Technical Report No. 170

06868-1-T

THE OPTIMAL CONTROL OF QUEUES, WITH APPLICATIONS TO COMPUTER SYSTEMS

by

Dennis W. Fife

Approved by: Thomas W. Butler Jr.
Thomas W. Butler, Jr.

for

COOLEY ELECTRONICS LABORATORY
Department of Electrical Engineering
The University of Michigan
Ann Arbor

Contract No. AF 30(602)-3553
United States Air Force
Rome Air Development Center
Rome, New York

and

Grant No. GK-176
National Science Foundation
Washington, D. C.

October 1965

This report was also a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in The University of Michigan, 1965.

ACKNOWLEDGMENTS

This research effort is the culmination of over seven years of graduate study and research at The University of Michigan. During this time, I have benefited from many associations, especially with the members of my doctoral committee. For this, and their helpful suggestions on this report, I am most appreciative. Professor Chuang deserves special recognition and gratitude for it was his early work which led me to investigate this subject matter, and he has been a constant source of advice and encouragement throughout the course of this work. To my other research associates I also express my thanks.

The rough draft of this dissertation was typed by a number of the girls from the Cooley Electronics Laboratory, whom I thank, but special mention should be accorded to Mrs. Joan Carstens and Miss Carol Wolanski. Ronald Kilgren did most of the drafting, and Ralph Olsen was very helpful in plotting some of the graphs. The final draft typing and printing has been handled by George Prosser and staff of the Office of Research Administration. I am grateful to them for a fine job done under pressure of time.

I am also indebted to the United States Air Force and the National Science Foundation for financial support under contracts AF 30(602)-2661, AF 30(602)-3553, and NSF grants GP-1381 and GK-176.

Finally, the unfailing encouragement of my wife, Joyce, has been of immeasurable value to me in all of my graduate study, including this research.

TABLE OF CONTENTS

Page

CHAPTER

TABLE OF CONTENTS (Concluded)

LIST OF TABLES

LIST OF TABLES (Concluded)

LIST OF FIGURES

LIST OF FIGURES (Concluded)

LIST OF SYMBOLS

Symbol	Meaning	Page Where First Used
α	A factor for discounting the one-step return.	54
α_i^k	Partial one-transition return attributable to new arrivals in the transition interval.	180
δ_i^k	Partial one-transition return attributable to the swapping and set-up time for a job.	180
$\delta_i(k)$	The test quantity of the Howard algorithm, for state i and action k.	126
γ_i^k	Total weight per unit of time-in-system for jobs in system at beginning of a transition, excluding the job to be executed.	180
$\Delta^\ell \pi$	A transformation of the policy π which advances the time sequence of decision by ℓ time steps.	87
$\eta(k\|i)$	A probability governing random selection of action k in state i.	66
$\eta_n(k\|i)$	The action selection probability specified for time n by a policy.	67
θ	Probability that a new arrival is a program requiring a set-up time.	155
λ	Reciprocal of the mean value of t_a.	27
μ_1,μ_2,μ_3	Reciprocals of the mean values of the phases in a three-phase hyperexponential distribution.	157
ν_i^k	Expectation of duration of a transition from state i with action k.	48
ξ_i	True difference in the test quantities in state i for some pair of policies.	130
ξ_i^k	Partial one-transition return attributable to execution time of a job.	180

LIST OF SYMBOLS (Continued)

Symbol	Meaning	Page Where First Used
π	An admissible control policy for the decision process.	67
π_d	A uniformly optimal stationary policy for the discounting optimization problem.	96
π_i	Stationary probability of state i of a Markov chain.	31
π_ν	The ν^{th} element of a sequence of policies from the set P.	83
$\sigma_1, \sigma_2, \sigma_3$	Probabilities in the definition of a three-phase hyperexponential distribution.	157
τ	The integral duration of a transition.	47
$\psi_o(j,v)$	Probability that the decision process begins in state j at time v.	73
$\psi_{ij}(n)$	Probability that the system enters state j at time n, conditional on initial state i for the process.	99
$\psi_n(j)$	Briefer notation for $\psi_n(j,o)$.	77
$\psi_n(j,v)$	Joint probability that $Z_n = j$ and $W_n = v$.	76
$\psi_n(j,v\|s_o,\pi)$	The distribution $\psi_n(j,v)$ with its dependence on s_o and π emphasized.	83
Φ	Probability of a job completion in an arbitrary time step.	189
b_i^k	The expectation of total return over one transition from state i with action k.	51
$d_i(e,\hat{e})$	The distance between elements e, $\hat{e}$ of E_i.	70
$d_P(\pi,\hat{\pi})$	The distance between elements π, $\hat{\pi}$ of the set P.	71
$d_S(s,\hat{s})$	The metric on the set S.	76

Symbol	Meaning	Page Where First Used
e_2	Accomplished execution time of jobs in Q_2.	150
e_3	Minimum accomplished execution time for a job in Q_3.	150
e_i	A vector of probabilities $\eta(k\|i)$ for state i.	66
$f_i^k(\tau)$	The unconditional probability distribution on the duration of a transition from state i with action k.	184
f_{ij}	Probability of reaching state j from state i.	106
$f_{ij}^k(\tau)$	Probability of duration τ for a transition between states i and j with action k.	47
f_n	Probability that a job is completed in the n^{th} time step of its execution.	171
$f_1(h_{n-1})$	Operator notation for the transformation which yields s_n as a function of s_{n-1} and a decision h_{n-1}.	77
g	The gain of the decision process with a stationary deterministic policy.	111
$g_\ell(i,\pi)$	The greatest lower bound on the subsequential limits of $A_T(i,\pi)$ for T = 1,2,3,... .	53
$g_\ell(s_o,\pi)$	The greatest lower bound on the subsequential limits of $V_T(s_o,\pi)/T$ for T = 1,2,... .	80
h	A set of vectors e_i, i = 1,2,...,N.	67
i	Integer index for a state of the decision process.	44
k	Integer index for a control action.	45
k_i	The index for the action to be taken in state i for a stationary deterministic policy.	101

Symbol	Meaning	Page Where First Used
ℓ	State variable giving the status of the user memory area.	161
ℓ_{jj}	The mean recurrence time of state j.	106
n_1, n_2, n_3	State variables giving the number of jobs in Q_1, Q_3, Q_3 respectively.	160
p_{ij}^{k}	The probability of passing from state i to state j in one transition with action k.	47
q	The maximum amount of execution time allowed on each pass for a job in Q_3.	150
$q_{ij}(\tau \mid k)$	Joint probability that the system passes from state i to j in one transition of τ units duration with action k.	47
$q_{ij}(\tau \mid h_n)$	Joint probability that a transition is from state i to state j with duration τ, when the decision h_n governs the transition.	77
$r_{ij}^{k}(m \mid \tau)$	The return for the m^{th} step of a transition having duration τ, between states i and j, with action k.	49
s	The ECP time involved in swapping two jobs between main memory and drum memory.	155
s_o	The element of S corresponding to the initial state distribution $\{\psi_o(j,v)\}$.	76
s_n	The element of the set S corresponding to the distribution $\{\psi_n(j,v)\}$.	76
t_a	Arrival interval for a single source or console.	25
t_e	Required execution time for a job in the computer model.	154

