INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of "sectioning" the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.

University Microfilms International
300 N. Zeeb Road
Ann Arbor, MI 48106

8314780

Meybodi, Mohammad Reza

LEARNING AUTOMATA AND ITS APPLICATION TO PRIORITY ASSIGNMENT IN A QUEUING SYSTEM WITH UNKNOWN CHARACTERISTICS

The University of Oklahoma PH.D. 1983

University
Microfilms
International 300 N. Zeeb Road, Ann Arbor, MI 48106

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __√__.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages ____✓
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Other _____

University
Microfilms
International

THE UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

LEARNING AUTOMATA AND ITS APPLICATION TO PRIORITY ASSIGNMENT IN A QUEUING SYSTEM WITH UNKNOWN CHARACTERISTICS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

DOCTOR OF PHILOSOPHY

BY

MOHAMMAD REZA MEYBODI

Norman, Oklahoma

1983

LEARNING AUTOMATA AND ITS APPLICATION TO PRIORITY ASSIGNMENT

IN A QUEUING SYSTEM WITH UNKNOWN CHARACTERISTICS

A DISSERTATION

APPROVED FOR THE SCHOOL OF ELECTRICAL ENGINEERING

AND COMPUTER SCIENCE

BY

S. Lakshmivarahan

Thesen

Jayaraj V

Calvin F. Minor

Leslie Mills

DISSERTATION COMMITTEE

To My Parents

ACKNOWLEDGMENTS

I am most grateful to Dr. S. Lakshmivarahan, my advisor and chairman, for the direction and encouragement given during the writing of this dissertation and the progression through the graduate program.

I also wish to express my appreciation for the constructive help given by the members of the dissertation committee: Dr. L. Miller, Dr. J. Minor, Dr. S. Dhall, and Dr. Vadiveloo.

A special thanks to Dr. S. K. Kahng, Chairman of the School of Electrical Engineering and Computer Science, for the help and thoughtful advice throughout my doctoral program.

Finally, I am deeply indebted to my wife, Malous, for her patience, understanding and enthusiasm.

ABSTRACT

Conditions for ε-optimality of a general class of absorbing barrier and strongly absolutely expedient learning algorithms are derived. As a consequence, a new class of learning algorithms having identical behavior under the occurrence of success and failure are obtained. An application of learning automata to the priority assignment in a queuing system with unknown characteristics is given.

TABLE OF CONTENTS

LEARNING AUTOMATA AND ITS APPLICATION TO PRIORITY ASSIGNMENT IN A QUEUING SYSTEM WITH UNKNOWN CHARACTERISTICS

CHAPTER I

INTRODUCTION

1.1 Learning System in Perspective

Learning has been studied by the psychologists for a long time, and in the last three decades by engineers and computer scientists. The aim of the psychologist or the mathematician has been to fit a mathematical model to the observed behavioral changes in an animal. A variety of mathematical models have been developed for this purpose. The work by Bush and Mosteller [B1], Luce [L1], Norman [N1], to mention a few, belongs to this category. The aim of the engineer and computer scientist, on the contrary, in the study of learning systems has been to build machines or write computer programs, perhaps in the context of pattern recognition or artificial intelligence, to learn a given task. For example, to build a machine to play chess, checkers, learn to read the letters of the English alphabet and so on (Fu [F1,F2], Tsypkin [T3], Mendel [M1], Waterman [W1], Findler [F3], Samuel [S1]).

Learning is defined as any relatively permanent change in behavior resulting from past experiences and a learning system is characterized by its ability to improve its behavior with time, in some sense tending towards an ultimate goal [N3]. The concept of learning makes it possible

to design systems which can gradually improve their performance during actual operation through a process of learning from past experience. Information feedback is inherent in learning. The concept of learning necessitates the need for a memory to store information about past experience which may be needed in the future. Thus the characteristic feature of learning is accumulation and usage of current information to eliminate the uncertainty due to insufficient a priori information and for the purpose of optimizing a certain performance criterion.

Usually the learning system uses information regarding the correctness and incorrectness of its response. This information is either derived by evaluation of the outcome of the selected course of action or else may be supplied by an external source called the teacher or supervisor. So, depending on whether or not external supervision is present, it is possible to distinguish two different types of learning systems: supervised learning and unsupervised learning. Unsupervised learning occurs in the case when the system does not receive any outside information except signals from the environment, and supervised learning occurs in the case when the system receives additional information from the outside during the learning process. In learning with supervision, it is assumed that at each instance of time it is known in advance the desired response of the learning system. We use the difference between the desired and actual response, that is, the error, to correct its behavior. Unsupervised learning, in general, is related to the problem of induction.

One of the most important principles in all learning theory is the law of reinforcement. This law governs how rewards and punishments produce changes in behavior. Reinforcement learning is a process by which

the response of a system is strengthened by reward and weakened by punishment. In the former situation, positive reinforcement occurs and increases the response probabilities, whereas in the latter situation negative reinforcement occurs and decreases the response probability. Reward and punishment represent favorable and unfavorable reactions to response, respectively.

In System Theory and Computer Science, learning has been implemented in various ways:

1. Heuristic programming technique
2. Bayesian technique
3. Inductive inferential technique
4. Stochastic approximation methods
5. Automata models.

In very complicated learning situations such as games of chess, checkers and programs for proving theorems, it is difficult to formulate the corresponding mathematical models. Heuristic approaches have been applied to this type of learning problem. A heuristic method is a rule of thumb, strategy, trick or any other kind of device which does not have a logical structure but which often leads to shortcuts. Heuristic methods emphasize achieving results that are good enough rather than optimal. Except for some introductory efforts [W1,F3], at the present time heuristics are all programmed in artificial intelligence. In other words, it is not the machine that discovers and selects the rules which are used in problem solving programs. A much better situation would be the one in which the heuristic processes are automated. Learning programs, initially inefficient and possibly even random in their actions would gradually formulate more and more heuristics on the basis of

experience. A particular procedure which has received the most attention learns to make decisions based on the values of weighted sums of factors related to that decision. Examples of this technique are the Samuel Polynomial, Signature tables and Move Phase table [G1]. Another machine learning technique which can be applied to the problem of learning heuristics is suggested by D. A. Waterman [W1]. Heuristics, represented as production rules, are created, evaluated and modified on the basis of information supplied by human or another program (explicit learning), or hypothesized during the normal course of problem solving (implicit training).

Bayes' theorem $P(A|B) = P(A)\ P(B|A)/P(B)$ is central to the Bayesian approach to learning. ($P(A)$, $P(B)$, $P(A|B)$ and $P(B|A)$ are the probability of A, probability of B, probability of A conditioned on B, and probability of B conditioned on A, respectively.) This theorem offers guidance for modifying judgments in the light of new experience. $P(A)$ is interpreted as the probability of a particular state of nature, future event or hypothesis, before obtaining additional information; that is, $P(A)$ is the a priori probability of A. $P(A|B)$ is interpreted as the revised value of this probability after receiving additional information and is known as posteriori probability. So, the value of $P(A)$ can be learned by repetitive application of Bayes' rule.

The problem in inductive inference is to detect (learn) regularities and common patterns in a body of data and use them for prediction. All induction problems can be shown to be equivalent to extrapolating a long sequence of symbols [S3, S4]. This sequence contains all data to be used in the prediction. Devising a grammar for

a language from a given set of strings (of symbols) from that language is an example of inductive inference. The constructed grammar can be used to determine if an arbitrary new string is a member of the language and also generates new strings belonging to that language.

The goal of many adaptive or learning systems is to find or learn the value of certain parameters which minimize a prespecified criterion function. The stochastic approximation method is a recursive scheme which can be applied for estimating the unknown parameters or to recover the unknown underlying distribution function when, due to the stochastic nature of the observation, measurements are subjected to random error.

Learning automata, the subject of this dissertation and stochastic approximation represent two related approaches to the learning problems; though both approaches introduce iterative procedure, updating at every stage is done in the parameter space in the stochastic approximation and probability space in automata models.

1.2 Learning Automata

The concept of learning automata was introduced by Tsetlin [T2]. He investigated the learning behavior of finite automata under stationary random environment and showed that finite automata which have appropriate properties "behave well" if memory capacity tends to infinity. A great deal of work in the Soviet Union and elsewhere has followed the trend set by this source paper. The work of Tsetlin was later followed in the Soviet Union by various authors: Krylov [K1], Krinski [K2] and Ponomarev [P1]. However, it is practically impossible to build automata with unbounded memory capacity. As an alternative,

Varshavskii and Vorontsova introduced variable structure automata [V1]. In this setup, state transition probabilities or state probabilities of the finite automata is updated in accordance with reinforcement scheme or learning algorithm.

The theory of learning automata is concerned with the analysis and synthesis of fixed or variable structure automata in a random environment.

Fixed Structure Automata: A fixed structure automata is a quintuple $\langle X, \phi, \alpha, F, G \rangle$.

I. X is the input set of the automata. The input of the automata at the instant k denoted by $x(k)$, is an element of set X. This set can be either a finite set or an infinite set, such as an interval on the real line, then

$$X = \{x_1, x_2, \ldots, x_n\} \quad \text{or} \quad x = [a,b]$$

where a,b are real numbers.

II. ϕ is the set of states of the automata. The state of the automata at any instant k, denoted by $\phi(k)$, is an element of the finite set $\phi = \{\phi_1, \phi_2, \ldots, \phi_s\}$ $(2 \leq s \leq \infty)$.

III. α is the output or action set of the automata. The action set of the automata at the instant k, denoted by $\alpha(k)$, is an element of the finite set $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_M\}$ $(2 < M < \infty)$.

IV. The transition function F: $\phi \times X \rightarrow \phi$ determines the state at the instant (k+1) in terms of the state and output at time k, that is,

$$\phi(k+1) = F[\phi(k), X(k)].$$

Function F can be either deterministic or stochastic.

V. The output function G: $\phi \rightarrow \alpha$ determines the output of the automata at any instant in terms of the state at that instant, that is,

$$\alpha(k) = G(\phi(k)).$$

G could be a stochastic function but there is not loss of generality in assuming it to be deterministic [P2]. By increasing the number of states it can be readily shown that an equivalent automata with the same input-output can be realized using a deterministic mapping G.

Remark 1.1: If the input set x of an automata contains only two elements 0 and 1, it is called a P-model. If x takes more than two but a finite number of values, each with a positive probability, then we have a Q-model learning automata. If x takes on values in a continuum, say for example $x \varepsilon [0,1]$, then we have an S-model learning automata.

Remark 1.2: The next state transition function F, may be specified by a transition matrix $[f_{ij}^{x}]$ for each $x \varepsilon X$. If these matrices contain only the elements 0 and 1, the automata is a fixed structure deterministic automata. If the elements lie in the interval [0,1], the automata is a fixed structure stochastic automata. The input to the automata depends on the environment in which the automata is operating.

Remark 1.3: In a fixed structure automata the output mapping G is generally assumed to be deterministic. G partitions the state set ϕ into r subsets, $m_i (i=1,2,\ldots,r)$, such that the elements of each subset m_i map into the same action α_k. When G is stochastic, a unique action need not correspond to a given state ϕ.

Remark 1.4: The output behavior of a learning automata for a given input

sequence can be obtained if the transition probabilities are known. However, it may frequently be important to know the probability with which the automata is in a particular state at a given instant. These probabilities are known as state probabilities and an alternative description of the operation of the automata can be given in terms of these probabilities. At any time instant k, the automata being in state $\phi(k) = \phi_i(k)$ is governed by the probability distribution $\pi_i(k)$ such that

$$\sum_{i=1}^{s} \pi_i(k) = 1 \qquad \text{for all } k.$$

Let $\pi(k) = (\pi_1(k), \pi_2(k), \ldots, \pi_s(k))^T$, where T denotes the transpose and

$$\pi_i(k) = \text{Prob}[\phi(k) = \phi_i(k)].$$

One can also consider the action probability vector P(k) where the i^{th} component $p_i(k)$ is given by

$$p_i(k) = \text{Prob}[\alpha(k) = \alpha_i].$$

If G is deterministic, then we can have

$$p_i(k) = \sum_{j \epsilon m_i} \pi_j(k).$$

If G is the identity mapping, there is a unique state associated with each action. In this case, the state probability uniquely determines the action probability.

Variable Structure Automata: In fixed structure automata, the probabilities of state transitions are fixed. Variable structure automata updates either the state transition probabilities or equivalently the state probabilities on the basis of the output from the environment.

The automata, in this case, is represented by the sextuple $< x, \phi, \alpha, p, G, T >$, where x is the input set, ϕ is the set of states, α is the action set, and p is the state probability vector governing the choice of the state at each stage (i.e., at each stage k, $p(k) = (p_1(k), p_2(k), \ldots, p_s(k))^T$), and G is the output mapping. In this dissertation, G is taken to be deterministic and one-to-one (i.e., the number of actions is equal to the number of states; state and action are regarded synonymous). T is called an updating scheme or learning algorithm which is used to modify the state probability vector, that is, it generates p(k+1) from p(k).

Environment: The environment in which the automata operates is represented by the triple $< \alpha, x, D >$. The environment has random response characteristics. It has input $\alpha(k) \in \{\alpha_1, \alpha_2, \ldots, \alpha_M\}$ and output belonging to the set x. In a P-model situation, where $x(k) \in \{0,1\}$ the environment is characterized by the success probability vector $D = (d_1, d_2, \ldots, d_M)$ where

$$d_i = \text{Prob}[x(k) = 1 \mid \alpha(k) = \alpha_i].$$

If the d_i's don't depend on k, the environment is said to be stationary, otherwise it is nonstationary. If $E_1, E_2, \ldots, E_n$ are stationary environments which can themselves be considered to be states of a Markov Chain, we have the Markovian Switching Environment [T2]. If the $d_i(k)$ vary periodically with time, the environment is called the Periodic Environment [T2][N7].

In the P-model which is our interest, when the automata is in state $\phi_i(k)$ at time k, the random variable x(k) takes only two values x(k) = 1, known as the success input with probability d_i and x(k) = 0

known as the penalty input with probability $c_i = 1-d_i$. In other words, the output x(k) of the environment is a random variable having a two-point binomial distribution

$$f[x(k) \mid \phi(k) = \phi_i] = d_i^{x(k)} \; c_i^{(1-x(k))}.$$

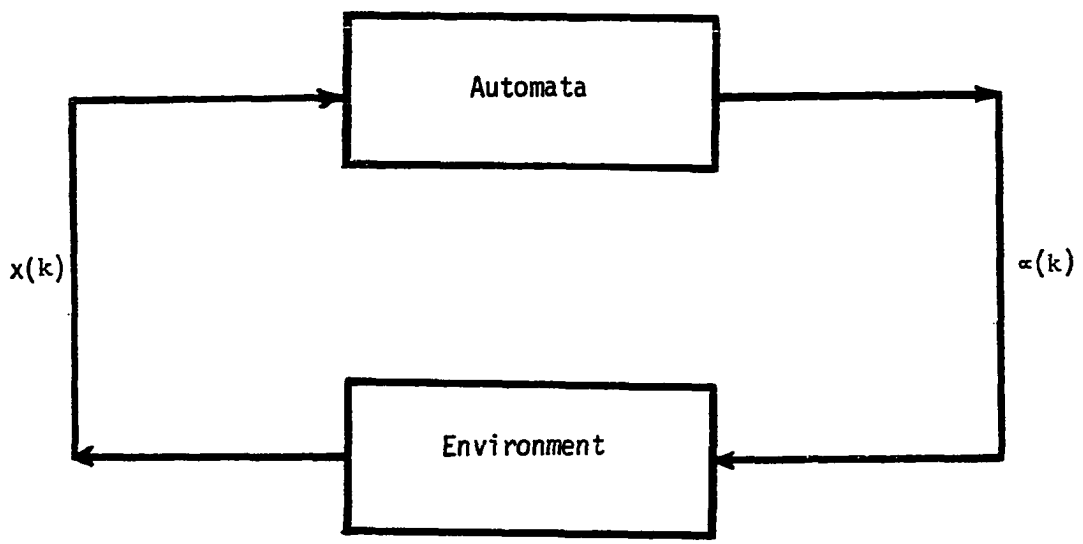
Let

$$d_1 > d_2 > d_3 \ldots > d_M \qquad (1.1)$$

and $d_i \in (0,1)$ for all i = 1 to M. For them let $D = (d_1, d_2, \ldots, d_M)$. It should be noted that the above assumptions are not really restrictive since when there is more than one state with the same d_i we can merge these states together.

So far we have considered the structure and properties of the environment and automata in isolation. Figure (1) represents a feedback connection of an automata and an environment. The actions of the automata form the inputs to the environment and the responses or outputs of the environment in turn are the inputs to the automata. Starting from an initial state $\phi(0)$, the automata generates the corresponding action $\alpha(0)$. The response of the environment to this input is x(0), which in turn changes the state of the automata to $\phi(1)$. This sequence of operations is then repeated to result in a sequence of states, actions and responses of the environment. In the case of Variable Structure Learning Automata, the action probability vector p(k) (or the state transition matrix) gets updated at each state $\phi(k)$.

Norms of Behavior: To judge the learning process objectively, it is necessary to set up quantitative norms of behavior. One quantity useful in judging the behavior of a learning automata is the average probability of success at time k given by



Learning Automata

Fig. 1.1

$$\eta(k) = E[x(k) \mid p(k)] = \text{Prob}[x(k) = 1 \mid p(k)]$$

$$= \sum_{i=1}^{M} \text{Prob}[x(k) = 1 \mid p(k), \alpha(k) = \alpha_i] \text{ Prob}[\alpha(k) = \alpha_i]$$

$$= \sum_{i=1}^{M} d_i \, p_i(k).$$

If no prior information is available, there is no basis on which the different action $\alpha_i (i=1,\ldots,M)$ can be distinguished. In such a case, one would choose each action with equal probability (i.e., by pure choice). In this case the value of average success is denoted by η_0 and is given

$$\eta_0 = \frac{1}{M} \sum_{i=1}^{M} d_i.$$

The use of the term learning automata can be justified if the average success is made greater than η_0, at least asymptotically. Such a behavior is called expedient and was introduced for the first time by Tsetlin [T2]. Tsetlin defined it in the context of deterministic learning automata. When a learning automata is expedient it only does better than the one which chooses actions in a purely random manner.

Definition 1.1: A learning automata is called expedient if

$$\lim_{k \to \infty} E[\eta(k)] > \eta_0.$$

Remark 1.6: In the case of fixed structure automata of Tsetlin, $\pi(k)$ and hence $\eta(k)$ tends to be constant with probability one and

$$\lim_{k \to \infty} E[\eta(k)] = \lim_{k \to \infty} \eta(k).$$

Definition 1.2: A learning automata is called optimal if

$$\lim_{k \to \infty} E[\eta(k)] = d_1.$$

From the definition of optimality it is evident that optimality

implies that the action α_1 associated with the maximum probability d_1 is chosen with probability one. While optimality appears very desirable in a stationary environment, it may not be possible to achieve it in a given situation. In such a case one might aim at a sub-optimal situation. One such property is given by ε-optimality.

Definition 1.3: A learning automata is said to be ε-optimal if

$$\lim_{k \to \infty} | E[\eta(k)] - d_1 | < \varepsilon$$

for any arbitrary $\varepsilon > 0$ by a proper choice of the parameters of the automata.

It is possible that ε-optimality holds only when the values of d_i's satisfy certain restrictions, for example, that they should lie in certain intervals. In such cases we have conditional ε-optimal. In practice, the success probabilities are often completely unknown, and would be interested in automata which exhibit the desired behavior in an arbitrary environment. The performance would also be superior if the increase of $E[\eta(k)]$ is monotonic. These requirements are met by an absolutely expedient automata which is defined below.

Definition 1.4: A learning automata is said to be absolutely expedient if

$$E[\eta(k+1) \mid p(k)] \geq \eta(k)$$

with probability one for all $p(k) \in S_M$ and for all D satisfying (1.1) with strict inequality holding good for all $p \in S_M^0$ where

$$S_M = \{p \mid p = (p_1, p_2, \ldots, p_M)^T, \quad 0 \leq p_i , \quad \sum_{i=1}^{M} p_i = 1\}$$

and

$$S_M^0 = \{p \mid p = (p_1, p_2, \ldots, p_M)^T, \quad 0 < p_i , \quad \sum_{i=1}^{M} p_i = 1\} .$$

The concept of absolute expediency was first introduced by Lakshmivarahan and Thathachar [L2]. Since its introduction, this concept has played a major role in the analysis and design of learning algorithms. Absolute expediency implies that $\eta(k)$ is a submartingale and that $E[\eta(k)]$ is strictly monotonically increasing within all stationary environment. If $\eta(k) > \eta_0$ for some k, absolute expediency implies expediency. Furthermore, it can be shown that absolute expediency implies ε-optimality in all stationary random environment [L3].

Definition 1.5: A learning automata is said to be strongly absolutely expedient if

$$E[\eta(k+1) \mid p(k)] \geq \eta(k)$$

with probability one for all d satisfying (1.1) and for all $p \in S_M$ with strict inequality holding good for all $p \in (S_M - V_M)$ where

$$V_M = \{e_i \mid i=1,2,\ldots,M\}$$

where $e_i = (0,0,0,\ldots,1,\ldots,0)$ be the i^{th} unit vector of dimension M.

Clearly, V_M corresponds to the corners of simplex S_M. S_M^0 is called the interior of S_M and $\delta S_M = S_M - S_M^0$ is called the boundary The concept of strongly absolutely expedient was introduced for the first time by Meybodi and Lakshmivarahan in [M5]. Except for the absolutely expedient and strong absolutely expedient, all the other norms of behavior are applicable to both fixed structure and variable structure automata. There is no absolutely expedient or strongly absolutely expedient fixed structure automata at the present time.

Learning Algorithm: Variable structure automata updates either the transition probabilities or the action probabilities on the basis of the output from the environment. It is evident from the description of the

learning automata that the crucial factor that affects the performance is the reinforcement scheme (learning algorithm) for updating the action probabilities.

A general scheme for updating the action probabilities can be represented as follows:

Let α_i be the action chosen at time k as a sample realization from the distribution p(k). Then p(k+1) is defined as follows:

$$\begin{aligned} p_i(k+1) &= p_i(k) + \theta f_i^i[p(k)] \\ p_j(k+1) &= p_j(k) - \theta f_j^i[p(k)] \quad j \neq i \quad \text{if success occurs} \end{aligned} \tag{1}$$

and

$$\begin{aligned} p_i(k+1) &= p_i(k) - \theta g_i^i[p(k)] \\ p_j(k+1) &= p_j(k) + \theta g_j^i[p(k)] \quad j \neq i \quad \text{if failure occurs} \end{aligned}$$

where $f_j^i : S_M \to [0,1]$, $g_j^i : S_M \to [0,1]$, $i,j \in \{1,2,\ldots,M\}$ are nonnegative continuous functions and $0 < \theta \leq 1$ is called the step length parameter. The following consistency condition

(C1)
$$\begin{aligned} &\underline{\text{either}} \quad f_j^i[p] \equiv 0 \qquad \text{for all } p \in S_M \\ &\underline{\text{or}} \quad\;\; f_j^i[p] \leq p_j \qquad j \neq i \quad \text{and} \\ &\qquad\quad f_i^i[p] = \sum_{j \neq i} f_j^i[p] \end{aligned}$$

and

(C2)
$$\begin{aligned} &\text{either} \quad g_j^i[p] \equiv 0 \qquad \text{for all } p \in S_M \\ &\text{or} \quad\;\; g_i^i[p] \leq p_i \qquad \text{and} \\ &\qquad\quad g_i^i[p] = \sum_{j \neq i} g_j^i[p] \end{aligned}$$

for all i, j = 1,2,...,M, implies that $p(k+1) \in S_M$ if p(k) does. However, in order to make the algorithm nontrivial and interesting either $f_j^i \equiv 0$

or $g_j^i \equiv 0$ but not both. The continuity assumption of f_j^i and g_j^i is one of mathematical convenience. The fact that both f, g are nonnegative maintains the reward and penalty nature of updating. If $f_j^i \not\equiv 0$ but $g_j^i \equiv 0$ then (1) is called the reward-inaction algorithm, if $f_j^i \equiv 0$ but $g_j^i \not\equiv 0$ then it is called the inaction-penalty algorithm, and if both f_j^i and g_j^i are nonzero then the algorithm is called the reward-penalty algorithm.

Let $I = \{1,2,\ldots,M\}$ and $E = \{\text{success,failure}\}$. The learning algorithm (1) defines a mapping

$$T : S_M \times I \times E \to S_M ,$$

where

$$p(k+1) = T[p(k), i(k), e(k)] ,$$

where $i(k) \in I$ denotes the action chosen and $e(k) \in E$ the event occurring at time k. E is called the event space, p(k) and e(k) are known as the state and event sequence respectively. Let T* be an extension of T defined as follows:

$$T^* : S_M \times (I \times E)^n \to S_M \quad \text{for all } n \geq 1 ,$$

where $(I \times E)^n$ refers to the n-fold cartesian product of $I \times E$ and

1) $T^* = T$ for $n = 1$

2) $T^*[p,\{(i_0,e_0), (i_1,e_1) \ldots (i_n,e_n)\}]$

$$= T[\ldots T[T[T[p, i_0, e_0], i_1, e_1] \ldots i_n, e_n]] ,$$

where $i_j \in I$ and $e_j \in E$ for all $j = 0,1,2,\ldots,n$ and $n \geq 2$. Define

$$\kappa: S_M \times (I \times E) \to [0,1] \text{ as}$$

$$\text{Prob}[(i,e) \mid p] = \kappa[p, (i,e)]$$

and

$$\text{Prob}[(i(k+1),e(k+1))|p(0)=p,(i(1),e(1)) \quad 1=0,1,\ldots,k]$$

$$= \kappa[p',\ (i(k+1),\ e(k+1))]\ ,$$

where

$$p' = T^*[p(0),\{(i(0),e(0)),(i(1),e(1)),\ldots,(i(k),e(k))\}]\ .$$

Clearly

$$\kappa[p,\ (i,e)] = p_i d_i \quad \text{if} \quad e = \{\text{success}\}$$

$$= p_i\ c_i \quad \text{if} \quad e = \{\text{failure}\}.$$

That is $\kappa[p,\cdot]$ is the event probability distribution.

Definition 1.6: A state $p \in S_M$ is called an absorbing state of the algorithm T if

$$T[p,\ i,\ e] = p$$

for all $(i,e) \in I \times E$ with probability one. A learning algorithm is said to be absorbing if and only if there is at least one absorbing state. If the absorbing states are the vertices of the simplex S_M, the algorithm is called the absorbing barrier algorithm.

The basic idea behind a learning algorithm is a rather simple one. If the automata selects an action α_i at time k and success $(x(k) = 1)$ results, the action probability $p_i(k)$ is increased and all the other components of vector p(k) are decreased. For failure $(x(k) = 0)$ $p_i(k)$ is decreased and all the other components are increased.

Learning algorithms are generally classified either on the basis of the behavior of the automata using the scheme, e.g., expedient, optimal, etc., or the nature of mapping T. That is, the nature of the function appearing in the algorithm, e.g., linear, nonlinear. If p(k+1) is a linear function of the component of p(k), the learning algorithm is said to be linear, otherwise it is nonlinear. It is also

possible to classify the learning algorithm depending on whether it is an absorbing barrier algorithm or non-absorbing barrier algorithm (Ergodic algorithm). The non-absorbing barrier algorithm generates a Markov process that is ergodic, whereas, the absorbing barrier algorithms result in Markov process with more than one absorbing barrier. Initial values of action probabilities do not affect the asymptotic behavior in the case of ergodic scheme, whereas in the case of absorbing barrier algorithms, the behavior of p(k) for large k is very much a function of initial values of the action probabilities. It can be shown [L3] that the distribution of p(k) generated by an ergodic algorithm after suitable normalization is normal; in other words, in the case of ergodic algorithm p(k) converges in distribution, but p(k) when generated by an absorbing barrier algorithm converges with probability one to a random vector whose range is a discrete (finite) set. It may be noted that a systematic study of ε-optimal Ergodic Algorithm is relatively new and for the first time discussed in the book by Lakshmivarahan [L3]. In [L3] a time varying algorithm where θ changes in time is discussed. Analysis of this latter class of algorithm depends on the stochastic approximation methods.

Various learning algorithms-- linear and nonlinear, have been reported in the literature. The linear reward-penalty (L_{R-P}) scheme is one of the earliest schemes considered in mathematical psychology. The properties of this scheme have been studied in detail by a number of researchers in this field [B1, V1, M4, C1, V2]. It is known that an automata using linear reward-penalty scheme is expedient in all stationary random environment. The linear reward-inaction (L_{R-I}) scheme

is another simple linear scheme which is derived by a modification of the L_{R-P} scheme. This scheme was considered first in mathematical psychology [B1] but was later independently conceived and introduced into the engineering literature by Shapiro and Narendra [S1, S10]. The characteristic of the scheme is not to change the action probabilities whenever an unfavorable response resulted from the environment, following a favorable response, however, the probability of the action is increased as in L_{R-P} scheme. Because of this property, a learning automata using this scheme has been called a "benevolent automata" by Tsypkin and Poznyak [T4]. The L_{R-I} scheme was originally reported to be optimal in all stationary random environment, but it is now known that it is only ε-optimal [V3, L11]. L_{R-P} and L_{R-I} schemes can be considered as prototypes for the behavior encountered in many other complex schemes that have been investigated. Other possible combinations such as L_{R-R}, L_{P-P} and L_{I-P} have been studied in [V5]. It is shown that L_{I-P} scheme is always expedient and L_{P-P} is expedient only when an action is less penalized when it causes a nonpenalty response than the case when it causes a penalty response from the environment. L_{R-R} scheme is expedient if an action is more rewarded when it produces a penalty than if it produces a non-penalty.

