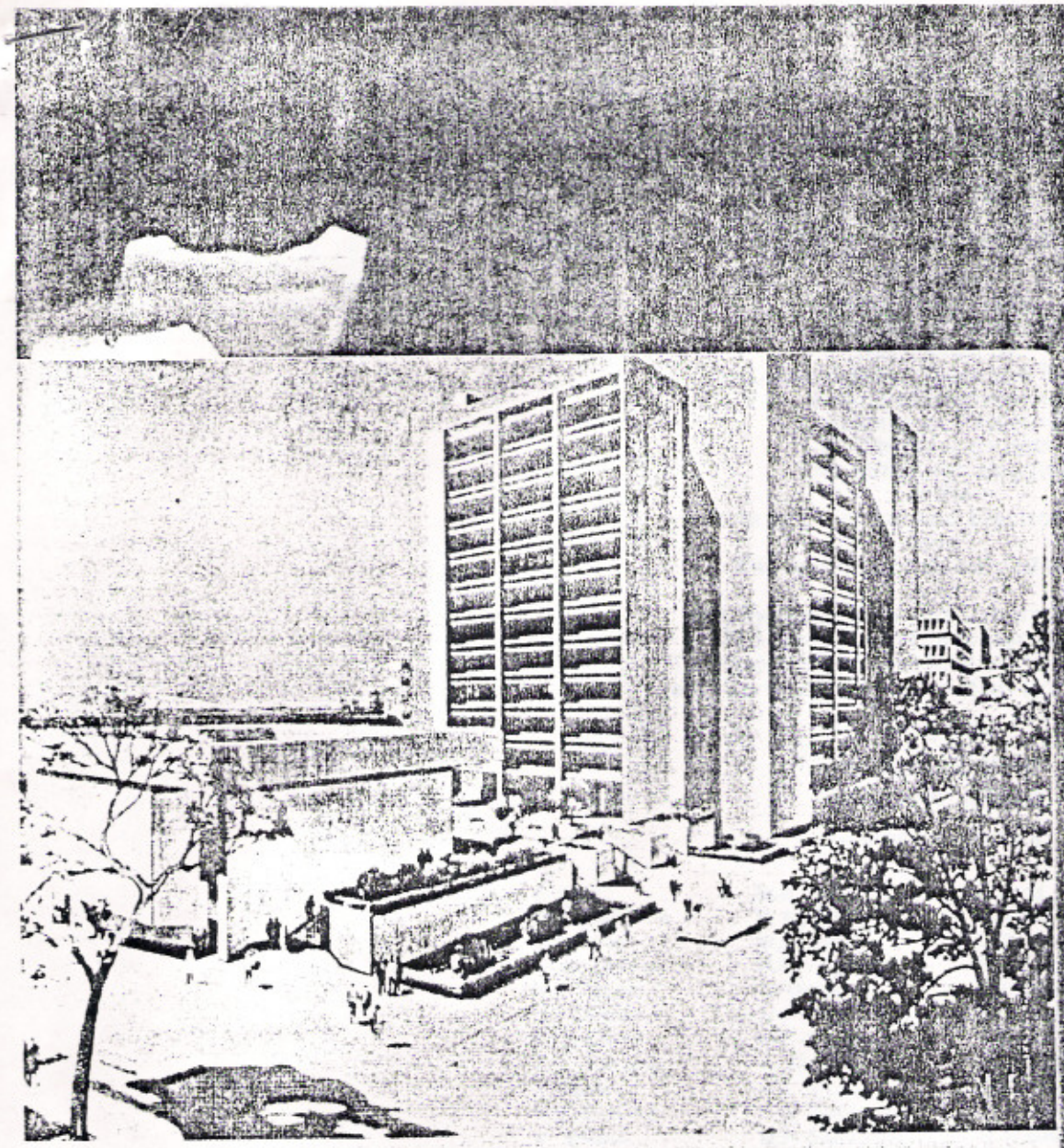


MODELING AND SIMULATION
VOLUME 15

Part 4: Computers and Applications

Proceedings of the Fifteenth Annual
Pittsburgh Conference

held April 19-20, 1984
School of Engineering

University of
Pittsburgh

published and
distributed by
Instrument Society
of America

MODELING AND SIMULATION
VOLUME 15
Part 4

Proceedings of the Fifteenth Annual Pittsburgh Conference

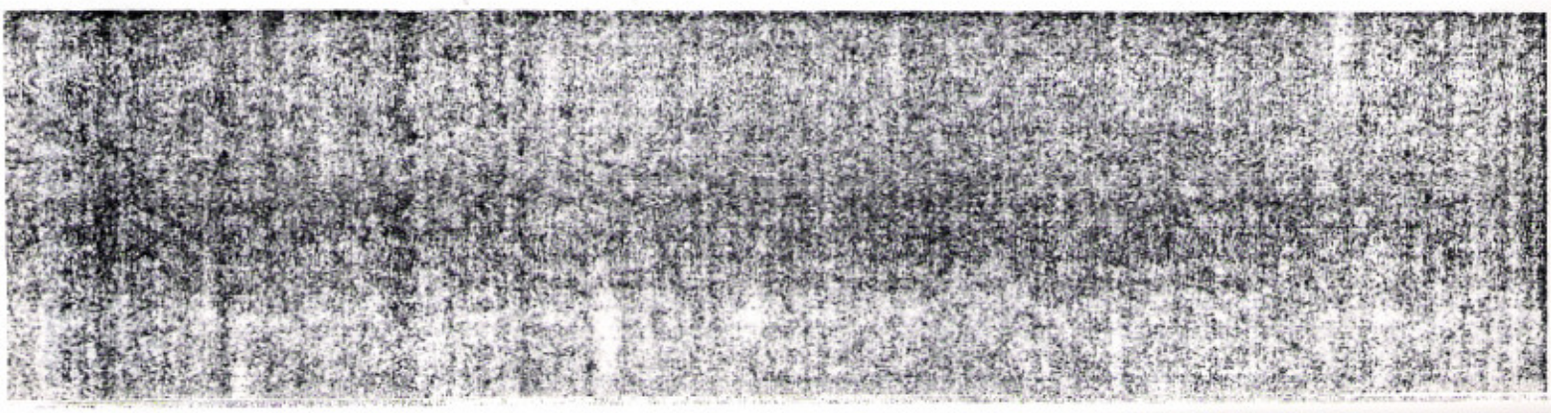
Held
April 19-20, 1984
University of Pittsburgh
Sponsored by
The Department of Electrical Engineering
School of Engineering
University of Pittsburgh
In Cooperation with
The Pittsburgh Sections
of the
Institute of Electrical and Electronic Engineers
and
The Instrument Society of America
and
The Systems, Man and Cybernetics Society
The Society for Computer Simulation
The International Association for Mathematics
and Computers in Simulation (formerly AICA)

Edited by

William G. Vogt
Marlin H. Mickle

Associate Editors

S. Abel
M.F. Aburdene
L. Anselin
N. Arunasalam
K.K. Bagchi
C.I. Bartfeld
M. Bayoumi
S.D. Bedrosian
J. Bezdek
B.B. Bhaumik
M. Bosetti
D. Boussalis
C.G. Brockus
C.L. Brooks
F. Calzonetti
R.E. Carter
E. Casetti
C.M. Chen
C.T. Chen
F.D. Chichester
L.W. Cornwell
F.C. Crosby
P.K. Das
G.J. Demko
D.R. Drew
M.A. Fahim
P. Fisher
M.S. Fogarty
R.H. Foglesong
G. Garduno
H.L. Gauthier
H.E. Ghobary
R. Giest
A.A. Girgis
R.G. Golledge
M.F. Goodchild
N.N. Greenbaun
R. Hanham
M.T. Hanna
A. Hargrove
K.E. Haynes
D. Hertsgaard
G.J. Hewings
R.C. Hoover
J.C. Howard
L.B. Huang
W. Isard
K.G. Janardan
D.W. Jones
P. Kalata
L. King
M. Kokar
D.O. Koval
R.K. Krzyweic
P.V. Lasala
S.S. Lasala
T.S. Lee
P. Luebbe
R.A. Maggio
P.C. Mann
B. Mashaw
R. H. McClure
E.M. McNertney
G.A. Mihram, Ph.D.
G.L. Moltyaner
G.W. Moser
N.S. N. Lam
A.A. Nadkarni
A. Nauda
M. O'Kelly
J. Odland
M. Rafiquzzman
F. Reifler
V.B. Robinson