Symbol	Meaning	Page Where First Used
$u(m\|i,k)$	The expectation of the one-step return received from a transition from state i with action k, m steps after it begins.	49
$u(m\|i,h_n)$	The expectation of the one-step return received from a transition from state i m steps after it begins, where the decision h_n governs the action taken at its beginning.	79
v_i	A constant for state i, known as a "relative value," involved in the asymptotic behavior of $V_T(i)$ with an ergodic policy.	121
w_1, w_2, w_3	Expectation of the weight per unit-of-time in systems for jobs in Q_1, Q_2, Q_3 respectively.	181
x_i	Computed value equal to v_i+g within limits of computational errors.	129
A_i	The set of integers $1,2,\ldots,K_i$ which is the set of control actions for state i.	45
$A_T(i,\pi)$	The average per unit time expectation of return over T steps, for policy π and initial state i.	53
CPU	Abbreviation for Central Processing Unit.	13
$D_\alpha(i,\pi)$	The expectation of total return for a decision process with discount factor α, policy π, initial state i.	54
$D_\alpha(j,v,\pi)$	The expectation of total discounted return from the decision process with policy π, conditional on the process beginning at time v in state j.	93
$D^*_\alpha(j,v)$	The expectation of total discounted return from an optimum decision process, conditional on the process beginning at time v in state j.	94

Symbol	Meaning	Page Where First Used
$D_\alpha(s_o,\pi)$	The expectation of total discounted return from the decision process with policy π and initial distribution s_o.	80
$D^*_\alpha(s_o)$	The expectation of total discounted return from an optimum decision process with initial distribution s_o.	87
ECP	Abbreviation for Executive Control Program.	14
E_i	The set of all vectors e_i.	67
$F(T)$	The probability distribution function on execution time, $Pr(t_e \leq T)$.	154
$G_{ij}(T)$	The probability of reaching state j from state i for the first time in T steps or less.	105
H	The set of all N-tuples h.	69
K_i	The number of possible control actions in state i.	45
L	Expectation of number of jobs in system at an arbitrary time.	187
$M_{ij}(T)$	The expectation of $U_j(T)$, conditional on the initial state i.	105
N	Total number of states in the decision process.	44
$U_j(T)$	The number of times state j is entered in the first T steps, excluding the initial state.	105
P	The set of all admissible policies π.	69
P'	The set of all stationary deterministic policies.	102
Q_1,Q_2,Q_3	Notation for queue levels 1, 2, 3 respectively.	152

Symbols	Meaning	Page Where First Used
$R_n(i,\pi)$	The expectation of return at the n^{th} step of the decision process with policy π, conditional on an initial state i.	52
$R_n(s_o,\pi)$	The expectation of the return for the n^{th} step of the decision process with policy π and initial distribution s_o.	78
S	The set of all joint probability distributions $\{\psi(j,v);\ j = 1,2,\ldots N;\ v = 0,1,2,\ldots\}$.	75
S	Set-up time for relocation of a job program.	148
T_e	Mean value of execution time t_e.	170
T_i^k	Expectation of the time interval between an arrival from a user and the end of the transition in progress.	184
T_u	Mean time for a user to generate a new request, equal to $1/\lambda$.	154
$V_T(i,\pi)$	The expectation of total return for T steps of the decision process with policy π, conditional on initial state i.	52
$V_T(s_o,\pi)$	The expectation of total return over T steps of the decision process with policy π and initial distribution s_o.	78
$W(t_e)$	A function giving a value for weighting the time-in-system of a job requiring execution time t_e.	177
W_i	The expectation of time a job spends waiting in $Q_i(i = 1,2,3)$ including swapping and set-up time.	189
W_n	The random time interval measured forward from time n to the beginning of a transition.	72
X_i	The number of jobs present in system at beginning of a transition from state i.	183

LIST OF SYMBOLS (Concluded)

Symbol	Meaning	Page Where First Used
Y_n	The state of the system at time n.	55
Z_n	The state entered at the first transition beginning at time n or later.	72

ABSTRACT

An important element in the operation of a system with queues is the procedure for controlling the use of system resources in processing the random demands for service. Many systems, especially those employing general purpose computers, admit a wide class of economically practical procedures. It is usually infeasible to use classical techniques of analysis to study every admissible control policy. The system engineer instead requires an optimization technique which will systematically isolate an optimal policy within the admissible class. This research pursues the theoretical development and application of Markov sequential control processes for this purpose.

Major effort is devoted to a discrete-time Semi-Markov decision process, and the optimization of performance return from this process over infinite time. Proceeding from an appropriate definition of optimization, it is established that a particularly simple optimal control policy exists. This policy is stationary and deterministic, and optimal for any initial condition on the process. This result simplifies both the computation of the optimal control and its implementation in physical systems.

A timely example for the application of the technique is provided by the scheduling control problem of multiple-access or time-shared computers. A Semi-Markov model is developed for a computer serving four remote consoles with three queueing levels for user jobs. Swap time and the set-up time for initial program loading are included, and rather general execution time distributions are treated. Two optimization criteria aimed at minimizing response time to commands are proposed. The computational method devised by Howard and Jewell is used to determine optimal policies. The execution intervals assigned to jobs are also studied.

Results for the computer model show that optimal policies are predominantly priority policies, although a much wider class is admissible. Also, a need is indicated for more rapid preemption of low priority jobs than used in existing time-shared systems. The mean response time for given execution time is compared for the optimal system and the control procedures now being used.

The practicality of the optimization results establish that this new technique can profitably be used in the design of stochastic service systems.

CHAPTER I

INTRODUCTION

1.1 ENGINEERING STUDY OF QUEUES

Since the early days of the telephone industry, electrical engineers have had a basic interest in the design and performance of systems with queues, or what have recently been called "stochastic service systems" [1]. A physical system may be called a service system if it exists to perform a function or a set of functions in response to discrete demands presented to it in the course of time from an external source. The functions performed are generally called "services" and the demands presented are called "arrivals."

Everyone is familiar, of course, with the service system represented by a theatre ticket booth. A telephone switching center is also a service system, in which the arrivals are subscriber's calls and the service performed is to establish and hold a communication path to the called party. A less apparent example is a nuclear radiation counter, a service system in which the arrivals are nuclear decay particles, and the service performed is to detect and record the occurrence of each particle. During the last decade, another class of service systems has evolved which will have a widening role in our technology and which are of specific interest in this study. These so-called on-line, remote access, or multiple-access computer systems.

The arrivals to these service systems are commands or digital data communicated directly from a remote device; the service performed is the execution of a computational process.