The first nonlinear scheme for two state automata by Varshavskii and Vorontsova [V1] was in terms of transition probabilities. This can be used in the action probability version of the scheme. This scheme is ε-optimal in a restricted random environment satisfying either $d_1 < 1/2 < d_2$ or $d_2 < 1/2 < d_1$. Several nonlinear schemes which are ε-optimal in all random environment have been suggested by Viswanthan

and Narendra [V2], Sawargi and Baba [S6], McMurtry and Fu [M2], Fu [F4], and Lakshmivarahan and Thathachar [L2]. Instead of purely nonlinear and purely linear schemes, it is possible to combine them to get what is called a hybrid scheme [V5].

In [A1], Aso and Kimura extended the class of the absolutely expedient learning algorithm. In the earlier works, the way of updating the probabilities used in selecting an action depended on reactions and dichotomy of actions: the selected and not selected; that is, the functions that are used in updating are independent of the action chosen [M2, N1-N3, F4, L2, L4, S6, V1]. In [A1] updating depends on which action is selected as well (Algorithm (1)). Interestingly, Aso and Kimura called this class of learning algorithm as "Stochastic Vector Automata" algorithm. The class of learning algorithms represented by algorithm (1) is quite a general one and subsumes most of the known schemes available. We give below some of the early schemes in terms of general learning algorithm (1).

Scheme 1 (Linear Reward-Inaction Scheme): In algorithm (1) if we set $f_j^i[p] = c\, p_j$ and $g_j^i[p] \equiv 0$ for all i, j, $i \neq j$, where $0 < c < 1$, we get the linear reward-inaction algorithm. This scheme is ε-optimal.

Scheme 2 (Beta Model): This scheme which was originally proposed by Luce [L1] in mathematical psychology as an alternative to Bush and Mosteller's L_{R-P} scheme (referred to as the α-model) can be obtained by setting

$$f_j^i[p] = \frac{(b-1)\, p_j\, (1-p_j)}{b(1-p_j) + p_j} \quad ,$$

$$g_j^i[p] = \frac{(b-1)\, p_j}{(1-p_j) + b\, p_j} \quad ,$$

where $b > 1$. This scheme is expedient in a restricted environment.

Scheme 3 (Vorontsova): This scheme can be obtained by setting

$$f_j^i[p] = a\,\phi(p_1, 1-p_1)p_j^{\theta+1}(1-p_j)^{\theta} ,$$

$$g_j^i[p] = b\,\phi(p_1, 1-p_1)p_j^{\theta+1}(1-p_j)^{\theta} ,$$

where

$$\phi(p_1, 1-p_1) = \phi(1-p_1, p_1) .$$

This scheme applies only to a two-action automata. It is, however, the first nonlinear absolutely expedient scheme put forward, though at that time the concept of absolute expediency was unknown. This scheme is also ε-optimal.

Scheme 4 (Viswanathan and Narendra): This scheme can be obtained by setting

$$f_j^i[p] = p_j[a_1 + a_2 p_j^{\theta+1}(1-p_j)^{\theta}] ,$$

$$g_j^i[p] = b\,p_j^{\theta+1}(1-p_j)^{\theta} ,$$

where the positive constants a_1, a_2, b are to be chosen properly to satisfy consistency conditions (C1)-(C2). This scheme is absolutely expedient.

Scheme 5 (Lakshmivarahan and Thathachar): By setting $f_j^i[p] = \lambda(p)p_j$ and $g_j^i[p] \equiv 0$ for all i and j, $i \neq j$, where $0 < \lambda(p) < 1$, and assuming that function $\lambda(p) = 0$ only if p is a unit vector we get the nonlinear reward-inaction reported in [L2]. This scheme is ε-optimal and absolutely expedient.

Aso and Kimura derived necessary and sufficient conditions for learning algorithm (1) to be absolutely expedient. Conditions on functions g_j^i and f_j^i given by Aso and Kimura do not guarantee that V_M is the

only set of absorbing barriers for the automata, and so they do not guarantee ε-optimality for the automata. At present there are a variety of algorithms which are ε-optimal. It is interesting to note that almost all of these algorithms have asymmetric behavior with respect to the occurrence of success and failure. It should be mentioned that the choice of functions f_j^i and g_j^i considered by Aso and Kimura do not always induce identical behavior by the algorithm (1) under success and failure. Very recently Herkenrath et al. [H1] have derived necessary and sufficient conditions for the same class of learning algorithm considered by Aso and Kimura to be an absorbing barrier algorithm.

This thesis consists of two parts: theory and application. In the first part necessary and sufficient conditions for Algorithm (1) to be strongly absolutely expedient are derived. As a consequence, ε-optimality of this class of algorithm is obtained. The choice of functions $g_j^i[p]$ given in this part induce identical behavior of the algorithm (1) under the occurrence of success and failure. Conditions on functions $g_j^i[p]$ and $f_j^i[p]$ presented guarantee that V_M is the only set of absorbing barriers for all three types of algorithms: reward-penalty, reward-inaction and inaction-penalty. This is in sharp contrast with the properties of currently available absolutely expedient learning algorithm [L10, L4, A1, S6] wherein the reward-penalty and reward-inaction algorithms are absorbing barrier type, but inaction-penalty is not. In fact, in all the inaction-penalty algorithms of the absolutely expedient type known so far, every state in $\delta S_M = S_M - S_M^0$ is an absorbing state. The modified definition of strong absolute expediency is in fact motivated by the existence of the absorbing

barrier algorithms of the reward-penalty, reward-inaction and inaction-penalty types. For algorithm (1) under the choice of functions given in part 1, if p(k) reaches the boundary of Simplex S_M, then p(k) will continue to remain in that boundary, that is the dimensionality of the algorithm is reduced automatically. It is shown that strong absolute expediency leads to ε-optimality in all stationary random environment. In part 2 of this thesis an application of learning automata to the priority assignment in a queuing system with unknown characteristics is given.

Mathematical tools needed for the analysis and design of learning automata are Martingale theory and theory of Markov process which are readily available in the literature [N1, D1, L3]. Pioneering work for the analysis of variable structure learning automata has been done by Norman [N1] for the L_{R-I} scheme for the two state automata, and later extended by Lakshmivarahan and Thathachar in two directions-- to non-linear schemes and to multi-state automata. The recent book by Lakshmivarahan [L3] provides a comprehensive treatment of this subject. The state sequence of fixed structure learning automata operating in a stationary random environment is a homogeneous Markov chain which is ergodic. The Markov chain corresponding to the variable structure learning automata is non-homogeneous. In the case of variable structure learning automata which updates its action probabilities, the action probability vector p(k) may itself be regarded as the state of the Markov process at instant k. The state space is now the unit simplex of dimension M (which is the same as the number of actions of the automata). For example, in a two action automata the state space reduces

to the unit interval [0,1] on the real line. If p(k+1) depends on p(k) and not explicitly on k, choice of this state space makes p(k) a homogeneous Markov process having a continuous state space but operating in discrete time.

Learning automata has found application in a number of situations such as: hypothesis testing [L9], game theory [L6, L7, L8, V4], pattern recognition [L5] and parameter optimization [S7, M2]. The most significant of the applications has been the problem of adaptive routing in large network. Learning approach to routing problem in telephone and data communication has been extensively studied in [N5, N6, S8, S9].

1.3 Organization of the Thesis

In chapter 2 necessary and sufficient conditions for strong absolute expediency, which is the principal theme of chapter 2, are derived. It turns out that these conditions are general but simple conditions of symmetry of the functions used in the learning algorithm. The convergence properties of this class of learning algorithm is studied. Bounds on the probability of convergence to the desired state is obtained. Using the bound it is shown that strong absolute expediency indeed is a sufficient condition for ε-optimality.

In chapter 3 an application of learning automata to the priority assignment in a queuing system with unknown characteristics is given. Different mathematical models for a queuing system in which the priorities are worked out directly without a priori knowledge of the input and service characteristics are presented. This situation is modeled in the framework of a single server priority queuing system when there are only two classes of job and preemption of the service

is not allowed. It is shown that these systems gradually improve their performance and finally converge to the right priority with probability as close to unity as desired. A variety of simulation results are also included.

The concluding chapter 4 summarizes the thesis and presents some suggestions for further work.

CHAPTER II

STRONG ABSOLUTE EXPEDIENCY AND ε-OPTIMALITY OF A GENERAL CLASS OF LEARNING ALGORITHM

2.1 Introduction

In the previous chapter, the basic formulation of learning automata has been briefly reviewed. In fixed structure automata, the probabilities of state transitions are fixed. Variable structure automata updates either transition probabilities or the state probabilities on the basis of the input to the automata. The learning algorithm has been proposed for modifying state transition probabilities or state probabilities. The new transition probabilities or state probabilities reflect the information which the automata has received from the input and consequently provide the ability to improve its performance.

In this chapter a new class of absorbing barrier algorithm of the reward-penalty type which has identical behavior under the occurrence of success and failure is discussed. Necessary and sufficient conditions for strong absolute expediency and convergence of the algorithm with probability one are established. As a consequence, ε-optimality of this class of algorithm is obtained. A number of computer simulation results are also given in this chapter.

2.2 Convergence with Probability One

In this section conditions on the algorithm 1.1 are derived such that the Markov process $\{p(k)\}\ k \geq 0$ converges with probability one. At first the consistency conditions (1.2) are rewritten in a form more suitable for the analysis. Let

$$\begin{aligned} f_i^i[p] &= \alpha[i,p](1 - p_i(k)) , \\ f_j^i[p] &= \beta[i,j,p]p_j(k) , \\ \sum_{j\neq i} \beta[i,j,p]p_j(k) &= \alpha[i,p](1-p_i(k)) , \end{aligned} \tag{C.1}$$

and

$$\begin{aligned} g_i^i[p] &= \gamma[i,p]p_i(k) , \\ g_j^i[p] &= \delta[i,j,p](1-p_j(k)) , \\ \gamma[i,p]p_i(k) &= \sum_{j\neq i} \delta[i,j,p](1-p_j(k)) , \end{aligned} \tag{C.2}$$

where $\alpha,\gamma : I \times S_M \to [0,1]$ and $\beta,\delta : I \times I \times S_M \to [0,1]$. In view of conditions (C.1) and (C.2), the class of updating scheme which is considered in this chapter is defined as follows:

Let α_i be the action chosen at time k as a sample realization from the distribution p(k). Then p(k+1) is defined as follows:

$$\begin{aligned} p_i(k+1) &= p_i(k) + \theta\alpha[i,p](1-p_i(k)) , \\ p_j(k+1) &= p_j(k) - \theta\beta[i,j,p]\ p_j(k) , \qquad j \neq i \end{aligned}$$

if the action chosen resulted in success, and (2)

$$\begin{aligned} p_i(k+1) &= p_i(k) - \theta\gamma[i,p]p_i(k) , \\ p_j(k+1) &= p_j(k) + \theta\delta[i,j,p](1-p_j(k)) , \quad j \neq i , \end{aligned}$$

if the action chosen resulted in failure, where $0 < \theta \leq 1$ is step

length parameter.

The basic rule that governs the choice of functions in the above algorithm (algorithm 2) is that if a term is subtracted from p_j, then it is made proportional to p_j, and if a term is added to p_j, then it is made proportional to $1-p_j$ irrespective of which action is chosen and whether the action results in success or failure.

Remark 2.1: the choice of the functions $g_j^i[p]$ is quite untraditional in the sense that in almost all the papers in mathematical pyschology [B1, C1, N1, I1] $g_j^i[p]$ is made proportional to p_j for all i and j, $j \neq 1$. Also, in almost all the papers on learning automata [L10, L4, A1, S6] $g_i^i[p]$ is made proportional to $1-p_i$ and $g_j^i[p]$ is made proportional to p_j for all $j \neq i$. Because of this there is a disparity in the behavior of the algorithm (1) under success and failure. However, the present choice of functions $g_j^i[p]$ for all i and j in (2) induce identical behavior of the algorithm under success and failure.

The following theorem 2.1 gives a set of necessary and sufficient conditions for the algorithm (2) to be an absorbing barrier learning algorithm.

Theorem 2.1: Necessary and sufficient conditions for the learning algorithm (2) to have V_M as the only set of absorbing states are:

(A.1) For all $p \in S_M - V_M$ there exists $1 \leq s \leq M$ such that

$$[\alpha[s,p] + \gamma[s,p]]p_s > 0$$

(A.2) For all $1 \leq s \leq M$, $\gamma[s, e_s] = 0$.

Proof: It can easily be seen that in order for V_M to be an invariant set for algorithm (2) the following conditions are necessary and sufficient.

(B.1) For all $P \in S_M - V_M$ there exists $1 \leq s, t \leq M$ such that

$$[\beta[s,t,p]p_t + \delta[s,t,p](1-p_t)]P_s > 0$$

(B.2) For all $1 \leq t \leq M$ and all $e_s \in V_M$, $1 \leq s \leq M$

$$\beta[s,t,e_s]p_t = \delta[s,t,e_s](1-p_t) = 0$$

First the equivalence of A.1 and B.1 is shown. Assume (B.1), then for all $p \in S_M - V_M$ there exist $1 \leq s \leq s_M$ such that

$$\{\sum_{t \neq s} (\beta[s,t,p]p_t + \delta[s,t,p](1-p_t))\}p_s > 0$$

or

$$\{\alpha[s,p](1-p_s) + \gamma[s,p]p_s\}p_s > 0 ,$$

which implies (A.1) since $p \notin V_M$. Assume (A.1) then for all $p \in S_M - V_M$ there exist $1 \leq s \leq M$ such that

$$\{\alpha[s,p](1-p_s) + \gamma[s,p]p_s\}p_s > 0 ,$$

which implies

$$\{\sum_{t \neq s} \beta[s,t,p]p_t + \sum_{t \neq s} \delta[s,t,p](1-p_t)\}p_s > 0$$

and therefore (B.1). Now let us proceed to show the equivalence of (A.2) and (B.2). Assume (A.2) then clearly for arbitrary s and for all $t(t \neq s)$, $\delta[s,t,e_s] = 0$ which implies

$$\delta[s,t,e_s](1-p_t) = 0 \qquad \text{for all } 1 \leq t \leq M$$

and since $p_s = 1$ then, $\alpha[s,p](1-p_s) = 0$ which implies

$$\beta[s,t,e_s]p_t = 0 \qquad \text{for all } 1 \leq t \leq M$$

and since s is chosen arbitrary the equivalence of (A.2) and (B.2) is established.

Remark 2.2: From theorem 2.1 it can be observed that if $p \in V_M$ then $p = e_s$ is an absorbing state if and only if $f_j^s[e_s] = g_j^s[e_s] = 0$ for all

$1 \le j \le M$; if $p \in S_M - V_M$, then p is absorbing if and only if $f_j^s[p]p_s = g_j^s[p]p_s = 0$ for all $1 \le s,j \le M$.

The typical choice of functions in algorithm (2) that satisfies the conditions (C.1) - (C.2) and (A.1) - (A.2) are given in the following examples.

Example 1:

$$\beta[i,s,p] = a_1(1-p_i)(1-p_s)$$

$$\alpha[i,p] - a_1 \sum_{s\neq i} p_s(1-p_s)$$

$$\delta[i,s,p] = a_2\, p_i\, p_s^{\,2}\,(1-p_i)$$

$$\gamma[i,p] = a_2(1-p_i) \sum_{s\neq i} p_s^2(1-p_s)$$

for all i, s = 1,2,...,M where $0 < a_1, a_2 < 1$

Example 2:

$$\beta[i,s,p] = a_1(1-p_i)^n(1-p_s)$$

$$\alpha[i,p] = a_1(1-p_i)^{n-1} \sum_{s\neq i} p_s(1-p_s)$$

$$\delta[i,s,p] = a_2\, p_i^m\, p_s^{\,2}\,(1-p_i)$$

$$\gamma[i,p] = a_2\, p_i^{m-1}\,(1-p_i) \sum_{s\neq i} p_s^{\,2}\,(1-p_s)$$

for all i, s = 1,2,...,M where $0 < a_1, a_2 < 1$, $m,n > 0$

Example 3:

$$\beta[i,s,p] = a_1(1-p_i)^n(1-p_s)^n$$

$$\alpha[i,p] = a_1(1-p_1)^{n-1} \sum_{s\neq i} (1-p_s)^n p_s$$

$$\delta[i,s,p] = a_2\, p_i^{\,m-1}\, p_s^{\,m}\,(1-p_i)$$

$$\gamma[i,p] = a_2\, p_i^{\,m-2}\,(1-p_s) \sum_{s\neq i} p_s^{\,m}\,(1-p_s)$$

for all $i,s = 1,2,\ldots,M$ where $0 < a_1, a_2 < 1$, $n,m > 0$

Example 4:

$$\beta[i,s,p] \equiv 0 \quad \text{for all } i,s$$

$$\gamma[i,s] \equiv 0 \quad \text{for all } i$$

$$\delta[i,s,p] = a_s^i (1-p_i)p_s$$

$$\gamma[i,p] = \begin{cases} \frac{1-p_i}{p_i} \sum a_s^i \, p_s(1-p_s) & \text{if } p_i > 0 \\ 0 & \text{if } p_i = 0 \end{cases}$$

for all $i,s = 1,2,\ldots,M$, where $0 < a_s^i < 1$ for all i,s and $a_s^i = a_i^s$.

Example 5:

$$\beta[i,s,p] = a_1(1-p_s)$$

$$\alpha[i,p] = \begin{cases} a_1 \frac{\sum_{s \neq i}(1-p_s)p_s}{1-p_i} & \text{if } p_i < 1 \\ 0 & \text{if } p_i = 1 \end{cases}$$

$$\delta[i,s,p] \equiv 0$$

$$\gamma[i,p] \equiv 0$$

for all $i,s = 1,2,\ldots,M$ where $0 < a_1 < 1$.

Example 6:

$$\beta[i,s,p] \equiv 0 \quad \text{for } i,s$$

$$\alpha[i,p] \equiv 0 \quad \text{for all } i$$

$$\delta[i,s,p] = a_s^i(1-p_i)p_s B_s^i$$

$$\gamma[i,p] = \begin{cases} \frac{\sum_{s \neq i} a_s^i(1-p_i)p_s B_s^i}{p_i} & \text{if } p_i > 0 \\ 0 & \text{if } p_i = 0 \end{cases}$$

for all $i,s = 1,2,\ldots,M$. Where $B : I \times I \times S_M \to [0,1]$ and $B_s^i = B_i^s > 0$ for

all i,s, and $0 < a_s^i < 1$ and $a_s^i = a_i^s$ for all s,i.

Example 7:

$$\beta[i,j,p] \equiv 0$$

$$\alpha[i,p] \equiv 0$$

$$\delta[i,j,p] = a_j^i(1-p_i)p_j\ [b_j^i\ p_i(1-p_j) + b_i^j\ p_j(1-p_i)]$$

$$\gamma[i,p] = \begin{cases} \dfrac{(1-p_i)\sum_{j\neq i} a_j^i\, p_j [b_j^i(1-p_j)p_i + b_i^j\, p_j(1-p_i)]}{p_i} & \text{if } p_i > 0 \\ 0 & \text{if } p_i = 0 \end{cases}$$

for all $i,j = 1,2,\ldots,M$ where $0 < a_j^i,\ b_j^i < 1$ for all i,j and $a_j^i = a_i^j$ for all i,j, $i \neq j$.

Remark 2.3: The demonstration of the symmetry in the properties of the algorithm (2) under success and failure alluded to in Remark 2.1 is evidenced by Theorem (2.1). Notice that V_M is the only set of absorbing states for (2) if $\alpha[i,p] \not\equiv 0$ and $\gamma[i,p] \equiv 0$, or $\alpha[i,p] \equiv 0$ and $\gamma[i,p] \not\equiv 0$, or both $\alpha[i,p]$ and $\gamma[i,p] \not\equiv 0$. This is in sharp contrast with the properties of the currently available absolutely expedient learning algorithm [L2, L4, A1], namely the reward-penalty and the reward-inaction algorithms have V_M as the only set of absorbing states but the inaction-penalty algorithm does not. In fact, in all the inaction-penalty algorithms of the absolutely expedient type known so far [L2], every state in δS_M is an absorbing state. The modified definition of strong absolute expediency is in fact motivated by the existence of learning algorithms of the reward-penalty, reward-inaction, and inaction-penalty types, each with V_M as the only set of absorbing states.

Remark 2.4: For some $1 \leq j \leq M$, if $p_j(k) = 0$, then it follows from (2) that $p_j(k^*) = 0$ for all $k^* \geq k$. In other words, during the learning process if p(k) reaches the boundary ($p_j = 0$) of the simplex S_M, then p(k) will continue to remain in that boundary.

Henceforth in this chapter learning algorithms with V_M as the only set of absorbing states, that is, the algorithm (2) under conditions (A.1) - (A.2) of Theorem 2.1 only, will be considered.

Necessary and sufficient conditions for the learning algorithm (2) to be strongly absolutely expedient are stated in the following theorem.

Theorem 2.2 Necessary and sufficient conditions for the learning algorithm (3) to be strongly absolutely expedient are

$$\sum_{j \neq i} p_j \beta[i,j,p] = \sum_{j \neq i} p_j \beta[j,i,p] \qquad (S.1)$$

and

$$\sum_{j \neq i} p_i (1-p_j) \delta[i,j,p] = \sum_{j \neq i} p_j (1-p_i) \delta[j,i,p] \qquad (S.2)$$

for all $i = 1,2,\ldots,M$.

Sufficiency: Define $\delta x(k) = x(k+1) - x(k)$, and let

$$\Delta\eta(k) = E[\delta\eta(k) \mid p(k)] = \sum_{i=1}^{M} E[\delta p_i(k) \mid p(k)] d_i \qquad (2.1)$$

It can be seen by direct computation that

$$\begin{aligned} E[\delta p_i(k) \mid p(k) = p] = {} & p_i(1-p_i) d_i \alpha(i,p) - P_i^2 c_i \gamma[i,p] \\ & - \sum_{j \neq i} p_j p_i d_j \beta[j,i,p] \\ & + \sum_{j \neq i} p_j (1-p_i) c_j \delta[j,i,p]. \end{aligned} \qquad (2.2)$$

Substituting (2.2) into (2.1), in view of (C.1) and (C.2), we obtain

$$\Delta\eta(k) = \Delta\eta_1(k) + \Delta\eta_2(k), \tag{2.3}$$

where

$$\Delta\eta_1(k) = \sum_{i=1}^{M} p_i d_i^2 \sum_{j\neq i} p_j \beta[i,j,p] - \sum_{i=1}^{M} p_i d_i \sum_{j\neq i} p_j d_j \beta[j,i,p] \tag{2.4}$$

and

$$\Delta\eta_2(k) = -\sum_{i=1}^{M} p_i d_i c_i \sum_{j\neq i} \delta[i,j,p](1-p_j)$$

$$+ \sum_{i=1}^{M} (1-p_i) d_i \sum_{j\neq i} p_j c_j \delta[j,i,p]. \tag{2.5}$$

Consider $\Delta\eta_1(k)$: It can be rewritten as

$$\Delta\eta_1(k) = \frac{b}{2},$$

where

$$b = \sum_{i=1}^{M} \sum_{j\neq i} \{p_i p_j d_i^2 \beta[i,j,p] + p_i p_j d_i^2 \beta[i,j,p] - 2p_i p_j d_i d_j \beta[j,i,p]\}.$$

Applying (S.1) to the second term within the curly braces, we obtain

$$b = \sum_{i=1}^{M} \sum_{j\neq i} \{p_i p_j d_i^2 \beta[i,j,p] + p_i p_j d_i^2 \beta[j,i,p] - 2p_i p_j d_i d_j \beta[j,i,p]\}$$

$$= p^T A p,$$

where $A = [A_{ij}]$, $A_{ii} = 0$, and

$$A_{ij} = d_i^2\{\beta[i,j,p] + \beta[j,i,p]\} - 2d_i d_j \beta[j,i,p] \quad \text{for } i \neq j.$$

If we define a matrix $B = A + A^T = [B_{ij}]$, then

$$B_{ii} = 0$$

and

$$B_{ij} = \{\beta[i,j,p] + \beta[j,i,p]\}\,(d_i - d_j)^2 \quad \text{for } i \neq j.$$

Since $b = \frac{1}{2} p^T B p$ and $B_{ij} \geq 0$, it readily follows that

$$\Delta\eta_1(k) = \frac{1}{4} \sum_{i=1}^{M} \sum_{j\neq i} p_i p_j (d_i - d_j)^2 \{\beta[i,j,p] + \beta[j,i,p]\} \geq 0 \tag{2.6}$$

with equality holding only if $p \in V_M$.

Consider $\Delta\eta_2(k)$: It can be written as

$$\Delta\eta_2(k) = Z + g, \tag{2.7}$$

where

$$g = \sum_{i=1}^{M} p_i d_i^2 \sum_{j\neq i} (1-p_j)\delta[i,j,p]$$

$$- \sum_{i=1}^{M} (1-p_i) d_i \sum_{j\neq i} p_j d_j \delta[j,i,p] \tag{2.8}$$

and

$$Z = -\sum_{i=1}^{M} d_i \{ \sum_{j\neq i} p_i(1-p_j)\delta[i,j,p] - \sum_{j\neq i} p_j(1-p_i)\delta[j,i,p]\} \tag{2.9}$$

$$\equiv 0 \quad \text{by the condition (S.2).} \tag{2.10}$$

If we define $y_i = 1 - p_i$ and $y = (y_1, y_2, \ldots, y_M)^T$, then g can be rewritten as

$$g = \frac{1}{2} \{ \sum_{i=1}^{M} p_i d_i^2 \sum_{j\neq i} y_i \delta[i,j,p] + \sum_{i=1}^{M} p_i d_i^2 \sum_{j\neq i} y_j \delta[i,j,p]$$

$$- 2 \sum_{i=1}^{M} y_i d_i \sum_{j\neq i} p_j d_j \delta[j,i,p]\}. \tag{2.11}$$

Applying (S.2) to the second term in the curly braces, we obtain

$$g = \frac{1}{2} \{ p^T E y + y^T F p \},$$

where

$$E = [E_{ij}], \quad E_{ii} = 0, \quad E_{ij} = d_i^2 \, \delta[i,j,p].$$

$$F = [F_{ij}], \quad F_{ii} = 0, \quad F_{ij} = (d_i^2 - 2d_i d_j) \, \delta[j,i,p]. \tag{2.12}$$

Since

$$g = \frac{1}{4} \{ p^T(E + E^T)y + y^T(F + F^T)p \}$$

it follows after simplifications that

$$g = \frac{1}{4} \sum_{i \neq 1}^{M} \sum_{j=i} p_i y_i (d_i - d_j)^2 \{\delta[i,j,p] + \delta[j,i,p]\} \geq 0 \qquad (2.13)$$

with equality holding only if $p \in V_M$. From (2.6) and (2.13) sufficiency follows.

Necessity: $\Delta\eta(k)$ can be represented as a quadratic and linear term in the vector D as follows:

$$\Delta\eta(k) = D^T A D + D^T B, \qquad (2.14)$$

where $A = [A_{ij}]$ and $B = [B_1, B_2, \ldots, B_m]^T$ with

$$\begin{aligned} A_{ii} &= p_i(1-p_i)\alpha[i,p] + p_i^2 \gamma[i,p], \\ A_{ij} &= -\{p_i p_j \beta[j,i,p] + (1-p_i)p_j \delta[j,i,p]\}, \end{aligned} \qquad (2.15)$$

and

$$B_i = p_i^2 \delta[i,p] + (1-p_i) \sum_{j \neq i} p_j \delta[j,i,p] \qquad (2.16)$$

for all i, j = 1,2,...,M. From the definition of strong absolute expediency it follows that $\Delta\eta(k)$ attains its minimum value zero either when all d_i are equal or when $p \in V_M$. Since every member of V_M is absorbing on V_M, it is easily seen that $\Delta\eta(k)$ attains its minimum value for all admissible D-vectors. In the following we shall derive conditions for the minimum of $\Delta\eta(k)$ when $d_i = d$ for all i = 1,2,...,M. $0 < d < 1$. Necessary conditions for a minimum are obtained by setting the derivative of $\Delta\eta(k)$ (with respect to d_i) at the point $d_i = d$ for all i = 1, 2,...,M equal to zero, that is,

$$\left.\frac{\partial \Delta\eta}{\partial d_i}\right|_{d_i = d} = 0 \quad \text{for all } i = 1,2,\ldots,M. \qquad (2.17)$$

From (2.14), the equation (2.17) takes the form

$$(A + A^T)d1 + B = 0, \qquad (2.18)$$

where $1 = (1,1,\ldots,1)^T$ is an M-dimensional column vector of all ones.