A. Rose
B.M. Rubin
W.K. Rudloff
H.A. Ryaciotaki-Boussalis
V. Selman
M. Shanblatt
S. Shelmani
C. Shiek
D.W. Smith
A.H. Stahler
S. Tarasiewicz
G.I. Thrall
T.K. Tran
C.R. Waits
R.H. Wang
C.E. Wells
C. Werner
J.O. Wheeler
M. William
M. Williams
C.J. Willmott
R.V. Wong
S.H. Young
N.S. Yung
X. Zhixang



Library of Congress Catalog Card Number 73-85004

ISBN Part 4 0-87664-828-6

Distributed by

INSTRUMENT SOCIETY OF AMERICA
67 Alexander Drive
P.O. Box 12277
Research Triangle Park, N.C. 27709
Phone: (919) 549-8411
Telex: 802 540 ISA DURM

Printed in U.S.A.
By
Technical Communication Services

A

William

Paul V.

N. A
Mic
C. M
C. T
P. F

An international co
success to many p
helped would likely
some who have bee

First, our special th
Cushing, for help i

The editors wish to
Department of Ele
Conference in man

The School of Eng
and Simulation Co
Dean Max Williams

A special thanks a
of whom helped w
and Barb Dippolo f

A final thanks goes
audio-visual equipr
those many situatio
Conference, but ea

LEARNING ALGORITHM FOR PRIORITY ASSIGNMENT IN A QUEUING SYSTEM WITH UNKNOWN CHARACTERISTICS

Mohammad Reza Meybodi
Computer Science Department
Western Michigan University
Kalamazoo, Michigan

ABSTRACT

An application of learning algorithm of the type studied in the context of learning automata 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 and three-class M/M/1 queuing system is given. 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. A number of interesting simulations are also presented.

INTRODUCTION

An important subject in queuing theory is that of priorities. In a priority queuing system there a 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 the 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 of present is always taken for service prior to the unit of 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.

