri 0.98 1.0			0.7721	0.2279			
			(e)				
Tip 0.01		0.6955	0.5003	0.4997	1.1096	0.3431	0.3520
Lri 0.98 1.0			0.8919	0.1081		0.6857	0.7064
			(f)				
Tip 0.01		0.6046	0.5963	0.4037	2.2618	0.3578	0.3589
Tip 0.01			0.6000	0.4000			
			(g)				

Table 8.6
Multiprocessor allocation scheme simulation results
2 job sources, mean time between jobs 0.625s
mean processing requirement 0.5cu
2 processors of speed 2.5cu/s and 1.0714cu/s
10000 iterations

CHAPTER 9 CONCLUSIONS AND FURTHER WORK

The work described in Chapter 2 is believed to be the first investigation of the Tsetlin and Krylov automata synthesised using digital electronics. These investigations quickly revealed practical weaknesses in these automata which have not been highlighted by theoretical analysis. The Tsetlin automaton is optimal as the memory size increases towards infinity provided one of the penalty probabilities is less than 0.5. However this work shows that for satisfactory performance the penalty probabilities should be about 0.5. For this reason Tsetlin automata with more than two actions are not considered practical. The Krylov automaton is designed to provide a better performance than the Tsetlin automaton by being optimal for all penalty probabilities as the memory size increases towards infinity. In contrast it has been found that the performance of the Krylov automaton has been unsatisfactory in all environments.

Also described in Chapter 2 is what is believed to be the first use of hierarchical automata. A second automaton has been used to monitor the performance of the first and controls its parameters to enable it to achieve the best performance in a non-stationary environment. The importance of this is increased when, in later chapters on non-autonomous environments, it is shown that in general automata do not converge to the optimum for the environment but to a steady state condition determined by their parameters.

The work in Chapter 2 has highlighted the deficiencies in the automata described as Type 2 in Table 1.1. These have a fixed structure and a deterministic output which means that the automata cannot provide good performance for the whole range of penalty probabilities. The automata proposed in Chapter 3 have a variable structure and the results have shown that adopting a variable structure can achieve good performances over the range of penalty probabilities.

Chapter 4 shows the disadvantages of a deterministic output function by considering non-autonomous environments, where a mixture of actions produces the best performance. To be able to investigate non-autonomous environments a model is required. The model proposed in Chapter 4 is more realistic than others in that it uses the information that would be available to an actual environment and also provides a model on which it is easy to carry out theoretical analysis. Simulations of automata with deterministic output functions has shown their unsuitability in non-autonomous environments. Simulations of automata with stochastic output functions operating in Narendra's non-autonomous environment has led to a theoretical analysis showing the unsuitability of this model. Theoretical analysis of the steady state conditions of the Lri and Lrp automata has been given and though these results have been presented elsewhere the method used here to achieve the results is different. These results show that the Lrp and Lri automata operate to satisfy their own steady state conditions and not the conditions for minimum average penalty.

Based on the conclusions of Chapter 4, three automata are proposed in Chapter 5 which have stochastic output functions. The operation of these automata has been analysed and their performance calculated and compared to the Lri automaton. The graphs presented in this and previous chapters giving the optimality, average penalty and mean switching times in a variety of stationary, non-stationary and non-autonomous environments is an attempt to give useful information about the performance of the automata and so enabling them to be compared directly. The analysis of the Lrp automaton required to produce these graphs is believed to be the first of its kind. Of the automata proposed in this chapter, the Trp and Tip have performances comparable to the Lri automaton.

Chapter 6 considers multi-action automata using the hierarchical learning system and automaton games. Though the use of the hierarchal learning system is not new the modified Tsetlin automata have not been used in this system before. Comparisons with the Lrp automaton have proved valuable and certain advantages of the modified Tsetlin automata highlighted. Attempts to cause the hierarchical learning system to converge to the incorrect actions were unsuccessful proving the practicality of the system.

The operation of various automata in games situations has been observed and was as expected from previous work. In cooperative games the convergence of the automata to the optimum penalty probability element has been tested and the best conditions for convergence established. In competitive games the operation of the automata has been observed and the conditions for their convergence or

non-convergence understood. Based on this, a penalty probability matrix has been devised to test the performance of the automata. From these tests the Lri automaton has provided the best performance but the Trp automaton also gave good results, in many cases better than Lrp automata.

In Chapter 7 the operation of the Tsetlin allocation scheme was investigated and it was found that it did not perform as expected. The reasons for this were identified and a modification to the system provided an improvement in performance by making the operation of the system closer to what was originally intended. Also tested was a more conventional learning automata scheme. Although this also provided an improvement in performance a more important practical consideration was discovered. In the system the automaton could not select between all its actions but for simplicity the feedback was applied to all actions. The result of this was that the automata algorithms were distorted so that the automata did not operate as expected. The effect on different automata was variable with the operation of some automata changing dramatically. This is obviously an effect that will occur in a variety of practical applications and should be remembered when future systems are being designed.

In Chapter 8 a scheduling discipline for a multiprocessor system with up to date information about the system is compared to an automaton system with no information about the system. The comparison is somewhat unfair as the superior performance of the fixed scheduling discipline shows. It is only when the system is disrupted a great deal that the adaptability of the automaton scheme becomes beneficial. A point worth

noting from these investigations is the performance of the Tri automata. When operating alone they have a poor performance but when many are used together their combined performance has been shown to be good.

The work presented here investigates the performance of a variety of automata and how they operate in various environments. It is felt that enough is known about the performance and characteristics of learning automata to allow their use in practical situations. However the selection of a suitable application for the use of learning automata is important. Learning automata learn by selecting the wrong actions and cope with non-stationary environments by continuing to select the wrong actions occasionally after they have learned. Because of this, learning automata cannot be used where the wrong actions would cause damage or have dangerous consequences. They are more suitable in non-autonomous environments where there are no wrong actions and the ratio of actions is important. Learning automata are best used in situations where feedback is available frequently and the feedback should be determined by the actions of the automaton as directly as possible. It is also important to be sure there are gains to be made by using learning automata. For example the use of learning automata within a single processor computer system cannot provide more processing time for the users, it can only distribute it between the users in different ways.

One example which illustrates all these points is the adaptive cancelling of sound [50] where a waveform is adaptively generated to cancel the noise of a diesel exhaust. In this example the performance is easily specified, an output from the environment is always available for feedback and an incorrect action by the automaton does not have

serious consequences.

One application of learning automata that is receiving considerable attention is the adaptive routing of calls in a telephone network [45,46,47,48,49]. To provide a link between the source of a call and its destination a number of links are made between exchanges. For a particular source and destination there are a variety of paths the call can take. The use of learning automata to route calls between exchanges has a twofold advantage. First the learning scheme can achieve near optimal performance and so match the performance of a conventional routing algorithm but in addition the learning scheme is also adaptive and so can maintain performance in a non-stationary situation. Secondly a learning scheme can cope with overloads in particular parts of the system by using unused capacity in other parts of the system, something a conventional routing algorithm cannot do. Although the operation of individual automata in such a system can be predicted much less is known about the overall performance and it is here that present work is being concentrated.

A similar application is the use of learning automata in a packet switched communication network. In this case a complete link from source to destination is not made, instead the message is split into standardised packets and sent from exchange to exchange. A packet switched communication network provides an even more complex system than the telephone network with a greater rate of feedback for the learning automata but also with more quickly changing characteristics. Once again the performance of individual learning automata in a decentralised system can be forecast from previous work but questions are still

unanswered regarding the global performance and how global characteristics can be used to control local automata.

Finally recent work [53] has drawn interest towards the use of a hierarchical automaton to control a PID three term controller. This combination provides a control unit with widely understood and trusted operating characteristics. However the use of a learning automaton to control the parameters of the controller provides a learning capability and an ability to adapt to changing environmental characteristics.

APPENDIX 1 CALCULATION OF STEADY STATE ACTION PROBABILITIES

The operation of an automaton can be described in terms of the state vector and the Markov transition matrix Pt [51,52] as

$$\phi(n+1) = Pt * \phi(n) \qquad \text{(A1.1)}$$

Given an initial condition $\phi(0)$, $\phi(n)$ can be calculated as

$$\phi(n) = P^{n} * \phi(0) \qquad \text{(A1.2)}$$

As $n \to \infty$ $\phi(n)$ approaches the steady state probability vector. This method of calculating the steady state probability vector is impractical because of the large number of matrix computations involved.

Equation (A1.1) describes a set of N simultaneous equations. By replacing the first equation by

$$1 = \phi_1 + \phi_2 + \ldots \qquad \text{(A1.3)}$$

the set of equations can be solved to give the steady state probabilities. Once the state probabilities are found and the relationship between the states of the automaton and the actions is known the action probabilities can be calculated.

APPENDIX 2 CALCULATION OF MEAN SWITCHING TIMES

For an automaton with 2n states operating in a switched environment initially favoring action 1 but changing to favor action 2 in response to a switch the mean switching time is given by [31]

$$t = \phi_1 \, m_{1\,n+1} + \phi_2 \, m_{2\,n+1} + \cdots \phi_n \, m_{n\,n+1} \qquad \text{(A2.1)}$$

where ϕ_i = probability of state i

$m_{i\,j}$ = mean first passage time from state i to state j

To find the mean first passage time from state i to state j consider the situation after one time epoch [52]. The automaton will have moved from state i to some other state k, which may be the final state j with probability $pt_{i\,k}$ thus

$$m_{i\,j} = 1 + \sum_{k=j} pt_{i\,k} \, m_{k\,j} \qquad \text{(A2.2)}$$

since when k=j $m_{k\,j} = 0$

where $pt_{i\,k}$ is an entry in the Markov transition matrix for the automaton after the switch in the environment. By considering equation (A2.2) for all values of i a set of simultaneous equations can be formed which when solved give passage times from the states i to state j. Using equation (A2.1) the mean switching time of the automaton can be found.