Rewriting (2.18) we get

$$dK_i + L_i = 0 \tag{2.19}$$

for all $i = 1,2,\ldots,M$ and all $0 < d < 1$, where

$$K_i = p_i(1-p_i)\,\alpha[i,p] + p_i^2\,\gamma[i,p] - p_i \sum_{j\neq i} p_j\beta[j,i,p] - (1-p_i) \sum_{j\neq i} p_i\delta[j,i,p]$$

and

$$L_i = p_i^2\,\gamma[i,p] + (1-p_i) \sum_{j\neq i} p_j\,\delta[j,i,p].$$

(2.19) is true for all $0 < d < 1$ only if

$$L_i = 0 \text{ and } K_i = 0 \quad \text{for all } i = 1,2,\ldots,M.$$

And $L_i = 0$ leads to the condition

$$p_i^2\,\gamma[i,p] = (1-p_i) \sum_{j\neq i} p_j\,\delta[j,i,p]. \tag{2.20}$$

Using (C.2), from (2.20) we obtain

$$p_i \sum_{j\neq i} (1-p_j)\,\delta[i,j,p] = (1-p_i) \sum_{j\neq i} p_j\,\delta[j,i,p], \tag{2.21}$$

which in fact is (S.2). Substituting (2.21) in $K_i = 0$, we obtain

$$p_i(1-p_i)\,\alpha[i,p] = p_i \sum_{j\neq i} p_j\,\beta[j,i,p]. \tag{2.22}$$

Once again, using (C.1), we get

$$p_1 \sum_{j\neq i} p_j\beta[i,j,p] = p_1 \sum_{j\neq i} p_j\,\beta[j,i,p], \tag{2.23}$$

which is the same as (S.1). Hence the theorem. □

Among the 7 examples previously given, examples 1, 3, 4, 6 and 7 not only satisfy conditions (C.1) - (C.2) and (A.1) - (A.2) but they satisfy conditions (S.1) - (S.2) and hence they are strongly absolutely expedient. Example 2 and 5 are examples of non-absolutely expedient but absorbing barrier learning algorithms.

Remark 2.5: By setting $\delta[i,j,p] \equiv 0$ and $\beta[i,j,p] \equiv c$ $(0 < c < 1)$ for all i,j, where c is a constant, we get linear reward-inaction scheme. It is easy to verify that this is the only linear scheme that satisfies conditions (C.1) - (C.2), (A.1) - (A.2) and (S.1) - (S.2). In fact, this scheme is the only one among all the linear scheme which is strongly absolutely expedient. Linear reward-inaction scheme can be obtained by setting $n = 0$ and $a_2 = 0$ in example 3.

Remark 2.6: By setting $\beta[i,j,p] = \lambda(p)$ and $\delta[i,j,p] \equiv 0$ for all i,j, where $0 < \lambda(p) < 1$, and assuming that the function $\lambda(p) = 0$ only if p is a unit vector, we get the reward-inaction scheme reported in [L2]. This scheme satisfies the conditions (C.1) - (C.2) and conditions of theorem (2.1) and (2.2), and therefore is strongly absolutely expedient.

Remark 2.7: If

$$\beta[i,j,p] = (1-p_i)(1-p_j)\,\lambda_{ij}(p) \quad \text{and} \quad \delta[i,j,p] \equiv 0$$

for all $i,j = 1,2,\ldots,M$, where $\lambda_{ij}(p) = \lambda_{ji}(p)$ and $\lambda_{ij}(p)\colon S_M \to [0,1]$ and if for all $p \in S_M - V_M$ there exist i and j such that $\lambda_{ij}(p) > 0$, then algorithm 2 is strongly absolutely expedient.

Proof of this remark is immediate from theorem (2.1) and (2.2). Clearly (C.1) - (C.2) and (S.1) - (S.2) are satisfied and (A.1) is satisfied if for all $p \in S_M - V_M$ there exist i,j such that $\lambda_{ij}(p) > 0$.

Theorem 2.2 established the equivalence of strong absolute expediency and conditions (S.1) - (S.2). The following theorem in turn proves the equivalence of conditions (S.1) - (S.2) and the sign definitions of Δp_1 and Δp_M. Δp_i is defined to be

$$\Delta p_i(k) = E[\delta p_i(k) \mid p(k)]$$

and α_1 and α_M are the actions associated with the largest and smallest

success probability, respectively.

Theorem 2.3: When a learning automata uses learning algorithm 2 and operates in any stationary random environment, conditions (S.1) - (S.2) are necessary and sufficient conditions for $\Delta p_1(k) < 0$ and $\Delta p_M(k) < 0$ for all $p(k) \in S_M - V_M$.

Sufficiency: Using (S.1) - (S.2), (2.2) can be rewritten as

$$\Delta p_i(k) = \sum_j (d_i - d_j) H_{ij} \tag{2.24}$$

where

$$H_{ij} = p_j(k)\, \beta[j,i,p] p_i(k) + p_j(k) \delta[j,i,p](1-p_i(k)) \tag{2.25}$$

The sufficiency is immediate, since for any j, $d_i - d_j \geq 0$ if i = 1 and for any j, $d_i - d_j \leq 0$ if i = M and $H_{ij} \geq 0$ for all i,j = 1,2,...,M.

Necessity: (2.2) can be rewritten as

$$\Delta p_i(k) = - r_i + (r_i + s_i) d_i + \sum_{j \neq i} (d_i - d_j)\, H_{ij} \tag{2.26}$$

where

$$r_i = p_i(k) \sum_{j \neq i} \delta[i,j,p](1-p_j(k)) - \sum_{j \neq i} p_j\, \delta[j,i,p](1-p_i(k))$$

and

$$s_i = p_i(k) \sum_{j \neq i} [\delta[i,j,p](1-p_j(k)) + \beta[i,j,p] p_j(k)]$$

$$- \sum_{j \neq i} p_j(k) [\delta[j,i,p](1-p_i(k)) + \beta[j,i,p] p_i(k)]$$

Let us fix the subscript i. Consider two environments $D^{(1)} = (d_1^{(1)}, d_2^{(1)}, \ldots, d_M^{(1)})$ and $D^{(2)} = (d_1^{(2)}, d_2^{(2)}, \ldots, d_M^{(2)})$ such that $d_i^{(1)} = d_i^{(2)} = x$, $d_j^{(1)} = y (j \neq i)$, and $d_j^{(2)} = z (j \neq i)$, where $1 \geq y > x > z \geq 0$. Let the corresponding Δp_i's be denoted by $\Delta p_i^{(1)}(k)$ and $\Delta p_i^{(2)}(k)$. Then

$$\begin{aligned} \Delta p_i^{(1)}(k) &= -r_i + (r_i + s_i)x + \sum_{j \neq i} (x-y) H_{ij} \\ \Delta p_i^{(2)}(k) &= -r_i + (r_i + s_i)x + \sum_{j \neq i} (x-z) H_{ij} \end{aligned} \tag{2.27}$$

From (2.27) it follows that $\Delta p_1(k) > 0$ and $\Delta p_M(k) < 0$ are equivalent to

$$-(y-x) \sum_{j \neq i} H_{ij} < r_i - (r_i + s_i) < (x-z) \sum_{j \neq i} H_{ij} . \qquad (2.28)$$

Since the inequality holds for any x, y, z, it should hold if $y \to x$ and $z \to x$. Hence we have $r_i - (r_i + s_i)x = 0$. In particular, this implication should hold for $x = 0$ and $x = 1$. Then we have $r_i = 0$ and $s_i = 0$ as $x = 0$ and $x = 1$ respectively. Since the above holds for any i, the proof is immediate. □

Remark 2.8: The sign definiteness of $\Delta p_1(k)$ ($\Delta p_M(k)$) implies that $E[p_1(k)]$ ($E[p_M(k)]$) is monotonically increasing (decreasing) with respect to k. Theorem 2.2 thus states that by using absolutely expedient schemes $p_1(k)$ and $p_M(k)$ we move in the right direction at least in an expected-value sense.

Corollary 2.1: The learning algorithm (2) is strongly absolutely expedient if and only if

$$\Delta p_1(k) > 0 \quad \text{and} \quad \Delta p_M(k) < 0$$

for all $p \in S_M - V_M$.

Proof: The proof is obvious from Theorem (2.2) and (2.3) □

Remark 2.9: Making $\Delta p_1(k) > 0$ and $\Delta p_M(k) < 0$ implies that $p_1(k)$ is a sub-martingale and $p_M(k)$ is a super-martingale (refer to Appendix A). In view of martingale theorem [D.1] we can conclude that $\lim_{k \to \infty} p_1(k) = p_1^*$ and $\lim_{k \to \infty} p_M = p_M^*$ exist with probability one. Since $\Delta p_i(k) = 0$ when $p_i = 0$ or 1, p_1^* and p_M^* approaches 0 or 1, each with positive probability.

We conclude this section by stating a fundamental result on the sample path behavior of the process $\{p(k)\}$.

Corollary 2.2: The Markov process $\{p(k)\}$ $k \geq 0$, as generated by the algorithm (2) under conditions (A.1) - (A.2), and (S.1) - (S.2) converges to V_M with probability one.

Proof: Let $F[x(1), x(2), \ldots, x(k)]$ be the σ-algebra generated by a random sequence $x(1), x(2), \ldots, x(k)$. As $\eta(k)$ is a linear function of $p(k)$, it easily follows that

$$F[\eta(1), \eta(2), \ldots, \eta(k)] \subset F[p(1), p(2), \ldots, p(k)]. \tag{2.29}$$

Further

$$F[p(k)] \subset F[p(1),p(2),\ldots,p(k)] . \tag{2.30}$$

From theorem (2.2) we know that

$$\begin{aligned} E[\delta\eta(k)|p(k)] = \Delta\eta(k) = &\frac{1}{4} \sum_{i=1}^{M} \sum_{j\neq i} p_i p_j (d_i-d_j)^2 \{\beta[i,j,p] + \beta[j,i,p]\} \\ &+ \frac{1}{4} \sum_{i=1} \sum_{j\neq i} p_i(1-p_i)(d_i-d_j)^2 \{\delta[i,j,p] \\ &+ \delta[j,i,p]\} \geq 0 . \end{aligned} \tag{2.31}$$

Using (2.30) we get

$$E[\eta(k+1) \mid F[p(1),p(2),\ldots,p(k)]] \geq \eta(k) . \tag{2.32}$$

Now taking conditional expectations on both sides of (2.31) with respect to $F[\eta(1), \eta(2), \eta(3), \ldots, \eta(k)]$ from the low of iterated conditional expectations and from (2.29) we get

$$E[\eta(k+1)|F[\eta(1),\eta(2),\ldots,\eta(k)]] \geq \eta(k) .$$

This in turn implies that $\eta(k)$ is a non-negative bounded submartingale and by the martingale theory $\eta(k)$ converges to a random variable η^* with probability one, and hence $\lim_{k\to\infty} p(k) = p^*$ exists with probability one. As $\delta\eta(k) = 0$ only on V_M, it follows that $p^* \in V_M$ with probability one. This implies that the set of all unit vectors

are the only absorbing barriers for the process $\{p(k)\}$. □

The above theorem stated that $\{p(k)\}$ corresponding to algorithm (2) under conditions (A.1) - (A.2) and (S.1) - (S.2) is a process with more than one absorbing barriers, that is

$$\lim_{k\to\infty} p(k) \in \{e_i : i = 1,2,\ldots,M\} \text{ with probability one.}$$

Out of all these M unit vectors only one corresponds to the optimal or desired action. In fact, the Markov process $\{p(k)\}$ converges with a positive probability to each one of these unit vectors. Therefore, strong absolute expediency implies that the algorithm (2) converges to the optimal action with a positive probability less than one. Our aim in the following section is to quantify this probability.

2.3 ε-Optimality

In the previous section, it was established that $p^* \in V_M$ with probability one. In this section we set out to quantify the distribution of p^*. We first derive lower bound on the probability with which strongly absolutely expedient scheme can converge to a desired state. Using the lower bound it will be shown that strong absolute expediency is indeed a sufficient condition for ε-optimality.

To this end define

$$\begin{aligned} \Gamma_i(p) &= \text{Prob}[p^* = e_i \mid p(0) = p] \\ &= \text{Prob}[p_i^* = 1 \mid p(0) = p] \end{aligned} \tag{2.33}$$

for $i = 1,2,\ldots,M$. Notice $\sum_{i=1}^{M} \Gamma_i(p) = 1$. Stated in words, corollary (2.2) states that if $p(k)$ is updated according to algorithm (2) then along each sample path asymtotically we will end up choosing only one action and action i will be chosen with probability $\Gamma_i(p)$ where $p(0)=p$.

In view of corollary (2.2) and (2.33), we obtain for all $p(0) = p \in S_M^0$.

$$\lim_{k \to \infty} E[\eta(k)] = \sum_{i=1}^{M} \Gamma_i(p) d_i \ . \tag{2.33a}$$

To characterize further properties of $\Gamma_i(p)$ we need a few definitions.

Let $C[S_M]$ be the class of all continuous functions from S_M to real line. If $f(\cdot) \in C[S_M]$ define the operator U as follows

$$U\, f[p] = E[f[p(k+1)] \mid p(k)=p] \tag{2.34}$$

$$= \int_{S_M} f(x)\ k[p,dx] \ ,$$

where $k[\cdot,\cdot]$ is the transition function defined as

$$K[p,A] = \text{Prob}[p(k+1) \in A \mid p(k) = p] \quad \text{for every } k \geq 0$$

where A is a subset of S_M. In our case the transition function is defined in terms of the learning algorithm (2).

Define

$$K^{(n)}[p,A] = \text{Prob}[p(k+n) \in A \mid p(k) = p]$$

for $n \geq 1$ and all $k \geq 0$, as the n step transition function. The following is the consequence of the definition of U [N2].

1. U is linear. That is, if $f_1(p)$, $f_2(p) \in C[S_M]$ and m_1, m_2 are real constants then

$$U(m_1 f_1[p] + m_2 f_2[p]) = m_1\ U\ f_1[p] + m_2\ U\ f_2(p) \ .$$

2. U is positive, that is, it preserves non-negative functions. Thus, if f(p) is non-negative so is U f(p).

3. $U^2 f(p) = U[U\, f\, [p]]$

$$= E[U\ f[p(k+1)] \mid p(k) = p]$$

$$= E[E[f[p(k+2)] \mid p(k+1)] \mid p(k) = p]$$

$$= \int_{S_M} k[p,dx] \int_{S_M} k[x,dy]\ f[y]$$

$$= \int_{S_M} k^{(2)}[p,dy]\, f[y] = E[f[p(k+2)] \mid p(k) = p] ,$$

where $k^{(2)}[p,A] = \int_{S_M} k[p,dx]\, k[x,A]$ is the well-known Chapman-Kolmogorov equation [D1]. In general for any $n \geq 2$.

$$U^n f[p] = E[f[p(k+n)] \mid p(k) = p]$$

$$= \int_{S_M} k^{(n)}[p,dx] f(x) ,$$

where $k^{(n)}[p,A] = \int_{S_M} k[p,dx] k^{n-1}[x,A]$ and U^n is called the n^{th} interate of U. As a result, if the n^{th} iterate of U is applied to the function f[p] it gives the expectation of f[p(n)] as a function of the initial value p(0).

Definition 2.1 A function f: $S_M \rightarrow$ (real line) is called superregular (regular, subregular) if

$$f(p) \geq (=, \leq)\ Uf(p) \tag{2.35}$$

for all $p \in S_M$.

Remark 2.10: It can easily be shown that super and sub regular functions are closed under addition and multiplication by non-negative constants. That is, if $f_1(\cdot)$ and $f_2(\cdot)$ are super (sub) regular functions and if m_1 and m_2 are real, non-negative constants, then $m_1 f_1(\cdot) + m_2 f_2(\cdot)$ is also super (sub) regular. Further, constants are both super and sub regular and hence are regular. Also if f(.) is a subregular function then $-f(\cdot)$ is super regular.

With these preliminaries, we now state and prove two propositions that lead to an algorithm for quantifying $\Gamma_i(p)$.

Proposition 2.1 $\Gamma_i(p)$ is the only continuous solution of the functional equation

$$U\, \Gamma_i(p) = \Gamma_i(p) , \tag{2.36}$$

satisfying the boundary conditions

$$\Gamma_i(e_i) = 1 \quad \text{and} \quad \Gamma_i(e_j) = 0 \tag{2.37}$$

for all i, j, $i \neq j$.

This proposition is proved in Appendix B.

Notice that $\Gamma_i(p)$ satisfying (2.36) is a regular function by definition. This functional equation is extremely difficult to solve. It may be surprising to note that except for certain very simple cases [N2] no one has yet been able to solve the above functional equation. Hence in the following we establish upper and lower bounds on $\Gamma_i(p)$.

Proposition 2.2 Let $f_i(p) \in C[S_M]$ be superregular (subregular) functions with $f_i(e_i) = 1$ and $f_i(e_j) = 0$ for all i, j, $i \neq j$. Then

$$f_i(p) \geq (\leq)\ \Gamma_i(p). \tag{2.38}$$

Proof: Let $f_i(p)$ be a superregular function, that is

$$f_i(p) \geq U f_i(p). \tag{2.39}$$

Since U is positive, we obtain

$$U f_i(p) \geq U^2 f_i(p) \geq \ldots \geq U^\infty f_i(p) . \tag{2.40}$$

But since $\lim_{k \to \infty} p(k) = p^*$

$$\begin{aligned} U^\infty f_i(p) &= E[f_i(p^*) \mid p(0) = p] \\ &= \sum_{j=1}^{M} f_i(e_j)\ \Gamma_j(p) \\ &= \Gamma_i(p). \end{aligned} \tag{2.41}$$

Combining (2.38), (2.39) and (2.40) we obtain

$$f_i(p) \geq \Gamma_i(p).$$

The result for subregular functions follow in a similar fashion. Hence the proposition. □

Thus, if we can find two functions $h_i^1(p)$ and $h_i^2(p)$ which are super and subregular functions respectively and satisfy the boundary conditions

$$\begin{aligned} h_i^1(e_i) &= h_i^2(e_i) = 1 \\ h_i^1(e_j) &= h_i^2(e_j) = 0 \qquad \text{for all } j\neq i\ , \end{aligned} \tag{2.42}$$

then from Proposition (2.2) it will follow that

$$h_i^2(p) \leq \Gamma_i(p) \leq h_i^1(p)\ .$$

In other words, any super and sub regular functions satisfying (2.42) form the upper and lower bound on $\Gamma_i(p)$. We now proceed to provide an algorithm to compute these bounds.

Consider a function

$$\psi_i[x_i,p] = e^{-x_i p_i/\theta} \tag{2.43}$$

where $x_i > 0$ is a parameter. Clearly

$$1 = \psi_i[x_i,e_j] > \psi_i[x_i,e_i] = e^{-x_i/\theta} \qquad \text{for all } i\neq j\ .$$

$$\text{Let } \phi_i[x_i,p] = \frac{1-e^{-x_i p_i/\theta}}{1-e^{-x_i/\theta}} \qquad \text{for } x_i > 0\ . \tag{2.44}$$

In view of remark (2.9) $\psi_i[x_i,p]$ is sub (super) regular, then $\phi_i[x_i,p]$ is super (sub) regular. Also notice

$$\phi_i[x_i,\ e_i] = 1 \quad \text{and} \quad \phi_i[x_i,e_j] = 0\ . \tag{2.45}$$

Let y_i and z_i be two positive constants such that

$$\phi_i[y_i,p] = \frac{1-e^{-y_i p_i/\theta}}{1-e^{-y_i/\theta}} \qquad \text{is subregular} \tag{2.46}$$

and

$$\phi_i[z_i,p] = \frac{1-e^{-z_i p_i/\theta}}{1-e^{-z_i/\theta}} \qquad \text{is superregular.}$$

It follows from proposition (2.2) that

$$\phi_i[y_i,p] \le \Gamma_i(p) \le \phi_i[z_i,p].$$

Note that function $\phi_i[x_i,p]$ depends on just one component of vector P and thus leads to conservative results. However, it gives rise to expressions which are easily manageable.

The problem of getting bounds on $\Gamma_i(p)$ now reduces to one of finding two positive constants y_i and z_i such that $\phi_i[y_i,p]$ is subregular and $\phi_i[z_i,p]$ is superregular. Further, from (2.33a) and the fact that action 1 has the highest success probability (d_1) it is clear that for ε-optimality we need to concentrate only on the lower bound of $\Gamma_1(p)$.

It can be seen that (dropping the subscript 1 from x, for convenience)

$$\begin{aligned} U\,\psi_1[x,p_1] - \psi_1[x,p_1] = e^{\frac{-x p_1}{\theta}} \Big\{ & [p_1 d_1 (e^{-x\alpha[1,p](1-p_1)} - 1)] \\ & + [p_1 c_1 (e^{x\gamma[1,p]p_1} - 1)] \\ & + [\sum_{j\neq 1} p_j d_j (e^{x\beta[j,1,p]p_1} - 1)] \\ & + [\sum_{j\neq 1} p_j c_j (e^{-x\delta[j,1,p](1-p_1)} - 1)] \Big\} \end{aligned} \tag{2.47}$$

Define

$$V[z] = \begin{cases} \dfrac{e^z - 1}{z} & \text{for } z \neq 0 \\ 1 & \text{for } z = 0. \end{cases} \tag{2.48}$$

It can be seen that

$$V[z] = \int_0^1 e^{zw}\, dw \tag{2.49}$$

for all real z. Since the integrand in (2.49) is positive $V[z] > 0$ for all real z. Taking the derivative of V[z], it can be seen that

$$V^{(k)}(z) = \int_0^1 w^k e^{zw}\, dw$$

where $v^{(k)}(z)$ is the kth derivative of v(z). Since $V^{(k)}(z) > 0$ for $k = 1,2,$ we have

(i) V[z] strictly monotonically increasing

(ii) V[z] is convex.

Let

$$H[z] = \ln V[z] = \log_e V[z]$$

since log of an increasing function is increasing we have

(iii) H[z] is strictly monotonically increasing

It can be shown that the second derivative H"[z] of H[z] is positive and hence

(iv) H(z) is convex,

and

(v) H'(z) , the first derivative of H(z) is increasing.

Expressing the terms in the right hand side of (2.47) in terms of V[•] function we have

$$U\,\psi_1[x,p_1] - \psi_1[x,p_1] = -\,x\,F_1[x,p]\,\psi_1[x,p]$$

where

$$F_1[x,p] = p_1(1-p_1)d_1\,\alpha[1,p]\,V[-x\alpha[1,p](1-p_1)] - p_1^2 c_1 \gamma[1,p]\,V[x\gamma[1,p]p_1]$$

$$- \sum_{j\neq 1} p_j p_1 d_j \beta[j,1,p] V[x\beta[j,1,p]p_1]$$

$$+ \sum_{j\neq 1} p_j (1-p_1) c_j \delta[j,1,p] V[-x\delta[j,1,p](1-p_1)] \cdot \tag{2.50}$$

Clearly

$$F_1[x,p] \geq 0 \tag{2.51}$$

implies $\phi_1[x,p]$ subregular. (2.51) is the governing inequality from which the required x is to be found. But before considering this most general case, we shall consider two special cases.

Case 1: The Reward-Inaction Scheme ($\beta[i,j,p] \not\equiv 0, \delta[i,j,p] \equiv 0$ for all i,j)

Setting $\delta[i,j,p] \equiv 0$ for all i,j in (2.50) it follows that $\psi_i[x_1 p]$ is subregular if

$$\begin{aligned} & p_1 d_1 \alpha[i,p](1-p_1) V[-x\alpha[1,p](1-p_1)] \\ & - \sum_{j\neq 1} p_j d_j \beta[j,1,p] p_1 V[x\beta[j,1,p]p_1] \geq 0 \cdot \end{aligned} \tag{2.52}$$

Clearly (2.52) can be written in the following form

$$p_1 d_1 \alpha[1,p](1-p_1) \geq \frac{1}{V[-x\alpha[1,p](1-p_1)]} \sum_{j\neq 1} p_j d_j$$

$$\beta[j,1,p] p_1 \, V[x\beta[j,1,p]p_1] \cdot \tag{2.53}$$

Define

$$G_{1j}[x,p] = \frac{V[x\,\beta[j,1,p]p_1]}{V[-x\,\alpha[1,p](1-p_1)]} \quad \text{for all } j\,,\ j\neq 1\,. \tag{2.54}$$

Define

$$\Delta_{1j}[x,p] = \log_e G_{1j}[x,p] = \int_{-b}^{a} H'[u]du \tag{2.55}$$

where $a = x\,\beta[j,1,p]p_1$ and $b = x\,\alpha[1,p](1-p_1)$. Using the properties of function H, the following inequality is true (Refer Appendix C).

$$\int_{-(a+b)}^{0} H'[u]du \le \int_{-b}^{a} H'[u]du \le \int_{0}^{a+b} H'[u]du \le \int^{\max(a+b)} H'[u]du. \quad (2.56)$$

Since H'[u] is increasing function therefore

$$\Delta_{1j}[x,p] \le \int_{0}^{x} H'[u]du$$

or

$$\Delta_{1j}[x,p] \le H[x]. \quad (2.57)$$

Taking exponentials throughout (2.57) we obtain

$$G_{1j}[x,p] \le V[x].$$

Therefore (2.53) can be written in the following form

$$\frac{p_1 d_1 \alpha[1,p](1-p_1)}{\sum_{j\neq 1} p_j d_j \beta[j,1,p]} \ge V[x]. \quad (2.58)$$

Using (C.1) and (S.1) we have

$$\frac{d_1}{\min_{j\neq 1}(d_j)} \ge \frac{p_1 d_1 \alpha[1,p](1-p_1)}{\sum_{j\neq 1} p_j d_j \beta[j,1,p]} . \quad (2.59)$$

Combining (2.58) and (2.59) we obtain

$$V[x] \le \frac{d_1}{\min_{j\neq 1}\{d_j\}} . \quad (2.60)$$

Setting

$$V[x] = \frac{d_1}{\min_{j\neq 1}\{d_j\}} . \quad (2.61)$$

Now for all $d \in D$ and since $d_1 = \max_j \{d_j\}$ we see that

$$\frac{d_1}{\min_{j\neq 1}\{d_j\}} > 1 .$$

As $V[u] < 1$ for $u < 0$ and $V[u] > 1$ for $u > 0$ and $V[u]$ is strictly monotonically increasing from (2.61) it follows that there exists

unique value for x that satisfies equation (2.61). Following the above procedure, upper bound on $\Gamma_1(p)$ can be easily obtained.

Case 2: Inaction-penalty algorithm ($\beta[i,j,p] \equiv 0$, $\delta[i,j,p] \not\equiv 0$ for all i,j)

Setting $\beta[i,j,p] \equiv 0$ for all i,j in (2.50) it follows that $\psi_i[x,p]$ is subregular if

$$-p_1\ c_1\gamma[1,p]\ p_1\ V[x\,\gamma[1,p]p_1] \leq \sum_{j\neq 1} p_j\ c_j\ \delta[j,1,p](1-p_1)\ V[-x\,\delta[j,1,p](1-p_1)]. \qquad (2.62)$$

Following the procedure described in case 1 we will have the following equation:

$$\frac{1}{V[x]} = \frac{c_1}{\max\limits_{j\neq 1}\{c_j\}} \cdot \qquad (2.63)$$

By similar reasoning there exists unique value for x that satisfies equation (2.63).

General Case: The Reward-penalty scheme ($\beta[i,j,p] \not\equiv 0, \delta(i,j,p) \not\equiv 0$ for all i,j)

Since $\alpha[i,p]$, $\gamma[i,p]$, $\delta[i,j,p]$ and $\beta[i,j,p]$ are all bounded above by unity and as the function $V[\cdot]$ is strictly monotonically increasing, we can have the following inequalities:

$$\begin{aligned} &V[-x\,\alpha[1,p](1-p_1)] > V[-x(1-p_1)], \\ &V[+x\,\gamma[1,p]\ p_1] < V[x,p_1], \\ &V[-x\,\delta[1,j,p](1-p_1)] > V[-x\,(1-p_1)], \\ &V[+x\,\beta[1,j,p]p_1] < V[x,p_1]. \end{aligned} \qquad (2.64)$$

Clearly $\phi[x,p]$ is a subregular function if $F_1[x,p] \geq 0$. In view of (2.64) and (2.50), $F_1[x,p] \geq 0$ holds if

$$G[x,p] = \frac{V[-x(1-p_1)]}{V[xp_1]}$$

$$\geq \frac{p_1^2 c_1 \gamma[1,p] + \sum_{j\neq 1} p_1 p_j d_j \beta[j,1,p]}{\sum_{j\neq 1} p_j(1-p_1)c_j\delta[j,1,p] + p_1(1-p_1)d_1\alpha[1,p]} \quad (2.65)$$

Define

$$\Delta[x,p] = \log_e G[x,p] = \int_{x p_1}^{-x(1-p_1)} H'[u]du \; . \quad (2.66)$$

Using (2.56) it can be seen that

$$\Delta[x,p] = \int_{x p_1}^{-x(1-p_1)} H'[u]du > -\int_0^x H'[u]du. \quad (2.67)$$

Taking exponential throughout (2.67) we obtain

$$G[x,p] \geq \frac{1}{V[x]} \; . \quad (2.68)$$

Further, using (C.1) - (C.2), it can be shown that

$$\frac{p_1^2 c_1 \gamma[1,p] + \sum_{j\neq 1} p_1 p_j d_j \beta[j,1,p]}{\sum_{j\neq 1} p_j(1-p_1)c_j\delta[j,1,p] + p_1(1-p_1)d_1\alpha[1,p]}$$

$$= \frac{c_1 \sum_{j\neq 1} p_1\delta[1,j,p](1-p_j) + p_1 \sum_{j\neq 1} p_j d_j \beta[j,1,p]}{\sum_{j\neq 1} p_j(1-p_1)c_j\delta[j,1,p] + p_1 d_1 \sum_{j\neq 1} \beta[1,j,p]p_j} \quad (2.69)$$

$$\leq \frac{c_1 \sum_{j\neq 1} p_1\delta[1,j,p](1-p_j) + p_1 \max_{j\neq 1}\{d_j\} \sum_{j\neq 1} p_j\beta[j,1,p]}{\sum_{j\neq 1} p_j(1-p_1)\min_{j\neq 1}\{c_j\}\delta[j,1,p] + p_1 d_1 \sum_{j\neq 1} p_j \beta[1,j,p]} \; .$$

It can be seen using (S.1) - (S.2) and inequality $d_1 > d_2 > d_3 \ \ldots > d_M$ that

$$e^* = \frac{c_1 A + d_2 B}{c_2 A + d_1 B} \geq \frac{p_1^2 c_1 \gamma[1,p] + \sum_{j\neq 1} p_1 p_j d_j \beta[j,1,p]}{\sum_{j\neq 1} p_j(1-p_1)c_j\delta[j,1,p] + p_1(1-p_1)d_1\alpha[1,p]} \; , \quad (2.70)$$

where

$$A = \sum_{j\neq 1} p_j(1-p_1)\ \delta[j,1,p]$$

$$B = \sum_{j \neq 1} p_1 p_j \beta[j,1,p] .$$

In view of the fact that

$$d_2 = \max_{j \neq 1} \{d_j\} ,$$

$$c_2 = \min_{j \neq 1} \{c_j\} ,$$

we have $e^* < 1$. From (2.68) and (2.70)

$$F_1[x,p] \geq 0 \quad \text{if} \quad \frac{1}{V[x]} \geq e^* .$$

Since $e^* < 1$, $V[x] = \frac{1}{e^*}$ has a unique solution $x > 0$ such that $\phi_1[x,p]$ is subregular.