A service system generally consists of a number of different types of service facilities. For economic or physical reasons, the facilities in almost all systems are limited, so that only a certain number of arrivals can be accomodated at one time. To avoid turning away demands which occur when the system is fully occupied, many systems have a means for holding arrivals until facilities become available. The waiting arrivals then constitute a queue.

Each arrival to a service system will require the use of various facilities for various periods of time before its service is completed. In a majority of the physical service systems one can imagine, moreover, there is a significant degree of uncertainty in regard to when arrivals will actually occur, and the time durations and facilities which will be required to service each. For example, neither the times at which subscribers will initiate telephone calls nor the duration of any particular call can be known beforehand to the telephone engineer. Indeed, in many service systems, including the telephone system, it is both convenient and proper to consider the occurrence of an arrival and the end of a service interval as randomly occurring events. It is for this reason that the term "stochastic service system" is applied.

The performance of a stochastic service system is degraded by the queueing delays which arrivals encounter in being serviced, and by the occurrence of a busy condition during which arrivals are turned away. The random nature of events during system operation demands that performance be described statistically, i.e., in terms of probabilities and average values. A prediction of performance which includes the stochastic effects is generally difficult, but often of such importance as to warrant extensive mathematical analysis. As a result, a large body of knowledge on stochastic processes and their application as models of physical service systems has developed in the last half century. This literature comprises the so-called "theory of queues." References [1] through [10] are examples of the more important works.

The theory of queues has a long and honorable history of providing solutions to practical problems in electrical engineering, as well as other fields. The theory had its origin in telephony in the early 1900's, and its early development was largely due to A. K. Erlang [2]. An early problem treated by Erlang was estimating the number of trunks required between toll centers in order to provide a satisfactory grade of service. The grade of service is the probability that a caller encounters a busy signal because all trunks are in use. Erlang's result for the grade of service, now known as the Erlang loss probability, is still in use [34]. Fry [5], Riordan [1], and Syski [9] give comprehen-

sive coverage of the queueing problems in telephony. More practical aspects can be studied through papers such as Wilkinson's [34] in the Bell System Technical Journal.

The increasing volume of long distance data transmission and message switching in recent years has given new applications for queueing analysis beyond the problems provided by telephone communication. For example, Otterman [35] has studied the grade of service when store-and-forward data transmission has shared access to trunks along with voice communication. Haenschke [36] has used queueing models to investigate several switching center designs for data transmission. The control of errors in digital data and resulting queueing problems have been of concern [37]. The study of routing procedures and priority schemes for ordering the transmission of messages over a communication network has recently produced some results of more general interest [38].

The use of computer control for high-volume message switching has brought queueing analysis into the design of computers, see [39] for example. Aside from this computer application, the trend in modern computer design generally is toward asynchronous, independently operating subsystems, and queueing studies are now seen valuable for general purpose computer design [40-44].

Surprisingly, the theory of queues also has application to devices for recording nuclear radiation, see [45] and [10], p. 205ff. This is because radiation detection elements are limited physically by a dead

time occurring after each particle detected. This dead time corresponds to a "service time" for each recorded particle. The problem for analysis is to determine the maximum average counting rate possible with random particle arrivals.

Beyond electrical engineering applications the mathematical study of queues has made important contributions to air traffic control, highway traffic control, organization of hospital facilities, assembly line manufacturing, and many other fields. Saaty [8] and Morse [7] give some indication of the additional breadth of applications.

To a large degree, the applications of the theory of queues have centered on determining the extent of the service facilities required to provide a specified quality of performance. But an equally important aspect to system design, especially when there are queues, is the overall operational control procedure for the service system. The operational control procedure consists of those actions which govern and coordinate the utilization of the system's service facilities by the arrivals. Basically it involves two kinds of decisions which occur sequentially in time:

1) selection of arrivals for service,

2) allocation of time and service facilities to the selected arrivals.

These decisions may provide the only flexibility available in controlling the performance of the system.

A common control procedure used in physical systems is "first-come, first-served." With this procedure, arrivals are selected in the time order of their occurrence and each thereupon receives whatever time and facilities are required for its completion. A great part of the theory of queues is limited to this rule. But it is widely recognized that other procedures can improve a system whenever there is large variation in the amount of time and facilities required to service each arrival. This is especially true when convenience to a human user is important. An illustration of this fact occurs in the modern supermarket, which provides an express checkout counter for customers with small orders.

An improved control strategy will generally be more complex than the simple "first-come, first-served" rule. The control actions may depend on a number of factors, including time, the number of arrivals in queue, and distinctions between arrivals, for example. The use of a more complex strategy will usually imply on additional investment in facilities to implement the control logic. Economics may therefore eliminate all but simple procedures like the first-come, first-served rule. However, when a principal system component is a general purpose electronic digital computer, the queue control function is conveniently implemented as a stored program. There is every reason then to investigate the performance advantage which may be realized by incorporating a sophisticated, dynamic decision procedure.

This study concerns the theory and application of a mathematical technique which has practical utility in finding optimum decision processes for controlling queues. It is particularly motivated by the queueing problems in large remote access computer systems. Such service systems are ideal for the application of the more sophisticated processes which the optimization technique allows.

The optimization of control of service systems is a logical direction in which to extend the theory of queues, yet fairly little has been accomplished in this area. This study appears to have merit as a general contribution in this respect. The results on optimization models of multiple-access computers should be of special value for computer system engineering. The technique advanced in this study should have additional applications as wide and varied as the theory of queues has had.

1.2 ORGANIZATION OF MULTIPLE-ACCESS COMPUTERS

The scope of operational control in a stochastic service system is well illustrated in a multiple-access computer system. A general discussion of these systems will give a better appreciation of the problems involved, and an introduction to the principal application of interest in this report.

1.2.1 Purpose

A multiple-access computer is a special case of a class which may be called "on-line" computers. This general class is characterized by

direct communication between the computer and many remotely located devices. These devices may serve to couple the computer to a physical process, as do temperature sensors and electronic counters; they may couple the computer to another computational process, as when the remote device is another computer; or, they may couple the computer to a human being, as when the remote device is a teletypewriter or a cathode-ray tube display. The existence of direct communication is essential. Conventional operation of digital computers today, the so-called "batch processing" mode, involves an intermediate conversion of program input from punched cards to magnetic tape. This includes much manual activity from a human machine operator, and results in considerable delay between receipt of the punched cards from the user and the start of execution of the user's program. The batch processing operation therefore does not permit the computer to be used in controlling a rapidly varying physical process, nor does it allow users of the computer to quickly detect and correct errors in their programs.

The term "multiple-access computer" has in the past two years come to mean an on-line computer aimed at providing computation facilities to human users on a "public utility" basis. From a narrow view, this simply means improving the convenience of using the computer. This is done by providing each user with a personal communication device, chiefly a teletypewriter at the present time, and making it possible for the user to activate a computer program and quickly receive the results, directly at his console. The speed of a modern computer is so

great that a user will usually spend much more time in the conception of each request than the computer spends in performing the requested computation. The public utility concept is therefore more economical if a large number of users are connected to the machine. Then in the time intervals when one user is deciding what to request next, the computer can be serving the requests of other users. Thus, as time proceeds, the users are actually sharing the computer in turn, and this mode of operating a computer is frequently called "time-sharing."