APPENDIX 3 MARKOV TRANSITION MATRICES OF AUTOMATA

Tsetlin Automaton

	1	2	3	4	.	N-1	N	N+1	N+2	.	2N-3	2N-2	2N-1	2N
1	$1-c_1$	c_1	0	0										
2	$1-c_1$	0	c_1	0										
3	0	$1-c_1$	0	c_1										
.														
.														
.														
N						$1-c_1$	0	c_1	0					
N+1						0	c_2	0	$1-c_2$					
.														
.														
.														
2N-2											c_2	0	$1-c_2$	0
2N-1											0	c_2	0	$1-c_2$
2N											0	0	c_2	$1-c_2$

where c_1 = penalty probability associated with action 1

and c_2 = penalty probability associated with action 2

and N = memory size of automaton

and 2N = number of states in automaton

	1	2	3	4	.	N-1	N	N+1	N+2	.	2N-3	2N-2	2N-1	2N
1	$1-c_1/2$	$c_1/2$	0	0										
2	$1-c_1/2$	0	$c_1/2$	0										
3	0	$1-c_1/2$	0	$c_1/2$										
.														
.														
.														
N						$1-c_1/2$	0	c	$1/2$		0			
N+1						0	$c_2/2$	0	$1-c_2/2$					
.														
.														
.														
2N-2												$c_2/2$	0	$1-c_2/2$ 0
2N-1												0	$c_2/2$	0 $1-c_2/2$
2N												0	0	$c_2/2$ $1-c_2/2$

Type 1 Modified Tsetlin Automaton

	1	2	3...
1	$(1-c_1)W_r + c_1(1-W_p)$	$(1-c_1)(1-W_r)+c_1 W_p$	0
2	$(1-c_1)W_r + c_1(1-W_p)$	0	$(1-c_1)(1-W_r)+c_1 W_p$
3	0	$(1-c_1)W_r + c_1(1-W_p)$	0
.			
.			
.			
N	$(1-c_1)W_r + c_1(1-W_p)$	0	$(1-c_1)(1-W_r)+c_1 W_p$	0
N+1	0	$(1-c_2)(1-W_r)+c_2 W_p$	0	$(1-c_2)W_r + c_2(1-W_p)$
.			
.			
.			
2N-2	0	$(1-c_1)(1-W_r)+c_1 W_p$	0
2N-1	$(1-c_2)(1-W_r)+c_2 W_p$	0	$(1-c_2)W_r + c_2(1-W_p)$
2N	0	$(1-c_2)(1-W_r)+c_2 W_p$	$(1-c_2)W_r + c_2(1-W_p)$

	...2N-2	2N-1	2N

where $W_p = 1/(2c_m)$ (3.3)

and $W_r = 1/2(1-c_m)$ (3.6)

and $c_m = (c_1 + c_2)/2$

Type 2 Modified Tsetlin Automaton

	1	2	3...
1	$(1-c_1)+c_1(1-W_p)$	$c_1 W_p$	0
2	$(1-c_1)W_r$	$(1-c_1)(1-W_r)+c_1(1-W_p)$	$c_1 W_p$
3	0	$(1-c_1)W_r$	$(1-c_1)(1-W_r)+c_1(1-W_p)$
.			
.			
.			
N	$(1-c_1)W_r$	$(1-c_1)(1-W_r)+c_1(1-wp)$	$c_1 W_p$ 0
N+1	0	$c_2 W_p$	$(1-c_2)(1-W_r)+c_2(1-W_p)$ $(1-c_2)W_r$
.			
.			
.			
2N-2	$(1-c_2)(1-W_r)+c_2(1-W_p)$	$(1-c_2)W_r$	0
2N-1	$c_2 W_p$	$(1-c_2)(1-W_r)+c_2(1-W_p)$	$(1-c_2)W_r$
2N	0	$c_2 W_p$	$(1-c_2)+c_2(1-W_p)$

	...2N-2	2N-1	2N

where $W_p = (1-c_m)/c_m$ (3.9)

and $W_r = c_m/(1-c_m)$ (3.12)

and $c_m = (c_1+c_2)/2$

Tri Automaton

	1	2	3...
1	$\frac{N-1+c_2}{N}$	$\frac{(1-c_2)}{N}$	0
2	$\frac{(N-2)(1-c_1)}{N}$	$\frac{(N-2)c_1+2c_2}{N}$	$\frac{(N-2)(1-c_1)}{N}$
.			
.			
.			
N-1		$\frac{2(1-c_1)}{N}$	$\frac{2c_1+(N-2)c_2}{N}$ $\quad$ $\frac{(N-2)(1-c_2)}{N}$
N		0	$\frac{(1-c_1)}{N}$ $\quad$ $\frac{N-1+c_1}{N}$
		...N-2	N-1 $\quad$ N

where N = number of states in automaton

Tip Automaton

	1	2	3...
1	$\frac{(N-1)(1-c_1)+1}{N}$	$\frac{(N-1)c_1}{N}$	0
2	$\frac{2c_2}{N}$	$\frac{(N-2)(1-c_1)+2(1-c_2)}{N}$	$\frac{(N-2)c_1}{N}$
.			
.			
.			
N-1		$\frac{(N-2)c_2}{N}$	$\frac{2(1-c_1)+(N-2)(1-c_2)}{N}$	$\frac{2c_1}{N}$
N		0	$\frac{(N-1)c_2}{N}$	$\frac{(N-1)(1-c_2)+1}{N}$
		...N-2	N-1	N

Trp Automaton

	1	2	3...
1	$\frac{(N-1)(1-c_1)+c_2}{N}$	$\frac{(N-1)c_1+1-c_2}{N}$	0
2	$\frac{(N-2)(1-c_1)+2c_2}{N}$	0	$\frac{(N-2)c_1+2(1-c_2)}{N}$
.			
.			
.			
N-1		$\frac{2(1-c_1)+(N-2)c_2}{N}$	0	$\frac{2c_1+(N-2)(1-c_2)}{N}$
N		0	$\frac{(1-c_1)+(N-1)c_2}{N}$	$\frac{c_1+(N-1)(1-c_2)}{N}$
		...N-2	N-1	N

REFERENCES

1. Booth T L
'Sequential Machines and Automata Theory'
John Wiley and Sons Inc., 1968

2. Aleksander I and Hanna K
'Automata Theory: An Engineering Approach'
Edward Arnold Computer Systems Engineering Series, 1976

3. Ribeiro S T
'Random-Pulse Macines'
IEEE Transactions on Electronic Computers, Vol. EC-16, no. 3, June 1967

4. Narendra K S and Thathachar M A L
'Learning Automata-A Survey'
IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-4, no. 4, July 1974, pp 323-334

5. Narendra K S and Lakshmivarahan S
'Learning Automata-A Critique'
Systems and Information Sciences Report no. 7703, Department of Engineering and Applied Science, Yale University, New Haven, Conneciticut, May 1977

6. Varshavskii V I and Vorontsova I P
'On the Behavior of Stochastic Automata with a Variable Strucure'
Translated from Avtomatika i Telemekhanika, Vol. 24, no. 3, March 1963, pp 353-360

7. Shapiro I J and Narendra K S
'Use of Stochastic Automata for Parameter Self-Optimization with Multimodal Performance Criteria'
IEEE Transactions on Systems Science and Cybernetics, Vol. SSC-5, no 4, October 1969

8. Mason L G
'An Optimal Learning Algorithim for S-model Environments'
IEEE Transactions on Automatic Control, October 1973, pp 493-496

9. Viswanathan R and Narendra K S
'Expedient and Optimal Variable Structure Automata'
Dunham Laboratory Technical Report CT-31, Department of Engineering and Applied Science, Yale University, April 1970

10. Viswanathan R and Narendra K S
'Simulation Studies of Stochastic Automata Models Part 1: Reinforcement Schemes'
Becton Center Technical Report CT-45, Department of Engineering and Applied Science, Yale University, December 1971

11. Viswanathan R and Narendra K S
'Comparison of Expedient and Optimal Reinforcement Schemes for Learning Systems'
Journal of Cybernetics, Vol. 2, no. 1, 1972, pp 21-37

12. Neville R G, Nicol C R and Mars P
'Synthesis of Stochastic Learning Automata'
Electronics Letters, Vol. 14, no. 6, 16th March 1978, pp 206-207

13. Neville R G, Nicol C R and Mars P
'Design of Stochastic Learning Automata using Adaptive Digital Logic Elements'
Electronics Letters, Vol. 14, no. 11, 25th May 1978, pp 324

14. Neville R G, Nicol C R and Mars P
'Design of Nonlinear Stochastic Learning Automata'
Electronics Letters, Vol. 14, no. 13, 22nd June 1978, pp 396

15. Tsetlin M L
'On the Behaviour of Finite Automata in Random Media'
Translated from Avtomatika i Telemekhanika, Vol. 22, no. 10, October 1961, pp 1345-1354

16. Krylov V U
'On One Stochastic Automaton which is Asymptotocally Optimal in a Random Medium'
Automation and Remote Control, 1963, Vol. 24, pp 1114-1116

17. Krinskii V I
'Asymptotically Optimal Automaton with Exponential Speed of Convergence'
Biofizika, Vol. 9, no. 4, 1964, pp 484-487

18. Ponomarev V A
'A Design for an Automaton that is Asymptotically Optimal in a Stationary Random Medium'
Biofizika, Vol. 9, no. 1, 1964, pp 104-110

19. Narendra et al
'Deterministic Automata'
Draft of Interim Report, Yale University, 1979

20. Langholtz G
'Behaviour of Automata in a Nonstationary Random Environment'
Electronics Letters, Vol. 7, no. 12, 17th June 1971, pp 348-349

21. Langholz G
'On a Class of Automata Models of Learning Machines'
International Journal of Man-Machine Studies, no. 4, 1972, pp 119-127

22. Langholz G
'Interaction Between Stochastic Automata and Random Environments'
International Journal of Man-Machine Studies, Vol. 9, 1977, pp 223-231

23. Devroye L P
'A Class of Optimal Performance Directed Probabilistic Automata'
IEEE Transactions on Systems, Man, and Cybernetics, November 1976, pp 777-783

24. Coutts M J and Mars P
'Study of a Modified-Estimation Automaton in Stationary Environments'
Electronics Letters, 22nd June 1978, Vol. 14, no. 13, pp 404

25. Coutts M J and Mars P
'Theory and Applications of a Modified-Estimating Automaton'
Proceedings of the 1st International Symposium on Stochastic Computing and its Applications, Toulouse, France, November 1978, paper 7.1, pp 303-320

26. Swan G
'Investigation and Design of an N State Learning Automaton'
BSc Thesis, Robert Gordon's Institute of Technology, June 1980

27. Golomb S W
'Shift Register Sequences'
Holden Day Inc. Series in Information Systems, 1967

28. Damashek M
'Shift Register with Feedback Generates White Noise'
Eletronics, 27th May 1976, pp 107-109

29. Miller A J, Brown A W and Mars P
'A Study of an Output Interface for a Digital Stochastic Computer'
International Journal of Electronics, Vol. 37, no. 5, 1974, pp 637-655

30. Miller A J and Mars P
'Optimal Estimation of Digital Stochastic Sequences'
International Journal of Systems Science, Vol. 8, no. 6, 1977, pp 683-696

31. Loui M C and Narendra K S
'Comparison of Learning Automata Operating in Nonstationaty Environments'
Becton Center Technical Report CT-65, Department of Engineering and Applied Science, Yale University, May 1975

32. Mendel J M and Fu K S
'Adaptive, Learning and Pattern Recognition Systems: Theory and Applications'
Mathematics in Science and Engineering, Vol. 66, Academic Press, 1970

33. Tsuji H, Mizumoto M, Toyada J and Tanaka K
'An Automaton in the Nonstationary Random Environment'
Information Sciences, no. 6, 1973, pp 123-142

34. Cox D R and Miller H D
'The Theory of Stochastic Processes'
Methuen and Co., London, 1965

35. Narendra K S and Thathachar M A L
'On the Behaviour of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing'
Proceedings of the 1st International Symposium on Stochastic Computing and Its Applications, Toulouse, France, November 1978, paper 7.1, pp 301-313 and
System and Information Science Report no. 7803, Yale University, October 1978

36. Kumar P R and Narendra K S
'Learning Algorithm Model for Routing in Telephone Networks'
Systems and Information Science Report no. 7903, Yale University, May 1979

37. Chrystall M S
'Learning Automata Applied to Switched Circuit Networks'
Internal Report, Robert Gordon's Institute of Technology, February 1980

38. Neville R G, Nicol C R and Mars P
'Hardware Design for a 128-State Stochastic Learning Automaton using a Hierarchical Structure'
Journal of Cybernetics and Information Science, Vol. 2, no. 1, Spring 1979, pp 30-35

39. Neville R G, Chrystall M S and Mars P
'Application of a Hierarchical Structure Stochastic Learning Automaton'
Systems and Information Science Report no. 7906, Department of Engineering and Applied Science, Yale University, September 1979

40. Narendra et al
'Two Player Tsetlin-Lri and Lri-Lri Games'
Draft Interim Report, Yale University, 1979

41. Tsetlin M L
'Automata Theory and Modeling of Biological Systems'
Volume 102 in Mathematics on Science and Engineering, Academic Press, 1973, pp 102-107

42. Madnick S E and Donavan J
'Operating Systems'
McGraw-Hill Computer Science Series, 1974

43. Pritsker A A B and Kiviat P J
'Simulation with Gasp II'
Prentice-Hall Inc., New Jersey, 1969

44. Colon-Osorio F C
'Scheduling in Multiple-Processor Systems with the Aid of Stochastic Automata'
Ph.d. Dissertation, University of Massachusetts , 1977

45. Narendra K S, Mason L G and Tripathi S S
'Applications of Learning Automata to Telephone Traffic Routing Problems'
Becton Center Technical Report CT-60, Department of Engineering and Applied Science, Yale University, January 1974

46. Narendra K S and Wright E A
'Application of Learning Automata to Telephone Traffic Routing Problems'
Becton Center Technical Report CT-69, Department of Engineering and Applied Science, Yale University, May 1976

47. Narendra K S, Wright E A and Mason L G
'Application of Learning Automata to Telephone Traffic Routing and Control'
IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-7, no. 11, November 1977, pp 785-792

48. Narendra K S and McKenna D M
'Simulation Study of Telephone Traffic Routing using Learning Algorithims: Part 1'
Systems and Information Science Report no. 7806, Yale University, December 1978.