Remark 2.11: In all three cases: reward-inaction, inaction-penalty and reward-penalty there exists a lower bound on $\Gamma_1(p)$. This is in sharp contrast with the existing result in the literature, wherein for the inaction-penalty (absolutely expedient) algorithms [L4] no lower bound on $\Gamma_1(p)$ has been established. One of the reasons for this anomaly is that none of currently available inaction-penalty (absolutely expedient) algorithms [L4] have V_M as the only set of absorbing states.

Having established the lower bound on $\Gamma_1(p)$, we now state our main result. The following theorem states that strong absolute expediency is a sufficient condition for ε-optimality.

Theorem 2.4. For every $\varepsilon > 0$ and $p(0) = p \in S_M^0$ there exists $0 < \theta^* < 1$ such that for all $0 < \theta < \theta^*$, the learning algorithm under the conditions (C.1) - (C.2), (A.1) - (A.2), and (S.1) - (S.2) is such that

$$\lim_{k \to \infty} | E [\eta(k)] - d_1 | < \varepsilon . \tag{2.71}$$

Proof. From the definition of $\eta(k)$ we have

$$E[\eta(k)] = \sum_{i=1}^{M} E[p_i(k)]d_i \tag{2.72}$$

also

$$\begin{aligned} \lim_{k\to\infty} E[\eta(k)] &= \lim_{k\to\infty} \sum_{i=1}^{M} E[p_i(k)]d_i \\ &= \sum_{i=1}^{M} d_i \lim_{k\to\infty} E[p_i(k)] \\ &= \sum_{i=1}^{M} d_i E[\lim_{k\to\infty} p_i(k)]. \end{aligned} \tag{2.73}$$

From corollary (2.2) it follows that $\lim_{k\to\infty} p(k) \in V_M$ with probability one, that is $\lim_{k\to\infty} p_i(k) = 1$ or 0 in limit. From (2.33) we have

$$\begin{aligned} &\text{Prob}\,[\lim_{k\to\infty} p_i(k) = 1 \mid p(0) = p] = \Gamma_i(p)\,, \\ &\text{Prob}\,[\lim_{k\to\infty} p_i(k) = 0 \mid p(0) = p] = 1 - \Gamma_i(p)\,. \end{aligned} \tag{2.74}$$

Therefore it can be easily shown that

$$E\,[\lim_{k\to\infty} p_i(k) \mid p(0) = p] = \Gamma_i(p)\,. \tag{2.75}$$

Substituting (2.75) into (2.73), we obtain

$$\lim_{k\to\infty} E\,[\eta(k)] = \sum_{i=1}^{M} d_i\, \Gamma_i(p)\,. \tag{2.76}$$

From (2.76) we can have

$$\lim_{k\to\infty} E[\eta(k)] - d_1 \le \Gamma_1(p)d_1 - d_1 + \max_{j\neq 1}\{d_j\} \sum_{j\neq 1} \Gamma_j\,. \tag{2.77}$$

Since $\max_{j\neq 1}\{d_j\} = d_2$ and $\sum_{j\neq 1} \Gamma_j(p) = 1 - \Gamma_1(p)$, combining these and (2.77) we have

$$\lim_{k\to\infty} E[\eta(k)] - d_1 \le (1-\Gamma_1(p))(d_2 - d_1)\,.$$

That is

$$\lim_{k\to\infty} |\,E[\eta(k)] - d_1| \le (1 - \Gamma_1(p))\,|\,d_2 - d_1|\,. \tag{2.78}$$

Since $\phi_1[x,p] \leq \Gamma_1(p)$, we get

$$\lim_{k\to\infty} | E[\eta(k)] - d_1 | \leq (1-\phi_1[x,p]) | d_2 - d_1 | . \qquad (2.79)$$

Also, for any $p \in S_M^0$, we know that

$$\lim_{\theta\to 0} \phi_1[x,p] = \lim_{\theta\to 0} \frac{1-e^{-x\,p_1/\theta}}{1-e^{-x/\theta}} = 1 . \qquad (2.80)$$

Combining (2.79) and (2.80) we see that for any $\delta > 0$ there exists $0 < \theta^* < 1$ such that for all $0 < \theta < \theta^*$

$$\lim_{k\to\infty} | E[\eta(k)] - d_1 | \leq | \delta(d_2 - d_1) | . \qquad (2.81)$$

The theorem follows by choosing

$$\delta = \frac{\varepsilon}{|d_2 - d_1|} . \qquad \square$$

Remark 2.12: The quantity $\Delta p_1(k)$ defined as

$$\Delta p_1(k) = E[\delta p_1(k) \mid p_1(k) = p] = O(\theta) \qquad (2.82)$$

is the conditional expected step size in $p_1(k)$ at time k. The expected step size $\Delta p_1(k)$ may be used as a convenient measure of the speed of convergence of a learning algorithm. If $\Delta p_1(k)$ for learning algorithm T1 is larger than $\Delta p_1(k)$ for learning algorithm T2 for all $p_1(k) \in (0,1)$, then the scheme T1 is said to be faster than the scheme T2. The validity of this definition is confirmed by the computer simulation [V5]. From (2.82) it can be said that the larger is θ, the higher is the speed of convergence. From theorem 2.4, increasing θ will result in smaller value for the lower bound of the probability of convergence. From the above discussion, there is a trade-off between the speed of convergence and probability of converging to the desired state. Thus a fastly converging scheme may with a high probability converge to an undesired state. It would be very desirable to design schemes with a

high probability of converging to the desired state but at the same time not sacrificing the speed of convergence too much.

2.4 Simulations: The algorithm (2) was simulated for the choice of the functions given in example 1. The average value of $\eta(k)$ averaged over 20 sample runs for various values of θ are given in the following Table 2.1.

Table 2.1

k	$E[\eta(k)]$		
	$\theta = 0.2$	$\theta = 0.1$	$\theta = 0.05$
0	0.5333	0.5333	0.5333
50	0.6379	0.5935	0.5668
100	0.6944	0.6510	0.6017
200	0.7505	0.7184	0.6598
500	0.7860	0.7693	0.7378
1000	0.7928	0.7865	0.7703
1500	0.7953	0.7913	0.7808
2000	0.7968	0.7938	0.7862
2500	0.7975	0.7952	0.7895
3000	0.7999	0.7996	0.7927

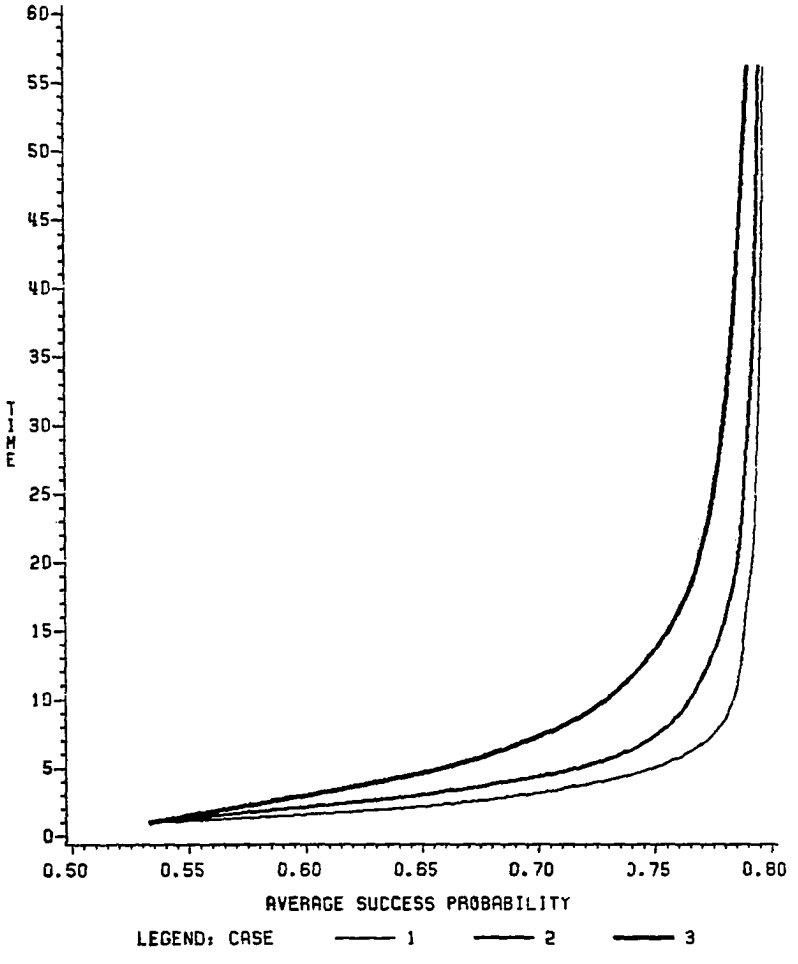
2.5 Conclusions

In this chapter conditions for ε-optimality of a general class of absorbing barrier and strongly absolutely learning algorithm are derived. As a consequence a new class of learning algorithmswhich has

identical behavior under the occurrence of success and failure is obtained. The concept of strong absolute expediency is introduced. The class of learning algorithms presented in this chapter is quite a general one and subsumes most of the well-known schemes available. The choice of functions $g_s^i[p]$ in (1) used in this chapter is quite untraditional in the sense that in almost all the papers in mathematical psychology $g_s^i[p]$ is made proportional to p_s for all i and s, $s \neq i$. Also in almost all the papers on learning automata, $g_i^i[p]$ is made proportional to $(1-P_i)$ and $g_s^i[p]$ is made proportional to p_s for all $s \neq i$. Because of this there is a disparity in the behavior of the algorithm (1) under success and failure. However, the choice of functions $g_s^i[p]$ for all i and s given in this thesis induces identical behavior of the algorithm (1) under success and failure.

All three types algorithms: reward-penalty, reward-inaction and inaction-penalty are absorbing barrier learning algorithms. This is in sharp contrast with the properties of currently available absolutely expedient learning algorithms wherein the reward-penalty and reward-inaction algorithms are absorbing barrier type, but the inaction-penalty is not. In fact, in all the inaction-penalty algorithms of the absolutely expedient type known so far, every state in $\delta S_M = S_M - S_M^0$ is an absorbing state. The modified definition of strong absolute expediency is in fact motivated by the existence of the absorbing barrier algorithms of the reward-penalty, reward-inaction and inaction-penalty types. For this class of algorithms if p(k) reaches the boundary ($p_j = 0$ for some j) of simplex S_M, then p(k) will continue to remain in that boundary, that is the dimensionality of the algorithm is reduced

EXAMPLE 1



Case #1 $\theta = 0.2$

Case #2 $\theta = 0.1$

Case #3 $\theta = 0.05$

Figure 2.1

automatically.

The convergence properties of this class of learning algorithms is studied. Bounds on the probability of convergence to the desired state is obtained. It is shown that for all three types of algorithms: reward-inaction, reward-penalty and inaction-penalty, there exists a lower bound on the probability of convergence to the desired action. This is in sharp contrast with the existing results in the literature, wherein for the inaction-penalty (absolutely expedient) algorithms no lower bound on probability of convergence to the desired action has been established. One of the reasons for this anomaly is that none of the currently available inaction-penalty (absolutely expedient) algorithms have V_M as the only set of absorbing states. In this chapter it is also shown that strong absolutely expedient algorithms lead to ε-optimality in all stationary random environments.

CHAPTER III

A LEARNING APPROACH TO PRIORITY ASSIGNMENT IN AN M/M/1 QUEUING SYSTEM WITH UNKNOWN PARAMETERS

3.1 Introduction

An important subject in queuing theory is that of priorities. In a priority queuing system there are k ($k > 1$) different classes of jobs and an arriving job belongs to one of these k different classes. These classes may be distinguished according to some measure of importance, and to indicate the relative measure of importance a priority index i is associated to each class. It is conventional that the larger the value of the index associated with a class, the lower is the priority associated with that class, that is, preferential treatment is given to the class with lower priority index. The discipline according to which the server selects the next unit from these classes is called a priority discipline.

Priority discipline is specified by two rules. The first rule indicates the manner in which a unit is selected for service. This rule may depend only on the knowledge of the priority class to which a unit belongs, or it may depend solely or partially on other considerations relating to the existing state of the system, e.g., the type of unit last served or the waiting time of units present. The former disciplines are called exogenous priority disciplines while the latter are called endogenous priority disciplines. In exogenous priority disciplines the

decision to select the next unit for service depends only on the priority class: a unit of the i^{th} class if present is always taken for service prior to the unit of the j^{th} class ($i < j$).

The second rule specifies the manner in which the unit is served after entering service. This rule may also be state-independent or state dependent. In preemptive and nonpreemptive disciplines the second rule is independent of the state of the system. A preemptive priority system is one in which a unit of higher priority takes, on arrival, immediate precedence over units of lower priority. The unit whose service is interrupted returns to the service point only when there are no higher priority units remaining in the system. Under non-preemptive disciplines, a job, once at the service, remains there until his service is complete, then the next unit for service is the one with highest priority among these classes. In "Discretionary Priority Discipline" [A2] where the server is permitted to use his discretion to continue or discontinue the service of a unit, the second rule is time-dependent.

The preemptive discipline can be broken down into the following categories, depending upon the manner in which the job is served on its reentry:

(i) Preemptive resume: The preempted unit resumes service from the point where it was interrupted.

(ii) Preemptive repeat without resampling: The preempted unit on its reentry requires the same amount of service as it required on its earlier entry. That is, the service time for a job is sampled only once, regardless of the number of

preemptions experienced.

(iii) Preemptive repeat-with resampling: The preempted unit on its reentry requires a random service time independent of past preemptions and wasted service time. That is, each time a single job returns to the service station a new service time is sampled.

There are other types of priority disciplines reported in the literature; among those: Alternating-priority discipline, priority discipline with reorientation time, priority with balking and reneging, round-robin discipline and dynamic priority discipline.

Alternating priority discipline: Under this discipline, if the server becomes free to take up a unit after completing service on a unit of type i ($1 \leq i \leq k$), the next unit to be taken for service will be an i-type if available in the queue. If no i-type unit is available, the server selects the unit with the lowest priority index. Note that this discipline is endogenous in character because the decision to select a unit depends not only on the priority class to which it belongs but also on the type of unit last serviced.

Priority discipline with reorientation time: In this discipline the server has to perform an orientation while switching from servicing one type of unit to another type of unit. This type of discipline arises from the fact that the assumption that the units of various classes can be serviced without any loss of time in reorienting the server to take up a unit may not be true.

Priority discipline with balking and reneging: A unit is said to balk if on arrival it does not join the queue, and is said to renege if

after joining the queue it gets impatient and leaves the queue without getting service. It is conceivable that in situations where more than one class of units are being serviced under a given priority discipline units of different priority exhibit balking and reneging behavior. Obviously, the lower priority class units will be more suspectible to such behavior than the higher-priority class units.

Dynamic priority discipline: Under static priority discipline such as preemptive or nonpreemptive discipline mentioned before, the distribution of priority indexes of jobs arriving at a particular queue is stationary. Under a dynamic priority discipline, the distribution of indexes changes over time. In this discipline each succeeding arrival has lower and lower probability of taking preference over a job already in queue. Dynamic priority discipline is endogenous in nature.

Round robin priority discipline: Under this discipline, each unit in the system is served for a specified time in a round robin fashion, i.e., serves each unit for a short period, leaves it, takes the next unit and serves it for the same period, and so on. This procedure automatically gives priority to those units which have shorter service requirements. The limiting case of round robin discipline in which time quantum is allowed to approach zero is called processor-sharing discipline [K3].

The evaluation of various characteristics of a priority system under given priority discipline is an important objective in any priority system; the queue-length distributions may be important from the design point of view, the waiting time from the job's point of view,

and the busy period from the server's point of view.

Below, we briefly summarize some major results on priority queuing that are pertinent to our present study.

It is shown [K4] that so long as the queuing discipline selects jobs in a way that is independent of their service time or any measure of their service time, then the distribution of the number of jobs in the system and average waiting time will be invariant to the order of service.

Cox and Smith [C2] considered a priority system with k independent Poisson streams and arbitrary service time distribution for each of the classes. Assuming a waiting cost per unit time of c_i units for each class i job and a nonpreemptive service discipline, they showed that a policy that ranks classes according to the "μc rule"

$$(1)\quad \mu_1 c_1 \geq \mu_2 c_2 \geq \ldots \geq \mu_i c_i \geq \ldots \geq \mu_k c_k$$

will minimize the average waiting cost. In (1) $\mu_i = \frac{1}{\mu_i^{-1}}$, where μ_i^{-1} is the mean service requirements of class i jobs. So the optimal priority assignment is in the decreasing order of $\mu_i c_i$, with the highest priority for the class with the highest value of $\mu_i c_i$. This policy will minimize the average cost of the system for preemptive resume discipline providing service time distributions are exponential. [J1]. Evidently if the unit delay costs for different classes of jobs are equal, assignment of priorities in increasing order of mean service time requirement will also minimize the waiting cost. This should be noted that if the unit delay cost is 1 for all the classes, total delay cost is essentially the same as total waiting time, so ordering according to average service time will minimize the average waiting time.

Mova and Ponamarenko [M7] analyzed an (M/M/C) priority system with k independent Poisson streams by means of Markov decision processes. They showed, by a numerical example, that the simple μc rule does not hold for a system with finite queue size, and that the optimal priority depends on the number of jobs in the system as well as on the arrival rates of all classes. The optimal stationary priority policy was computed by linear programming.

In order to develop a meaningful priority scheme, we need to know the probabilistic characteristics of the jobs in various classes, such as the arrival and service time distributions. But in many practical situations this information may not be known in advance. An interesting question in this connection is that, if the probabilistic (arrival and service time) characteristics of jobs in various classes are not known in advance, how to go about developing a meaningful priority assignment among the various classes. That is, how to design a priority queuing system in which the priorities are worked out directly without a priori knowledge of the input and service characteristics of the system.

Varshavskii, Meleshina and Tsetlin [V6] first considered this problem and proposed a solution using fixed structure learning automata where one automaton is used for each class. This latter class of algorithm was developed by Tsetlin [T2] in the context of modelling of collective behaviors. The state transitions of automaton are indirectly controlled by the service requirements of jobs in each class. At any given instance the class corresponding to the automata with highest state number is chosen for service. The graph of transitions within the set of automata states, corresponding to a single class is shown in Figure 3.1.

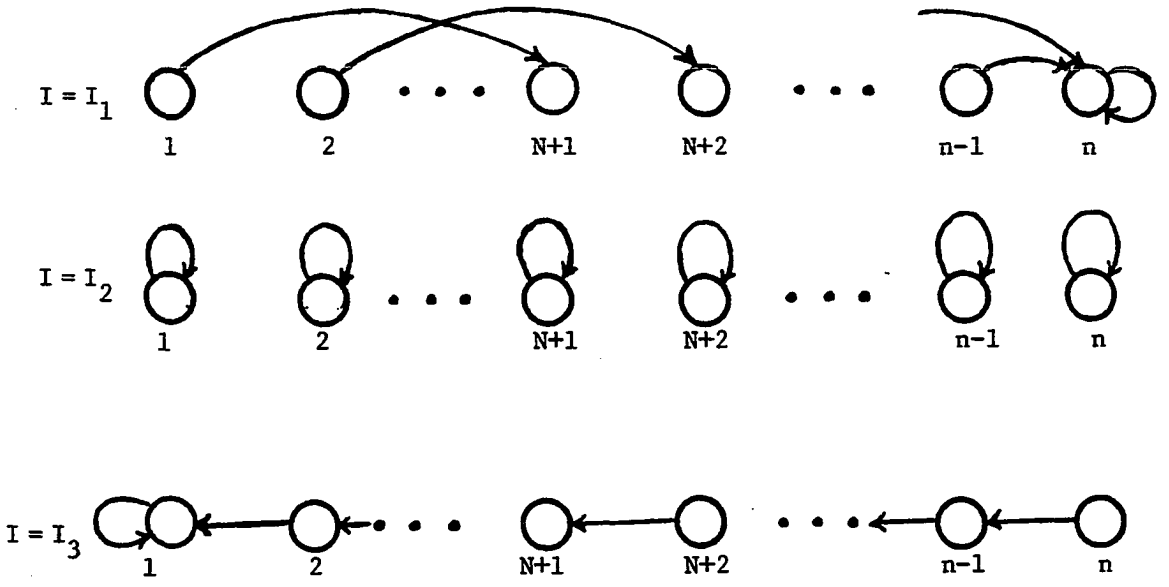


Figure 3.1

Input I_1 corresponds to the selection of a job by the server with the automata turned on; the Input I_2 corresponds to the case where the automata is turned off and Input I_3 corresponds to the case where the server is busy and the automata is turned on. Thus, when a job is selected by a class, states with the numbers k (k=1, ..., n-N) transfer to the state having the number k+N, and the states with number n-N+1, ..., n transfer to the state with number n. n is the total number of states of the automata and N may be chosen to be, for example, equal to the average serving time, averaged over all classes. When the server is busy, the number of the state of the automata is lowered by one during each time cycle. State 1 is transferred into itself. The higher the state number of the automata, the more favored the corresponding class is at a given time.

This approach leads to a Markov chain with very complex transition structure which does not lend itself to formal analysis. In view of this difficulty Varshavskii et al. [V5] resorted to simulation to evaluate the effectiveness of this approach. They showed through extensive simulation that if the number of states of each automaton is chosen to be large to start with, their algorithm would asymptotically approach the priority assignment that would be used if the probabilistic characteristics of the system to be known.

In this chapter, using a class of learning algorithms very similar to those that are used in Mathematical Psychology [N2, N8, L3, N3] we formally show that a randomized choice of classes for service would indeed asymptotically converge to the classical priority assignment rule with a probability as close to unity as desired. That is,

the proposed system will asymptotically assign the highest priority to the class with the shortest average service time. These situations are modeled in the framework of single server priority queuing system when there are only two classes of jobs and preemption of the service is not allowed. In section 2,we describe the algorithm and state the main theorem. A proof of this theorem is presented in section 3. A number of interesting variations of this system are presented in section 4 and part of section 3. Extensions to more than two classes are given in section 5.

3.2 A Learning Algorithm and Statement of the Problem

The priority assignment queuing system considered in this chapter is represented schematically in Figure 3.2. We have considered a queuing system with two classes of jobs. There is only one server, no preemption and queue length of each class is allowed to be infinite. Within each class, the service is on a first come first served basis. Jobs arrive into two queues from a group of 2 different classes of infinite population according to independent Poisson process with constant rate λ_i (>0) $i = 1,2$. The service time requirement for jobs in class i and j ($\neq i$) are independent. Further, within each class the service times are independent and are identically distributed with exponential distribution having mean μ_i^{-1}, $i = 1,2$. The following assumption is very crucial in our analysis.

(A1) The parameters λ_i and μ_i^{-1}, $i = 1,2$ are not known. Without loss of generality let $\mu_1^{-1} < \mu_2^{-1}$.

Let γ^i be the random service time of a typical job from $i = 1,2$. Consider a stage when k_i jobs from class i have been served. Then,

$k = k_1 + k_2$ is the total number of jobs served by the system. Define a dynamic threshold

$$T(k) = 1/2[T^{(1)}(k_1) + T^{(2)}(k_2)] ,$$

where

$$T^{(i)}(k_i) = \frac{1}{k_i} \sum_{j=1}^{k_i} \gamma^{(i)}(j) , \quad T^{(i)}(0) = 0 ,$$

and $\gamma^{(i)}_{(j)}$ is the random service time for the j^{th} job in the i^{th} class. T(k) is essentially the average service time taken over both classes up to time k. From estimation theory [M6] it is evident that

$$T^{(i)}_{(1)}, T^{(i)}_{(2)}, \ldots, T^{(i)}(k_i) , \qquad i = 1,2 ,$$

is a sequence of estimators of μ_i^{-1} , i = 1,2, and

$$\mathrm{Var}[T^{(i)}(k_i)] = E[(T^{(i)}(k_i) - \mu_i^{-1})^2] , \qquad i = 1,2 ,$$

and

$$\lim_{k_i \to \infty} \mathrm{Var}\,[T^{(i)}(k_i)] = 0 , \qquad i = 1,2$$

and hence the sequence $\{T^{(i)}\}$, i = 1,2, is a mean-squared error consistant sequence of estimator of μ_i^{-1}, and also for every $\varepsilon > 0$

$$\lim_{k_i \to \infty} \mathrm{Prob}\,[\mu_i^{-1} - \varepsilon < T^{(i)}(k_i) < \mu_i^{-1} + \varepsilon] = 1.$$

From the above discussion it is evident that for large k_1 and k_2, the difference between T(k) and $1/2[\mu_1^{-1} + \mu_2^{-1}]$ is small with a very high probability.

3.2.1 A Dichotomy of Service Characterization

It is assumed that class i has a service time distribution with exponential density function

$$f_i(t) = \mu_i e^{-\mu_i t} .$$

Given that k jobs have been given service, k = 1,2,3,..., define

$$\begin{cases} x(k) = 1 & \text{if} \quad \gamma^{(i)}(k+1) \leq T(k) \\ x(k) = 0 & \text{otherwise.} \end{cases}$$

Clearly

$$d_i(k) = \text{Prob}[x(k) = 1] = \int_0^{T(k)} f_i(t)d_t$$
$$= 1-e^{-\mu_i T(k)}$$

and

$$c_i(k) = \text{Prob}[x(k) = 0] = 1-\int_0^{T(k)} f_i(t)d_t$$
$$= e^{-\mu_i T(k)}.$$

Since T(k) is a random variable, so is $d_i(k)$ i = 1,2. Clearly, $d_i(k)$ [$c_i(k)$] is the probability that $(k+1)^{th}$ job will have its service time less (more) than T(k) if that job is from class i. Define d(k) = $(d_1(k), d_2(k))$. Notice

$$0 < d_i(k) < 1$$

with probability one for i = 1,2 and all $k \geq 1$. Further, since $\mu_1^{-1} < \mu_2^{-1}$ (assumption A1) it is easily seen that

$$d_1(k) > d_2(k) \tag{3.1}$$

with probability one for all k. From this it follows that

$$\eta = \sup_k \frac{d_2(k)}{d_1(k)} < 1.$$

Also for large k_1, k_2, the difference $d_i(k)$ and $1-e^{-\frac{\mu_i}{2}[\mu_1^{-1} + \mu_2^{-1}]}$ is small with a high probability.

Remark 3.1: It should be noted that the components of vector d are not constants, as assumed in the abstract automata formulation of Chapter I.

Rather the comparisons that define the probabilities are statistically dependent, so that d_i's are time varying parameters, dependent in fact on the particular process that is evolving.

Remark 3.2: Dynamic threshold at any time will partition the service time density function into two regions. Depending on in which region the service time of $(k+1)^{th}$ job lies, x(k+1) takes one of the values 0 or 1 (0 is called failure and 1 is called success). If we fix the threshold before the operation of the system starts, using information from the past experience, process of portioning will be done only once and remain fixed throughout the operation of the system.

3.2.2 Learning Algorithm

Let $p_j(k)$ be probability with which the DD in figure 3.2 decides to pick up the $(k+1)^{th}$ job for service from class j. Define $p(k) = (p_1(k), p_2(k))$ where $p_1(k) + p_2(k) = 1$ and $k \geq 0$. Initially, $p_1(0) = p_2(0) = \frac{1}{2}$ with probability one.