From a broader view, however, multiple-access computing aims to increase the usefulness of a computer, beyond mere convenience. What is sought is what Licklider [11] has called "man-computer symbiosis." Man-computer symbiosis is achieved when the man can exploit the tremendous speed and logical capability of the computer as fully and easily as he would use a human assistant. The premise is that in circumstances permitting free and rapid interchange of information between man and machine, the essence and solution of a problem is more quickly exposed. See [12,13] for examples of this approach to computer usage.

Beyond direct communication, successful man-computer symbiosis requires a large repertoire of programming languages and computing utility programs. It is by means of this "library" that the man can "converse" with the machine, and have it respond with "intelligence" analogous to that of the human assistant. The size of such a library, and the need for rapid access to any part of it, implies that the multiple-access computer should have a very large capacity, fast-access

memory unit. Magnetic drum and disc memories have been found suitable.

Although military and business organizations have been interested in special purpose on-line computers for some time [14,15] and despite rather early speculation on general purpose systems [16], much credit for achieving a useful multiple-access system belongs to F. J. Corbató and associates [17,18,19] at the Massachusetts Institute of Technology. As a principal effort of Project MAC[1] at MIT, Corbató's original concept has been expanded in the last two years to more closely approach the scale appropriate to the public utility concept. Presently, there are about 100 remote-units on the MIT campus, and as many as 25 can be simultaneously in use [19]. There are and have been a number of other multiple-access computing projects in progress throughout the country [20-27], and much interest in such systems exists among organizations now ordering new computing equipment.

1.2.2 Equipment and Use

Let us turn now to more description of the equipment and use of a multiple-access computer. The discussion will occasionally refer to particular characteristics of currently existing systems, but the principal intent is to exhibit the common characteristics of them.

---

[1] An acronymn meaning both multiple-access computing, and "machine-aided cognition," a term equivalent to man-computer symbiosis.

Figure 1.1 depicts the equipment and principal data flow in the typical multiple-access computer existing today. The equipment used is in most respects commonplace even in batch processing systems, undoubtedly because multiple-access computing is such a new development.[2]

The central processor is typical of modern general purpose computers. When executing a particular program, the central processor obtains instructions and data for the program from the main memory. The input-output processor, sometimes called a "data channel," is a special purpose digital computer for controlling data transmission with a peripheral device—in this case, the secondary memory. The I/O processor can independently transmit data or computer programs between the main memory and the secondary memory. The secondary memory is a large capacity magnetic disc or magnetic drum unit, capable of storing at least a million computer words. This unit holds almost all of the system library. Some programs however are permanently held in the main memory. The latter constitute part of the operational control for the system.

The communication processor is a special purpose digital computer which provides the interface between common carrier telephone and telegraph lines and the data processing complex. The IBM 7741 [30] is an example of such a unit. Together with suitable line termination devices the communication processor performs such functions as conversion

[2]This is bound to change as new computers are developed specifically for multiple-access computing. A suggestion of things to come is contained in Dennis' paper [28], and a paper by Galler, Arden, Westervelt and O'Brien [29].

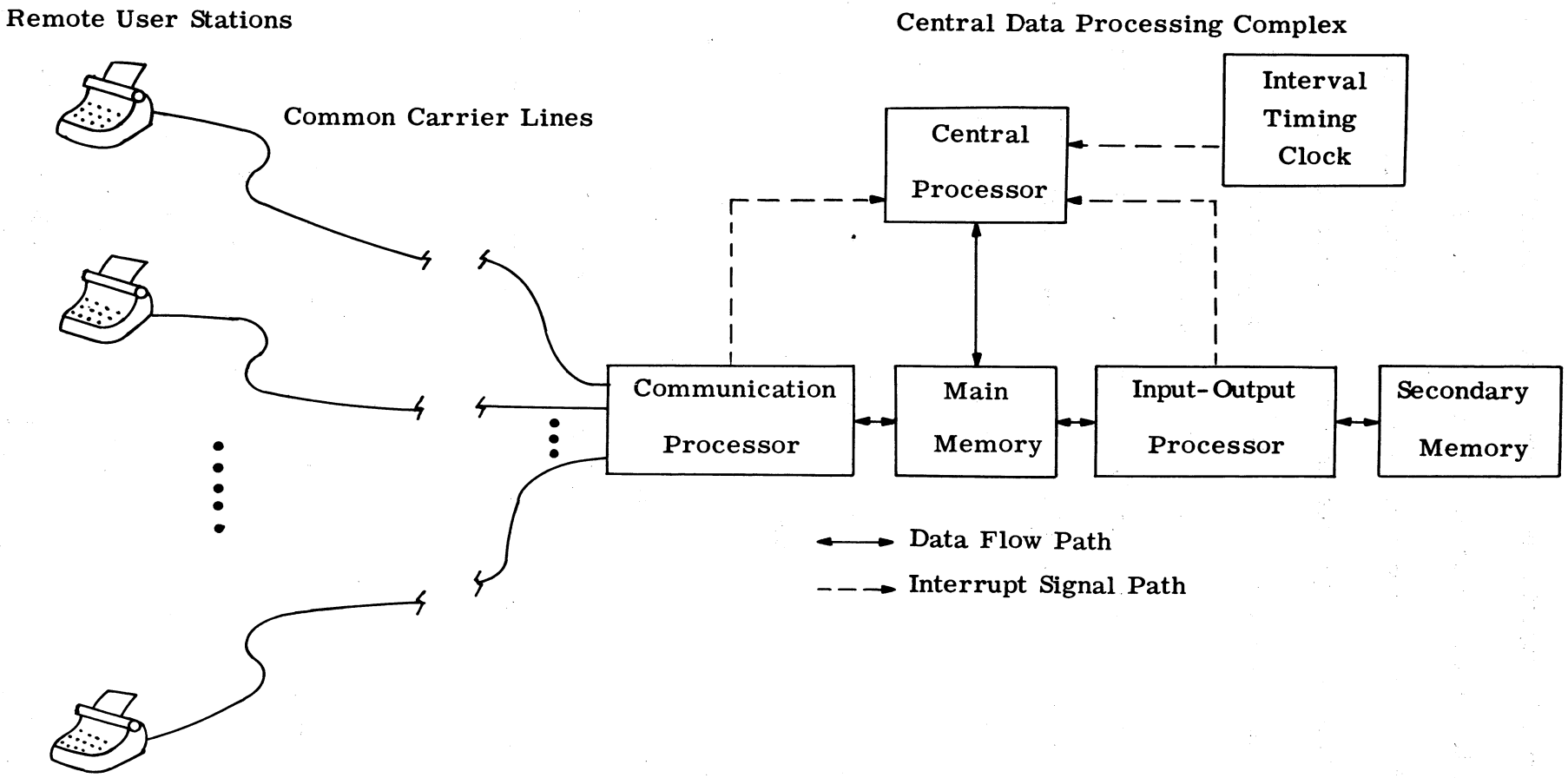


Fig. 1.1. Equipment units and data paths in a multiple-access computer.

of signal levels, conversion of teletype code to computer code, etc. As each teletype character is received from remote users, the communication processor stores the character and the user line number as a single word in the main memory. The central processor is thereby relieved of many time-consuming but necessary communication activities.