49. Narendra K S, Mars P and Chrystall M S
'Simulation Study of Telphone Traffic Routing using Learning Algorithims'
Systems and Information Sciences Report no. 7907, Department of Engineering and Applied Science, Yale University, October 1979

50. Anon.
'Adaptive Sound Cancelling'
New Scientist, 18th February 1980, pp 657

51. Hiller F S and Liberman G J
'Introduction to Operations Research'
Holden-Day Inc., 1967

52. Kemeny J G and Snell J L
'Finite Markov Chains'
D Van Nostrand Company Inc., New Jersey, 1960

53. Neville R G
'Synthesis of Stochastic Learning Automata'
Ph.d. Thesis, Robert Gordon's Institute of Technology, October 1980

BIBLIOGRAPHY

General

1. Feller W
'An Introduction to Probability Theory and its Applications: Vol. 1'
3rd Edition, John Wiley and Sons, 1968

2. Hartley
'Digital Simulation Methods'
IEE Monograph Series 15, Peter Peregrinus Ltd, 1975

3. Howard R A
'Dynamic Probabilistic Systems: Vol. 1 Markov Models'
John Wiley and Sons Inc., New York, 1971

4. Vincent C H
'Random Pulse Trains: Their Measurment and Statistical Properties'
IEE Monograph Series 13, Peter Peregrinus Ltd., 1973

Queues

5. Black J R and Even J C Jr.
'Computer Solution of Queueing Models'
Simulation, October 1973, pp 113-116

6. Cox D R and Smith W L
'Queues'
Methuen and Co. Ltd., London, 1961

7. Gross D and Harris C M
'Fundamentals of Queueing Theory'
John Wiley and Sons, 1974

8. Kleinrock L
'Queueing Systems: Vol. 1 Theory'
John Wiley and Sons, 1975

9. Kleinrock L
'Queueing Systems: Vol. 2 Computer Applications'
John Wiley, 1976

10. Lee A M
'Applied Queueing Theory'
Macmillan, London, 1966

11. Saaty T L
'Elements of Queueing Theory, with Applications'
McGraw-Hill, New York, 1961

12. White J A, Schmidt J W and Bennett G K
'Analysis of Queueing Systems'
Academic Press, 1975

Communication Networks

13. Franks R L and Richel R W
'Optimum Network Call-Carrying Capacity'
The Bell System Technical Journal, September 1973, pp 1195-1214

14. Franks R L and Rishel R W
'Overload Model of Telephone Network Operation'
Bell System Technical Journal, November 1973, pp 1589

15. Glorioso R M and Greneich G R
'A Training Algorithim for Systems Described by Stochastic Transition Matrices'
IEEE Transactions on Systems, Man and Cybernetics, January 1971, pp 86-87

16. Hamsher D H
'Communication System Engineering Handbook'
McGraw-Hill, 1967

17. Kleinrock L K
'Communication Nets: Stochastic Message Flow and Delay'
McGraw-Hill Inc., 1964

18. Kleinrock L K
'On Communications and Networks'
IEEE Transactions on Computers, Vol. C-25, no. 12, December 1976

19. Macurdy W B and Ritchie A E
'The Network: Forging Nationwide Telphone Links'
Bell Laboratories Record, January 1975, pp 5-15

20. Prosser R T
'Routing Procedures in Communication Networks, part 1: Random Procedures, part 2: Directory Procedures
IRE Transactions on Communications Systems, CS-10(4), December 1972, pp 322-335

21. Rubin M and Haller C E
'Communication Switching Systems'
Reinhold (Chapman and Hall Ltd. London), 1966

22. Weber J H
'Some Traffic Characteristics of Communications Networks with Automatic Alternate Routing'
Bell System Technical Journal, March 1962, pp 769-796

Computer Systems

23. Badel M, Gilenbe E, Levoudier J and Potier D
'Adaptive Optimization of the Performance of a Virtual Memory Computer'
Computer Architectures and Networks, North Holland Publishing Co., 1974

24. Badel M, Gelenbe E, Leroudier J and Potier D
'Adaptive Optimization of a Time-Sharing System's Performance'
Proceedings of the IEEE, Vol. 63, no. 6, June 1975, pp 958-965

25. Bhandarkar D P
'Analysis of Memory Interference in Multiprocessors'
IEEE Transactions C-24, 1975, pp 897-908

26. Cardenas A F, Presser L and Marin M
'Computer Science'
Wiley-Interscience, 1972

27. Censier L M and Feautrier P
'Coherence Problems in Multi-Cache Systems'
IEEE Transactions on Computers, Vol. 27, no. 12, December 1978

28. Donavan J J
'Systems Programming'
McGraw-Hill Computer Science Series, 1972

29. Ferrari D
'Computer Systems Performance Evaluation'
Prentice-Hall Inc., New Jersey

30. Hellerman H and Conroy T F
'Computer System Performance'
McGraw-Hill Computer Science Series

31. Hide J
'Multi-Processing'
Microprocessors, Vol. 1, no. 6, August 1977

32. Hunter T J
'Construction and Control of a Multi Access Simulator'
MSc thesis, Victoria University of Manchester, October 1968

33. Muntz R R
'Analytic Modeling of Interactive Systems'
Proceedings of the IEE, Vol. 63, no. 6, June 1975, pp 946-953

34. Pin-Shan Chen P
'Queueing Network Model of Interactive Computing Systems'
Proceedings of the IEEE, Vol. 63, no. 6, June 1975, pp954-957

35. Potier D, Gelenbe E and Lenfant J
'Adaptive Allocation of Central Processing Unit Quanta'
Journal of the Association for Computing Machinery, Vol. 23, no. 1, January 1976, pp 97-102

36. Ramamoothy C V, Chandy K M and Gonzolez M J Jr
'Optimal Scheduling Strategies in a Multiprocessor System'
IEEE Transactions on Computers, Vol. C-21, no. 2, February 1972, pp 137-146

37. Ramamoorthy C V and Gonzalez M J Jr
'A Survey of Techniques for Recognizing Parallel Processable Streams in Computer Programs'

Fall Joint Computer Conference

38. Rao G S, Stone H S and Hu T C
'Assignment of Tasks in a Distributed Processor System with Limited Memory'
IEEE Transactions on Computers, Vol. C-28, no. 4, April 1979, pp 291-299

39. Smith A J
'An Analytic and Experimental Study of Multiple Channel Controllers'
IEEE Transactions on Computers, Vol. C-27, no. 1, January 1979, pp 38-49

40. Smith J L
'An Analysis of Time-Sharing Computer Systems using Markov Models'
Proceedings Spring Joint Computer Conference, 1966

41. Stone H and Bokhari S H
'Control of Distributed Processes'
Computer, July 1978, pp 97-106

42. Stone H S
'Critical Load Factors in Two-Processor Distributed Systems'
IEEE Transactions on Software Engineering, Vol. SE-4, no. 3, May 1978

43. Stone H S
'Multiprocessor Scheduling with the Aid of Network Flow Algorithms'
IEEE Transactions of Software Engineering, Vol. SE-3, no. 1, January 1977, pp 85-93

44. Walker M J
'The Design of a Special Purpose Digital Machine'
MSc thesis, Victoria University of Manchester, October 1968

45. Wallace V L and Rosenberg R S
'Markovian Models and Numerical Analysis of Computer System Behavior'
Proceedings Spring Joint Computer Conference, 1966, pp 141-148

46. Wilkes M V
'Time-Sharing Computer Systems'
Macdonald/American Elsevier Computer Monographs, Second Edition, 1972

47. Willis P J
'Derivation and Comparison of Multiprocessor Contention Measures'
IEE Journal on Computers and Digital Techniques, Vol. 1, no. 3, August 1978, pp 93-98

48. Wittle L
'Micronet'
Simulation, Vol. 31, no. 5, November 1978

Publications

Conf. n° 7-3

STOCHASTIC AUTOMATA IN NON-STATIONARY ENVIRONMENTS

by

Neil J Mackie and Philip Mars

School of Electronic and Electrical Engineering
Robert Gordon's Institute of Technology
Schoolhill
ABERDEEN AB9 1FR, Scotland

Proceedings of the First International Symposium on
Stochastic Computing and its Applications
Toulouse, France, November 1978

The application of digital stochastic computing techniques to the hardware synthesis of Tsetlin and Krylov automata is considered. Experimental results and measurements are presented for the performance of the Tsetlin automaton in non-stationary random environments. Contrary to previous work the Krylov automaton is shown to possess serious disadvantages in non-stationary environments. The results of simulations for two new automata based on those of Tsetlin and Krylov are given.

1 Introduction

Tsetlin in a pioneering paper[1] described a fixed structure learning automaton with a linear tactic, operating in a random environment.

Recently the suggestion has been made that the automaton described by Tsetlin is more suitable than other automata for use in non-stationary environments[2]. Elsewhere the Krylov automaton[3] has been proposed as an automaton which was asymptotically optimal for any environment, rather than being asymptotically optimal only for environments with one penalty probability c_i less than half and the other c_i greater than half as for the Tsetlin automaton.

The operation of an automaton is governed by an algorithm F which, in a fixed structure automaton, relates the state of the automaton $\phi(n)$ to $\phi(n + 1)$ and can be either deterministic or stochastic. In a variable structure automaton, F relates $p(n)$, the state probability vector, to $p(n + 1)$ while it is $p(n)$ which relates $\phi(n)$ to $\phi(n + 1)$. $\phi(n)$ is related to the action of the automaton $\alpha(n)$ by an output function G, which also can be either deterministic or stochastic. The Tsetlin automaton considered later is a fixed structure deterministic automaton while the Krylov automaton is a fixed structure stochastic automaton. Both have a deterministic output function. The discussion will be confined to automata classified as p model and with action sets limited to two elements.

For learning automata operating in non-stationary environments a measure of performance is the mean adjustment or switching time[2], defined as the average number of epochs, after a sudden change of the penalty probabilities from $c_1 < c_2$ to $c_1 > c_2$, till p_1 changes from being less than p_2 to being greater than or equal to p_2. For linear learning automata the mean switching time is the average number of epochs, after a sudden reversal of the penalty probabilities, until action 2 is reached, assuming the automaton was correctly providing an action 1 input immediately prior to the switch in the penalty probabilities.