With the above preliminaries, basic operation of DD can be explained as follows: Increase the probability of choosing the i^{th} class at time (k+1) if a job selected from i^{th} class has service time less than T(k) (resulted in success), and decrease the probability of choosing the i^{th} class if a job selected from that class has service time greater than T(k) (resulted in failure). The increase and decrease are called reward and penalty, respectively. The probability of choosing another class is adjusted to keep total probability equal to one. How these probabilities are adjusted is dictated by an updating algorithm of the type discussed in Chapter II, when they are updated is dictated by a timing updating algorithm. That is, updating algorithm decides how to

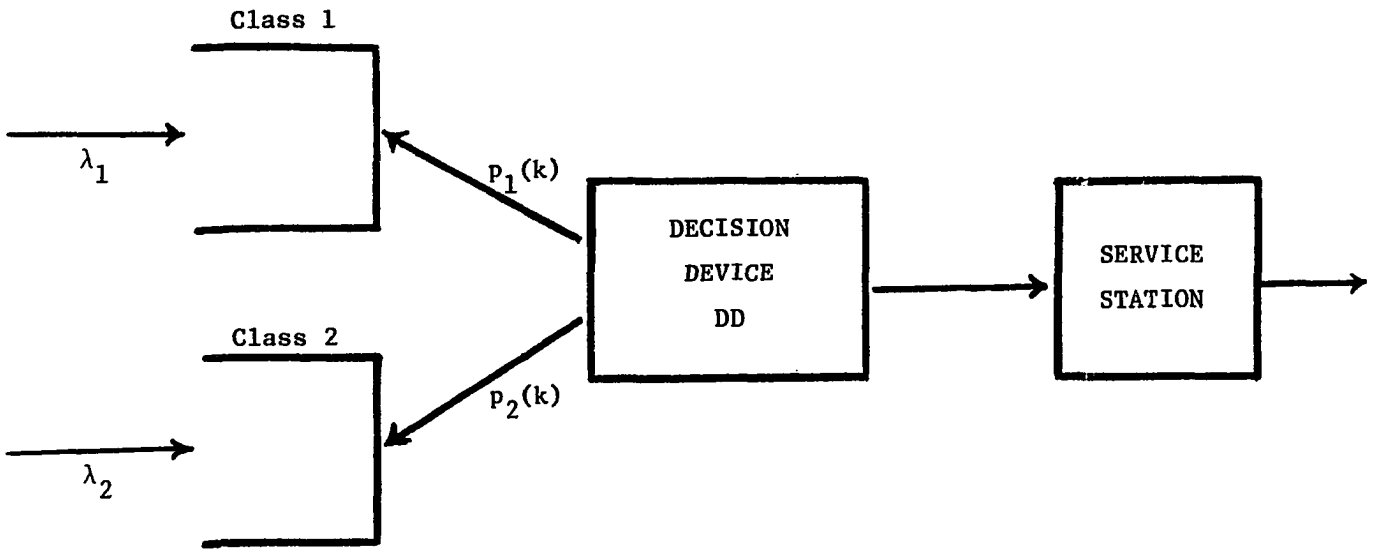

Class 1
λ_1
$p_1(k)$
DECISION
DEVICE
DD
SERVICE
STATION
Class 2
λ_2
$p_2(k)$

Figure 3.2

update p(k) if to be updated, and the update timing algorithm decides as to when to update p(k). Jobs are selected from class 1 and 2 according to the probability vector P. Dynamic threshold T(k) is updated whenever service to the currently being serviced job is finished. In this chapter we propose two different update timing algorithms. We show any of these two algorithms together with the linear reward-inaction updating algorithm will guarantee convergence of the system to the desired class.

Updating Algorithm: Linear reward-inaction updating algorithm is given below:

$$P_i(k+1) = p_i(k) + \theta(1-p_i(k))\ J_i - \theta p_i(k)\ J_j \qquad j \neq i\ ,$$

where J_s denotes the indicator of the event and $0 < \theta < 1$ is step length parameter.

$J_s = 1$ if the probability of choosing class s is increased

$= 0$ otherwise.

Linear reward-inaction algorithm is the only linear algorithm which is strongly absolutely expedient (refer Remark 2.5).

Update Timing Algorithm 1: If a class is chosen as a sample realization from distribution P, and is found to be non-empty, update the probability of choosing that class provided another class is also non-empty. If the selected class is empty, choose a job from another class and do not update the probability vector P.

Update Timing Algorithm 2: If a class is selected as a sample realization from distribution P and is found to be non-empty, update the probability of choosing that class. If the selected class is empty,

select a job from another class and do not update the probability vector P.

In timing algorithm 2, the probability of choosing a class is updated regardless of the status (being empty or non-empty) of the other class, whereas in timing algorithm 1, the status of the other class affects updating of the probability of choosing the selected class. In both algorithms whenever a class is chosen for job selection and is found to be empty, DD will select a job from another class without adjusting the probability vector P. If both classes are empty, DD remains off until it receives a signal that a job is entered into one of the classes. DD will be off during serving a job and also during updating dynamic threshold.

The following algorithm (L1) describes the operation of the system when the system uses timing algorithm 1 and the linear reward-inaction updating algorithm.

Algorithm L1:

Step 1: Let i be the class from which the $(k+1)^{th}$ job is chosen for service as a sample realization from p(k), for $k \geq 0$. At this instant, if the queue of this chosen class is non-empty, then select the job in front of the queue for service. Otherwise, choose the job in front of the other queue. Complete the service and note $\gamma^{(i)}(k+1)$.

Step 2: The update timing algorithm

At the instant class i is chosen for service (in step 1 as a sample realization from p(k)) if the queues corresponding to both class i as well as that of class j ($\neq$i) are both non-empty

then update p(k) using step 3. Otherwise p(k+1) = p(k) and go to step 4.

Step 3: Update Algorithm for p(k):

If the class i is chosen for service, then

$$\left.\begin{aligned} p_i(k+1) &= p_i(k) + \theta(1-p_i(k)) \\ p_j(k+1) &= p_j(k) - \theta P_j(k) \quad , \quad j \neq i \end{aligned}\right\} \text{ if } \gamma^{(i)}(k+1) \leq T(k)$$

and

$$p(k+1) = p(k) \qquad \text{if} \qquad \gamma^{(i)}(k+1) > T(k).$$

Step 4: Update T(k) using $\gamma^{(i)}(k+1)$ as follows:

$$T(k+1) = 1/2[T^{(i)}(k_i+1) + T^{(j)}(k_j)] \qquad i \neq j$$

where

$$T^{(i)}(k_i+1) = \frac{1}{k_i+1} \sum_{\ell=1}^{k_i+1} \gamma^{(i)}(\ell).$$

Step 5: Go to step 1 until one of the components of p(k) is unity.

The following restrictive assumption is needed to simplify the analysis.

(A2). Except for the actual service needed by the jobs in step 1 all other overhead operations such as the decision to pick a class from p(k), checking whether the two queues are non-empty, the updating of p(k) and T(k) are all instantaneous.

In other words, there is no time delay involved in these overhead operations. Obviously this is a restrictive assumption. However, by employing fast microprocessors, the overall overhead operation can be made very small, if not zero. We state our main result.

Theorem 3.1: Under the assumptions A1 and A2, if p(k) evolves according to the above learning algorithm (L1) then for every $\varepsilon > 0$, there exists

a θ^* such that $0 < \theta^* < 1$ and for all $0 < \theta < \theta^*$

$$\text{Prob}\left[\lim_{k \to \infty} p_1(k) = 1\right] \geq 1 - \epsilon.$$

Stated in words the above theorem asserts that the above learning algorithm (L1) would evolve to the now classical assignment rule with a probability as close to unity as desired.

Remark 3.3: Obviously, there are other possible choices for update algorithms. The above update algorithm has been extensively studied in the context of Mathematical Psychology [N2, N8] and in the context of learning automata [L3, N3]. Very recently similar learning algorithms have been applied to the two-person zero sum games [L8], two person decentralized team [L6] and decentralized routing in telephone networks [S8].

3.3 Proof of the Main Result

The proof of theorem 1 is presented in various steps. Let $\rho_i(k)$ be the probability that the queue of the class i is non-empty at the time when the $(k+1)^{th}$ job is chosen for service. Clearly, $\rho_i(k)$ depends on λ_i, μ_i, $i = 1,2$ and on $p(s)$, $0 \leq s \leq k$ and also we assume that $0 < \rho_i(k) < 1$ with probability one for all $k \geq 0$ and $i = 1,2$. Let $\alpha_i(k)$ be the probability with which $p(k)$ is updated. Then

$$\alpha_1(k) = \rho_1(k)\rho_2(k)[p_1(k)\ d_1(k) + p_2(k)\ d_2(k)] > 0.$$

Define

$$F_k = \sigma[p(0), \rho(0), d(1); p(1), \rho(1), d(2); \ldots, p(k), \rho(k)]$$

be the smallest Borel-field [D1] generated by the indicated random variables. That is, F_k contains all the information regarding the choices of classes in step 1; states of the queues in step 2 and the

results of the comparison of the service time with the dynamic threshold up to the instant when the $(k+1)^{th}$ job has been given service. Let

$$\Delta p_i(k) = E[p_i(k+1) \mid F_k] - p_i(k). \tag{3.1a}$$

The following lemma is of immediate consequence.

Lemma 3.1: When $p_i(k)$ is defined according to a linear reward-inaction algorithm and timing algorithm 2, the increment in the conditional expectation of $p_1(k)$ is always positive, that is, $\Delta p_1(k) \geq 0$ for all $p_1(k) \in (0,1)$ with equality holding only when $p_1(k) = 0$ or 1.

Proof: Using linear reward-inaction update algorithm, we have

$$\begin{aligned}\Delta p_1(k) &= E[\ \theta(1-p_1(k))J_1 - \theta p_1(k)J_2 \mid F_k] \\ &= \theta(1-p_1(k))\ E[J_1 \mid F_k] + \\ &\quad \theta p_1(k)\ E[J_2 \mid F_k].\end{aligned} \tag{3.2}$$

In view of the update timing algorithm 1, $E[J_1 \mid F_k]$ and $E[J_2 \mid p_1(k) = p_1]$ can be written as

$$E[J_1 \mid F_k] = p_1(k)\ \rho_1(k)\ \ {}_2(k)\ d_1(k) \tag{3.3}$$

and

$$E[J_2 \mid F_k] = p_2(k)\rho_1(k)\ \ {}_2(k)\ d_2(k). \tag{3.4}$$

Substituting (3.3) and (3.4) in (3.2) we would have

$$\Delta p_1(k) = \theta\rho_1(k)\rho_2(k)\ p_1(k)\ p_2(k)[d_1(k) - d_2(k)]\ .$$

The lemma follows from the properties of $d_i(k)$ $i = 1,2$, given in (3.1).

□

Corollary 3.1: $\lim_{k\to\infty} p_1(k) = p_1^*$ exists and $p_1^* \in \{0,1\}$ with probability one.

Proof: Lemma 3.1 implies that $\{p_1(k)\}$ is a submartingale. By the martingale theorem, since $\{p_1(k)\}$ is non-negative and uniformly bounded,

limit $p_1(k) = p_1^*$ exists with probability one. Further, if $p_1(k) \notin \{0,1\}$
$k \to \infty$
then $p_1(k+1) \neq p_1(k)$ with nonzero probability for all k. Hence
$p_1^* \in \{0,1\}$ which constitutes the absorbing set for the process $\{p_1(k)\}$.□

Define

$$\psi[p,x,\theta] = e^{-\frac{x p_1}{\theta}},$$

where x is real, and

$$v[u] = \frac{e^u - 1}{u} \qquad \text{if } u \neq 0$$

$$= 1 \qquad \text{if } u = 0.$$

From Chapter II, V[u] is non-negative increasing and a convex function.

Lemma 3.2: There exists a positive, real number y such that

$$E[\psi[p_1(k+1),y,\theta] \mid F_k] \leq \psi[p_1(k),y,\theta].$$

Proof: By direct computation we obtain

$$E[\psi[p_1(k+1),\ x,\ \theta] \mid F_k] =$$

$$p_1(k)\ \rho_1(k)\ \rho_2(k)\ d_1(k)\ \psi[p_1(k)+\theta(1-p_1(k)),\ x,\ \theta]$$

$$+ p_1(k)\ [\rho_1(k)\ \rho_2(k)\ c_1(k)+B(k)]\ \psi[p_1(k),\ x,\ \theta]$$

$$+ p_2(k)\ \rho_1(k)\ \rho_2(k)\ d_2(k)\ \psi[p_1(k) - \theta p_1(k),\ x,\ \theta]$$

$$+ p_2(k)\ [\rho_1(k)\ \rho_2(k)\ c_2(k)+B(k)]\ \psi[p_1(k),\ x,\ \theta],$$

where

$$B(k) = [(1-\rho_1(k))\rho_2(k) + \rho_1(k)(1-\rho_2(k)) + (1-\rho_1(k))(1-\rho_2(k))].$$

From this, after some algebraic simplification, we obtain

$$E[\psi[p_1(k+1),\ x,\ \theta] \mid F_k] - \psi[p_1(k),\ x,\ \theta] =$$

$$-x\ p_1(k)\ p_2(k)\ \rho_1(k)\ \rho_2(k)\ \psi[p(k),x,\theta]\ F[p(k),\ x],$$

where

$$F[p(k),x] = V[-x(1-p_1(k))]\ d_1(k) - V[x\ p_1(k)]d_2(k).$$

Lemma 3.2 follows if we can show that

$$F[p(k),x] \geq 0 ,$$

that is, if

$$\frac{V[-x(1-p_1(k))]}{V[x\,p_1(k)]} \geq \frac{d_2(k)}{d_1(k)} . \tag{3.5}$$

Defining $H[u] = \ell_n V[u]$ (where ℓ_n stands for natural logarithm) and from the convexity properties of H[u] it can be shown that (Refer to Appendix D).

$$\frac{V[-x(1-p_1(k))]}{V[x\,p_1(k)]} \geq \frac{1}{V[x]} . \tag{3.6}$$

Combining (3.1) and (3.6) we see that (3.5) is true if

$$\frac{1}{V[x]} \geq \eta . \tag{3.7}$$

From the properties of function V[u] it follows that there exists a real number $y > 0$ such that for all $x \in (0,y)$, inequality (3.7) will be true. By setting $x = y$, the lemma follows.

Now consider the function

$$h[z, x, \theta] = \frac{e^{\frac{x}{\theta} z} - 1}{e^{\frac{x}{\theta}} - 1} ,$$

where $h[j, x, \theta] = j$ where $j = 0,1$

and

$$0 < h[z, x, \theta] < 1 \quad \text{for all } 0 < z < 1.$$

Lemma 3.3:

$$E[h[p_2(k+1), y, \theta] \mid F_k] \leq h[p_2(k), y, \theta] , \tag{3.8}$$

where y is defined in lemma 2.

Proof: It can be seen that

$$h[p_2(k+1),\ y,\ \theta\,] = \frac{e^{-\frac{y}{\theta}} p_1(k+1) - e^{-\frac{y}{\theta}}}{1 - e^{-\frac{y}{\theta}}} \ . \qquad (3.9)$$

Taking conditional expectations on both sides of (3.9), from the properties of conditional expectation and from lemma 3.2 the inequality (3.8) follows.□

Since $h[z,\ x,\ \theta\,]$ is continuous in z it follows that

$$\lim_{k\to\infty} h[p_2(k+1),\ y,\ \theta] = h[1-p_1^*,\ y,\ \theta\,]$$

with probability one. Since $p_1^* \in \{0,1\}$ with probability one, we obtain

$$h[1-p_1^*,\ y,\ \theta\,] = 1-p_1^* \quad \text{with probability one.} \qquad (3.10)$$

Proof of Theorem 3.1:

Taking expectations of both sides of (3.8), we obtain

$$h[p_2(0),y,\ \theta] \geq E[h[p_2(1),y,\ \theta]] \geq E[h[p_2(2),y,\theta]] \geq \ldots .$$

Thus

$$\begin{aligned} h[p_2(0),y,\theta] &\geq \lim_{k\to\infty} E[h[p_2(k),y,\theta]] \\ &= E[\lim_{k\to\infty} h[p_2(k),y,\theta]] \\ &= E[h[1-p_1^*,y,\theta]] \\ &= E[1-p_1^*] \\ &= \text{prob}\ [p_1^*=0]\ . \end{aligned} \qquad (3.11)$$

From the definition, it follows that for $0<z<1$, $y>0$

$$\lim_{\theta\to\infty} h[z,\ y,\ \theta] = 0\ . \qquad (3.12)$$

Now given $\varepsilon>0$, from (3.12) it follows that there exists a θ^*, $0<\theta^*<1$ such that for all $0<\theta<\theta^*$

$$\varepsilon \geq h[p_2(0),\ y,\ \theta]. \qquad (3.13)$$

Combining (3.11), (3.13)

$$\varepsilon \geq \text{Prob } [p_1^* = 0] = 1 - \text{Prob } [p_1^* = 1],$$

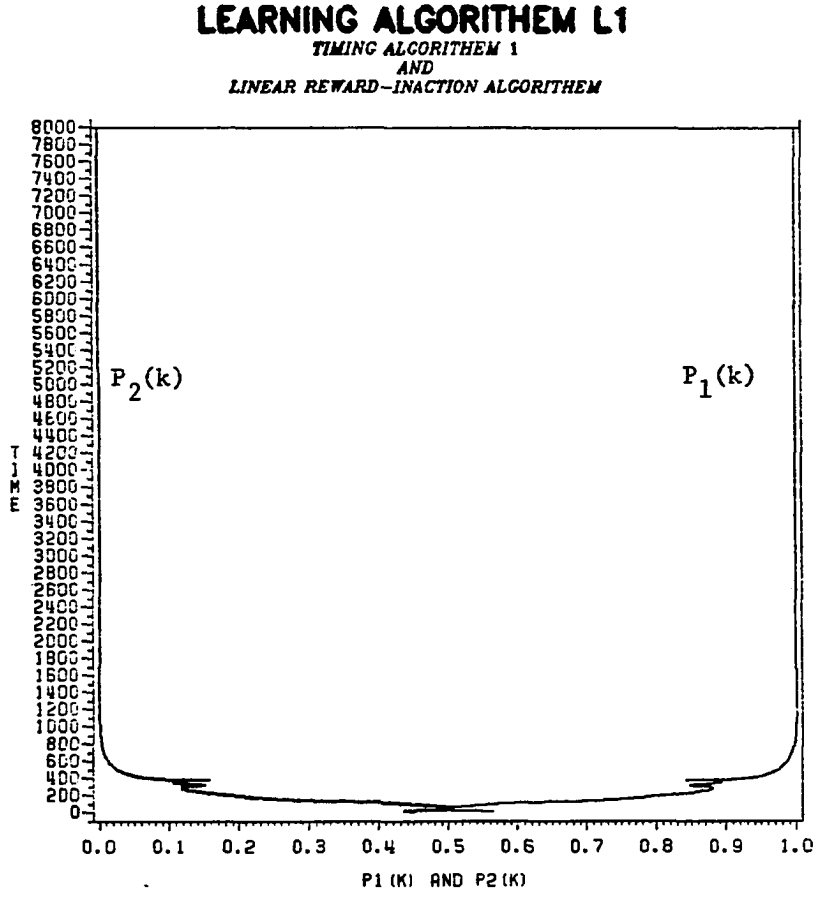
and the theorem is proved. □

Extensive computer simulation has been carried out to investigate the convergence properties of learning algorithm L2 under various choice of parameters λ_i, μ_i^{-1} $(i = 1,2)$. A selected set of these simulation results is presented to validate some of the theoretical assertions put forward in this chapter for learning algorithm L1. The plots for $p_1(k)$ and $p_2(k)$ versus k for the various choice of parameters λ_i, μ_i^{-1} $(i = 1,2)$ are given in Figure 3.3.

Remark 3.4: Our method of proof of the theorem 3.1 is very similar to the one given in [B2] in the context of the analysis of learning automata operating in a nonstationary environment as well as the one used in chapter 5 in [L3] for the analysis of the two-person zero sum games with incomplete information.

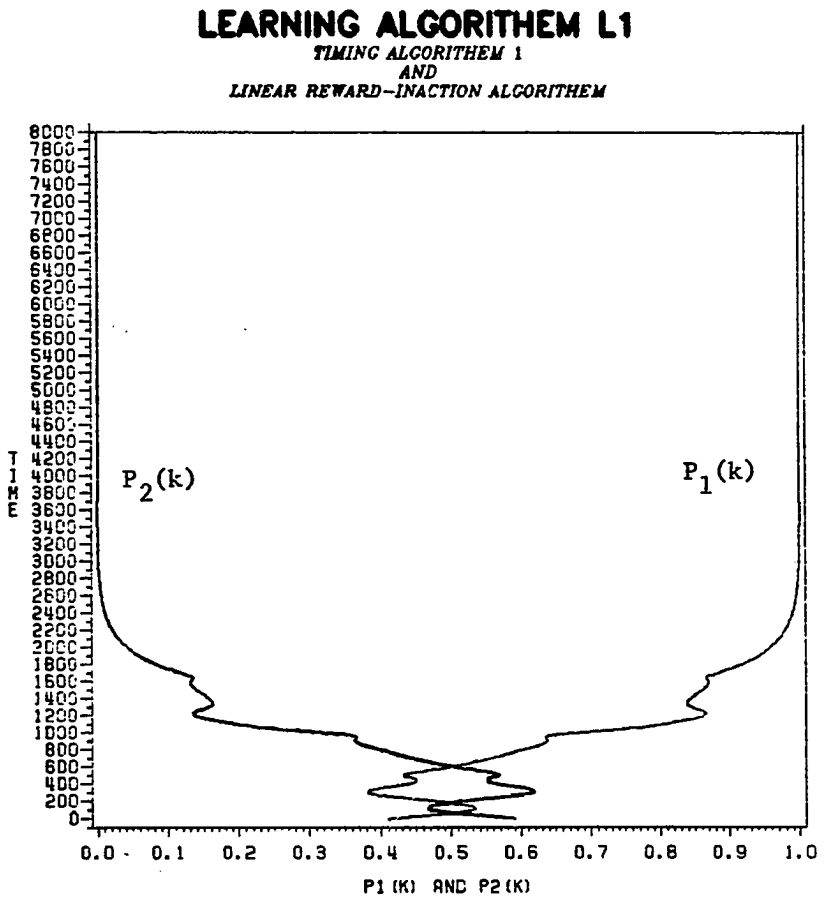
Remark 3.5: When θ is small, the algorithm in step 3 is called the small step learning algorithm. This latter class of algorithm has been extensively studied by Norman [N1] in the context of Mathematical Psychology. A major difference, however, is that in our analysis as in [B2], we don't need p(k) to be a Markov process.

Remark 3.6: The process $\{p(k), k \geq 1\}$ corresponding to the linear reward-inaction update algorithm and timing update algorithm 1 and 2 is a process with two absorbing barriers. Out of these absorbing barriers only one corresponds to the desired class. In fact, the system converges with a positive probability to each one of these absorbing states. From theorem (3.1) it is evident that there is a



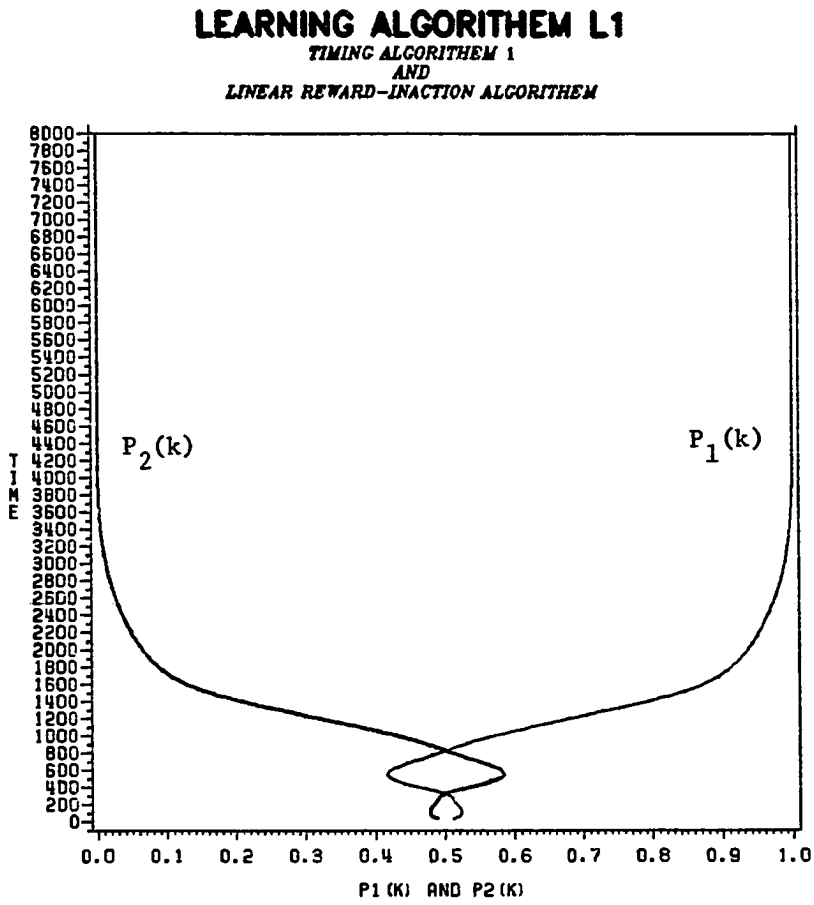
$\lambda_1 = 1/3$ $\lambda_2 = 1/6$ $\mu_1^{-1} = 1$ $\mu_2^{-1} = 5$

Figure 3.3a



$\lambda_1 = 1/3$ $\lambda_2 = 1/15$ $\mu_1^{-1} = 2$ $\mu_2^{-1} = 3$

Figure 3.3b



$\lambda_1 = 1/5$ $\lambda_2 = 1/20$ $\mu_1^{-1} = 1$ $\mu_2^{-1} = 8$

Figure 3.3c

trade-off between the speed of convergence and probability of converging to the desired class. Thus, a fastly converging learning algorithm may with a high probability converge to an undesired class. Larger value for θ will result in faster convergence for the system, but at the same time will decrease the probability of convergence to the right class.

Remark 3.7: The priority system reported here is an endogenous type priority system because the probabilities of choosing class 1 and 2 are changing over time; in other words, the probability that one of the class has higher priority over the other class is changing with time. Exogenous priority discipline is a special case of priority discipline conceived in this chapter, and can be obtained by setting $p_1(k) = 1$ and $p_2(k) = 0$ for all k or by setting $p_1(k) = 0$ and $p_2(k) = 1$ for all k depending on whether the higher priority is given to the first class or second class.

Now consider the update timing algorithm 2 for p(k). That is in step 2, if the class chosen for service (as a sample realization from p(k) in step 1) is non-empty, then update p(k) using step 3; otherwise p(k+1) = p(k). The probability $\alpha_2(k)$ with which p(k) is updated according to this modified update rule is

$$\alpha_2(k) = \rho_1(k)\ p_1(k)\ d_1(k) + \rho_2(k)\ p_2(k)\ d_2(k).$$

Since $0 < p_i(k) < 1 \quad i = 1,2$, it follows that $\alpha_1(k) < \alpha_2(k)$. That is, according to this modified rule p(k) is updated more frequently compared to update timing algorithm 1. Learning algorithm L1 with step 2 replaced by update learning algorithm 2 is called learning algorithm L2. Admittedly, update timing algorithm 2 results in a faster convergence. However, it is not without further problems. We need an extra

condition such as

(C1) $\rho_1(k) > \rho_2(k)$ for all k

for the lemma 1 and 2 and hence theorem 3.1 to be true.

Theorem 3.2: Under the assumption (A1) and (A2), and condition (C1), if p(k) evolves according to the learning algorithm L2, then for every $\varepsilon > 0$, there exists a θ^* such that $0 < \theta^* < 1$ and for all $0 < \theta < \theta^*$

$$\text{Prob}\left[\lim_{k\to\infty} p_1(k) = 1\right] \geq 1 - \varepsilon .$$

Proof: Proof for theorem 3.2 follows the same line as theorem 1. For the sake of completeness, we prove lemma 3.1 and lemma 3.2 when the system uses learning algorithm 2.

Lemma 3.1': Under condition C1, $\Delta p_1(k) \geq 0$ for all $k > 0$ with equality holding only when $p_1(k) = 0$ or 1.