The interval timing clock is a digital device which will count out a prescribed interval and signal the CPU[1] at its completion. The signal is an "interrupt" signal which forces the CPU to branch from the instruction sequence currently being executed to another designated instruction location. This action is intimately associated with the overall operational control, as we will presently describe. Other interrupt signals originate from the communications processor and input-output processor to notify the CPU of the completion of operations assigned by it, or equipment faults detected by these units.

During operation of the multiple-access computer, users at their remote stations are free to independently type in commands and input data to the computer. The commands call for the computer to execute a designated program, and input data may be required from the user during the subsequent execution. References [18,20,23,24,25,26] contain extensive illustrations of user commands and machine responses in a session at the console of a multiple-access computer.

---

[1]Central processor unit.

1.2.3 Executive Control

The central processor of a multiple-access computer can execute only one program at a time. However there is no constraint on the users which prevents two or more from attempting to have a program executed at the same time. Moreover there is nothing to prevent a user from initiating the execution of an incorrect program which will never terminate. One cannot rely on the user to recognize such a fault in time to prevent serious interference with other users of the computer. Hence to ensure orderly use of the available facilities there must exist an operational control process which supervises user program activities and resolves conflicting demands. Part of this control function is implemented in the "hardware" of the system. However, the principal operational control functions of the system reside in "software," i.e., a computer program, commonly referred to as the Executive Control Program (ECP) or Supervisor. The decision to implement the major control in a stored program stems of course from the desire to be able to readily alter the procedure after observing the system in operation.

The ECP goes into execution in the central processor whenever an event occurs which requires a control action to be taken. Such events are indicated by either an "interrupt" signal, or the execution of a transfer instruction. The interrupt forces the CPU to leave the user program being executed and transfer to a designated entry point in the ECP. The ECP may also be entered by a "voluntary" transfer from the

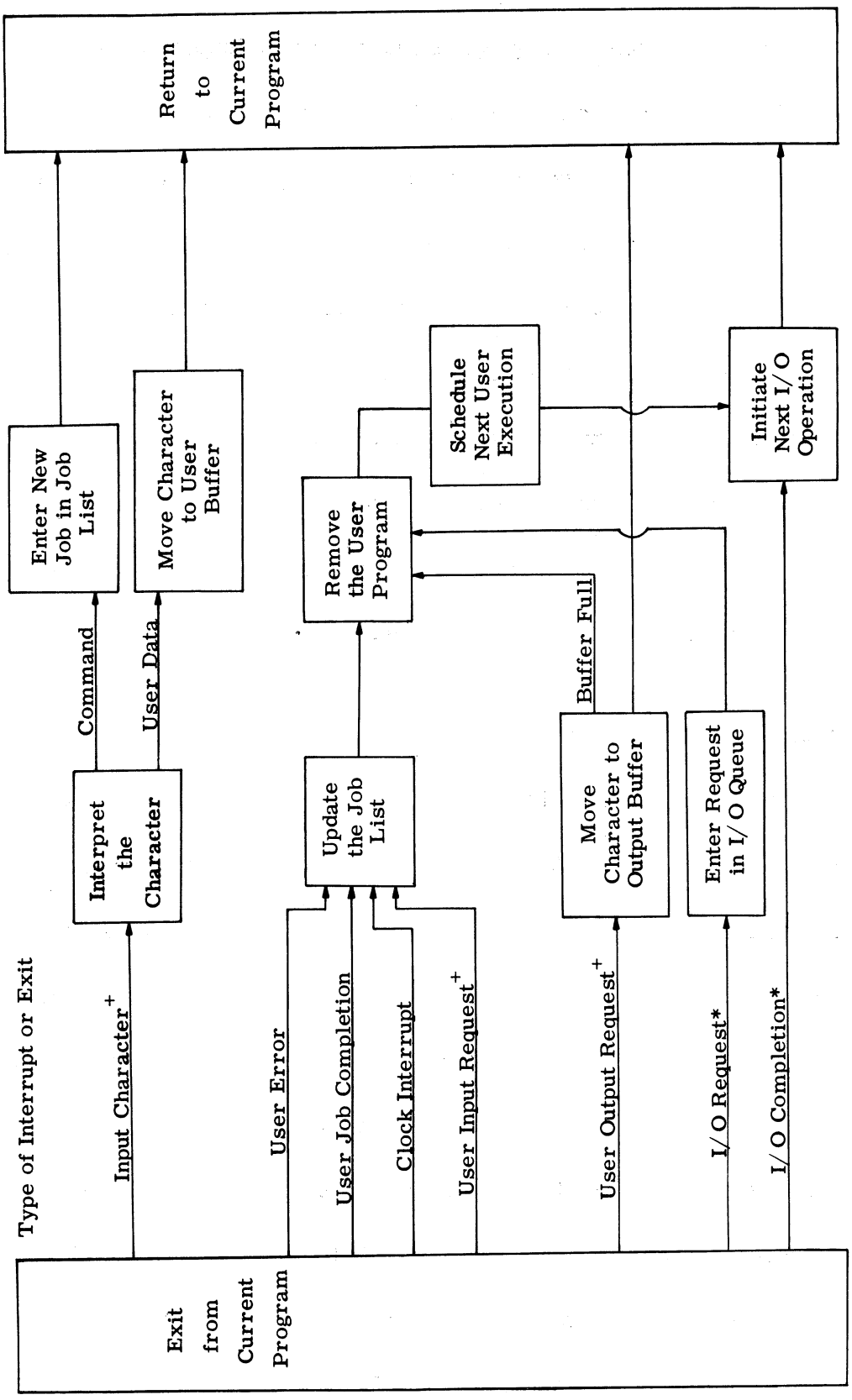
user program, as at the completion of the user program. During its execution, the ECP initiates control actions which will depend upon the system conditions existing at the time. In addition to the knowledge conveyed by the type of interrupt, the ECP uses information stored in tables held in main memory to determine all pertinent system conditions at the time of the interrupt. One important table in particular is the Job List. This lists each user who has a program in process and the status of the user's program. The status might indicate for example that the program was in execution before the interrupt. Alternatively, it may have been waiting to be executed, or waiting for input data from the user. Additional information about the program may also be included, such as the total execution time it has received, and the total size of the program in computer words. Such information is useful in "scheduling" of the central processor. This is the decision process that determines which user program will be placed in execution at any time, and how long it will be allowed to run before being superceded by another user program. This aspect of executive control is particularly of interest in this report.