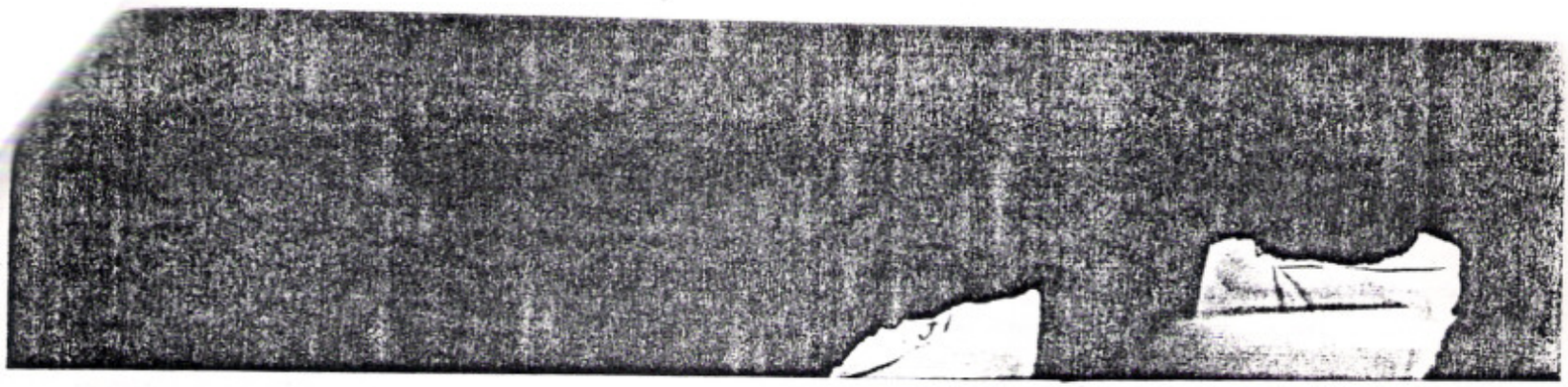
Below, we briefly summarize some major results on priority queuing that are pertinent to our present study.

It is shown [1] 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 [2] 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"

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

will minimize the average waiting cost. In (1) $\mu_1 = \frac{1}{\mu_i^{-1}}$, 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 [3]. 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. Nova and Ponamarenko [4] 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 [5] 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 [6] 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.

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. [5] 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.

Meybodi and Lakshmivarahan [10] used a class of learning algorithm very similar to those that are used in Mathematical Psychology [7, 11, 12] and formally showed that a randomized choice of classes for service would indeed asymptotically coverge to the classical priority assignment rule with a probability as close to unity as desired. That is the proposed system will asymtotically assign the higher priority to the class with the shortest average service time. They modelled this in the framework of single server pricrity queuing system when there are only two classes of jobs and preemption of the service is not allowed. In this paper, the result in [10] is extended to more than two classes. Number of interesting simulations for the case of two classes in presented.

THE MODEL

The priority assignment queuing system considered in this paper is represented schematically in Figure 1. There are m 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 m queues from a group of m different classes of infinite population according the independent Poisson process with constant rate λ_i (>0) i=1, 2, ..., m. The service time requirement for jobs in different classes 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, ..., m. The following assumption is very crucial in our analysis.

(A1) The parameters λ_i and μ_i^{-1}, i=1, 2, ..., m are not known. Without loss of generality let $\mu_1^{-1} < \mu_2^{-1} < \mu_3^{-1} < \ldots < \mu_m^{-1}$.

Let γ^i be the random service time of a typical job from i=1, 2, ..., m. Consider a stage when k_i jobs from class i have been served. Then, $k=k_1+k_2+\ldots+k_m$ is the total number of jobs served by the system. Define a dynamic threshold

$$T(k)=1/m[\sum_{i=1}^{m} T^{(i)}(k_i)] \quad \text{where} \quad T$$

and $\gamma^{(i)}(j)$ is the random service time for the average service time taken over both c

it is evident that for large $k_1, k_2, \ldots k$ is small with a very high probability.

It is assumed that class i has a service t

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

Given that k jobs have bee

$$x(k)=1 \quad \text{if} \quad \gamma^{(i)}(k+1) \le T(k)$$
$$x(k)=0 \quad \text{otherwise.}$$

Clearly

$$d_i(k)=\text{Prob}[x(k)=1]=\int_0^{T(k)} f_i(t)dt \quad i=$$

Since T(k) is a random variable, so is d_i probability that $(k+1)^{th}$ job will have it from class i. Define $d(k)=(d_1(k), d_2(k),$

for i=1, 2, ..., m, and all k≥1. Further easily seen that $d_1(k)>d_2(k)>d_3(k)>\ldots>d_m$ follows that $\eta=\sup_k(d_j(k)/d_1(k))<1$, j=1,2,

Also for large k_i, i=1, 2, ..., m, the di high probability.

Remark 1: Dynamic threshold at any time to two regions. Depending on in which re takes one of the values 0 or 1 (0 is call threshold before the operation of the sys ence, process of partitioning will be do tion of the system.

3. Learning Algorithm

3.1 Case 1: Two classes

Let $p_j(k)$ be probability with which the l service from class j. Define $p(k) = (p_1$ Initially, $p_1(0) = p_2(0) = \frac{1}{2}$ with probab

With the above preliminaries, basic oper

Increase the probability of choosing the class has service time less than T(k) (r of choosing the i^{th} class if a job selec T(k) (resulted in failure). The increas tively. The probability of choosing ano equal to one. How these probabilities a of the type discussed in [7], when they algorithm. That is, updating algorithm update timing algorithm decides as to wh and 2 according to the probability vectc vices to the currently begin serviced jc update timing algorithms. It can be sho linear reward-inaction updating algorit desired class.