2 The Tsetlin Automaton

2.1 Theory of Operation

The operation of the Tsetlin automaton can be seen with reference /

reference to Figure 1 which shows a two action automaton with a memory size of n, and has one action corresponding to internal states 1 to n and the other corresponding to internal states n + 1 to 2n. When the automaton takes action 1 the environment outputs a stochastic sequence of value c_1, while action 2 corresponds to a stochastic sequence of value c_2. When the automaton receives a penalty the automaton moves towards states n and n + 1 while, in response to a reward, the automaton moves towards end state 1 or 2n. Thus, with output action 1 the automaton performs a simple random walk between its internal states, with a reflecting barrier beyond state 1 and with output action 2 the automaton performs a simple random walk between its internal states, with a reflecting barrier beyond state 2n. If an action has associated with it a c_i, the value of which is greater than half, the automaton will tend to move towards states associated with the alternative action while, if the value of c_i is less than half, the automaton will tend to move towards the end state associated with the action it is already taking.

The operation of the Tsetlin automaton will fall into one of three modes depending on the environment. If the c_i's are about a half, one action will tend to make the automaton move towards states associated with the other action, while the other action will tend to make the automaton move towards the corresponding end state. Thus one action is stable while the other is unstable and the automaton works well. If the c_i's are both greater than a half, both actions will tend to make the automaton move towards states associated with the other action. Thus both actions are unstable, the automaton moves between states n and n + 1 frequently and works poorly. If the c_i's are both less than a half, both actions will tend to make the automaton move towards the end state associated with that action. Thus both actions are stable, with the automaton only moving from one action to another due to variance in the penalty probability causing it to be temporarily greater than a half over a long enough time to allow the automaton to move from one action to the other. If the largest penalty probability is not close to a half, or if the memory size is large, the automaton can output the wrong action for long periods of time and the automaton works poorly.

2.2 /

2.2 Hardware Design

A Tsetlin automaton, the block diagram of which is shown in Figure 2 was implemented using digital stochastic techniques[4][5][6]. The heart of the automaton is a 12-bit binary counter allowing up to 4096 states or memory sizes up to 2048. The most significant bit of the counter is taken as the action of the automaton and is input to the environment which outputs the appropriate penalty probability. The output of the environment and the action of the automaton are fed into combinational logic to convert these into an up/down control signal for the counter. The up/down signal is in turn fed into more combinational logic along with the state of the automaton and signals representing the memory size to provide a disable signal to prevent the counter exceeding the required memory size.

2.3 Experimental Results

The performance of a 2048 state memory Tsetlin automaton was investigated with a switched environment. Figure 3 shows the operation of the automaton with the central trace in each case indicating the switching instants for a reversal of penalty probabilities c_i. Figure 3(a) shows the satisfactory operation of the automaton with c_i's of $\frac{15}{16}$ and $\frac{1}{16}$. Figure 3(b) shows the effects of change of c_i's to $\frac{15}{16}$ and $\frac{3}{4}$, i.e. both greater than $\frac{1}{2}$. It is evident that the automaton fails to operate. Finally Figure 3(c) illustrates the characteristics with c_i's of $\frac{3}{16}$ and $\frac{1}{16}$. In this case since the c_i's are both less than $\frac{1}{2}$ the automaton again operates poorly and locks onto one action. These results are entirely consistent with the theoretical predictions.

Figure 4 shows experimental and theoretical results for mean switching times. The theoretical results given agree basically with previous predictions[2], but a compensation factor has been included to prevent the possibility of switching times less than one epoch which were not included in the experimental results. As may be seen from Figure 4 good correlation is obtained between theoretical and experimental results.

3 The Krylov Automaton

3.1 Theory of Operation

The /

2.2 Hardware Design

A Tsetlin automaton, the block diagram of which is shown in Figure 2 was implemented using digital stochastic techniques[4][5][6]. The heart of the automaton is a 12-bit binary counter allowing up to 4096 states or memory sizes up to 2048. The most significant bit of the counter is taken as the action of the automaton and is input to the environment which outputs the appropriate penalty probability. The output of the environment and the action of the automaton are fed into combinational logic to convert these into an up/down control signal for the counter. The up/down signal is in turn fed into more combinational logic along with the state of the automaton and signals representing the memory size to provide a disable signal to prevent the counter exceeding the required memory size.

2.3 Experimental Results

The performance of a 2048 state memory Tsetlin automaton was investigated with a switched environment. Figure 3 shows the operation of the automaton with the central trace in each case indicating the switching instants for a reversal of penalty probabilities c_i. Figure 3(a) shows the satisfactory operation of the automaton with c_i's of $\frac{15}{16}$ and $\frac{1}{16}$. Figure 3(b) shows the effects of change of c_i's to $\frac{15}{16}$ and $\frac{3}{4}$, i.e. both greater than $\frac{1}{2}$. It is evident that the automaton fails to operate. Finally Figure 3(c) illustrates the characteristics with c_i's of $\frac{3}{16}$ and $\frac{1}{16}$. In this case since the c_i's are both less than $\frac{1}{2}$ the automaton again operates poorly and locks onto one action. These results are entirely consistent with the theoretical predictions.

Figure 4 shows experimental and theoretical results for mean switching times. The theoretical results given agree basically with previous predictions[2], but a compensation factor has been included to prevent the possibility of switching times less than one epoch which were not included in the experimental results. As may be seen from Figure 4 good correlation is obtained between theoretical and experimental results.

3 The Krylov Automaton

3.1 Theory of Operation

The /

The Krylov automaton is very similar to the Tsetlin automaton in that it has a series of states 1 to 2n, with states 1 to n being associated with one action and states n + 1 to 2n being associated with the other. It is in the movement between the states that the Krylov and Tsetlin automata differ as shown in Figure 5. In response to a reward the Krylov automaton acts as the Tsetlin and moves deterministically towards an end state but, in response to a penalty, the automaton acts in a stochastic manner and either moves towards states n and n + 1 or towards the end states with probability $\frac{1}{2}$.

The action of the Krylov automaton can be related to that of the Tsetlin automaton. If an automaton performs an action such that it receives a penalty with probability c_1 then

$$\text{penalty probability} = c_1$$

$$\text{reward probability} = 1 - c_1$$

If a reward response is taken as a movement towards states 1 or 2n and if a penalty response is taken as a movement towards states n and n + 1 then for the Krylov automaton

$$\text{penalty response probability} = \frac{1}{2}c_1$$

$$\text{reward response probability} = 1 - \frac{1}{2}c_1$$

and a similar argument applies to c_2.

Equating response probabilities we see that a Krylov automaton receiving penalty probabilities in the range 0 , 1 is equivalent to a Tsetlin automaton receiving penalty probabilities in the range 0 , $\frac{1}{2}$. However, it has been shown above that the Tsetlin automaton does not function correctly with penalty probabilities both less than $\frac{1}{2}$ and so it was expected that the Krylov automaton would not work well over the complete range of c_i's.

3.2 Hardware Design

A Krylov automaton was designed using digital stochastic computing techniques and its schematic diagram is shown in Figure 2. The circuit is identical to that used in the Tsetlin automaton except that instead of deterministically converting a penalty response from the environment into an up/down control signal for the counter, a stochastic sequence of probability $\frac{1}{2}$ is sampled and used as the control signal.

3.3 /

3.3 Experimental Results

Figure 6(a) shows a Krylov automaton with memory size of 2045, initially with output action 1, operating in a switched environment with c_2's of 0 and $\frac{15}{16}$. As predicted the result is similar to a Tsetlin automaton working with both c_i's less than a half with the automaton locked into the output of one action. This locking is in fact a function of the memory size. The automaton has two stable states, with the state corresponding to the lower c_i being more stable than the other with stability increasing as the memory size increases. Variance in the penalty probabilities causes movement between the states and the time spent in a state depends on its stability. Thus while both states are stable, for small memory sizes, variance should cause movement between the states with the automaton spending more time in the most stable state. This can be seen in Figure 6(b) which shows a Krylov automaton with memory size of 8 with c_1 of $\frac{7}{8}$ and c_2 of $\frac{5}{8}$ moving from states corresponding to c_2 to states corresponding to c_1, remaining in those states for a time then moving back. Finally Figure 6(c) shows a Krylov automaton with memory size of 8 working in a switched environment with c_i's of $\frac{3}{4}$ and $\frac{5}{8}$. Since when the switching trace is high the automaton trace should be low it can be seen that it works poorly.

4 The Modified Tsetlin Automata, Types 1 and 2

Though the results of testing the Krylov automata were disappointing the Krylov automaton proved to be the basis of two new learning automata. The aim in designing these was to retain the good qualities of the Tsetlin automaton but also to produce automata which would operate well for c_i's about any value rather than the value of half which the Tsetlin automaton is limited to. The Krylov automaton took penalty probabilities which were greater than a half and produced penalty response probabilities which were less than a half. The modified Tsetlin automata take two penalty probabilities of greater than a half but about a value c_m and, by using a stochastic response to a penalty, produce one penalty response probability which is less than a half and one which is greater than a half. Further, by using a stochastic response to a reward, two penalty probabilities /

probabilities both less than a half but about a value c_m will produce one penalty response probability which is greater than a half and one which is less than a half. This is illustrated in Figure 7. Thus provided c_m is known any pair of penalty probabilities can be transformed to be about a half producing a Tsetlin type response.

4.1 Theory of Operation

The modified Tsetlin automata are similar to the Tsetlin automaton in that they have a series of states 1 to 2n with states 1 to n being associated with one action and states n + 1 to 2n being associated with the other. However, the movement between the states is more complex and is shown in Figure 8.

For the modified Tsetlin automaton, type 1 shown in Figure 8(a), and penalty probabilities about a c_m value greater than a half, as shown in Figure 7(a), to obtain penalty response probabilities c'_1 and c'_2 spaced about a half

$$c'_m = 0.5 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

Using a stochastic response to a penalty with probability W_p of moving towards states n and n + 1 and assuming a deterministic response to a reward then

$$c'_m = c_m \times W_p \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2)$$

Substituting equation (2) into equation (1)

$$W_p = \frac{1}{2\ c_m}$$

W_p is to be a stochastic variable and so has a maximum value of 1 thus

$$W_p = \frac{1}{2\ c_m} \quad \text{if} \quad \frac{1}{2\ c_m} < 1 \quad \ldots\ldots\ldots\ldots\ldots \quad (3)$$

$$= 1 \quad \text{if} \quad \frac{1}{2\ c_m} > 1$$

For penalty probabilities about a c_m value less than a half as shown in Figure 7(c), to obtain penalty response probabilities c'_1 and c'_2 spaced about a half

$$c'_m = 0.5 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4)$$

Using a stochastic response to a reward with probability W_r of /

of moving towards the end state associated with the action output by the automaton and assuming a deterministic response to a penalty, an assumption justified by equation (3) then

$$c'_m = c_m + (1 - W_r)(1 - c_m) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (5)$$

substituting equation (5) into equation (4)

$$W_r = \frac{1}{2(1 - c_m)} \quad \text{if} \quad \frac{1}{2(1 - c_m)} < 1 \quad \ldots\ldots \quad (6)$$

$$= 1 \quad \text{if} \quad \frac{1}{2(1 - c_m)} > 1$$

For c_m greater than a half $W_r = 1$, so justifying the assumption made in forming equation(2)

For the modified Tsetlin automaton, type 2, shown in Figure 8(b) in addition to penalty and reward responses we have an inaction response. If an inaction response is counted as half a penalty response, for penalty probabilities about a c_m value greater than a half as shown in Figure 7(a) to obtain penalty response probabilities c'_1 and c'_2 spaced about a half

$$c'_m = 0.5 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7)$$

Using a stochastic response to a penalty with probability of W_p of moving towards states n and (n + 1) and $(1 - W_p)$ of remaining in the same state, and assuming a deterministic response to a reward then

$$c'_m = c_m W_p + \frac{1}{2} c_m (1 - W_p) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (8)$$

Substituting equation(8) into equation (7)

$$W_p = \frac{1 - c_m}{c_m} \quad \text{if} \quad \frac{1 - c_m}{c_m} < 1 \quad \ldots\ldots\ldots \quad (9)$$

$$= 1 \quad \text{if} \quad \frac{1 - c_m}{c_m} > 1$$

For penalty probabilities about c_m value less than a half, as shown in Figure 7(c), to obtain penalty probabilities c_1 and c_2 spaced about a half

$$c'_m = 0.5 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (10)$$

Using a stochastic response with probability W_r of moving towards the end state associated with the action output by the automaton and assuming a deterministic response to a penalty, an /

an assumption justified by equation (9), then

$$c'_m = c_m + \frac{1}{2}(1 - c_m)(1 - W_r) \quad \ldots\ldots\ldots\ldots\ldots \quad (11)$$

Substituting equation (11) into equation (10)

$$W_r = \frac{c_m}{1 - c_m} \quad \text{if} \quad \frac{c_m}{1 - c_m} < 1 \quad \ldots\ldots\ldots\ldots\ldots \quad (12)$$

$$= 1 \quad \text{if} \quad \frac{c_m}{1 - c_m} > 1$$

For c_m greater than a half $W_r = 1$, so justifying the assumption made in forming equation (8).