Proof: By routine computation we obtain

$$\Delta p_1(k) = \theta p_1(k)\, p_2(k)\, (d_1(k)\, \rho_1(k) - d_2(k)\, \rho_2(k)) .$$

The lemma follows from the properties of $d_i(k)$, $i = 1,2$ given in (3.1). □

Lemma 3.2': Under condition (C1), there exists a positive, real number y such that

$$E[\psi[p(k+1),y,\theta] \mid F_k] \leq \psi[p(k),y,\theta] . \tag{3.13}$$

Proof: It can be shown that

$$\begin{aligned} E[\psi[p(k+1), x, \theta] \mid F_k] = \\ & \rho_1(k)\, p_1(k)\, d_1(k)\, \psi[p_1(k) + \theta(1-p_1(k)), x, \theta] \\ & + \rho_1(k)\, p_1(k)\, c_1(k)\, \psi[p_1(k), x, \theta] \\ & + \rho_2(k)\, p_2(k)\, d_2(k)\, \psi[p_1(k) - \theta\, p_1(k), x, \theta] \\ & + \rho_2(k)\, p_2(k)\, c_2(k)\, \psi[p_1(k), x, \theta] . \end{aligned}$$

After simplification,

$$E[\psi[p(k+1),x,\theta] \mid F_k] - \psi[p(k),x,\theta] =$$

$$-y\,\psi[p(k),x,\theta][\rho_1(k)\,p_1(k)\,p_2(k)\,d_1(k)\,V[-x(1-p_1(k)] - \rho_2(k)\,p_1(k)\,p_2(k)\,d_2(k)\,V[x\,p_1(k)]]$$

$$+ (\rho_1(k)\,p_2(k) + \rho_2(k)\,p_2(k) - 1)\,\psi[p(k),x,\theta]\,.$$

Since $\rho_i(k) < 1$, $i = 1,2$ and $p_1(k) + p_2(k) = 1$ for all k, we have

$$\rho_1(k)\,p_1(k) + \rho_2(k)\,p_2(k) - 1 < 0 \quad \text{for all } k\,.$$

So for lemma 3.2' follows we must have

$$\rho_1(k)\,p_1(k)\,p_2(k)\,d_1(k)\,V[-x(1-p_1(k))] - \rho_2(k)\,p_2(k)\,p_1(k)\,d_2(k)\,V[x\,p_1(k)] \geq 0\,, \tag{3.14}$$

or

$$\frac{V[-x(1-p_1(k)]}{V[x\,p_1(k)]} \geq \frac{\rho_2(k)\,d_2(k)}{\rho_1(k)\,d_1(k)}\,. \tag{3.15}$$

From the convexity properties of H[u] we will have

$$\frac{V[-x(1-p_1(k))]}{V[-x\,p_1(k)]} \geq \frac{1}{V[x]}\,. \tag{3.16}$$

From condition (C1) and properties of $d_i(k)$, $i = 1,2$, it can easily be seen that

$$\eta = \sup \frac{\rho_2(k)\,d_2(k)}{\rho_1(k)\,d_1(k)} < 1\,. \tag{3.17}$$

From (3.17) and (3.16), inequality (3.15) is true if

$$\frac{1}{V[x]} \geq \eta\,. \tag{3.18}$$

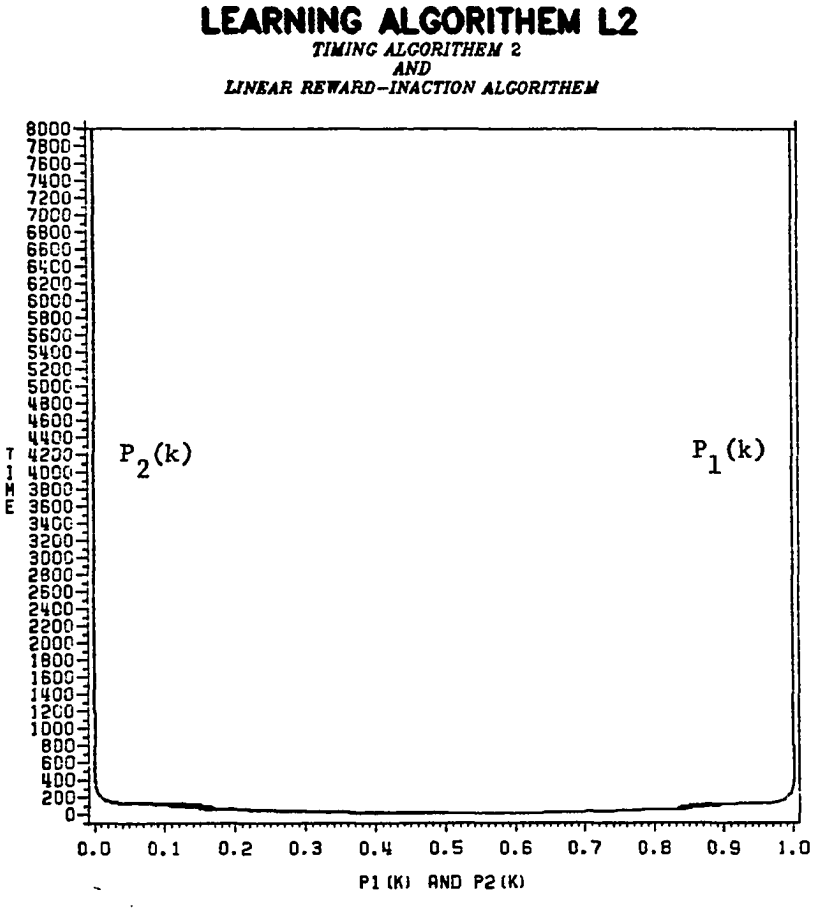
From the properties of the function V[u], it follows that there exists a real number $y > 0$ such that for all $x \in (0,y)$, inequality (3.18) will be true. By setting $x = y$ the lemma follows.□

An extensive number of simulations are performed using learning

algorithm L2 and various values of μ_i^{-1}, $\lambda_i (i = 1/2)$. A fact which is brought out from these simulations is that if μ_i/λ_i is greater than $\lambda_j/\mu_j (i \neq j)$ and for relatively small value of θ, the system will always converge to class i where $\mu_i^{-1} = \min_{j \neq i} (\mu_i^{-1})$. The plot for $p_1(k)$ and $P_2(k)$ versus k for various choices of parameters λ_i, μ_i^{-1} $(i = 1,2)$ are given in figure 3.4.

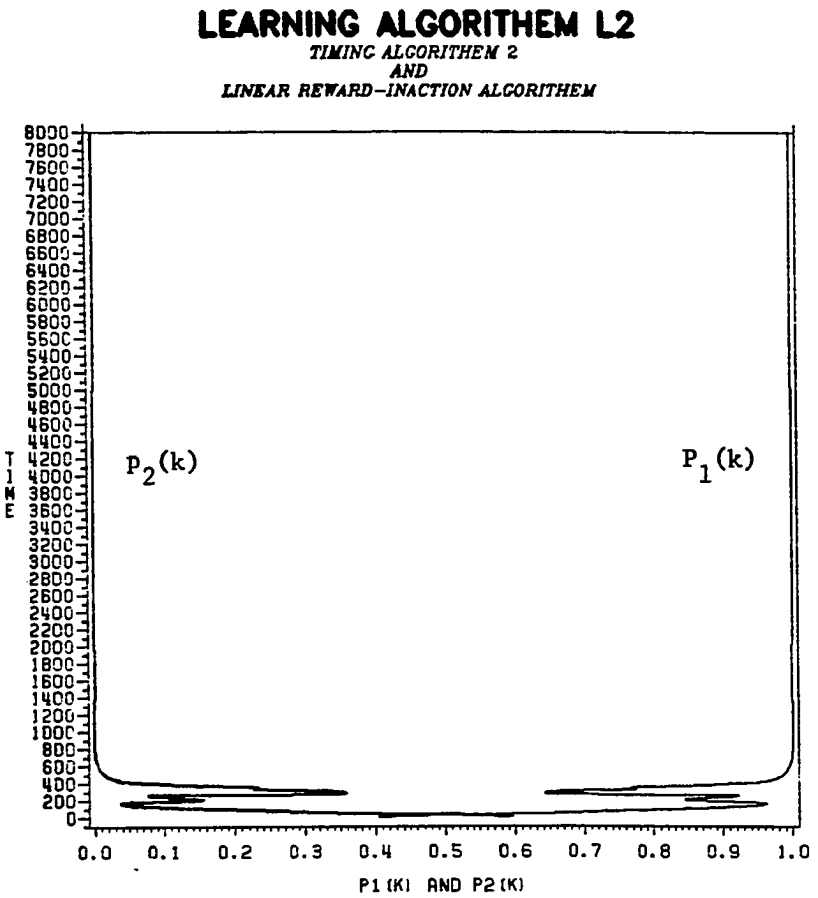
Remark 3.8: When the system uses learning algorithm L2, $\rho_1(k)p_1(k)d_1(k)$ and $\rho_2(k)\ p_2(k)\ d_2(k)$ are probabilities that p_1 is increased and decreased respectively. So a small value of $\rho_1(k)$ may cause the system to converge to the wrong class or result in prolonging the convergance time. For example, if due to long interarrival time of jobs entering into the first class, $\rho_1(k)$ is very small, the system may converge to the class with longer service time. If learning algorithm L1 is used, $\rho_1(k)\ \rho_2(k)\ p_1(k)\ d_1(k)$ is the probability that p_1 is increased and $\rho_1(k)\ \rho_2(k)\ p_2(k)\ d_2(k)$ is the probability that p_1 is decreased at time k. In this case low value of $\rho_1(k)$ may only result in prolonging the convergence time. Figure 3.5 illustrates the effect of interarrival time of jobs in the class with smaller service time on the convergence of the system.

Remark 3.9: The difference $(\mu_2^{-1} - \mu_1^{-1})$ affects the behavior of the system. If changes in μ_1^{-1} or μ_2^{-1} result in an increase in the difference $(d_1(k) - d_2(k))$, the system converges faster. It should be noted that a larger value for $(\mu_2^{-1} - \mu_1^{-1})$ does not always imply a larger value for $(d_2(k) - d_1(k))$ because the difference $(d_1(k) - d_2(k))$ depends not only on μ_1^{-1} and μ_2^{-1} but on the behavior of T(k) over time. Changing μ_1^{-1} and μ_2^{-1} may also change $\rho_1(k)$ and $\rho_2(k)$ which in turn



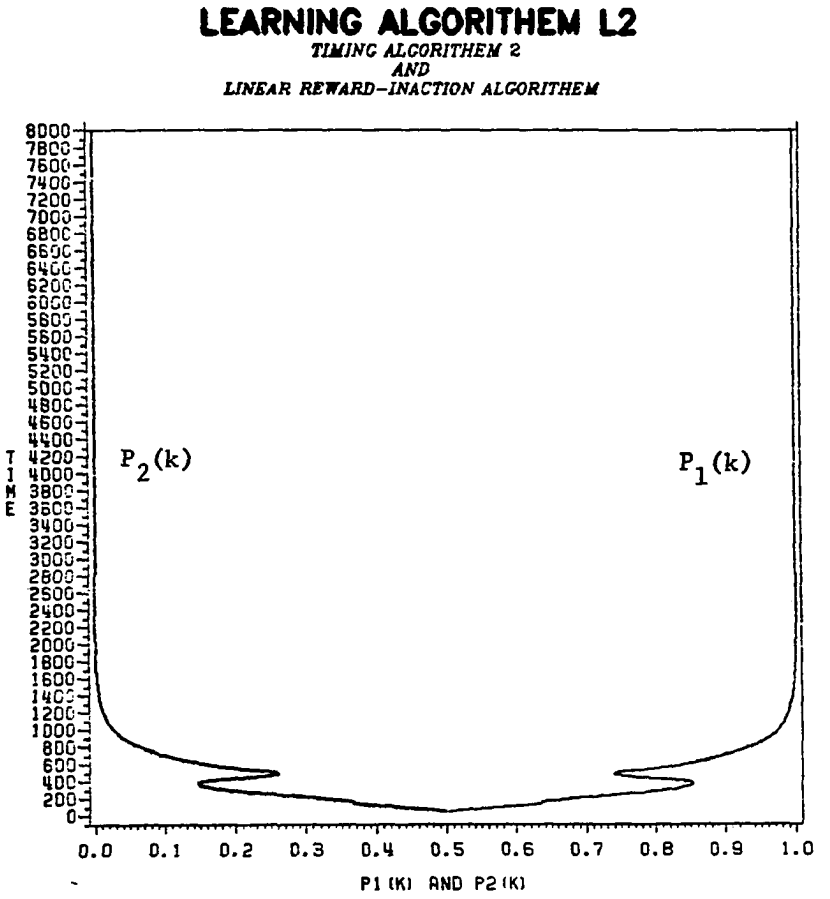
$\lambda_1 = 1/3$ $\lambda_2 = 1/6$ $\mu_1^{-1} = 1$ $\mu_2^{-1} = 5$

Figure 3.4a



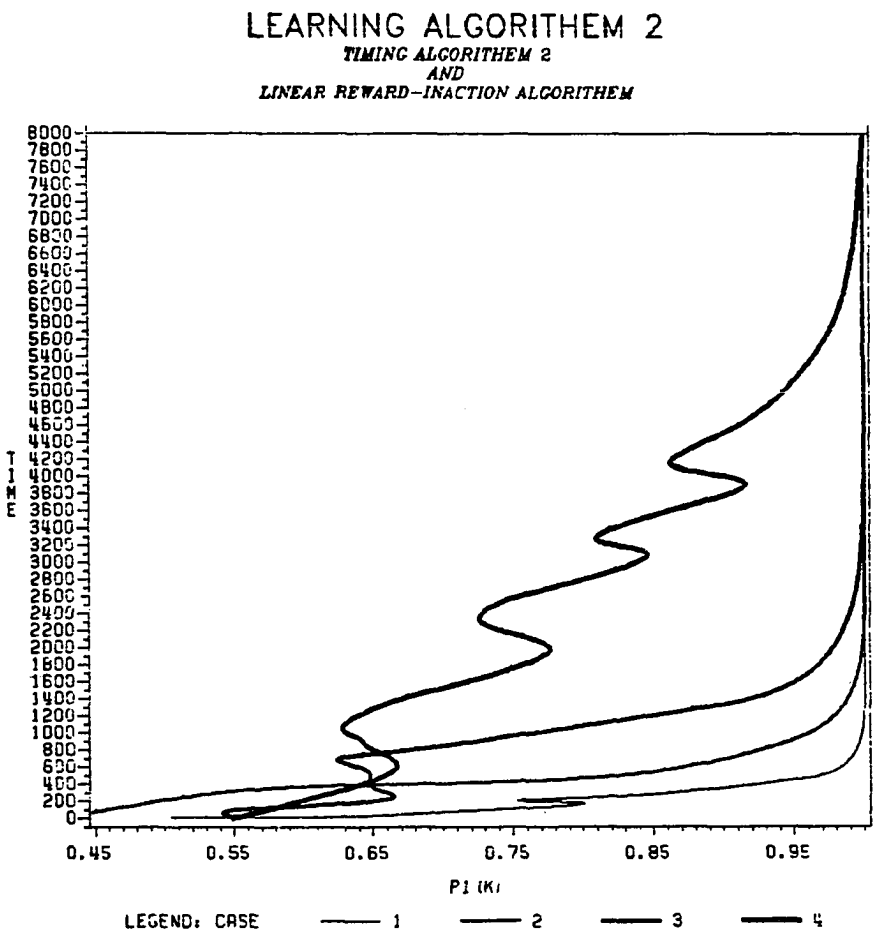
$\lambda_1 = 1/3$ $\lambda_2 = 1/15$ $\mu_1^{-1} = 2$ $\mu_2^{-1} = 3$

Figure 3.4b



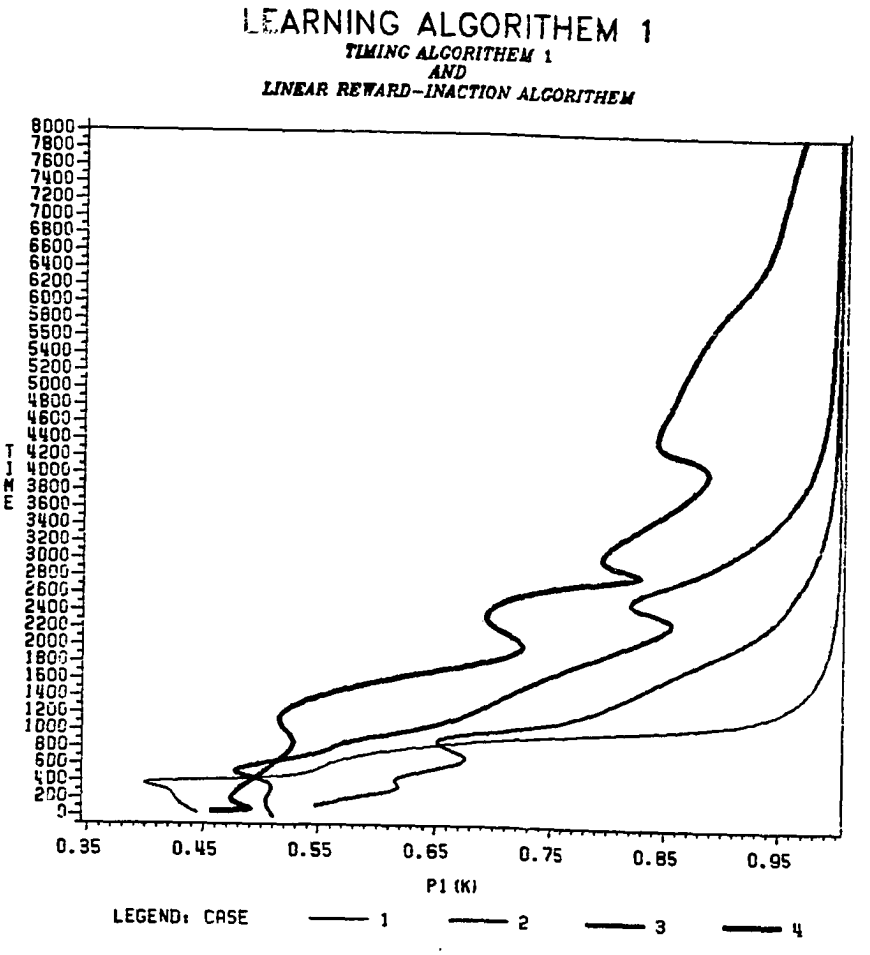
$\lambda_1 = 1/20$ $\lambda_2 = 1/40$ $\mu_1^{-1} = 1$ $\mu_2^{-1} = 5$

Figure 3.4c



Case #1	$\lambda_1 = 1/10$	$\lambda_2 = 1/90$	$\mu_1^{-1} = 1$	$\mu_2^{-1} = 4$
Case #2	$\lambda_1 = 1/20$	$\lambda_2 = 1/90$	$\mu_1^{-1} = 1$	$\mu_2^{-1} = 4$
Case #3	$\lambda_1 = 1/40$	$\lambda_2 = 1/90$	$\mu_1^{-1} = 1$	$\mu_2^{-1} = 4$
Case #4	$\lambda_1 = 1/70$	$\lambda_2 = 1/90$	$\mu_1^{-1} = 1$	$\mu_2^{-1} = 4$

Figure 3.5a



Case #1 $\lambda_1 = 1/1$ $\lambda_2 = 1/9$ $\mu_1^{-1} = 0.1$ $\mu_2^{-1} = 0.4$

Case #2 $\lambda_1 = 1/3$ $\lambda_2 = 1/9$ $\mu_1^{-1} = 0.1$ $\mu_2^{-1} = 0.4$

Case #3 $\lambda_1 = 1/6$ $\lambda_2 = 1/9$ $\mu_1^{-1} = 0.1$ $\mu_2^{-1} = 0.4$

Case #4 $\lambda_1 = 1/7$ $\lambda_2 = 1/9$ $\mu_1^{-1} = 0.1$ $\mu_2^{-1} = 0.4$

Figure 3.5b

will affect the convergence behavior of the system.

The following remark gives some insight into the behavior of ρ_i, i = 1,2, during the operation of the system.

Remark 3.10: It was earlier said in this chapter (Remark 3.7) that the exogenous priority scheme corresponds to $\{p_1(k) = 1$ and $p_2(k) = 0$ for all k$\}$ or $\{p_1(k) = 0$ and $p_2(k) = 1$ for all k$\}$, depending on whether the highest priority is assigned to the first or second class. Let ρ_i^* be the probability that class i is nonempty when $\{p_1(k) = 1$ and $p_2(k) = 0$ for all k$\}$. Now consider the case where $p_1(k) = a$ and $p_2(k) = 1-a$ for all k, where a is a positive constant less than one. Let ρ_i^a be the probability that class i is nonempty when $p_1(k)$ is fixed to constant a throughout the operation of the system. It is clear that

$$\rho_1^a > \rho_1^*$$

and

$$\rho_2^a < \rho_2^* .$$

Because by decreasing (increasing) the probability of choosing class 1 (class 2) from 1 (0) to a(1-a), the probability that class 1 (class 2) is nonempty will increase (decrease). In view of the fact that $\{p_1(k)\}$ is a sub-martingale and $\{p_2(k)\}$ is a super-martingale, we have

$$E[\rho_1(k+1) \mid p_1(k)] < \rho_1(k)$$

and

$$E[\rho_2(k+1) \mid p_2(k)] > \rho_1(k) .$$

That is, $\{\rho_1(k+1)\}$ and $\{\rho_2(k+1)\}$ are super and sub-martingale, respectively.

And since

$$\lim_{k\to\infty} p_1(k) = 1$$

and

$$\lim_{k\to\infty} p_2(k) = 0 ,$$

we have

$$\lim_{k\to\infty} \rho_1(k) = \rho_1^*$$

$$\lim_{k\to\infty} \rho_2(k) = \rho_2^* ,$$

Remark 3.11: If w_i^* and w_i^a are the average waiting time for the exogenous priority scheme and the case where $p_1(k) = a$ for all k where $0 < a < 1$, respectively, then we have

$$W_1^* = \frac{(\lambda_1/\mu_1^2) + (\lambda_2/\mu_2^2)}{(1-\lambda_1/\mu_1)} ,$$

and

$$W_2^* = \frac{(\lambda_1/\mu_1^2) + (\lambda_2/\mu_2^2)}{(1-\lambda_1/\mu_1)(1-\lambda_1/\mu_1-\lambda_2/\mu_2)} ,$$

and also

$$W_1^a > W_1^* \quad \text{and} \quad W_2^a < W_2^* .$$

In view of the fact that $\{p_1(k)\}$ is a submartingale and $\{p_2(k)\}$ is a supermartingale, we have

$$E[W_1(k+1) \mid p(k)] < W_1(k) ,$$

$$E[W_2(k+1) \mid p(k)] > W_2(k) ,$$

where $W_i(k)$, $i = 1,2$, is the average waiting time for class i at time k. Also

$$\lim_{k\to\infty} W_i(k) = W_i^* .$$

Since

$$\lim_{k \to \infty} p_1(k) = 1$$

and

$$\lim_{k \to \infty} p_2(k) = 0 .$$

Simulation studies were conducted and it was discovered that the speed of convergence of $W_i(k)$, $i = 1,2$, to W_i^*, $i = 1,2$, is very dependent on the speed of convergence of the learning algorithm.

3.4 Priority Assignment from Another Point of View

In the previous section we discussed the case where the highest priority is assigned to the class with the shortest average service time. This priority scheme can reduce the overall mean waiting time of the system. This ordering not only minimizes the overall waiting time of the system but minimizes the total waiting cost if the unit waiting cost (c_i, $i = 1,2$) for different classes are equal . In this section the case where the unit waiting cost for different classes are different will be discussed. It was mentioned earlier in the chapter that ordering of priority assigned to the classes of arriving jobs with the highest priority for the class with the highest value of μc will minimize the total waiting cost.

In order to assign priority according to the μc rule, we need to ascertain both μ_i's and c_i's. In the following setup we assume that all c_i's are known but μ_i's are all unknown. We assume that jobs arrive into two queues independently at constant Poisson rate λ_i (> 0) ($i = 1,2$). The service time requirement ($\gamma^{(i)}$, $i=1,2$) for class i jobs are independent and identically distributed random variables with exponential density function $f_i(t)$ ($i=1,2$) with mean μ_i^{-1} ($i=1,2$). We also assume that no preemption is allowed and the queue length of

each class is allowed to be infinite. Without loss of generality, it is assumed

(A3) $\quad c_1 \mu_1 > c_2 \mu_2 .$

Dynamic threshold at stage k for this case is defined as

$$T(k) = 1/2 \, [T^{(1)}(k) + T^{(2)}(k)] ,$$

where

$$T^{(i)}(k) = \frac{1}{k_i} \sum_{j=1}^{k_i} \frac{\gamma_j^{(i)}}{c_i} \quad \text{and } T_i(0) = 0 \quad , \; i = 1,2 ,$$

where $k = k_1 + k_2$ and c_i is the unit waiting cost for class i. In view of the fact that service time $\gamma^{(i)}$ (i=1,2) are exponentially distributed with mean μ_i^{-1}, it can easily be seen that

$$\text{Prob} \left[\frac{\gamma_j^{(i)}}{c_i} < x \right] = 1 - e^{-c_i \mu_i x} .$$

So random variable $\frac{\gamma^{(i)}}{c_i}$ (i=1,2) is exponentially distributed with density functions

$$g_i(x) = c_i \mu_i e^{-c_i \mu_i x} , \quad (i=1,2),$$

clearly

$$d_i(k) = \text{Prob} \left[\frac{\gamma^{(i)}(k+1)}{c_i} - T(k) \leq 0 \right]$$

$$= \int_0^{T(k)} c_i \mu_i e^{-c_i \mu_i x} = 1 - e^{-\mu_i c_i T(k)} , \quad (i=1,2)$$

and

$$c_i(k) = \text{Prob} \left[\frac{\gamma^{(i)}(k+1)}{c_i} > T(k) \right] = e^{-c_i \mu_i T(k)} , \quad (i=1,2) .$$

In view of assumption (A3) it can be shown that $d_1(k) \geq d_2(k)$ with probability one for all k. Let us call algorithm L1 and L2 with the dynamic threshold replaced by the modified dynamic threshold of this

section, algorithm L3 and L4, respectively. Corresponding to algorithm L3 and L4 we can state the analogues of theorem 3.1 and 3.2, respectively. Stated in words, algorithm L1 and L2 when using the modified dynamic threshold of this section, will approach the μc rule, that is, they will assign the highest priority to the class with the highest value of μc.

3.5 Extension to More Than Two Classes

3.5.1 Extension to three classes:

We first present an extension of the learning algorithm described in section 2.2 to a priority queuing system with 3 different classes of jobs. All the assumptions which we made in section 2.2 are also made for this case. Without loss of generality, assume

$$\mu_1^{-1} < \mu_2^{-1} < \mu_3^{-1}. \tag{A4}$$

Define the following:

1. Probability vector $p(k) = (p_1(k), p_2(k), p_3(k))$, where $\sum_{i=1}^{3} p_i = 1$. $p_j(k)$ is the probability that the system decides to pick up the $(k+1)^{th}$ job for service from class j when all the classes are non-empty. Initially $p_1(0) = p_2(0) = p_3(0) = \frac{1}{3}$ with probability one.
2. Probability vector $p^i(k) = (p_r^{(i)}(k) + p_s^{(i)}(k))$ $r \neq s \neq i$, where $p_r^{(i)}(k) + p_s^{(i)}(k) = 1$ for all i. $p_j^{(i)}(k)$ is the probability of choosing class j for service when the i^{th} class is empty. Initially $p_r^{(i)}(0) = \frac{1}{2}$ for all i, r, $i \neq r$ with probability one.
3. $T(k) = \frac{1}{3} \sum_{i=1}^{3} T^{(i)}(k_i)$

and

$$T_{(i)}(k) = \frac{1}{2}\left[T^{(r)}_{(i)}(k_r) + T^{(s)}_{(i)}(k_s)\right] \qquad s \neq r \neq i ,$$

where

$$T^{(i)}(k_i) = \frac{1}{k_i}\sum_{\ell=1}^{k_i} \gamma^{(i)}(\ell) \qquad i = 1,2,3 ,$$

and

$$T^{(q)}_{(i)}(k_r) = \frac{1}{k_q}\sum_{\ell=1}^{k_q} \gamma^{(q)}(\ell) \qquad q \neq i ,$$

where k_i is the number of jobs which are served from class j and $\gamma^{(i)}(\ell)$ is the actual service time of the ℓ^{th} job from class i.

Learning Algorithm GL1

Let i be the class from which $(k+1)^{th}$ job is chosen for service as a sample realization from p(k) for $k \geq 0$.

1. If the queue of this chosen class is non-empty then

{ choose a job from in front of the queue for service, complete the service and note $\gamma^{(i)}(k+1)$.

The update timing and update algorithm for p(k):

If the queues corresponding to the other two classes are also nonempty then update p(k) using the following update algorithm.

$$\left.\begin{aligned} p_i(k+1) &= p_i(k) + \theta(1-p_i(k)) \\ p_j(k+1 &= p_j(k) - \theta p_j(k) \qquad j \neq 1 \end{aligned}\right\} \quad \text{if } \gamma^{(i)}(k+1) \leq T(k)$$

and

$$p(k+1) = p(k) \qquad \text{if } \gamma^{(i)}(k+1) > T(k)$$

else

$$p(k+1) = p(k)$$

Update $T(k)$ and $T_{(j)}(k)$ $j \neq i$ using $\gamma^{(i)}(k+1)$,
goto 1 }

else

{ Let n be the class from which $(k+1)^{th}$ job is chosen for service as a sample realization from $p^{(i)}(k)$.

If the queue of this chosen class is non-empty, then

{select the job from in front of the chosen queue for service, complete the service and note $\gamma^{(n)}(k+1)$.