Updating algorithm: Linear reward-inac

$$P_i(k+1) = p_i(k) + a(1-p_i(k))\ J_s$$

where J_s denotes the indicator of the e

t is in the decreasing order
est value of $\mu_i c_i$. This policy
resume discipline providing ser-
the unit delay costs for different
creasing order of mean service time
should be noted that if the unit delay
ntially the same as total waiting
ll minimize the average waiting time.
system with k independent Poisson
nowed, by a numerical example, that the
queue size, and that the optimal
s well as on the arrival rates of all
omputed by linear programming.

eed to know the probabilistic charac-
arrival and service time distributions.
not be known in advance. An inter-
babilistic (arrival and service time)
wn in advance, how to go about devel-
ous classes. That is, how to design a
rked out directly without a priori
the system.

ed this problem and proposed a solution
omaton is used for each class. This
] in the context of modelling of col-
n are indirectly controlled by the ser-
n instance the class corresponding to
service.

ex transition structure which does not
iculty Varshavskii et al. [5] resorted
pproach. They showed through extensive
aton is chosen to be large to start
he priority assignment that would be
em to be known.

ing algorithm very similar to those that
formally showed that a randomized choice
overge to the classical priority assign-
sired. That is the proposed system will
s with the shortest average service time.
pricrity queuing system when there are
ce is not allowed. In this paper, the
Number of interesting simulations for

this paper is represented schematically
only one server, no preemption and queue
hin each class, the service is on a first
from a group of m different classes of
n process with constant rate
t for jobs in different classes are inde-

es are independent and are identically
n μ_i^{-1} , i=1, 2, ..., m. The following

are not known. Without loss

μ_m^{-1}.

from i=1, 2, ..., m. Consider a stage
$=k_1+k_2+\ldots+k_m$ is the total number of

old

$$T(k)=1/m[\sum_{i=1}^{m} T^{(i)}(k_i)] \qquad \text{where} \qquad T^{(i)}(k_i) = \frac{1}{k_i} \sum_{j=1}^{k_i} \gamma^{i}(j), \; T^{(i)}(0)=0$$

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 [17] it is evident that for large $k_1, k_2, \ldots k_m$, the difference between T(k) and $1/m \left[\sum_{i=1}^{m} \mu_i^{-1}\right]$ is small with a very high probability.

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, ..., define

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

Clearly

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

Since T(k) is a random variable, so is $d_i(k)$, i=1, 2, ..., m. 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 $d(k)=(d_1(k), d_2(k), \ldots, d_m(k))$. Notice $0<d_i(k)<1$ with probability one for i=1, 2, ..., m, and all $k\geq 1$. Further, since $\mu_1^{-1} < \mu_2^{-1} < \ldots \mu_m^{-1}$ (assumption A1) it is easily seen that $d_1(k)>d_2(k)>d_3(k)>\ldots>d_m(k)$ with probability one for all k. From this it follows that $\eta=\sup_k(d_j(k)/d_1(k))<1$, j=1,2,...,m.

Also for large k_i, i=1, 2, ..., m, the difference $d_i(k)$ and $1-e^{\frac{-\mu_i}{2}\sum_{i=1}^{m}\mu_i^{-1}}$ is small with a high probability.

Remark 1: 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 partitioning will be done only once and remain fixed throughout the operation of the system.

3. Learning Algorithm

3.1 Case 1: Two classes

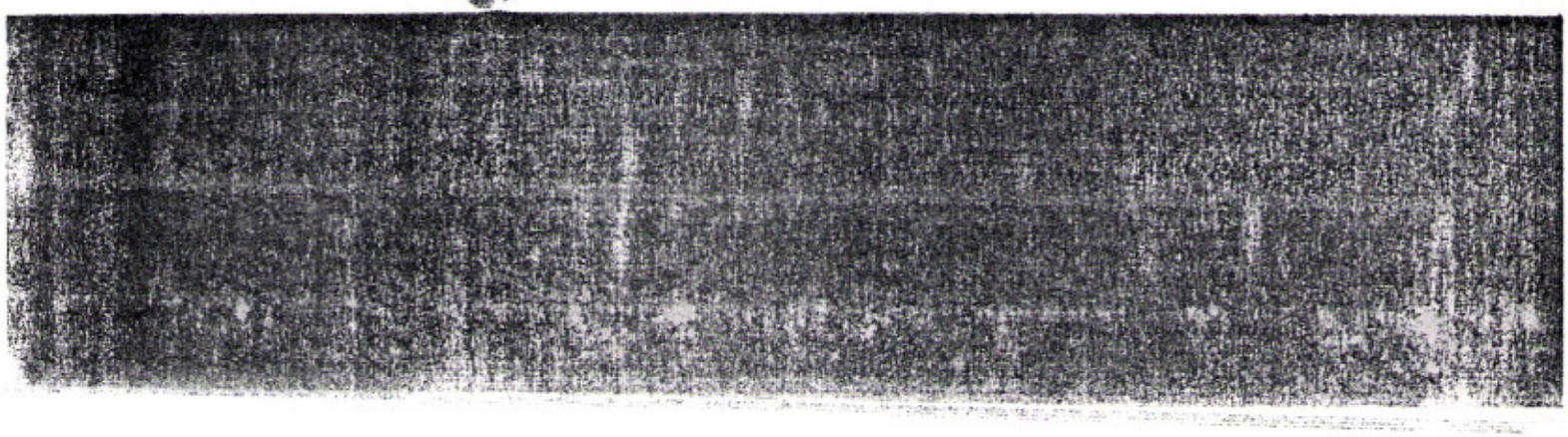
Let $p_j(k)$ be probability with which the DD in figure 2 decides to pick up the $(k+1)^{th}$ job for service from class j. Define $p(k) = (p_1(k), p_2(k))$ where $p_1(k) + p_2(k) = 1$ and $k \geq 0$. Initially, $p_1(0) = p_2(0) = \frac{1}{2}$ with probability one.