Equations (3), (6), (9) and (12) require a value for c_m. This is taken as the mean of estimated values for c_1 and c_2 obtained from two adaptive digital circuit elements (ADDIES)[6] which respond to the reward/penalty signals obtained from the environment, these signals being fed to the ADDIE estimating c_1 when action 1 is output and to the ADDIE estimating c_2 when action 2 is output. It was predicted that the type 2 automaton with the inaction response would have less variance than the type 2 automaton and would be more nearly optimal for the same memory size.

4.2 Software Simulation

The modified Tsetlin automata, types 1 and 2 were simulated on a computer rather than the hardware synthesis used for the Tsetlin and Krylov automata because of the relative complexity of the automata structures. For the purpose of comparison the Tsetlin and Krylov automata were also simulated.

Figure 9 shows results from a simulation of a modified Tsetlin automaton, type 1, with memory size of 10, ADDIE counter size of 32 and operating in an environment with penalty probabilities of 0.6 and 0.9. It can be seen in Figure 9(a) that the automaton initially moves between actions 1 and 2 frequently but later moves to states associated with the action corresponding to the lower penalty probability. Initially, with the estimates of the c_i's in the ADDIES being zero, both actions are unstable but as the estimates of the penalty probabilities rise the actions become less unstable until in the steady-state the action corresponding to $c_{i(min)}$ is stable and the other unstable. The learning time is limited by the speed of response /

response of the ADDIES. Between Figures 9(a) and 9(b) the environment has been switched and it can be seen that the automaton reacts quickly to the change. It can be seen that the switching time in Figure 9(b) is shorter than the learning time in Figure 9(a). This is because the switching time is governed by the memory size which is small, rather than the counter size of the ADDIES which is larger.

Figure 10 shows results from a simulation of a modified Tsetlin automaton, type 2, with the same parameters as the type 1 considered above but operating with penalty probabilities of 0.1 and 0.4, again with the environment switching between Figures 10(a) and 10(b). The automaton operates satisfactorily and the lower variance of the type 2 automaton can be seen.

Figure 11 shows results of simulations of Tsetlin and Krylov automata. Figure 11(a) shows the Tsetlin automaton operating with penalty probabilities of 0.6 and 0.9 and moving frequently between states n and n + 1, both actions being unstable, while Figure 11(b) shows the Krylov automaton remaining in the incorrect state after a switch in the environment to 0.65 and 0.35.

Figure 12 illustrates a problem that can occur with either of the modified Tsetlin automata. Figure 12(a) shows a type 2 modified Tsetlin automaton operating in an environment with penalty probabilities of 0.35 and 0.65. In Figure 12(b) these have been switched but it is a relatively long time before the automaton changes its output action. This is due to two causes. The first is variance in the ADDIES. At the time of the switch the ADDIES held estimates of the penalty probabilities which were higher than normal. This resulted in a higher than normal value for c_m causing the stability of the actions of the automaton to be increased leading to a longer switching time. The second cause is the speed of response of the ADDIES being too fast in comparison with the speed of the speed of the counter. In this example the memory size was 10 and the ADDIE counter size 32. After the switch the automaton has as an input the higher c_i. If the ADDIES are small their response time is fast and the penalty probability estimate in the ADDIE has increased significantly before the automaton counter has had time to output the action corresponding to $c_{i(min)}$. Because of this, the value of c_m increases causing the stability of both actions to increase and result in longer switching times. /

times. In order to maintain short switching times the ADDIES used in the modified Tsetlin automata should not be too small so that they have low variance and response times longer than the automata's counters.

5 Conclusions

In the course of investigating the Tsetlin automaton in a deterministically switched environment, comparisons have been made with other automata structures. Optimal automata are a severe disadvantage when operating in a non-stationary environment because an automaton which is nearly optimal takes the correct action with a probability very nearly unity. Thus if the c_i's change so that the previously correct action becomes the wrong action, the automaton will continue to output the previously correct action and will not cause the environment to output the c_i corresponding to the current correct action and so the change in the c_i's can go unnoticed by the automaton for a long time. Decreasing the optimality will cause the wrong action to be output more often and so any change in the c_i's will be noticed by the automaton sooner. There is a trade-off between optimality and mean switching time.

The Tsetlin automaton provides good mean switching times and a near optimal performance but with a severe limitation on the environment, the c_i's having to be about a half, but it is the restrictions on the c_i's that gives the good optimality and learning times. When operating in a switched environment the Tsetlin automaton does not have to sample the wrong state in order to determine whether the environment has switched or not. Because the c_i's are about a half when the switch occurs, a c_i which was less than a half is now greater than a half and the automaton moves towards states associated with the other action no matter the degree of optimality and so high optimality is not a great penalty. The Tsetlin automaton seems to have only a small trade-off between optimality and mean switching time but has a severe trade-off between optimality, mean switching time and limitiations on the range of c_i's that can be used.

The Krylov automaton was believed to be asymptotically optimal in arbitrary random environments. Experimental evidence clearly shows that the automaton possesses an unsatisfactory performance /

performance in non-stationary random environments.

The modified Tsetlin automata types 1 and 2 have been shown to be capable of good learning characteristics with no restrictions on the penalty probabilities that can be used whilst retaining the short mean switching times and near optimal performance that characterises the Tsetlin automaton. It is hoped these automata will be of use in non-autonomous environments where their ability to reject actions that correspond to penalty probabilities above any value of c_m and their short switching times should prove valuable.

Acknowledgement : The authors wish to gratefully acknowledge the support of a U K Science Research Council grant.

References

1 TSETLIN M L,
'On the Behaviour of Finite Automaton in Random Medium'
Automation and Remote Control, 1961, 22, pp 1345 - 1354

2 LOUI, M C and NARENDRA, K S
'Comparison of Learning Automata Operating in Nonstationary Environments'
Becton Centre, Yale University, 1975, Report CT-65

3 KRYLOV, V U,
'On One Stochastic Automaton which is Asymptotically Optimal in a Random Medium'
Automation and Remote Control, 1963, 24, pp 1114 - 1116

4 GAINES, B R,
'Stochastic Computing'
AFIPS SJCC, 1967, 30, pp 149 - 156

5 MILLER, A J and MARS, P,
'Theory and Design of a Digital Stochastic Computer Random Number Generator'
Trans IMACS, 1977, 19, pp 198 - 216

6 MILLER, A J and MARS, P
'On Optimal Estimation of Digital Stochastic Sequences'
Int J Syst Sci, 1977, 8, pp 683 - 696

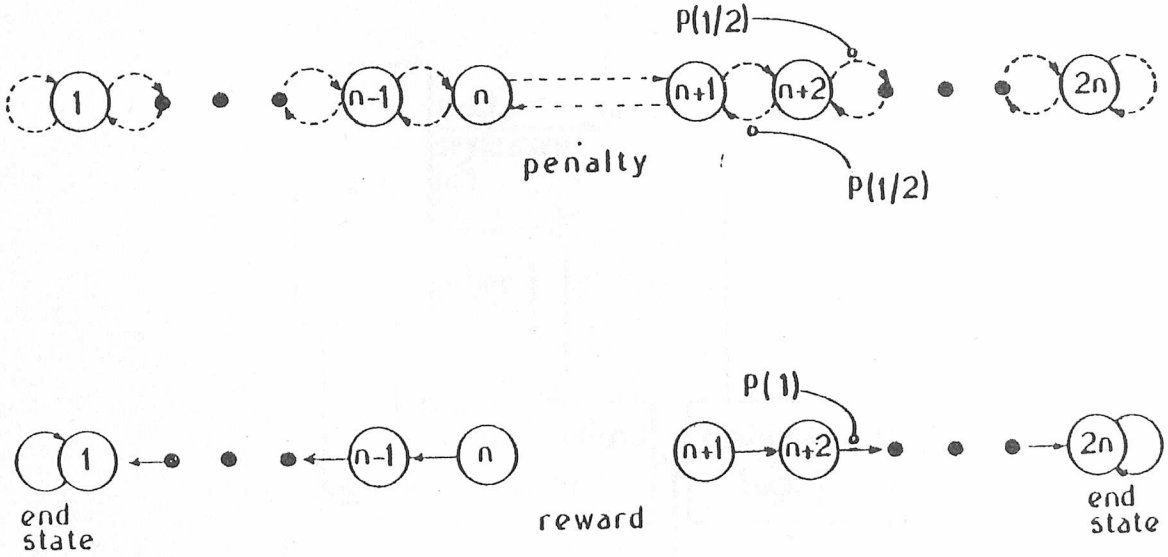


Figure 1 Operation of Tsetlin Automaton

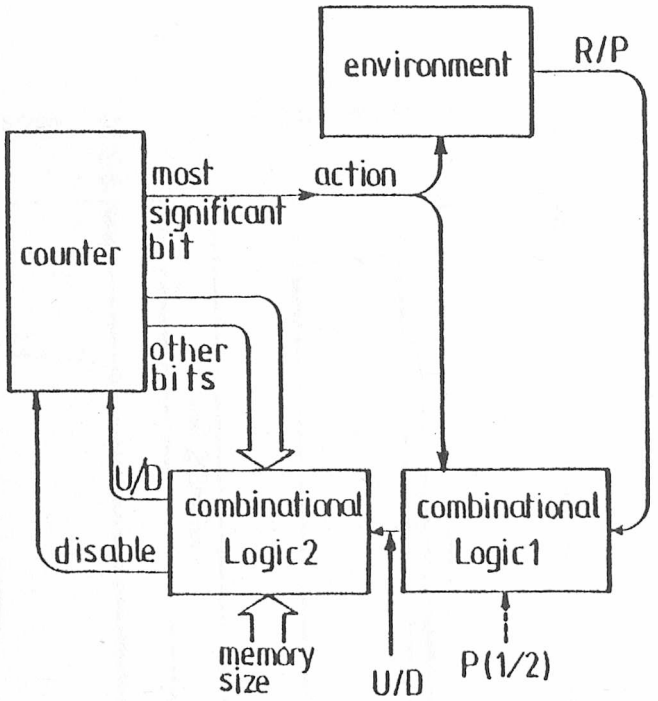


Figure 2 Schematic Diagram of Tsetlin Automaton with Modification for Krylov Automaton shown dotted