There are also so-called "buffer" areas in main memory which hold information the ECP may use or alter during its execution. The Input Buffer, accessed by both the ECP and the communication processor, receives input characters from the users. The ECP inspects each of these and interprets them relative to the user's current status. For example, if the user program is waiting for input, the received characters are

simply passed on by the ECP to the user program. Otherwise, a single character might signify a particular command, to which the ECP must respond by entering the user in the Job List. The Output Buffer stores characters being passed by the ECP from the user program to the communication processor.

Figure 1.2 gives a somewhat simplified picture of the overall flow of the Executive Control Program, and the functions performed in response to various interrupts. We have essentially covered already the ECP response when an input character is received from the communications processor. The main effect of the ECP response to a user error, user job completion, a clock interrupt, or a user program request for input is to remove the user program from main memory and schedule another user program for execution. When removed, the program is saved in the secondary memory unit of the computer. From there it can be retrieved later for further execution, or for debugging analysis by the user from his console. The latter would involve other commands which use the saved program as input data. The process of removing one program from main memory and substituting another from the secondary memory is referred to as "swapping".

Note that a user program may be removed from execution whenever it requires input or output with the user through his typewriter, or with the secondary memory through the input-output processor. In the case of a request for input from the user, this is done because the user may require an appreciable time to type in this data. Meanwhile, the CPU can



* Secondary Memory Input-Output
\+ Typewriter Input-Output

Fig. 1.2. Executive control functions.

serve other users. For a program request to send output to the user, swapping occurs only if the user program fills up the Output Buffer, which can hold only a certain number of characters. Swapping here allows the communication processor to clear the Output Buffer. In the case of I/O with the secondary memory, swapping is done because such I/O can be carried out concurrently with program execution in most systems, due to the capability of the I/O processor.

The removal of a user program at the occurrence of the clock interrupt is associated with the scheduling process for the central processor. The importance of this aspect here warrants a special discussion of existing techniques and their limitations.

1.2.4 Scheduling

The Job List of the Executive Control Program constitutes a queue of user programs which are waiting to be executed. The number of jobs in the list will fluctuate randomly in time as jobs receive execution and are completed, and as new commands are typed in by the users. Perhaps more often than not, a new user job entering the Job List will find another user's program already waiting. Before the arriving job is executed it will generally have to wait in queue for some interval of time. This delay increases the "response time" which the user observes. Response time is the interval from the time when the user completes typing a command, to the time when the machine completes typing back the results for that command.

The average response time that a user observes should not be uncomfortably long if multiple-access computing is to accomplish its avowed purpose. For each command however, the response time must be at least as long as the execution time required by the command. Now, effective use of man-machine interaction in multiple-access computing tends to encourage short execution time. The great speed of a modern computer also contributes, so that job execution time will frequently be only a fraction of a second. Nevertheless, because of either logical errors in programming or the complexity of some computing problems, execution time could be many seconds in duration or even unending.

This fact makes it clear that first-come, first-served scheduling of job execution is unsatisfactory for multiple-access computing. One must instead impose a maximum time allocation for execution of each job. Its value moreover should be consistent with desirable response time values, not more than a few seconds. Yet, a program is not necessarily faulty if it is not completed within the allocated time interval.

These observations have led to widespread use [20,23,24,26] of a scheduling process known as "round-robin" service. In round-robin, new jobs are entered in order by arrival at the end of the queue. Jobs are selected for execution one at a time from the top of the queue, and executed for some "quantum" of time. The same amount of execution is given to each job, unless of course it terminates itself during the quantum. If uncompleted at the end of the quantum, the job is placed at the end of the queue, behind any which arrived. The processor con-

tinues in this way, cycling through the queue. In each cycle the first job to receive its quantum is also the earliest arrival present. Figure 1.3 depicts the process.

In most systems today only one executable user program occupies main memory at any time. To switch execution to another user program, the input-output processor must swap two programs between the main and secondary memories. The swap time is not productive per se, since no job receives execution during this time.

A problem in round-robin service is to determine the optimum quantum to be allocated, in order to balance the loss of production in swapping against desirable response time values. One can see that if the quantum is very short, say 50 msec, the machine will spend less than half the time in productive processing. On the other hand, if the quantum is very long, say 5 seconds, then an arriving user program might wait many seconds in queue before receiving its first pass of execution. No complete solution has been obtained for this problem. What has been done is a rough setting of the quantum at 200-500 msec. This makes swapping relatively less inefficient than for shorter values, and allows a short response time provided the queue does not get large.

Two other approaches to the problem have led to modifications of round-robin service. The System Development Corporation technique [25] takes a more direct view of the response time obtained by the user. A response time cycle of about 2 seconds is established, and the quantum allocated to each job is determined by equally dividing this interval

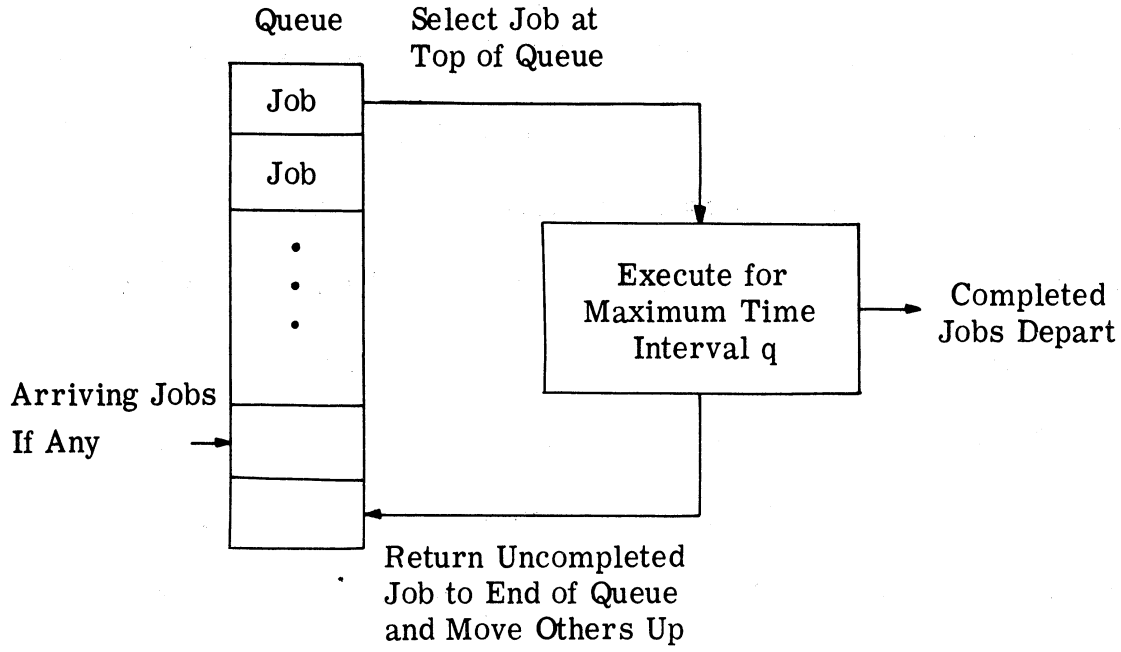


Fig. 1.3. Representation of round-robin service.

among the jobs in queue when it is executed. Thus jobs receive a longer quantum when there are few of then in queue, and a very short job is completed in close to 2 seconds. The process becomes inefficient for long queues however, since the swapping time takes a larger portion of the response cycle. To counteract this, a minimum quantum duration is established and not every program is processed in each cycle. This scheduling technique might be called "dynamic round-robin," since the allocated time changes with the number of user programs in queue.