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 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 of the type discussed in [7], when they are updated is dictated by a timing updating algorithm. That is, updating algorithm decides how to 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 services to the currently begin serviced job is finished. In this paper we use two different update timing algorithms. It can he shown that any of these two algorithms together with the linear reward-inaction updating algorithm [7] will guarantee convergence of the system to the desired class.

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

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

where J_s denotes the indicator of the event and $0 < \vartheta < 1$ 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 [8] [9].

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 for 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

$p_i(k+1) = p_i(k) + a(1-p_i(k))$ if $\gamma^{(i)}(k+1) \geq T(k)$

$p_j(k+1) = p_j(k) - aP_j(k)$, $j \neq i$

and

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

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

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

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.

Theorem 1: Under the assumptions A1 and A2, if p(k) evolves according to the above learning algorithm (L1) then for every $\epsilon > 0$, there exists a $a^* > 0$ such that

$0 < a < a^*$ and for all $0 < p_i(0) < 1$

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

Proof: Refer to [10].

Remark 1: The above theorem asserts that the above learning algorithm (L1) would evolve to the now classical assignment rule wit

Remark 2: Obviously, there are other po date algorithm has been extensively stu and in the context of learning automata been applied to the two-person zero sum decentralized routing in telephone netwo

Now consider the update timing algorithm for service (as a sample realization fro using step 3, otherwise p(k+1) = p(k). according to this modified update rule i

$$\alpha_2(k) = \alpha_1(k)\, p_1(k)\, d_1(k) + \alpha_2(k)$$

Since $0<p_i(k)<1$ $i = 1,2$, it follows th fied rule p(k) is updated more frequentl algorithm L1 with step 2 replaced by upd gorithm L2. Admittedly, update timing it is not without further problems. We

(C1) $\alpha_1(k) > \alpha_2(k)$ for a

for theorem 1 to be true. $\alpha_1(k)$ to the

Theorem 2: Under the assumption (A1) a to the learning algorithm L2, then for and for all $0 < p_i < 1$

$$\text{Prob}\left[\lim_{k \to \infty}\right.$$

Remark 3: The process p(k), $k \geq 1$, co gorithm and timing update algorithm 1 a these absorbing barriers only one corre verges with a positive probability to e it is evident that there is a trade-off converging to the desired class. Thus, high probability converge to an undesir convergence for the system, but at the gence to the right class.

Remark 4: The priority system reported the probabilities of choosing class 1 a probability that one of the class has n time. Exogenous priority discipline is paper, and can be obtained by setting p $p_1(k) = 0$ and $p_2(k) = 1$ for all k depen first class or second class.

Extensive computer simulation has been of learning algorithm L1 and L2 under v selected set of these simulation result assertions put forward in this paper fo and $p_2(k)$ versus k for the various cho 3-a, to 3-f. A fact about learning al is that if α_i/α_j is greater than α_i/α_j will always converge to class i where

Remark 5: When the system uses learni are probabilities that p_i is increased $\alpha_1(k)$ may cause the system to converge vergence time. For example, if due to first class, $\alpha_1(k)$ is very small, the time. If learning algorithm L1 is use p_i is increased and $\alpha_1(k) \alpha_2(k) p_2(k)$ p_i. In this case low value of $\alpha_1(k)$ ma

ncreased

orithm which is strongly absolutely

a sample realization from distribution
ity of choosing that class provided an-
s is empty, choose a job from another

is a sample realization from distribution
ity of choosing that class. If the se-
ass and do not update the probability

a class is updated regardless of the stat-
hereas in timing algorithm 1, the status
ity of choosing the selected class. In
selection and is found to be empty, DD
ing the probability vector P. If both
a signal that a job is entered into one
) and also during updating dynamic

n of the system when the system uses
pdating algorithm.

job is chosen for service as a sample
nstant, if the queue of this chosen class
of the queue for service. Otherwise,
. Complete the service and note

se (in step 1 as a sample realization for
lass i as well as that of class j (≠i)
step 3. Otherwise p(k+1)=p(k) and go to

$_1)_{(k+1)} \leq T(k)$

).

f p(k) is unity. The following restric-
analysis.
the jobs in step 1 all other overhead
a class from p(k), checking whether the
of p(k) and T(k) are all instantaneous.
hese overhead operations. Obviously this
fast microprocessors, the overall over-

) evolves according to the above learning
ϑ^* such that

above learning algorithm (L1) would evolve

to the new classical assignment rule with a probability as close to unity as desired.

Remark 2: Obviously, there are other possible choices for update algorithms. The above update algorithm has been extensively studied in the context of Mathematical Psychology [11] and in the context of learning automata [12]. Very recently similar learning algorithms have been applied to the two-person zero sum games [13], two person decentralized team [14] and decentralized routing in telephone networks [15].

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

for theorem 1 to be true. $\rho_i(k)$ is the probability that class i is nonempty at time k.