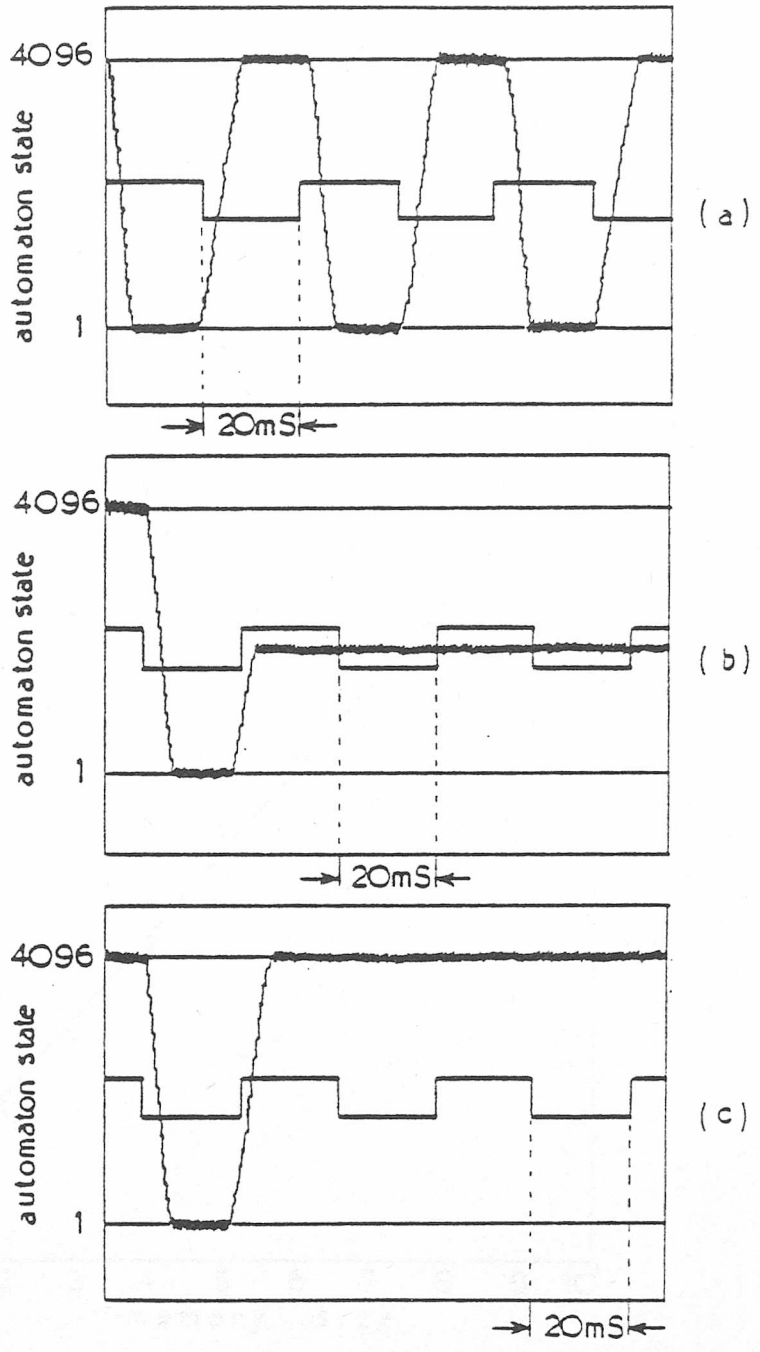


Figure 3 Examples of Operation of Tsetlin Automaton in Switched Environments

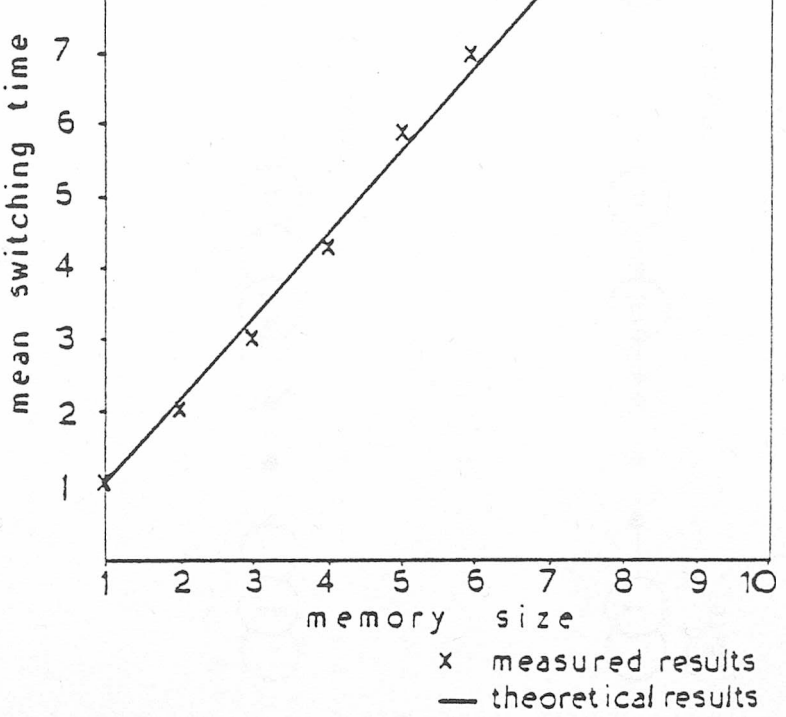
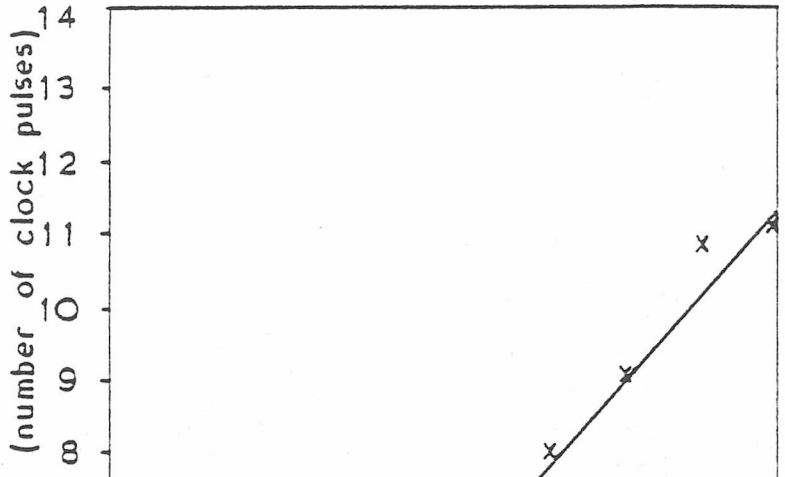