The most sophisticated technique in use is that developed by Corbató, et al. [17]. The MIT system, more than any other, is faced with wide variability in the service requirements of users. For example, user programs can range up to 32,000 words in size. Swap times can therefore approach several seconds. Execution times can be as small as several milliseconds, while for programs with complex decision trees, such as Samuel's checker player [32], the execution time can be on the order of minutes.

The technique devised by Corbató may be called a multi-level priority scheduling process. There are a fixed number, say M, of queueing levels for jobs in process. That is to say, the set of jobs in queue may be separated into M disjoint subsets, some of which may be empty. The levels are ranked by priority, with level 1 having highest priority. This means that whenever there is a job in level 1, it will be executed in preference to a job at any other level. A job at level 2 will be executed in preference to a job of level 3 or lower, etc. There is a max-

imum time allocation for execution associated with each level, and if a job is not completed in this time it is moved to the next lower priority level. The lowest level queue is processed by a round-robin procedure. Within each level, jobs are processed in order of arrival.

The size of a program in words determines its priority level at arrival, according to a general formula. Let W denote the size of a program, and W_0 a specified number of words. A job enters level ℓ on arrival if

$$\left[\log_2\left(\left[\frac{W}{W_0}\right]+1\right)\right] = \ell \qquad \ell = 0,1,2,\ldots,M-1$$

Here $[\cdot]$ denotes the integral part of the enclosed quantity. Presently W_0 is taken as 4000 words [33]. The maximum time allocated to each job processed from level ℓ is $2^{\ell}q$ seconds, where presently q is 4 seconds [33].

This procedure has two notable advantages. First, the variable time allocation allows a continual balance to be achieved between swapping time and productive processing. Second, the priority scheme allows small, short jobs to receive short response with less interference from large or long jobs. Its principal disadvantage is that large short jobs obtain comparatively poor response.

The round-robin scheduling procedure has considerable appeal because it seems to directly fit the idea of time-sharing. One can see that in a round-robin each user gets equal treatment from the central processor, an equal share of its time as it were. But the objective of

scheduling in multiple-access computing is not merely to ensure each user an equal share of central processor time. Rather it is to provide each user with as rapid a response as possible, consistent with the execution time the user has requested by his command. The multiple queue scheme, such as advanced by Corbató, appears to have greater flexibility for achieving a comprehensive optimization of response time performance.

The techniques in use today have resulted from good engineering judgment and limited experimentation with the actual systems. The success of the multiple-access computing concept speaks well for the ingenuity of the pioneers of the field. But with such large and complex systems there is great value in supplementing the designer's intuition with whatever theoretical and experimental evidence that can be made available. Experiments are so expensive and difficult to control, however, that a purely experimental approach is not likely to establish the general principles that the designer may seek. The study of a theoretical model can lead to such results, as engineers are well aware.

This report has this objective in mind. In Chapters V and VI we will consider a theoretical model of scheduling in a time-shared computer. This model will include a wider class of scheduling decision processes than have been used in practice. By an optimization procedure which is established in Chapters III and IV, we are able to determine some general principles governing the optimum control of queues in such computer systems.

1.3 ILLUSTRATIVE ANALYSIS OF A SIMPLE PROBLEM

The nature of the theory of queues is best illustrated by outlining the solution of a simple queueing problem. This will also permit introducing some fundamental concepts to be used later. Moreover it will provide background for contrasting the analysis approach with the optimization approach which is the principal technique of this study.

1.3.1 Problem Description

Consider a service system in which there are two distinct sources for arrivals. Each source originates only one arrival at a time, and no more until service of its preceding arrival is completed. The arrivals from both sources compete for use of a single service facility on a first-come, first-served basis. A queue will exist if one source generates an arrival when the server is already occupied. Figure 1.4 depicts the situation. Classically this is viewed as a "repairman" problem, where the sources are machines which fail and the service facility is a repairman. See Takács [10] for a more general treatment of this type of problem.

Let us assume for this illustration that the service time for each arrival is always of one unit duration. We are also concerned with the time interval t_a which elapses between the end of service of an arrival from one source and the time when it generates its next arrival. Let us assume that this interval is a random variable. Moreover, assume that the two sources have the same statistical properties for t_a, and that

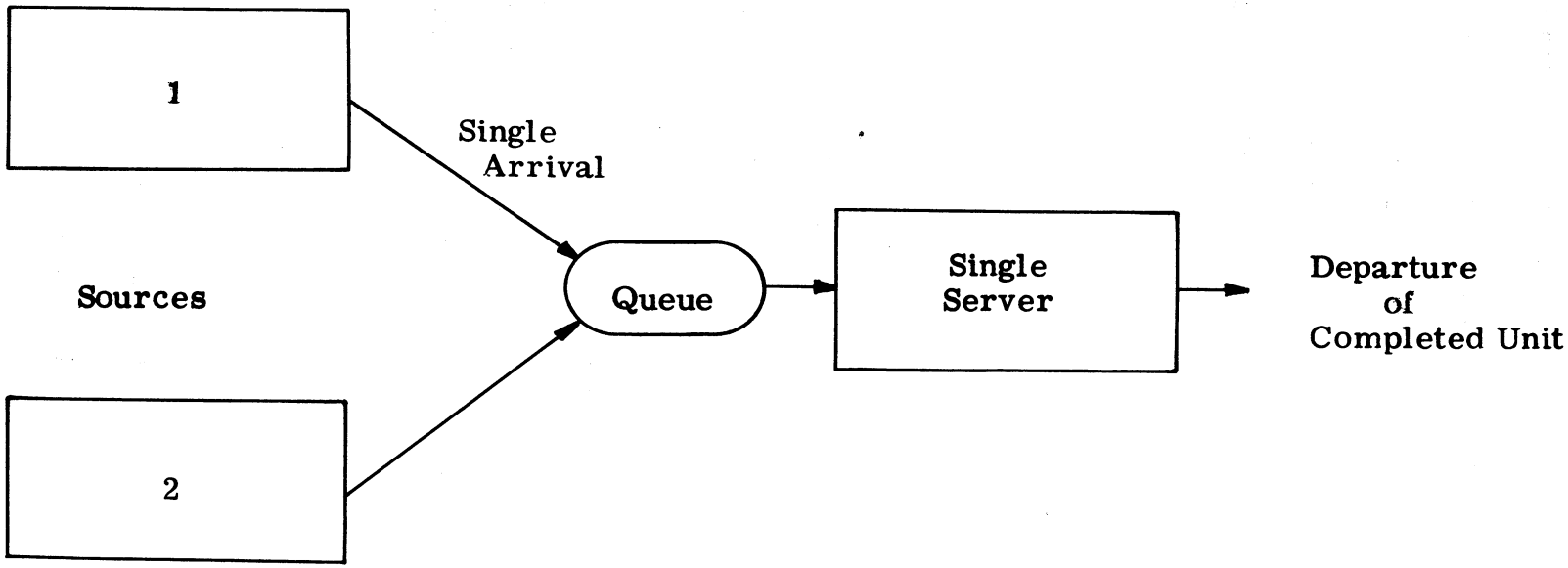


Fig. 1.4. Representation of a simple queueing problem.