Theorem 2: Under the assumption (A1) and (A2), and condition (C1), if p(k) evolves according to the learning algorithm L2, then for every $\epsilon > 0$, there exists a ϑ^* such that $0 < \vartheta^* < 1$ and for all $0 < \vartheta < \vartheta^*$ $\qquad \text{Prob}\left[\lim_{k \to \infty} p_1(k) = 1\right] \geq 1 - \epsilon.$

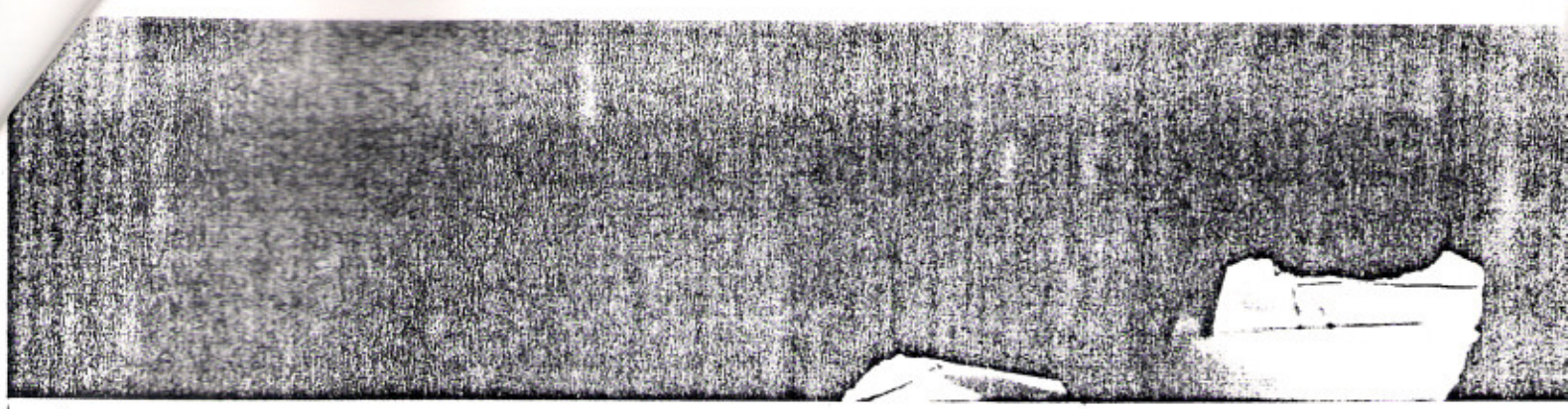
Remark 3: The process {p(k), k ≥ 1} corresponding to the linear reward-inaction update algorithm and timing update algorithm 1 and 2 is a process with two absorbing barriers. Out of these absorbing barriers only one corresponds to the desired class. In fact, the system converges with a positive probability to each one of these absorbing states. From theorem (1) it is evident that there is a 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 4: 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 priority discipline conceived in this paper, 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.

Extensive computer simulation has been carried out to investigate the convergence properties of learning algorithm L1 and 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 paper for learning algorithm L1 and L2. 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 Figures 3-a, to 3-f. A fact about learning algorithm L2 which is brought out from these simulation is that if λ_i/μ_i is greater than λ_j/μ_j (i≠j) and for relatively small value of ϑ, the system will always converge to class i where $\mu_i^{-1} = \min_{j \neq i} (\mu_j^{-1})$.

Remark 5: 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 convergence 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 4 illustrates the effect of interarrival time of jobs in the class with smaller service time on the convergence of the system.



Remark 6: 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 $p_1(k)$ and $p_2(k)$ which in turn will affect the convergence behavior of the system.

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

Remark 7: It was earlier said (Remark 4) 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

$$E[\rho_1(k+1) \mid p_1(k)] < \rho_1(k) \quad \text{and} \quad E[\rho_2(k+1) \mid p_2(k)] > \rho_1(k).$$

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

$$\lim_{k \to \infty} p_1(k) = 1 \text{ and } \lim_{k \to \infty} p_2(k) = 0$$

we have $\lim_{k \to \infty} \rho_1(k) = \rho_1^*$, $\lim_{k \to \infty} \rho_2(k) = \rho_2^*$.

Remark 8: 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

$$w_1^* = \frac{(\lambda_1/\mu_1^2) + (\lambda_2/\mu_2^2)}{(1 - \lambda_1/\mu_1)}, \quad \text{and} \quad W_2^* = \frac{(\lambda_1/\mu_1^2) + (\lambda_2/\mu_2^2)}{(1-\lambda_1/\mu_1)(1-\lambda_1/\mu_1-\lambda_2/\mu_2)} \quad \text{and also}$$

$$w_1^a < w_1^* \quad \text{and} \quad w_2^a > w_2^*.$$

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

$$\lim_{k \to \infty} p_1(k) = 1 \quad \text{and} \quad \lim_{k \to \infty} p_2(k) = 0.$$

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.

3.2 Case 2: Three classes

All the assumptions which we made for the previous case are also made for this case. Without loss of generality, assume

$$\mu_1^{-1} < \mu_2^{-1} < \mu_3^{-1} \qquad (A4)$$

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 $p_1(0) = p_2(0) = p_3(0) =$ [illegible] with probability one.

2. Probability vector $p^i(k) = (p_r^{(i}$ for all i. $p_j^{(i)}(k)$ is the prob class is empty. Initially $p_r^{(i)}$

3. $T(k) = 1/3 \sum_{i=1}^{3} T^{(i)}(k_i)$

and

$T_{(i)}(k) = 1/2 [T_{(i)}^{(r)}(k_r) + T_{(i)}^{(s}$

where

$$T^{(i)}(k_i) = \frac{1}{k_i} \sum_{k=1}^{k_i} \gamma^{(i)}(k)$$

and

$$T_{(i)}^{(q)}(k_r) = \frac{1}{k_q} \sum_{k=1}^{k_q} \gamma^{(q)}(k)$$

where k_i is the number of jobs actual service time of the th

Learning Algorithm GL1

Let i be the class from which $(k+1)^{th}$ jo $p(k)$ for $k \geq 0$.