Figure 4 Mean Switching Times of Tsetlin Automaton


(number of clock pulses)
14
13
12
11
10
9
8

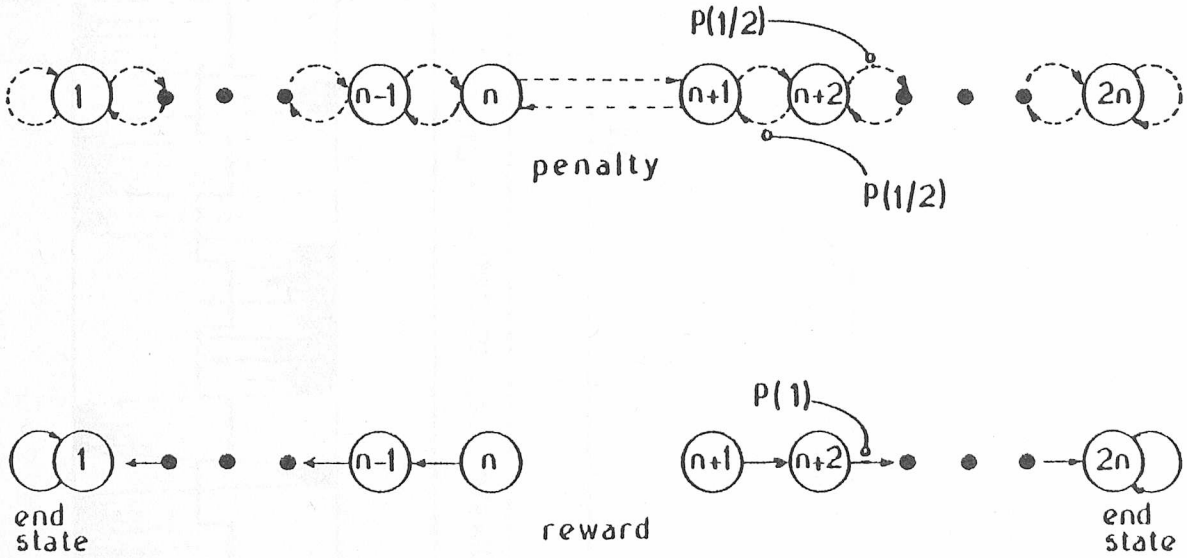


Figure 5 Operation of Krylov Automaton

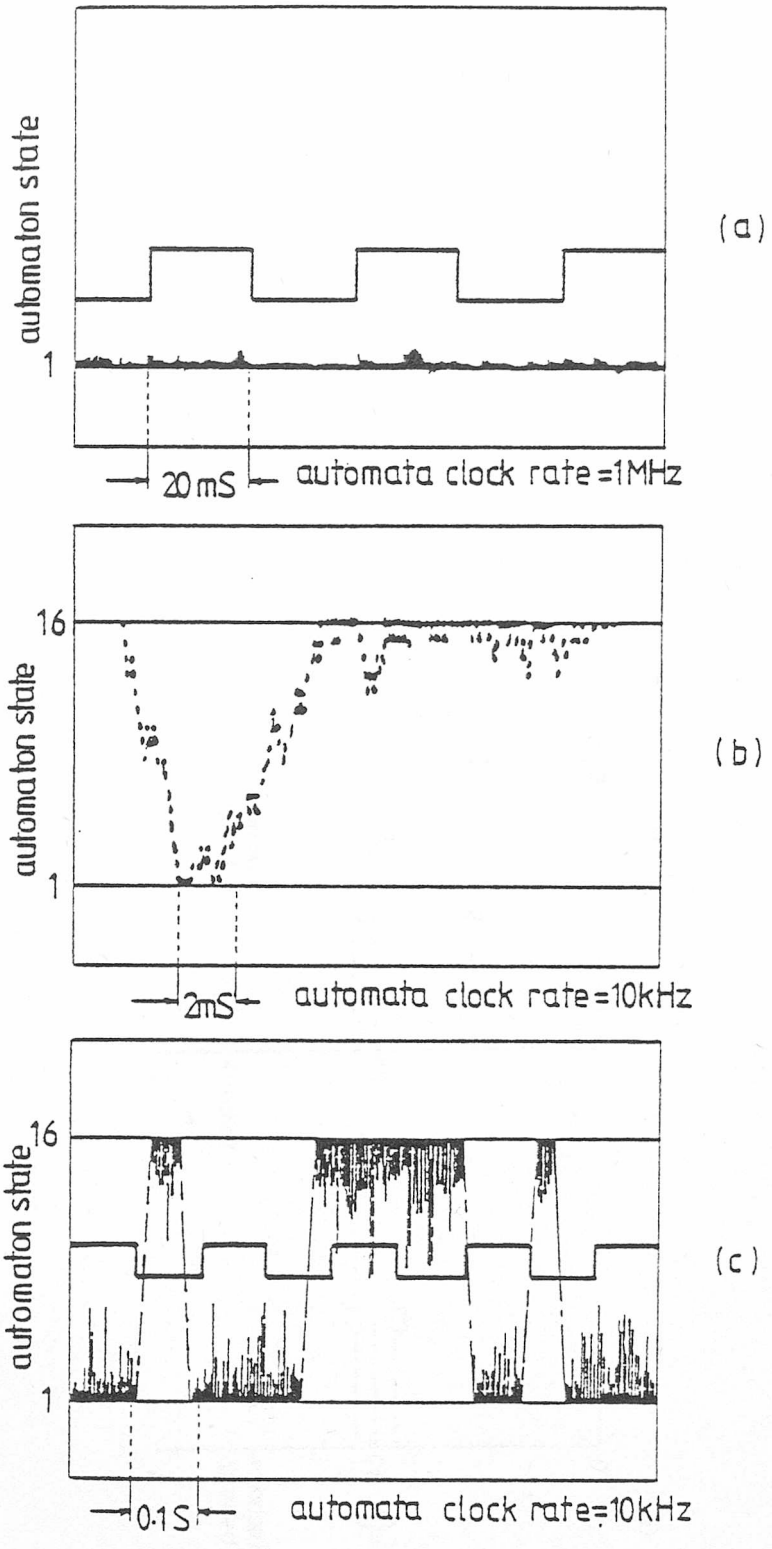


Figure 6 Examples of Operation of Krylov Automaton in Switched Environments

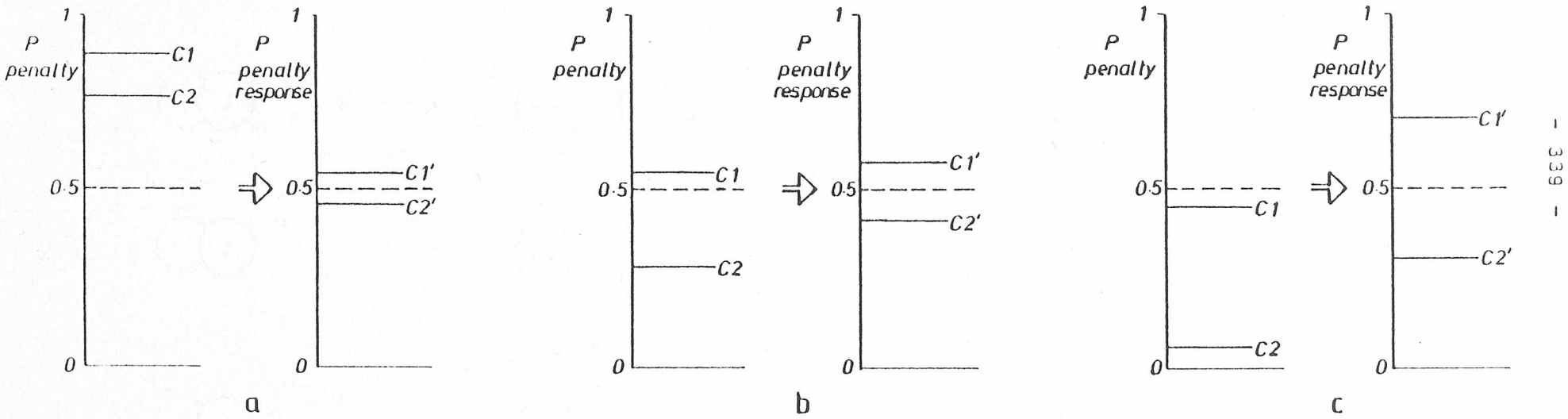


Figure 7 Effect of Modified Tsetlin Automata in Converting Penalty Probabilities into Penalty Response Probabilities

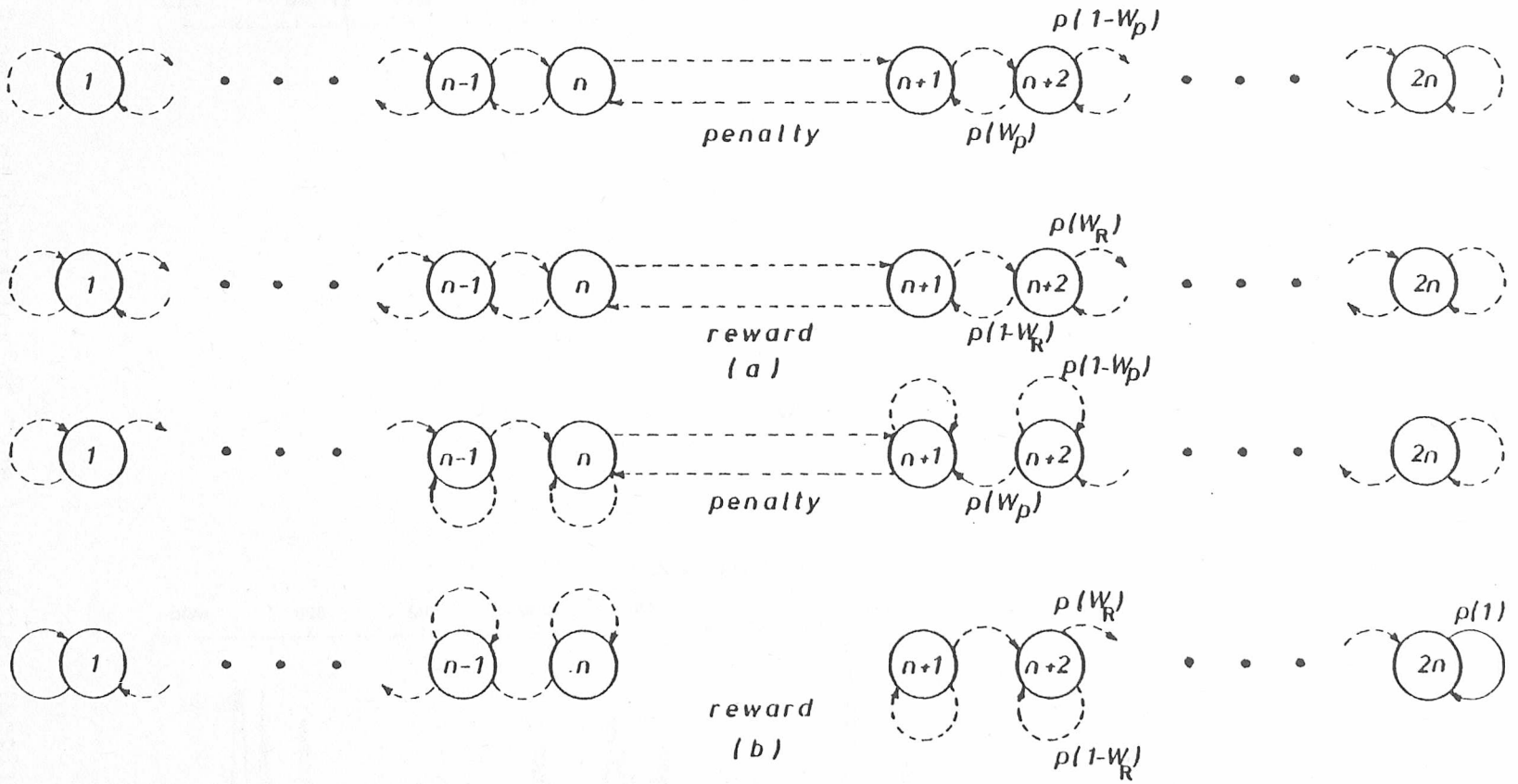


Figure 8 Operation of Modified Tsetlin Automata
(a) Type 1 (b) Type 2

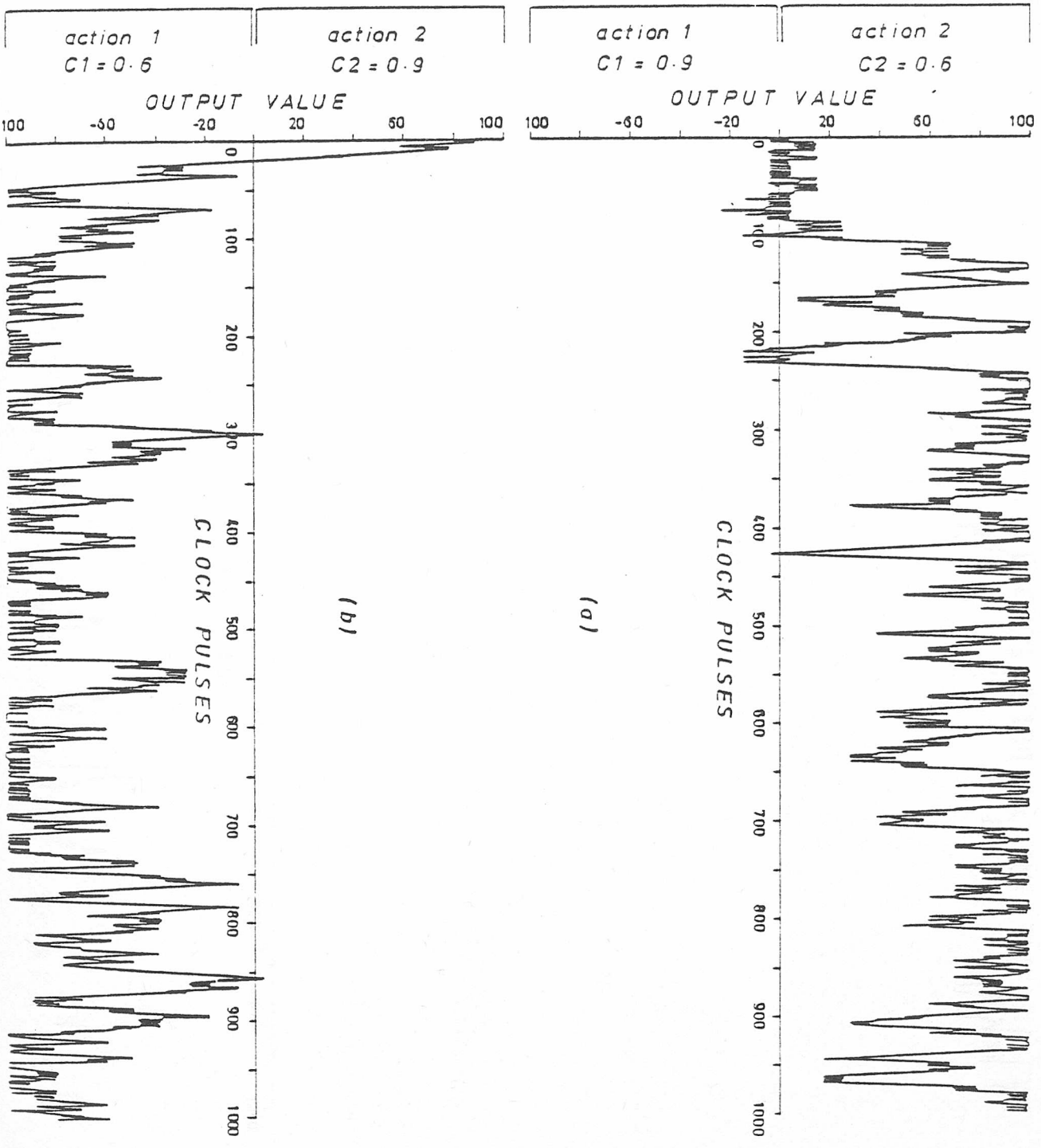


Figure 9 Example of Operation of Type 1 Modified Tsetlin Automaton in Switched Environment