The update timing and update algorithm for $p^{(i)}(k)$:

At the instant when class i is chosen for service if the queue corresponding to both the class n as well as that of class m ($m \neq n \neq i$) are both non-empty then update $p^{(i)}(k)$ using the following update algorithm:

$$\left.\begin{aligned} p_n^{(i)}(k+1) &= p_n^{(i)}(k) + \theta(1-p_n^{(i)}(k)) \\ p_m^{(i)}(k+1) &= p_m^{(i)}(k) - \theta p_m^{(i)}(k) \\ & \qquad m \neq n \neq i \end{aligned}\right\} \text{ if } \gamma^{(n)}(k+1) \leq T(k)$$

and

$$p^{(i)}(k+1) = p^{(i)}(k) \quad \text{if} \quad \gamma^{(n)}(k+1) > T(k)$$

else

$$p^{(i)}(k+1) = p^{(i)}(k)$$

update $T_{(j)}(k)$, $j \neq n$ and $T(k)$ using $\gamma^{(n)}(k+1)$
go to 1 }

else

{select the job from in front of class m ($m \neq n \neq i$) complete the service and note $\gamma^{n}(k+1)$.

update $T(k)$ and $T_j(k)$ $j \neq m$ using $\gamma^{(m)}(k+1)$.

goto 1 }

}

Theorem 3.3: Under the assumptions (A2) and (A4) if the system evolves according to the learning algorithm GL1, then for every ε, $\varepsilon_r > 0$, $r = 1,2,3$, there exists θ^*, θ_r^*, $r = 1,2,3$, such that $0 < \theta^*$, $\theta_r^* < 1$ $r = 1,2,3$ and for all $0 < \theta < \theta^*$, $0 < \theta_r < \theta_r^*$, $r = 1,2,3$.

a. Prob $[\lim_{k \to \infty} p_i(k)] \geq 1-\varepsilon$ if $\mu_i^{-1} = \min_{j \neq i} (\mu_j^{-1})$,

b. Prob $[\lim_{k \to \infty} p_i^{(r)}(k)] \geq 1-\varepsilon$, $r = 1,2,3$,

if $\mu_i^{-1} = \min_{\substack{j \neq i \\ j \neq r}} (\mu_j^{-1})$.

Proof: Proof is immediate from theorem 2.1.

While the result of section 3.2 of this chapter can be generalized to priority assignment system with any number of classes, the learning algorithm needed becomes more complex and messy as the number of classes increases. Due to this problem, in the following subsection we use a class of multi-input automata studied by Felrov [F5] to formulate the priority assignment system of pervious sections when there are m classes of jobs ($m \geq 2$).

3.5.2 Extension to m classes:

The aim of priority assignment system is to distinguish that class of jobs that has minimum average service time, or in other words to distinguish that random quantity whose mean value is smallest in magnitude. The simplest procedure to this job is that of sampling m distributions (population) in turn, and then calculating the sample

mean value for each of the distributions, as estimates of the true mean values, and using these estimates at any time to choose that random variable that has the smallest mean. After k rounds of sampling (after a total of mk sample measurements) the sample mean estimates are given by

$$T^{(i)}(k) = \frac{1}{k} \sum_{j=1}^{k} Y^{(i)}(j),$$

where $T^{(i)}(k)$ is the sample mean of size k from the i^{th} distribution (class), and $Y^{(i)}(j)$ denotes the j^{th} sample (job) from the i^{th} random variable (class). These estimates $T^{(i)}(k)$, $i = 1,2,\ldots,m$, are random variables with probability distribution function $F_i(T^{(i)}(k))$ commonly referred to as sampling distributions.

It is known that if original distributions are normally distributed, then the distribution of $T^{(i)}(k)$ for all k will be normally distributed with

$$E[T^{(i)}(k)] = \mu_i^{-1}$$

and

$$\text{Variance } [T^{(i)}(k)] = \frac{\sigma_i^2}{k},$$

where σ_i^2 and μ_i^{-1} are variance and mean of the i^{th} distribution respectively. If the original distributions are not Gaussian, then from the implications of central limit theorem, the distribution of

$$Z^{(i)}(k) = \frac{T^{(i)}(k) - E[T^{(i)}(k)]}{\sqrt{\text{Var } [T^{(i)}(k)]}} = \frac{T^{(i)}(k) - \mu_i^{-1}}{\sigma/\sqrt{k}}$$

approaches the standard normal distribution as k approaches infinity.

We now describe a priority assignment system that uses sample mean to select among m classes. Define

$$\bar{N}(k) = (N^{(1)}(k), N^{(2)}(k), \ldots, N^{(m)}(k)) ,$$

$$\bar{S}(k) = (T^{(1)}(k), T^{(2)}(k), \ldots, T^{(m)}(k)) ,$$

$$\bar{N}(0) = (0, 0, \ldots, 0) \quad , \quad \bar{S}(0) = (0, 0, \ldots, 0).$$

$N^{(i)}(k)$ indicates how many times class i was selected prior to time k and $T^{(i)}(k)$ is the sample average service time for the i^{th} class, and

$$T^{(i)}(k) = \frac{1}{N^{(i)}(k)} \sum_{j=1}^{K} \gamma_j^{(i)} ,$$

where $\gamma_j^{(i)}$ is the service time for the j^{th} job chosen from the i^{th} class.

This priority assignment system can be represented by Figure 3.6. Automata A keeps track of the number of times that class i is chosen (vector $\bar{N}$) and the magnitude of the average service time for all classes computed by AV (vector $\bar{S}$). State of the automata at time k is represented by

$$\psi(k) = (\bar{N}(k), \bar{S}(k)) .$$

The components of state ψ are transformed in the following manner during the operation of the system: after the service to a job from class c_i, selected at time k, is completed, the number $N^{(i)}(k)$ is incremented by 1, and the average service time $T^{(i)}(k+1)$ assumes the value

$$T^{(i)}(k+1) = \frac{N^{(i)}(k)\, T^{(i)}(k) + \gamma^{(i)}}{N^{(i)}(k) + 1} .$$

Remark 3.12: Process (N,S) is a homogenous Markov chain. Each state of this chain is irreversible, that is, if $p_{ij} > 0$ for $i > j$, then $p_{ji} = 0$ where p_{ij} is the probability of going from state i to state j.

Automata A operates in two modes R_1 and R_2, called learning and

selection mode, respectively. In mode R_1, each class is chosen k times (cycle). In mode R_2, the automata chooses at each cycle that class which has minimum sample mean $T^*(k)$ where

$$T^*(k) = \min_i \{T^{(i)}(k)\} .$$

The operation of the automata consists in a successive alternation of modes R_1 and R_2. First the automata operates $CL_1(R_1)$ cycles in mode R_1, then it operates for $CL_1(R_2)$ cycles in mode R_2, then again $CL_2(R_1)$ cycles in mode R_1, then for $CL_2(R_2)$ cycles in mode R_2, etc. The sequence $CL_i(R_1)$, $CL_i(R_2)$ are selected in such a way that the ratio $CL_i(R_2)/CL_i(R_1)$ increases without bounds when $i \to \infty$; that is for any positive value h there exists an i_0 such that for all $i \geq i_0$,

$$\frac{CL_i(R_2)}{CL_i(R_1)} > h .$$

Depending on the nature of the sequences $CL_i(R_2)$ and $CL_i(R_1)$, it is possible to have various strategy for mode alternation of the automata. For example

$$CL_i(R_2) = 2^i ,$$

$$CL_i(R_1) = mk ,$$

is one possible strategy for the automata. k is an integer constant denoting the number of times that each class is selected when the automata is operating in learning mode.

Remark 3.13: Automata described in this way is a deterministic Moor automata because the output (action) of the automata is uniquely determined by the state of the automata.

It is assumed that the service time distribution for either

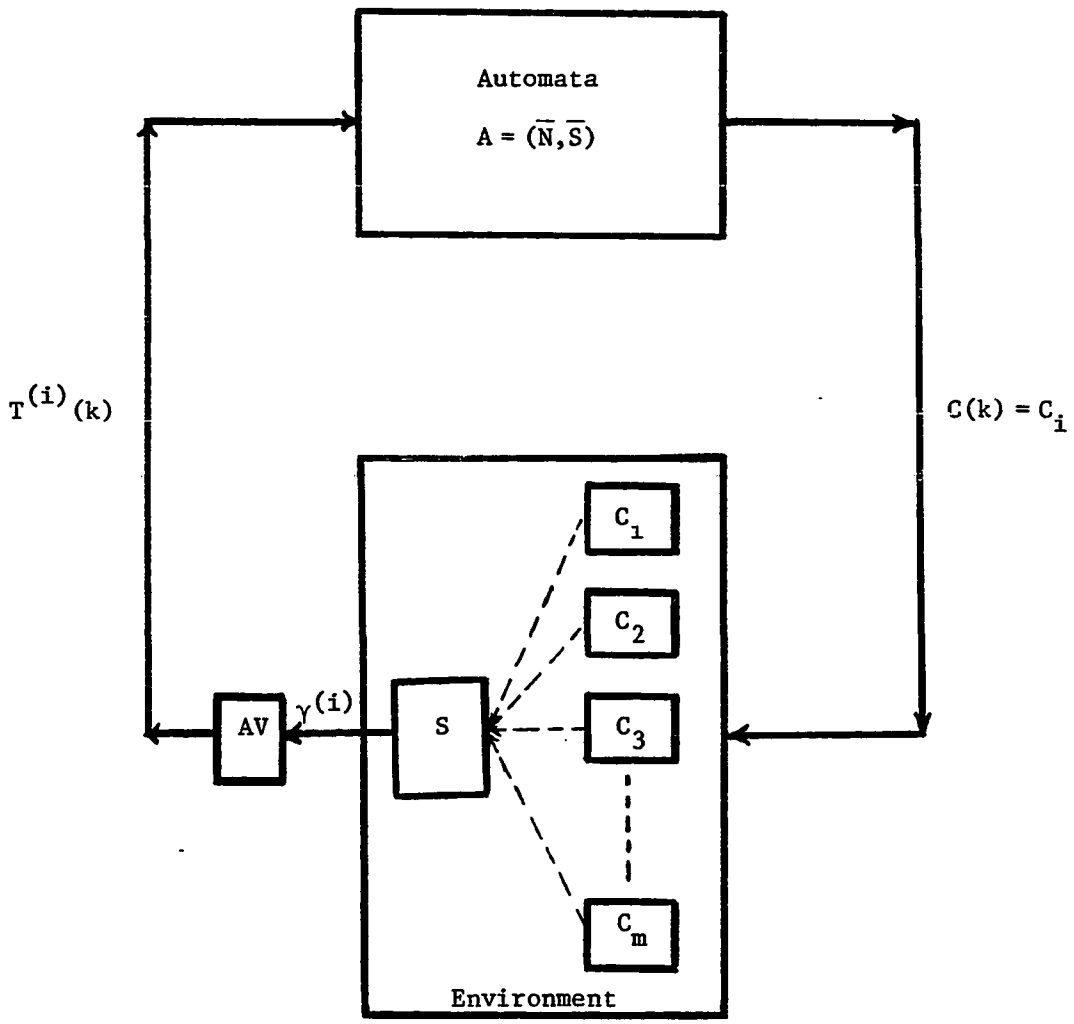


Figure 3.6

classes is stationary. This implies that the automata is operating in a stationary random environment. Without loss of generality we also assume that $\mu_1^{-1} < \mu_2^{-1} < \mu_3^{-1} < \ldots, < \mu_m^{-1}$.

Theorem 3.4: For a sufficiently long operation the automata will select a class that minimizes the average waiting time with a frequency as close to unity as desired.

Proof: Since $CL_i(R_1) \geq m\,k$, it follows that $N^{(i)}(k)$ $(i = 1,2,\ldots,m)$ increases without bound as $k \to \infty$, therefore $T^{(i)}(k)$ $(i = 1,2,\ldots,m)$ will be a consistant and unbiased estimate of μ_i^{-1} $(i=1,2)$, so that for any positive ε there exists a k^* such that

$$\text{Prob}\,\{T^{(1)}(k) = \underset{i}{\text{Min}}\,\{T^{(i)}(k)\}\} > 1 - \varepsilon\,,$$

for all $k \geq k^*$, or

$$\lim_{k\to\infty} \text{Prob}\,\{T^{(1)}(k) = \underset{i}{\text{Min}}\,\{T^{(i)}(k)\}\} = 1\,.$$

That is, over an infinite time the service time sample means will tend to arrange themselves with probability one, in the same order as the quantities μ_i^{-1} $(i=1,2,\ldots,m)$. In other words, as $k \to \infty$ the automata operating in mode R_2 will choose class C_i that minimizes the average waiting time with a probability which is as close as desired to 1. Assuming that the automata has changed its modes ℓ times from the beginning of the operation to time k, we have

$$f^i(k) = \frac{\sum_{j=1}^{\ell} CL_j(R_i)}{\sum_{j=1}^{\ell} CL_j(R_1) + \sum_{j=1}^{\ell} CL_j(R_2)}\,,$$

where $f^{(i)}(k)$ is the frequencies of the operation of the automata in

mode $R_i (i = 1,2)$. Knowing the fact that the ratio $CL_i(R_2)/CL_i(R_1)$ increases without bound when $i \to \infty$ we have

$$\lim_{k \to \infty} f^1(k) = 0 ,$$

and

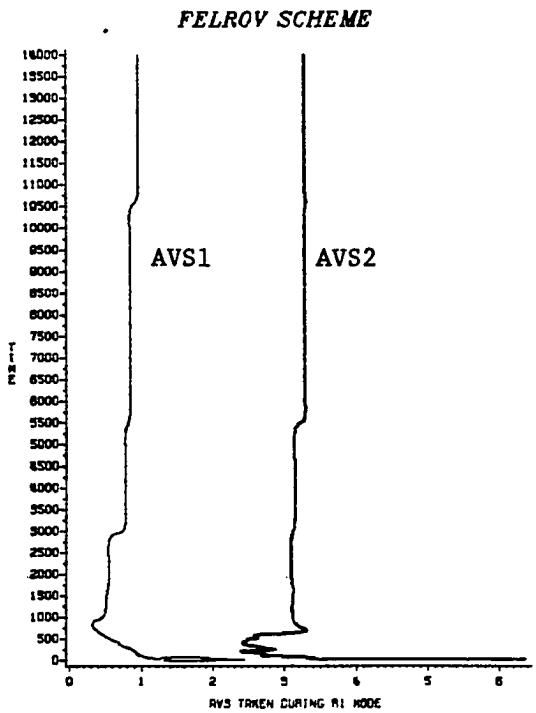
$$\lim_{k \to \infty} f^2(k) = 1.$$

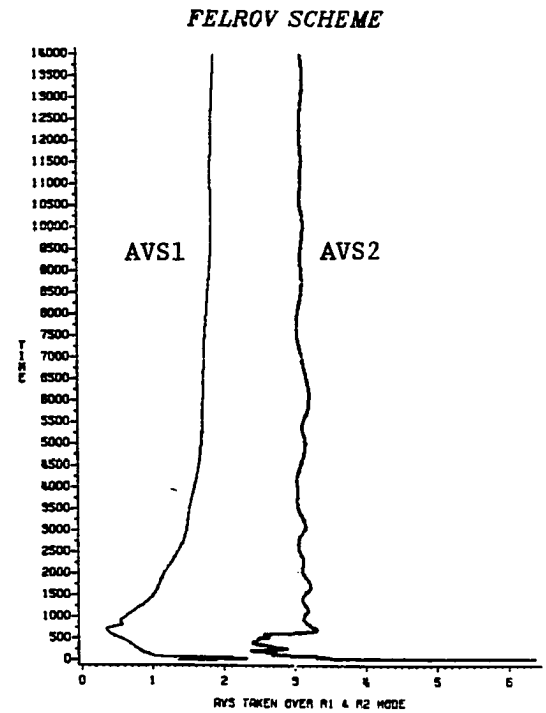
So we can conclude that for a sufficiently long operation the automata will select a class that minimizes the average waiting time with a frequency as close to unity as desired.

Felrov's scheme has been simulated for the case of 2 classes. The plots for average service time (AVS) taken over learning and selection mode, and average service time taken over learning mode for different sets of parameters λ_i, μ_i^{-1}, $i = 1,2$, are given in Figure 3.7.

Remark 3.14: The learning automata approach may be regarded as a selective sampling, where sampling is not on a regular cyclic basis (like Felrov scheme), but rather on a learned basis, where "better" population (distribution with the smallest mean) is sampled with increasingly higher frequencies as compared with the poorer population. In overall comparison, the intuitive expectation is that the learning automata approach because of more learning capability should be superior to the Felrov approach.

Remark 3.15: The time of final decision for the Felrov scheme (entering into selection mode forever) is based on a confidence statistical approach. In the case of learning automata, it is usual to terminate the process when one of the components of vector p(k) has

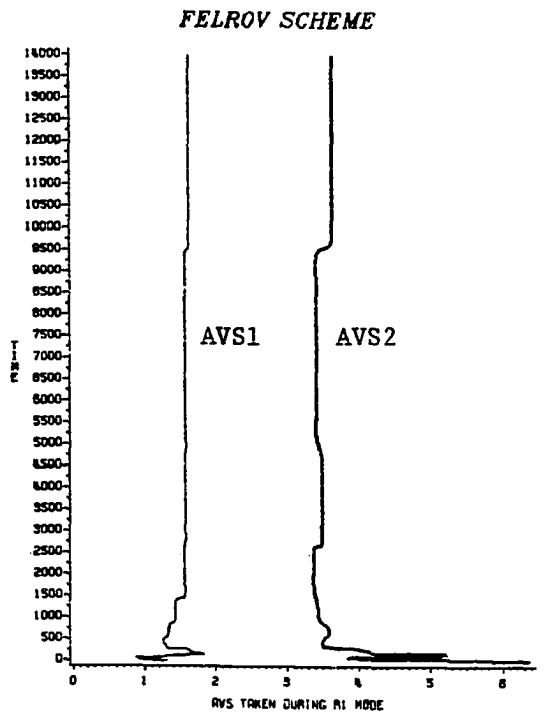


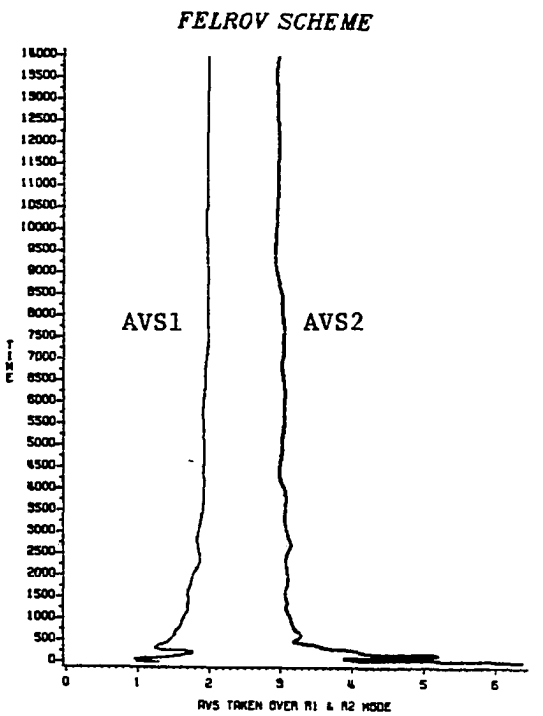


$\lambda_1 = 1/3$ $\lambda_2 = 1/15$ $\mu_1^{-1} = 2$ $\mu_2^{-1} = 3$

$CL_i(R_2) = 2^i$, $CL_i(R_1) = 2k$, $k = 10$

Figure 3.7a

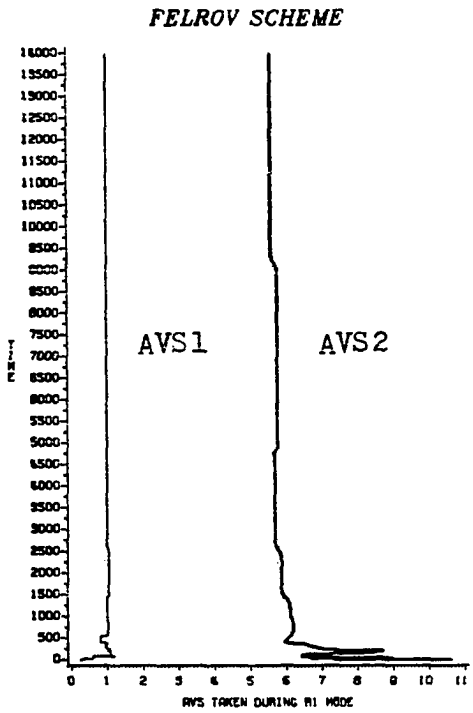


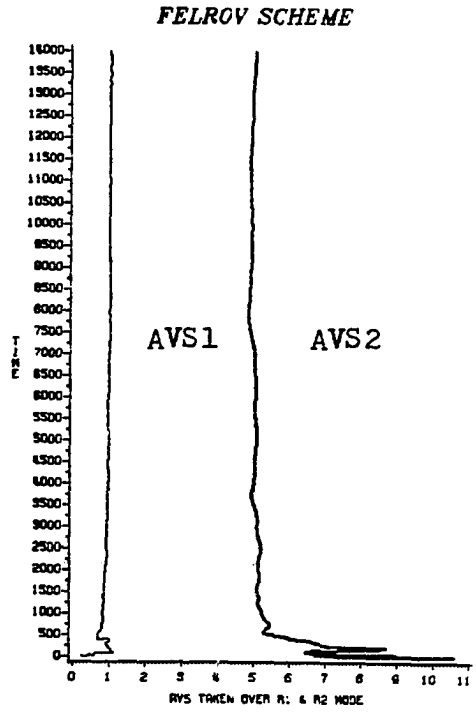


$\lambda_1 = 1/3$ $\lambda_2 = 1/6$ $\mu_1^{-1} = 2$ $\mu_2^{-1} = 3$

$CL_i(R_2) = 2^i$, $CL_i(R_1) = 2k$, $k = 10$

Figure 3.7b





$\lambda_1 = 1/3$ $\lambda_2 = 1/6$ $\mu_1^{-1} = 1$ $\mu_2^{-1} = 5$

$CL_i(R_2) = 2^i$, $CL_i(R_1) = 2k$, $k = 10$

Figure 3.7c

reached some suitable value such as 0.9 or 0.95. It is clear that the use of confidence intervals to specify the terminal time with some preassigned level of confidence implies a corresponding probability of error for the final decision. In principle, using the Gaussian properties of the sampling distributions (in view of central limit theorem), the conversion of confidence intervals may easily be computed. The calculation of error probability for the automata method can be performed in principle only if the distribution of p(k) is available. In view of the above discussion, the following criteria may be used to compare the speed of convergence of the Felrov scheme with learning automata approach.

Definition: Two schemes have the same speed of convergence if the number of samples needed to achieve specific level of accuracy are equal.

3.6 Conclusion

This chapter describes an application of variable structure learning automata to priority assignment in a queuing system with unknown parameters. As a consequence, a learning algorithm for assigning priorities is presented. The proposed priority assignment system keeps track of the average length of the service taken over both classes called 'Dynamic Threshold' (T(k)). T(k) is used to obtain binary responses required for a P-model learning automata. If a job, selected from a particular class, has service time less than T(k) then the probability of choosing that class for the next time cycle will be increased, and if the real service time is longer than the dynamic threshold, probability of choosing will be decreased. It is shown that the system will converge to the classical priority rule

with a probability as close to unity as desired. It has been shown that by modifying the dynamic threshold, the same learning algorithm would assign priorities in such a way that minimizes the system's total average waiting cost. The problem of priority assignment under unknown parameters is also formulated using a class of multi-input automata which uses sample mean to select between different actions. It is shown that for a sufficiently long operation, the automata will select a class that minimizes the average waiting time with a frequency as close to unity as desired.

CHAPTER IV

CONCLUSION

This chapter presents a summary of the results presented in this thesis and suggests some problems for further work in the field of learning automata and its application.

The aim of this thesis is the theoretical investigation of learning automata and its application to the priority assignment in a queueing system with unknown characteristics. The major mathematical tools used are the various martingale convergence theorems and theory of Markov process. Many of the algorithms reported in this thesis can be applied to pattern recognition, game theory, hypothesis testing, parameter optimization, etc. [L3].

First, in this thesis the behavior of stochastic automata operating in an unknown stationary random environment is discussed. A general class of learning algorithms, which has identical behavior under the occurrence of failure and success, for updating state probability of the automata (p(k)) is presented. This new class of learning algorithms subsumes most of the well-known schemes available in the literature. The choice of functions ($f_j^i[p]$, $g_j^i[p]$) which are used in the learning algorithm is quite untraditional and is needed to induce identical behavior of the learning algorithm under success and failure. The new concept of strong absolute expediency is introduced and the necessary

and sufficient conditions for strong absolute expediency are given. These conditions turn out to be very general but simple conditions of the symmetry of functions figuring in the learning algorithm.

The strongly absolute expedient schemes reported are such that the set of all unit vectors V_M form the absorbing barriers of the Markov process $\{p(k)\ ,\ k \geq 0\}$ defined by the learning algorithm. It is shown that strong absolute expediency implies that $\{p(k),\ k \geq 0\}$ in fact converges to the set of all unit vectors V_M with probability one, and thus the learning algorithm converges to the desired state with a non-zero probability. To prove ε-optimality of the proposed learning algorithm, lower bound on the probability of convergence to the desired state is obtained. Using the lower bound, it is shown that by properly choosing the parameters in learning algorithm one can make the probability of converging to the desired state as close to unity as desired. In other words, strong absolute expediency implies ε-optimality. It is shown that for all three types of algorithm: reward-penalty, inaction-penalty and reward-inaction, lower bound on the probability of convergence to the desired state can be computed. This is in sharp contrast with the existing results in the literature wherein for the inaction-penalty algorithm no lower bound on probability of convergence to the desired action has been established. The reason for this anomaly is that for all currently available absolutely expedient learning algorithm only reward-penalty, reward-inaction algorithms have V_M as the only set of the absorbing states. For this class of algorithm, when one of the p_i's is zero then dimensionality of the algorithm is reduced automatically.

An application of learning algorithms of the type studied in Chapter II of this thesis to the priority assignment in a queuing system with unknown characteristics is presented. As a consequence, a learning algorithm for assigning priorities for a two-class M/M/1 queuing system is given. The priority assignment system keeps track of the average service time taken over both classes. If a job chosen from one of the classes has service time less than this average, then the probability of choosing that class will be increased, and decreased if the actual service time of the job is greater than this average. The learning algorithm for assigning priority (updating probability of choosing different classes) has two major steps: updating algorithm and timing algorithm. The update algorithm which is of the type discussed in Chapter II, decides as to how to update probability of choosing the class (p(k)) if to be updated, and update timing algorithm decides as to when to update the probability of choosing different classes. It is formally shown that the system would asymptotically converge to the classical priority assignment with a probability as close to unity as desired. That is, the proposed system will asymptotically assign the highest priority to the class with the shortest service time. The learning algorithm is modified to minimize total average waiting cost for the system. The learning algorithm for assigning priority is also applied to a three class M/M/1 priority queuing system. Due to the increasing complexity of the learning algorithm as the number of classes increase a class of multi-input automata which uses sample mean to select among different classes is used to formulate the m-class M/M/1 priority assignment. It is proved that for a

sufficiently long operation this class of automata will arrange classes in such a way that minimizes the average waiting time with a frequency as close to one as desired.

Suggestions for further work:

In what follows, we mention some of the problems which have arisen in the course of research described in the previous chapters.

1. The functional equation obtained from conditions (S.1) - (S.2) should be solved to provide us a direct procedure for designing the ε-optimal learning algorithm.

2. Derivation of an analytical measure of the speed of convergence of a given learning algorithm is imperative for further analysis of the learning algorithm.

3. Application of the general class of learning algorithm reported in Chapter II to game theory and learning automata operating in a multi-teacher environment should be studied.

4. In theorem (3.1) and (3.2) we assumed that service time distribution is exponential. The non-exponential service time remains to be investigated.

5. As mentioned in Chapter III, the learning algorithm needed for assigning priority becomes more complex and messy as the number of classes increases. A more efficient algorithm (timing and update algorithm) must be searched for the case where there are more than two classes of job.

6. The priority assignment system reported in Chapter III is a stand-alone system. Its application to the networks such as telephone network or data communication network must be investigated.

7. A closed form formula for ρ_1, ρ_2, W_1 and W_2 when ($p_1(k) = a$, $p_2(k) = 1-a$ for all k) should be derived. The next step in this direction is to find ρ and W when there are more than two classes in the system.

8. Application of S and Q model learning automata to priority assignment under unknown parameters should be studied.

9. The result of Chapter II must be extended to the situation where either queue length is finite or preemption is allowed.

10. Learning automata and Felrov approach to priority assignment must be compared in terms of their speed of convergence (refer to remark 3.15).

REFERENCES

[A1] Aso, H. and Kimura, M., "Absolute Expediency of Learning Automata." Information Science, Vol. 17, pp. 91-112, 1979.

[A2] Avi-Itzhak, B., Brosh, I. and Naor, P., "On Discretionary Priority Queuing." Z. Angew. Math. Mech. 6, pp. 235-242, 1964.

[B1] Bush, R. R. and Mosteller, F., Stochastic Models for Learning. Wiley, 1958.

[B2] Baba, N. and Sawaragi, Y., "On Learning Behavior of Stochastic Automata under Non-Stationary Environment." IEEE Transactions on Systems, Man and Cybernetics, Vol. 5, pp. 273-275, 1975.

[C1] Chandrasekran, B. and Shen, D.W.C., "On Expediency and Convergence in Variable Structure Automata." IEEE Transactions on System Science and Cybernetics, Vol. 4, pp. 52-60, 1968.

[C2] Cox, D. R. and Smith, W. L., Queues. Methven, London, 1961.