1. If the queue of this chosen class is { choose a job from in front of t $\gamma^{(i)}(k+1)$.

The update timing and update a

If the queues corresponding to $p(k)$ using the following update

$$p_i(k+1) = p_i(k) + \theta(1-p_i(k$$

$$p_j(k+1) = p_j(k) - \theta p_j(k)$$

and

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

else

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

Update T(k) and $T_{(j)}(k)$ j

goto 1 }

else

{ Let n be the class from wh realization from $p^{(i)}(k)$.

If the queue of this chosen cl

{ select the job from in fr service and note $\gamma^{(n)}(k+1)$

The update timing and upd

At the instant when class both the class n as well update $p^{(i)}(k)$ using the

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

$$p_m^{(i)}(k+1) = p_m^{(i)}(k) - \theta$$

and

f the system. If changes in
$= d_2(k)$), the system converges
$\neq \lambda_2^{-1}$) does not always imply a
$(k+1) = d_2(k)$) depends not only on
hanging λ_1^{-1} and λ_2^{-1} may also change
ence behavior of the system.

vior of ρ_i, i = 1,2, during the opera-
ogenous priority scheme corresponds to
$p_2(k) = 1$ for all k], depending on
or second class. Let ρ_1^* be the
and $p_2(k) = 0$ for all k]. Now consider
here a is a positive constant less than
pty when $p_1(k)$ is fixed to constant a
that $\rho_1^a > \rho_1^*$ and $\rho_2^a < \rho_2^*$.
choosing class 1 (class 2) from 1 (0)
nonempty will increase (decrease). In
$\{p_2(k)\}$ is a super-martingale, we have
$p_2(k)] > \rho_1(k)$.
tingale, respectively. And since

for the exogenous priority scheme and
espectively, then we have

$$\frac{\lambda_2/\lambda_2^2)}{\lambda_1)(1-\lambda_1/\mu_1-\lambda_2/\mu_2)} \quad \text{and also}$$

nd $\{p_2(k)\}$ is a supermartingale, we have
$W_2(k)$, where $W_i(k)$, i = 1,2, is the
mit $W_i(k) = W_i^*$. Since

d that the speed of convergence of
the speed of convergence of the learning

se are also made for this case. With-

k)), where $\sum_{i=1}^{3} p_i = 1$. $p_j(k)$ is the
up the $(k+1)^{th}$ job for service from
Initially $p_1(0) = p_2(0) = p_3(0) = 1/3$

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 k. $p_j^{(i)}(k)$ 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. $T(k) = 1/3 \sum_{i=1}^{3} T_{(i)}^{(i)}(k_i)$

and

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

where

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

and

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

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 GLI

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.

$$p_i(k+1) = p_i(k) + \theta(1-p_i(k)) \quad \Big\} \text{ if } \gamma^{(i)}(k+1) \leq T(k)$$
$$p_j(k+1) = p_j(k) - \theta p_j(k) \quad j \neq 1$$

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 $T_{(j)}(k)$ $j \neq 1$ 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:

$$p_n^{(i)}(k+1) = p_n^{(i)}(k) + \theta(1-p_n^{(i)}(k)) \quad \Big\} \text{ if } \gamma^{(n)}(k+1) \leq T(k)$$
$$p_m^{(i)}(k+1) = p_m^{(i)}(k) - \theta p_m^{(i)}(k) \qquad m \neq n \neq i$$

and

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

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

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

b. Prob [$\lim_{k \to \infty} p_i^{(r)}(k)$] $\geq 1-\epsilon$, $r=1,2,3$ if $\omega_i^{-1} = \min_{\substack{j \neq i \\ j \neq r}} (\omega_j^{-1})$.

Proof: Proof is immediate from theorem 1.

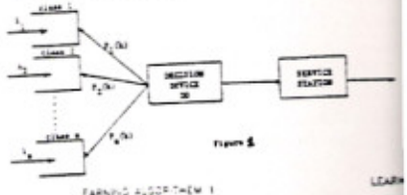
Remark 9: While the result of this paper can be generalized to priority assignment system with any number of classes, the learning algorithm needed becomes more complex and messy as the number of classes increases. Due to this problem, a more efficient algorithm (timing and update algorithm) must be searched for.

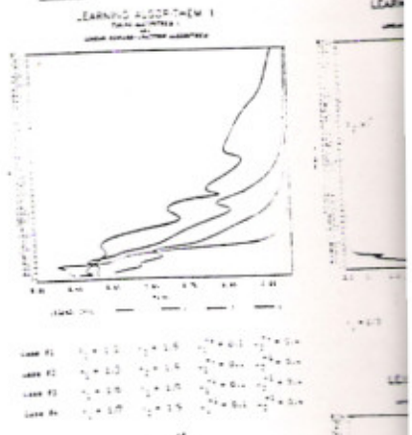
4. Conclusions

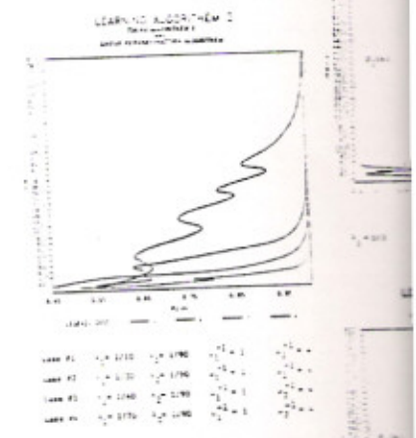
An application of variable structure learning automata to priority assignment in a queuing system with unknown parameters has been studied. As a consequence, a learning algorithm for assigning priorities is presented.