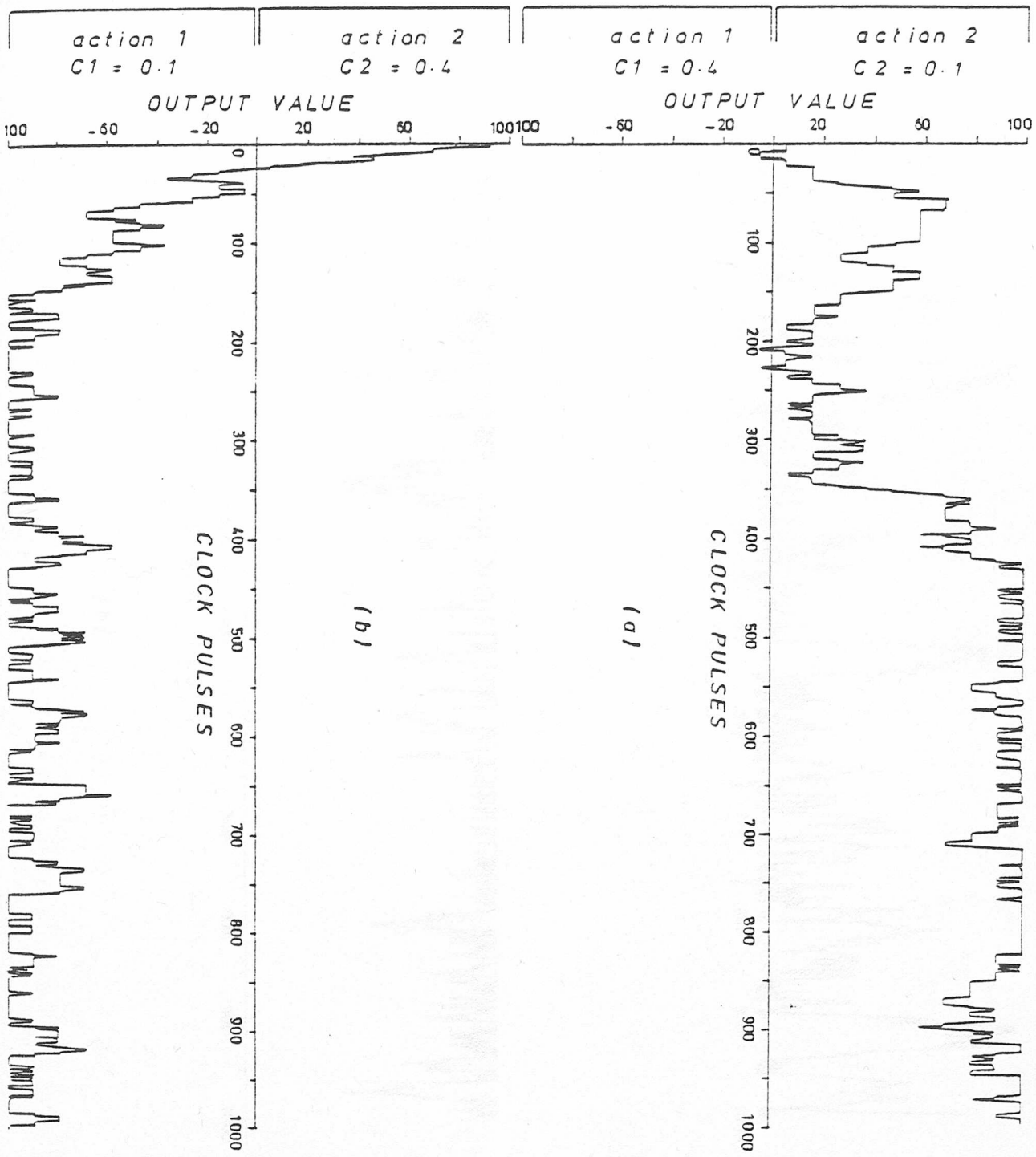


Figure 10 Example of Operation of Type 2 Modified Tsetlin Automaton in Switched Environment

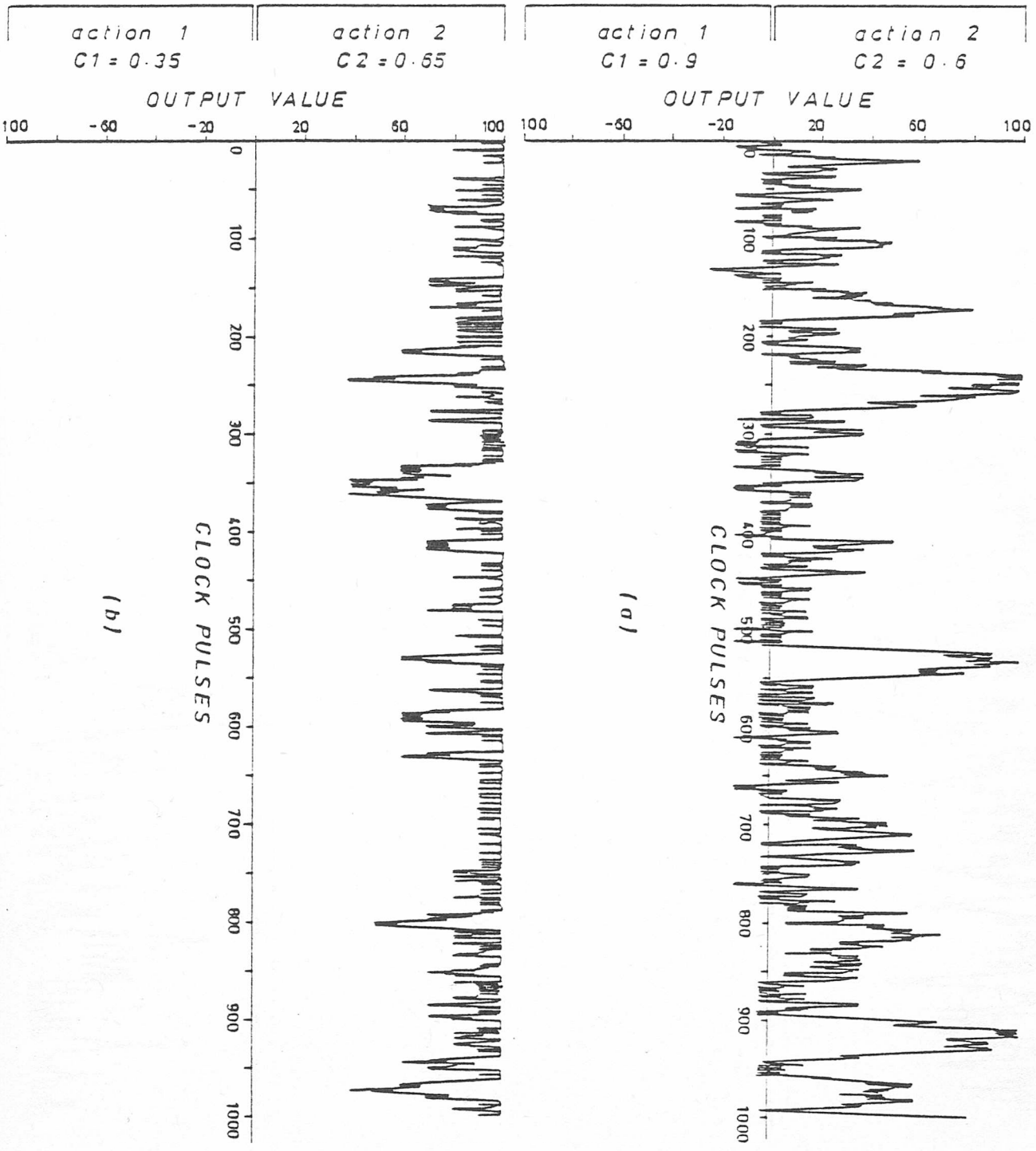


Figure 11 Examples of Operation of Tsetlin and Krylov Automata

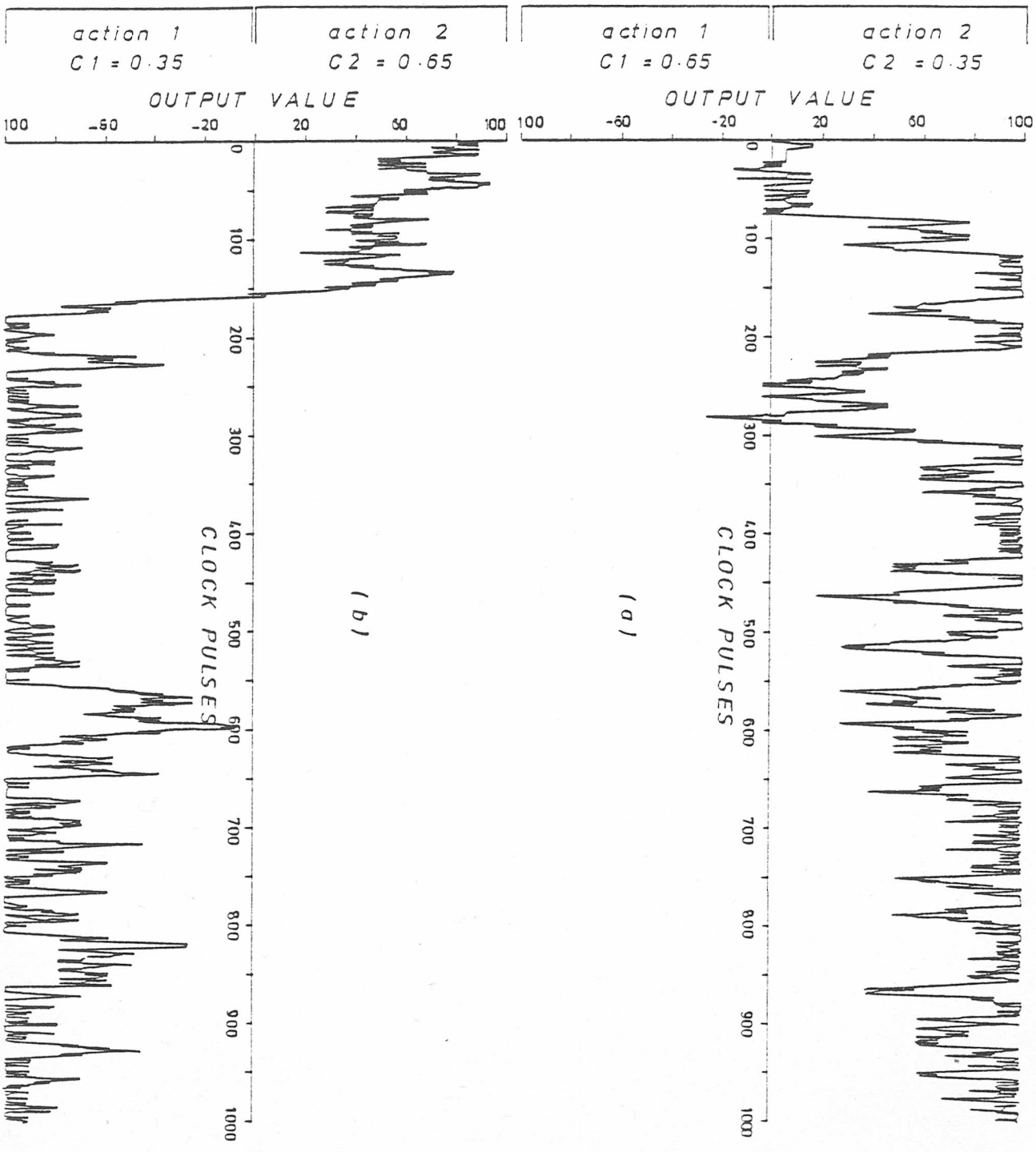


Figure 12 Example of Operation of Type 2 Modified Tsetlin Automaton showing Delayed Switching