[D1] Doob, J. L., Stochastic Processes. Wiley, 1951.

[F1] Fu, K. S., "Learning Control Systems. Review and Outlook." IEEE Transactions on Automatic Control, Vol. 15, pp. 210-221, April 1970.

[F2] Fu, K. S., "Learning Control Systems and Intelligent Control Systems: An Intersection of Artificial Intelligence and Automatic Control." IEEE Transaction on Automatic Control, Vol. 16, pp. 70-72, 1971.

[F3] Findler, N. V., "Studies in Machine Cognition Using the Game of Poker." CACM, Vol. 20-4, pp. 230-245, April 1977.

[F4] Fu, K. S., "Stochastic Automata Models for Learning Systems." In Computers and Information Science II (Ed) by J. T. Tou, Academic, 1967.

[F5] Felrov, Yu. A., "Some Classes of Multi-Input Automata." Journal of Cybernetics, Vol. 2, pp. 112-122, 1972.

[G1] Griffith, A. K., "A Comparison and Evaluation of Three Machine Learning Procedure as Applied to the Game of Checkers." Artificial Intelligence, Vol. 5, pp. 137-148, 1974.

[H1] Herkenrath, U., Kalin, D. and Lakshmivarahan, S., "On a General Class of Absorbing Barrier Learning Algorithm." Information Sciences, Vol. 23, 1981.

[I1] Iosifescu, M. and Theodorescu, R., Random Processes and Learning. Springer Verlag, 1969.

[J1] Jaiswal, N., Priority Queues. Academic Press, New York, 1968.

[K1] Krylov, V. Yu., "On One Stochastic Automation which is Asymptotically Optimal in a Random Media." Automation and Remote Control, Vol. 24, pp. 1114-1116, 1963.

[K2] Krinskii, V. I., "An Asymptotically Optimal Automata with Exponential Convergence." Bio Physics, Vol. 9, pp. 484-487, 1964.

[K3] Kleinrock. L., "Time-Shared System: A Theoretical Treatment." Journal of the Association for Computing Machinery 14(2), pp. 242-261, 1967.

[K4] Kleinrock, L., Queueing System. Volume II: Computer Applications, John Wiley, New York, 1976.

[L1] Luce, R. D., Individual Choice Behavior. Wiley, 1959.

[L2] Lakshmivarahan, S. and Thathachar, M.A.L., "Absolutely Expedient Learning Algorithm for Stochastic Automata." IEEE Transactions on Systems, Man and Cybernetics, Vol. 3, pp. 281-286, 1973.

[L3] Lakshmivarahan, S., Learning Algorithms: Theory and Applications. Springer Verlag, New York, 1981.

[L4] Lakshmivarahan, S. and Thathachar, M.A.L., "Bounds on the Probability of Convergence of Learning Automata." IEEE Transactions on Systems, Man and Cybernetics, Vol. 6, pp. 756-753, 1976.

[L5] Li, T. J. and Fu, K. S., "Automata Games, Stochastic Automata and Formal Language." Purdue University, Lafayette, Inc., Tech. Rep. TR-EE69-1, January 1969.

[L6] Lakshmivarahan, S., "Two Person Decentralized Team with Incomplete Information." Applied Mathematics and Computation, Vol. 8, pp. 51-78, 1981.

[L7] Lakshmivarahan, S., and Narendra, K.S., "Learning Algorithms for Two-Person Zero Sum Stochastic Games with Incomplete Information - A Unified Approach." SIAM Journal on Control and Optimisation, Vol. 20, No. 4, pp. 541-552, 1982.

[L8] Lakshmivarahan, S. and Narendra, K.S., Learning Algorithms for Two Person Zero Sum Stochastic Games with Incomplete Information." Mathematics of Operations Research, Vol. 6. No. 4, 1981.

[L9] Lakshmivarahan, S. and Thathachar, A.L., "Hypothesis Testing Using Variable Structure Stochastic Automata." In 1973 IEEE Decision and Control Conference, Preprints, San Diego, CA.

[L10] Lakshmivarahan, S. and Thathachar, A.L., "Absolutely Expedient Learning Algorithms for Stochastic Automata." IEEE Transactions on Systems, Man and Cybernetics, Vol. 3. pp. 281-286, 1973.

[L11] Lakshmivarahan, S., "Learning algorithms for Stochastic Automata." Ph. D. Dissertation, Dept. Elec. Engr., India Institute of Science, Bangalore, India, Jan. 1973.

[L12] Luce, R. D., Individual Choice Behavior. Wiley, 1959.

[M1] Mendel, J. M. and Fu, K. S., Adaptive Learning and Pattern Recognition Systems. Academic Press, 1970.

[M2] McMurtry, G. J. and Fu, K. S., "A Variable Structure Automata Used as a Multi-Modal Search Technique." IEEE Transactions on Automatic Control, Vol. 11, pp. 379-387, 1966.

[M3] Meybodi, M. R. and Lakshmivarahan, S., "ε-optimality of a General Class of Learning Algorithms." Information Science, Vol.28, pp. 1-20, 1982.

[M4] Meybodi, M. R. and Lakshmivarahan, S., "A Learning Approach to Priority Assignment in a Two Class M/M/1 Queueing System With Unknown Parameters." Technical Report, School of Electrical Engineering and Computer Science, University of Oklahoma, Jan. 1983.

[M5] Meybodi, M. R. and Lakshmivarahan, S. "On a Class of Learning Algorithm with Symmetric Behavior Under Success and Failure." Proceedings of the Conference on Mathematical Learning Models - Theory and Applications. Held at the University of Bonn, W. Germany, May 3-7, 1982. Springer Verlag, Lecture Notes in Statistics.

[M6] Mood, A. M., Graybill, F. A. and Boes, D. C., Introduction to the Theory of Statistics. McGraw Hill, 1974.

[M7] Mova, V. V. and Ponamarenko, L. A., "On the Optimal Assignment of Priorities, Depending on the State of a System with Limited Number of Waiting Places." Engr. Cybern., Vol. 12, No. 5, 1974.

[M8] McLaren, R. W., "A Stochastic Automata Model for Synthesis of Learning System." IEEE Trans. Syst. Sci. Cybern., Vol. SSC-2, pp. 109-114, Dec. 1966.

[N1] Norman, M. F., Markov Processes and Learning Models. Academic Press, 1972.

[N2] Norman, M. F., "On Linear Models with Two Absorbing Barriers." Journal of Mathematic Psychology. Vol. 5, pp. 225-241, 1968.

[N3] Narendra, K. S. and Thathachar, M.A.L., "Learning Automata - A Survey." IEEE Trans. Systems, Man and Cybernetics, Vol. 4, pp. 323-334, 1974.

[N4] Narendra, K. S. and Lakshmivarahan, S., "Learning Automata, A Critique." Journal of Cybernetics and Information Sciences - Special Issue on Learning Automata. Vol. 1, pp. 53-66, 1978.

[N5] Narendra, K. S., Wright, E. and Mason, L. G., "Application of Learning Automata to Telephone Traffic Routing." IEEE Trans. on Systems, Man and Cybernetics, pp. 785-792, 1977.

[N6] Narendra, K. S. and Thathachar, M.A.L., "On the Behavior of a Learning Automation in a Changing Environment with Routing Applications." IEEE Trans. on Systems, Man and Cybernetics, Vol. 10, pp. 262-269, 1980.

[N7] Narendra, K. S. and Viswanathan, R., "A Two-Level System of Stochastic Automata for Periodic Random Environment." IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-2, pp. 285-289, 1972.

[N8] Norman, M. F., "Some Convergence Theorems for Stochastic Learning Models with Distance Diminishing Operators." Journal of Mathematical Psychology. Vol. 5, pp. 61-101, 1968.

[N9] Narendra, K. S. and Thathachar, M.A.L., Learning Automata. Forthcoming book.

[P1] Ponomarov, V. A., "A Construction of an Automata which is Asymptotically Optimal in a Stationary Random Media." Bio Physics, Vol. 9, pp. 104-110, 1964.

[P2] Paz, A., Introduction to Probabilistic Automata, Academic Press. New York, 1971.

[S1] Sklansky, J., "Learning System for Automatic Control." IEEE Trans. on Automatic Control, Vol. 11, pp. 6-19, 1966.

[S2] Solomonoff, R., "Some Recent Work in Artificial Intelligence." Proc. of the IEEE, Vol. 54, Dec. 1966.

[S3] Solomonoff, R., "A Formal Theory of Inductive Inference." Part II, Information and Control. pp. 224-254, June 1964.

[S4] Solomonoff, R., "A Formal Theory of Inductive Inference." Part I, Information and Control, pp. 1-22, March 1964.

[S5] Samuel, A., "Some Studies in Machine Learning Using the Game of Checkers." IBM J. Res. Develop., Vol. 11, pp. 601-617, Nov. 1967.

[S6] Sawargi, Y. and Baba, N., "Two ε-optimal Non-linear Reinforcement Schemes for Stochastic Automata." IEEE Trans. on Systems, Man and Cybernetics, Vol. 4, pp. 126-131, 1974.

[S7] Shapiro, I. J. and Narendra, K. S., "Use of Stochastic Automata for Parameter Self-Optimization with Multimodal Performance Criteria." IEEE. Trans. on Systems, Man and Cybernetics, Vol. 5, pp. 352-360, 1969.

[S8] Srikantakumar, P. R. and Narendra, K. S., A Learning Model for Routing in Telephone Networks." SIAM Journal on Control and Optimization, Vol. 20, No. 1, Jan. 1982.

[S9] Srikantakumar, P. R., "Learning Models and Adaptive Routing in Communications Network." Ph.D. Dissertation, Yale University, August, 1980.

[S10] Shapiro, I. J., "The Use of Stochastic Automata in Adaptive Control." Ph.D. Dissertation, Dept. Engr. and Applied Sci., Yale Univ., New Haven, Conn., 1969.

[T1] Tsypkin, Ya Z., Adaptation and Learning in Automatic System. Academic Press, 1971.

[T2] Tsetlin, M. L., Automation Theory and Modelling of Biological Systems. Academic Press, 1973.

[T3] Tsypkin, Ya Z., Foundations of the Theory of Learning Systems. Academic Press, 1973.

[T4] Tsypkin, Ha Z. and Poznyak, A. S., "Finite Learning Automata." Engr. Cyber., Vol. 10, pp. 478-490, May-June 1972.

[V1] Varshavskii, V. I. and Vorontsova, I. P., "On the Behavior of Stochastic Automata with Variable Structure." Automation and Remote Control, Vol. 24, pp. 327-333, 1963.

[V2] Viswanathan, R. and Narendra, K. S., "Expedient and Optimal Variable Structure Stochastic Automata." Dunham Lab. TR-37, 1970. Yale University, New Haven, Conn.

[V3] Viswanathan, R., and Narendra, K. S., "A Note on Linear Reinforcement Scheme for Variable Structure Stochastic Automata." IEEE Trans. on Systems, Man and Cybernetics. Vol. 2, pp. 292-294, 1972.

[V4] Viswanathan, R. and Narendra, K. S., "Games of Stochastic Automata." IEEE Trans. on Systems, Man and Cybernetics, Vol. 4, pp. 131-135, 1974.

[V5] Viswanathan, R., "Learning Automata: Model and Application." Ph. D. Dissertation, Dept. Elec. Engr. & Applied Sci., Yale Univ., New Haven, Conn., 1972.

[V6] Varshavskii, V. I., Meleshina, M. V. and Tsetlin, M. L., "Priority Organization in Queuing Systems Using a Model of Collective Behaviour of Automata." Problemy Pereduchi Informatsii, Vol. 4, pp. 73-76, 1968.

[W1] Waterman, D. A., "Generalization Learning Techniques for Automating the Learning of Heuristics." Artificial Intelligence. Vol. 1, Nos. 1 and 2, pp. 121-170, 1970.

APPENDICES

APPENDIX A

Definition A: Let $X_n \geq 0$, $n = 0,1,2,3,\ldots$ be a sequence of real valued non-negative random variables and let R_n, $n = 0,1,2,3 \ldots$ be a non-decreasing sequence of σ-algebra.

(i) (X_n, R_n), $n=0,1,2,3,\ldots$ is called a martingale if for all $n \geq 0$

$$E[X_{n+1} \mid R_n] = X_n$$

(ii) (X_n, R_n), $n=0,1,2,3,\ldots$ is called a submartingale if for all $n \geq 0$

$$E[X_{n+1} \mid R_n] \geq X_n$$

(iii) (X_n, R_n), $n=0,1,2,3,\ldots$ is called a super-martingale if for all $n \geq 0$

$$E[X_{n+1} \mid R_n] \leq X_n$$

Theorem A.1: Let (X_n, R_n) be a submartingale

(i) If $E[X_n] < \infty$, then $\lim_{n \to \infty} X_n = X_\infty$

exists with probability one and X_∞ is a random variable such that $E[X_\infty] < \infty$.

(ii) If X_n's are uniformly integrable, then $\sup_n E[X_n] < \infty$ and $\lim_{n \to \infty} E[|X_n - X_\infty|] = 0$

Theorem A.2: Let (X_n, R_n) be a super-martingale

(i) If $E[X_n] < \infty$, then $\lim_{n \to \infty} X_n = X_\infty$ exists with probability one and X_∞ is a random variable such that $E[X_\infty] < \infty$.

(ii) If the X_n's are uniformly integrable, then $\lim_{n \to \infty} E[|X_n - X_\infty|] = 0$.

APPENDIX B

In this appendix we shall prove the proposition 2.1 of Chapter II. In the following we consider only 2-state stochastic automata. We first observe that stochastic automata with variable structure defines a non-homogeneous Markov chain on the states of stochastic automata. The associated reinforcement scheme (2) defines the evolution of a process $\{p_1(k),\ k \geq 0\}$ (as $p_1(k) + p_2(k) = 1$ we shall consider $p_1(k)$ alone) which is also Markov in nature. But this process $\{p_1(k),\ k \geq 0\}$ is a continuous state space, discrete parameter stationary Markov process. The random variable $p_1(k)$ for every $k \geq 0$ itself is a probability and hence $p_1(k) \in [0,1]$ for every k. Let us call $\overline{S} = [0,1]$, a subset of the real line, the state space of the Markov process. If x and y belong to $\overline{S}$ define $d(x,y) = |x-y|$, the absolute value of the difference between x and y. Clearly $(\overline{S}, d)$ is a metric space. As $\overline{S}$ is a closed and bounded subset, in fact $(\overline{S}, d)$ is a compact metric space. It is shown in Chapter II that the process $\{p_1(k),\ k \geq 0\}$ as defined by (2) converges with probability one to 0 and 1. The process $\{p_1(k),\ k \geq 0\}$ is said to be in state $s_k \in \overline{S}$ if $p_1(k) = s_k$.

$$\text{Let } K^{(k)}(s,B) = \text{Prob } \{s_{k+1} \in B \mid s_1 = s\} \tag{B.1}$$

for all $k \geq 0$ and $s \in \overline{S}$. Let $T_k(s)$ be defined as

$$T_k(s) = \{s' : K^{(k)}(s,\{s'\}) > 0\} \tag{B.2}$$

for $k \geq 0$, that is, it is the set of all states that can be reached from

s after k-transitions. An absorbing barrier is one, that once reached cannot be left, that is,

$$K(s, \{s\}) = 1 ,$$

or
$$T_k(s) = \{s\} . \tag{B.3}$$

If B and C are any two subsets of $\bar{S}$

$$d(B,C) = \inf_{x\in B, y\in C} d(x,y) . \tag{B.4}$$

If C is a unit set $\{c\}$, then d(B,C) is written as

$$d(B, C) = d(B, c) . \tag{B.5}$$

For the Markov process under consideration there are finite number of absorbing states, $a_1, a_2, \ldots, a_N$ such that for any state $s \in \bar{S}$ there is an $a_{j(s)}$ for which

$$\lim_{k\to\infty} d(T_k(s), a_{j(s)}) = 0 . \tag{B.6}$$

Let $C(\bar{S})$ be the space of all continuous functions on $\bar{S}$. If $\psi(x) \in C(\bar{S})$, then the norm of ψ denoted by $|\psi|$ is given by

$$|\psi| = \sup_{x\in\bar{S}} |\psi(x)| . \tag{B.7}$$

Let U be the operator as defined in (2.34). The following 2 lemmas are due to Norman [N1].

Lemma B.1: The operator U has no eigenvalue of modules 1 other than 1

Proof: Let $\psi \in C(\bar{S})$ and λ be such that $|\lambda| = 1$ and $\lambda \neq 1$. We shall show that such a λ cannot be an eigenvalue of U. To this end let $s_0 \in S$ be a state to which

$$|\psi(s_0)| = |\psi| . \tag{B.8}$$

Notice that the left hand side is the absolute value of the function at the point s_0 whereas the right hand side is the norm of the function ψ.

Let

$$C_k = \{s : \psi(s) = \lambda^k \psi(s_0)\} . \tag{B.9}$$

for k = 0,1,2,3,....

Now

$$U^k\psi(s_0) = \lambda^k\psi(s_0) . \tag{B.10}$$

Thus

$$K^{(k)}(s_0, C_k) = 1 . \tag{B.11}$$

and $T_k(s_0) \subset C_k$. By (B.6) there exists a sequence t_k such that $t_k \in T_k(s_0)$ and $\lim_{k\to\infty} d(t_k, a_j(s_0)) = 0$. (B.12)

Hence $\lim_{k\to\infty} \psi(t_k)$ converges and in fact it is equal to $\psi[a_j(s_0)]$.

But $t_k \in C_k$ and so

$$\psi(t_k) = \lambda^k\psi(s_0) . \tag{B.13}$$

But as λ is such that $|\lambda| = 1$ and $\lambda \neq 1$ the right hand side of (B.13) converges only if $\psi(s_0) = 0$. In view of (B.8) $\psi(s_0) = 0$ implies $|\psi| = 0$ which in turn implies that $\psi(s) \equiv 0$. Thus λ is not an eigenvalue of U.□

In what follows let $A = \{a_0, a_1, \ldots, a_{N-1}\}$ be the set of N absorbing barriers.

Lemma B.2: If $b_0, b_1, \ldots, b_{N-1}$ are any N scalars, there is one and only one $\psi \in C(\bar{S})$ such that $U\psi = \psi$ and $\psi(a_i) = b_i$, i = 0,1,2,...,N-1.

Proof: Uniqueness: To prove the uniqueness we shall first prove a maximum modulus principle. If $\psi(s) \in C(\bar{S})$ and

$$U\psi = \psi , \tag{B.14}$$

then all maxima of $|\psi(\cdot)|$ occur on A and possibly elsewhere. Let s_0 be a state such that $|\psi(s_0)| = |\psi|$ and let $C = \{s: \psi(s) = \psi(s_0)\}$. Since, in view of (B.14) $U^k\psi(s_0) = \psi(s_0)$, $K^{(k)}(s_0, C) = 1$ and so

$T_k(s_0) \subset C$. By (B.6) there exists a sequence t_k such that $t_k \in T_k(s_0)$ and $\lim_{k\to\infty} d(t_k, a_{j(s_0)}) = 0$. Hence $\lim_{k\to\infty} \psi(t_k) = \psi(a_{j(s_0)})$ and $|\psi(a_{j(s_0)})| = |\psi|$. In view of the definition of the norm of ψ, it follows that $|\psi(a_{j(s_0)})|$ is maximum. This is the required maximum modules principle.

To show the uniqueness, suppose ψ and $\psi' \in C(S)$ and are such that

$$U\psi = \psi \quad , \quad U\psi' = \psi', \tag{B.15}$$

and

$$\psi(s) = \psi'(s) \quad \text{for all } s \in A \ . \tag{B.16}$$

Let

$$\psi'' = \psi - \psi'. \tag{B.17}$$

Then, clearly $\psi'' \in C(\bar{S})$ and $U\psi'' = \psi''$ since U is a linear operator. But in view of (B.16) $\psi''(s) = 0$ for $s \in A$. This implies that $|\psi''(s)| = 0$ for all $s \in A$ which in turn implies $|\psi''| = 0$ and so $\psi(s) \equiv \psi'(s)$. Thus the uniqueness is proved.

Existence: Since

$$U^k\psi = E[\psi(s_{k+1}) \mid s_0 = s], \tag{B.18}$$

we have

$$U^k \psi(s) = \psi(s) \ , \tag{B.19}$$

for all $s \in A$, for all $n \geq 0$ and $\psi \in C(\bar{S})$. Thus

$$U_1 \psi(s) = \lim_{k\to\infty} U^k \psi(s) = \psi(s) \ , \tag{B.20}$$

for $s \in A$.

Let $W_1, W_2, \ldots, W_N \in C(\bar{S})$ be such that $W_i(a_j) = \delta_{ij}$, for example

$$W_i(s) = \left[1 - \frac{d(s,a_i)}{\varepsilon} \right]^+ ,$$

where $\varepsilon = \min_{j\neq i} \{d(a_i,a_j)\}$ and x^+ is x or 0,

depending on whether $x \geq 0$ or ≤ 0. It will be shown that

$$\Gamma(s) = \sum_{i=0}^{N-1} b_i U_1 W_i(k),$$

is the function sought. Clearly $\Gamma \in C(\bar{S})$ and

$$\begin{aligned} \Gamma(a_j) &= \sum_{i=0}^{N-1} b_i U_1 W_i(a_j) \\ &= \sum_{i=0}^{N-1} b_i W_i(a_j) \qquad (B.21) \\ &= b_j . \end{aligned}$$

Finally we have

$$\begin{aligned} U\Gamma &= \sum_{i=0}^{N-1} b_i U U_1 W_i \\ &= \sum_{i=0}^{N-1} b_i U_1 W_i \\ &= \Gamma . \qquad \square \qquad (B.22) \end{aligned}$$

Remark B.1: Though the lemma B.2 is true for a general case of N absorbing barriers, only a special of 2 absorbing barriers, that is, N=2 and $a_0 = 0$ and $a_1 = 1$ is of our interest. In this case it can be seen that

$$\begin{aligned} \varepsilon &= \min_{i \neq j} \{d(a_i, a_j)\}, \; i, j = 0, 1 , \\ &= 1 . \end{aligned} \qquad (B.23)$$

Also $\quad d(s, a_0) = d(s, 0) = s ,$

and $\quad d(s, a_1) = d(s, 1) = 1-s. \qquad (B.24)$

From (B.23) and (B.24) it is clear that

$$W_0(s) = (1-s),$$

and

$$W_1(s) = s . \tag{B.25}$$

In what follows we shall identify $s_k = P_1(k)$ and the operator as defined (2.47). By setting $\psi(x) = x$ we have the following lemma.

Lemma B.3: The sequence $\{S_k\}_{k=1}^{\infty}$ as defined by

$$Us = E[s_k + 1 \mid s_k = s], \tag{B.26}$$

converges to a random variable S_∞ with probability one.

Proof: Replacing $p_1(k)$ by s_k and in view of theorem (2.3) we have

$$E[s_{k+1} \mid s_k] > s_k . \tag{B.27}$$

Inequality (B.27) implies that s_k is a non-negative submartingale and according to submartingale theorem s_k converges with probability one, that is, $\lim_{k\to\infty} s_k = s_\infty$ exists with probability one and is a random variable. □

Remark B.2: It is further established in Chapter II that $\lim_{k\to\infty} p_1(k) = p_1^*$ cannot take any other value other than 0 and 1 with probability one. Thus when $s_k = p_1(k)$, it is clear that s_∞ can take on only two values 0 and 1 with probability one.

Now that it is shown s_∞ exists and it can take on only two values 0 and 1 with probability one, we have the following lemma.

Lemma B.4: For $i = 0, 1, \ldots, N-1$, let Γ_i be the continuous function such that $\Gamma_i(a_j) = \delta_{ij}$ and $U\Gamma_i = \Gamma_i$, then

$$E[\psi(s_\infty) \mid s_0 = s] = \sum_{i=1}^{N-1} \Gamma_i(s)\, \psi(a_i), \tag{B.28}$$

where $A = \{a_0, a_1, a_2, \ldots, a_{N-1}\}$ is the set of absorbing barriers.

Proof: For any $\psi \in C(\bar{S})$ let

$$\bar{\psi}(s) = \sum_{i=0}^{N-1} \Gamma_i(s)\ \psi\ (a_i)\ , \tag{B.29}$$

and

$$\bar{\psi}'(s) = U_1\ \psi(s)\ , \tag{B.30}$$

where U_1 is defined in (B.20).

Clearly $\bar{\psi}$ and $\psi' \in C(\bar{S})$ and

$$\psi'(a_j) = \psi(a_j) = \bar{\psi}(a_j); \qquad j = 0,1,\ldots,N-1.$$

Also

$$U\,\bar{\psi} = \sum_{i=0}^{N-1} \psi(a_i)\, U\,\Gamma_i$$

$$= \sum_{i=0}^{N-1} \psi(a_i)\ \ \Gamma_i$$

$$= \bar{\psi}\ , \tag{B.31}$$

and

$$U\,\psi' = U\,U_1\psi = \psi'. \tag{B.32}$$

In view of the uniqueness of the solution $U\Gamma = \Gamma$ as proved in lemma B.2, we have $\bar{\psi} = \psi'$ which is in fact the assertion (B.28) of the lemma B.4. □

Remark B.3: In view of lemma B.3 and B.4 it is clear that the solution of $U\,\Gamma = \Gamma$ admits the representation

$$\Gamma(s) = E[\ \psi(s_\infty) \mid s_0 = s]$$

$$= \sum_{i=1}^{N-1} \Gamma_i(s)\ \psi(a_i). \tag{B.33}$$

Clearly in view of the fact $\Gamma_i(a_j) = \delta_{ij}$, we have

$$\Gamma(a_j) = \psi(a_j) = b_j\ . \tag{B.34}$$

Remark B.4: The assertion (B.28) of the lemma B.4 implies that the random variable s_∞ takes the value a_i with the probability $\Gamma_i(s)$, that is

$$\text{Prob}\ [s_\infty = a_i \mid s_0 = s] = \Gamma_i(s)\ . \tag{B.35}$$

Setting $b_0 = 0$ and $b_1 = 1$, it is seen from (B.33) in our case

$$\Gamma(s) = \text{Prob } [s_\infty = 1 \mid s_0 = s]. \tag{B.36}$$

Thus, with these boundary conditions, namely, $\psi(0) = b_0 = 0$ and $\psi(1) = b_1 = 1$. The unique solution of $U\Gamma = \Gamma$ has the interpretation given in (B.34). In view of the above lemmas, we have the proposition 2.1 of Chapter II.

APPENDIX C

Lemma C: If $H'[u] < 0$ for all real u and $H'[u]$ is strictly monotonically increasing with $H'[0] = 1/2$, then for any real $a > 0$, $b > 0$, we have

$$\int_{-(a+b)}^{0} H'[u]du < \int_{-b}^{a} H'[u]du < \int_{0}^{a+b} H'[u]du. \tag{C.1}$$

Proof: In view of the hypothesis we have the following inequality

$$H'[-(a+b)] < H'[-b] < H'[0] = \frac{1}{2} < H'[a] < H'[a+b]. \tag{C.2}$$

We shall first prove the right hand inequality in (C.1). On rewriting the right hand inequality becomes

$$\int_{-b}^{0} H'[u]du + \int_{0}^{a} H'[u]du < \int_{0}^{a} H'[u]du + \int_{a}^{a+b} H'[u]du. \tag{C.3}$$

After simplification the above inequality becomes

$$\int_{-b}^{0} H'[u]du < \int^{(a+b)} H'[u]du. \tag{C.4}$$

Since $H'[u]$ is strictly monotonically increasing we have

$$\int_{-b}^{0} H'[u]du < b\, H'[0] = \frac{b}{2}, \tag{C.5}$$

and

$$b\, H'[a] < \int_{a}^{(a+b)} H'[u]du. \tag{C.6}$$

As $H'[a] > \frac{1}{2}$, putting (C.5) and (C.6) together we have (C.4) which is the same as the right hand inequality in (C.1). The left inequality can be similarly proved. □

APPENDIX D

From the definition of convex function[1] it can be shown that

$$H[-x\ p_2(k)] \leq p_2(k)\ H[-x]\ , \tag{D.1}$$

and

$$H[x\ p_1(k)] \leq p_1(k)\ H[x]\ . \tag{D.2}$$

Using (D.1) and (D.2) we obtain

$$H[x\ p_1(k)] - H[-x\ p_2(k)] \leq p_1(k)\ H[x] - p_2(k)\ H[-x]\ . \tag{D.3}$$

Since $0 \leq p_i(k) \leq 1$ for all k, i = 1,2 and H[u] is a strictly monotonically increasing function, we have

$$p_1(k)\ H[x] - p_2(k)\ H[-x] \leq H[x]\ , \tag{D.4}$$

From (D.3) and (D.4) we have

$$H[x\ p_1(k)] - H[x\ p_2(k)] \leq H[x]\ .$$

In view of H[u] = ln v[u] and after simplification we obtain

$$\frac{v[-x\ p_2(k)]}{v[x\ p_1(k)]} \geq \frac{1}{v[x]}\ .$$

---

(1) A function ψ defined on an open interval (a,b) is said to be convex if for each x, $y \in (a,b)$ and each λ, $0 \leq \lambda \leq 1$ we have

$$\psi(\lambda x + (1-\lambda)y) \leq \lambda\,\psi(x) + (1-\lambda)\,\psi(y).$$