REFERENCES

1. Kleinrock, L., Queueing System. VolumeII: Computer Applications, John Wiley, New York, 1976.
2. Cox, D.R. and Smith, W.L., Queues. Methven, London, 1961.
3. Jaiswal, N., Priority Queues. Academic Press, New York, 1968.
4. Nova, 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.
5. 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 Informatisii, Vol. 4, pp. 73-76, 1968.
6. Tsetlin, M.L., Automation Theory and Modelling of Biological System. Academic Press,1973.
7. Lakshmivarahan, S., Learning Algorithms: Theory and Applications. Springer Verlag, New York, 1981.
8. Meybodi, M.R. and Lakshmivarahan, S., "E-optimality of a General Class of Learning Algorithms." Information Science, Vol. 28, pp. 1-20, 1982.
9. Meybodi, M.R. and Lakshmivarahan, S. "On a class of Learning Algorithm with Symmetric Behavior Under Success and Failure." Lecture Notes in Statistics, Springer Verlag, 1982.
10. Meybodi, M.R. and Lakshmivarahan, S. "A Learning Approach to Priority Assignment in a Two class M/M/1 Queueing System with Unknown Parameters." Proceeding of the Third Yale Workshop on Applications of Adaptive System Theory, New Haven, Connecticut, June 15-17, 1983.
11. Norman, M.F., "Some Convergence Theorems for Stochastic Learning Models with Distance Diminishing Operators." Journal of Mathematical Psychology., Vol. 5, pp.61-101, 1968.
12. Narendra, K.S. and Thathachar, M.A.L., "Learning Automata - A Survey," IEEE Trans. System, Man and Cybernetics, Vol. 4, pp. 323-334, 1974.
13. Lakshmivarahan, S. and Narendra, K.S., Learning Algorithms for Two Person Zero Sum Stochastic Games with Incomplete Information." Mathematics of Operations Research., Vol. 7, No. 4, 1981.
14. Lakshmivarahan, S., "Two Person Decentralized Team with Incomplete Information." Applied Mathematics and Computation, Vol. 8, Pp. 51-78, 1981.
15. Srikantakumar, P.R. and Narendra, K.S. works." SIAM Journal on Control and O
16. Baba, N. and Sawaragi, Y. "On Learnin Stationary Environment." IEEE Transac pp. 273-275, 1975.
17. Mood, A.M., Graybill, F.A. and boes, D McGraw Hill, 1974.







(k+1) ≥ T(k)

...(m)(k+1), go to 1

...lass m (m ≠ n ≠ i) complete the service and
...) j ≠ m using ...(m)(k+1), goto 1

If the system evolves according to the
ery ..., ...$_r$>0, r = 1,2,3, there exists
... < 1,r = 1,2,3, and for all 0 < ... < ...*,

...$^{-1}_i$ = min$_{j≠i}$ (...$^{-1}_j$) ,

r=1,2,3

if ...$^{-1}_i$ = min$_{j≠i, j≠r}$ (...$^{-1}_j$).

...e generalized to priority assignment system
...thm needed becomes more complex and messy as
...roblem, a more efficient algorithm (timing

...automata to priority assignment in a queuing
...l. As a consequence, a learning algorithm for

...NCES

... Computer Applications, John Wiley, New York,

...n, London, 1961.
...ess, New York, 1968.
...Optimal Assignment of Priorities, Depending on
...of Waiting places." Engr. Cybern.,Vol.12,No.5,

...rlin, M.L., "Priority Organization in Queuing
...our of Automata." Problemy Pereduchi

...ling of Biological System. Academic Press,1973.
...Theory and Applications. Springer Verlag,

...ptimality of a General Class of Learning
..., pp. 1-20, 1982.
... class of Learning Algorithm with Symmetric
...ure Notes in Statistics, Springer Verlag, 1982.
...arning Approach to Priority Assignment in a
...wn Parameters." Proceeding of the Third Yale
...m Theory, New Haven, Connecticut, June 15-17,

...or Stochastic Learning Models with Distance
...atical Psychology., Vol. 5, pp.61-101, 1968.
...arning Automata - A Survey," IEEE Trans. Sys-
...334, 1974.
...rning Algorithms for Two Person Zero Sum
...on." Mathematics of Operations Research, Vol.

...ted Team with Incomplete Information." Applied
...-78, 1981.

15. 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.
16. Baba, N. and Sawaragi, Y. "On Learning Behavior of Stochastic Automata Under Non-Stationary Environment." IEEE Transactions on Systems, Man and Cybernetics, Vol. ..., pp. 273-275, 1975.
17. Mood, A.M., Graybill, F.A. and Boes, D.D., Introduction to the Theory of Statistics, McGraw Hill, 1974.

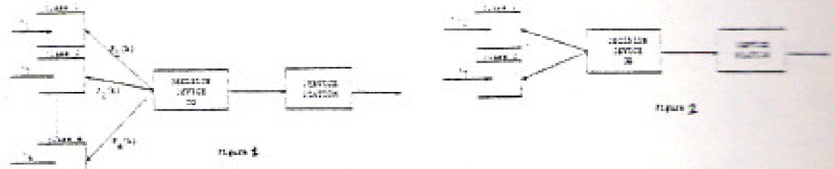


Figure 1 Figure 2