separate observations of t_a are mutually independent, and drawn from a common distribution. This distribution is assumed to be described by the "negative exponential" probability distribution function

$$\Pr(t_a \leq T) = 1 - e^{-\lambda T}, \quad 0 < T < \infty \tag{1.1}$$

Here $1/\lambda$ is the mean value of t_a.

The negative exponential distribution plays a very prominent role in queueing theory. This stems in part from the "memoryless" property which it alone possesses of all possible probability distribution functions. This property is evident from the conditional distribution function

$$\begin{aligned} \Pr(t_a \leq T+\tau \mid t_a > \tau) &= \frac{e^{-\lambda\tau} - e^{-\lambda(T+\tau)}}{e^{-\lambda\tau}}, \quad \tau > 0 \\ &= 1 - e^{-\lambda T} \\ &= \Pr(t_a \leq T) \end{aligned} \tag{1.2}$$

This result states that at any time τ after the interval t_a commences, the remaining time in the interval has the same distribution as the entire interval. Thus τ is irrelevant, and the distribution is said to be memoryless with respect to τ.

As this system operates, there is a possibility at any time of there being 0,1, or 2 arrivals present for service. If there are 2, one arrival must be waiting in queue. If there are 0 or 1, then at least one source is in process of generating a new arrival. It is convenient to say the source is "idle" in this case.

The classical problem of queueing analysis is to determine the probabilities P_0, P_1, and P_2 which govern the number of arrivals present at any time. Generally these probabilities will change with time due to the influence of the initial condition of the system. In a great many problems of interest, however, the probabilities approach constant values as time proceeds. This is called the "steady state" or "statistical equilibrium". The values P_0, P_1, and P_2 are then called the "steady state" or "equilibrium" probabilities.

1.3.2 Analysis

As a means of finding P_0, P_1, and P_2, let us first consider quite specific instants of time during the system operation, namely the successive instants when a service is completed. Figure 1.5 depicts a sample of the time behavior of the system, and the points mentioned. Immediately after a service completion, the number of arrivals present can only be either 0 or 1. Consider any two successive instants. Given the number present at the first point[1], we can readily determine the probability of 0 or 1 present at the second. For example, if 1 is present at the first, then the second point occurs one time unit later. The probability that one arrival is present then is simply the probability that an arrival occurs from the idle source during the unit interval.

[1]Although the phrase "at the first point" or "at the second point" is used, we mean of course immediately after the time point. As shown in Fig. 1.5 there is a step change in the number present at the time points in question.

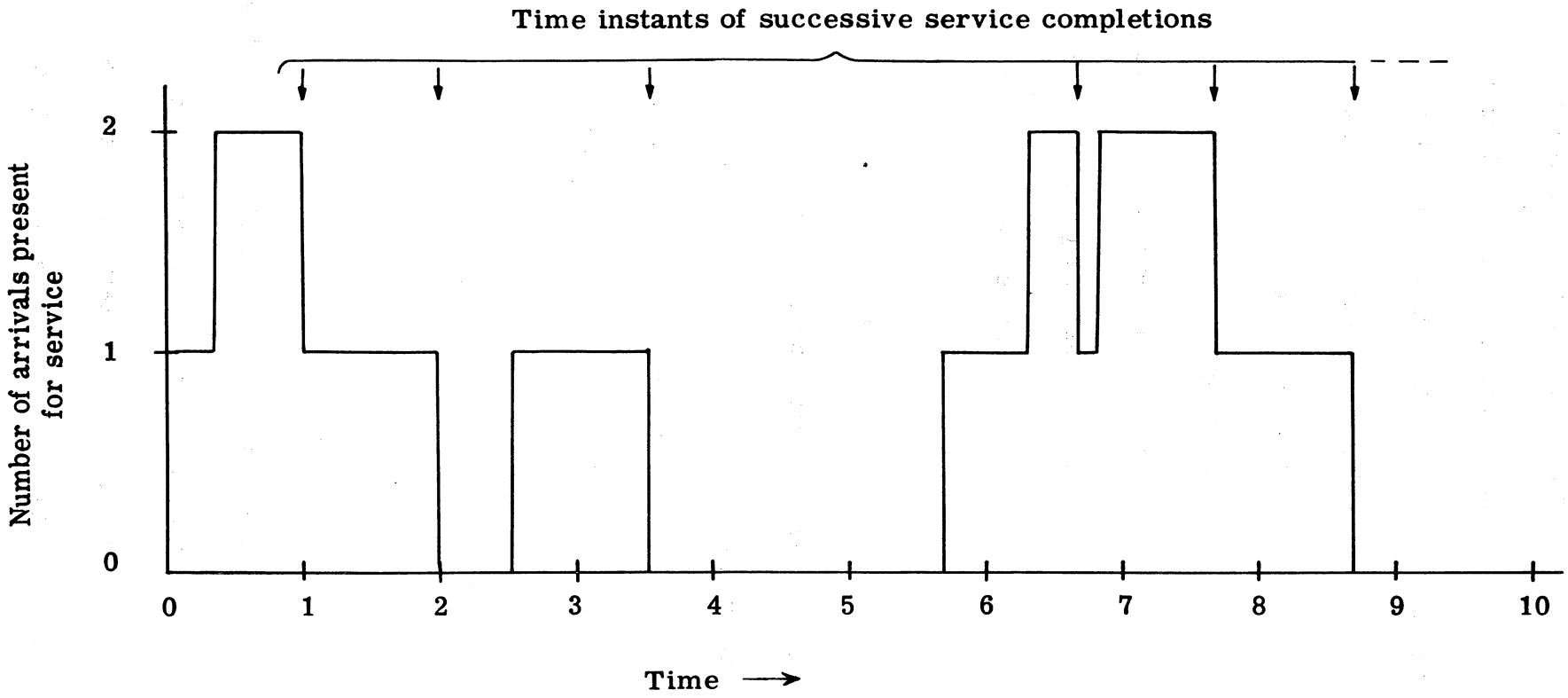


Fig. 1.5. A sample of system behavior showing service completion times.

The memoryless property of the negative exponential distribution on t_a now enters to assure us that we need not know how long the source has been idle. The probability that t_a ends in any unit of time is always $1-e^{-\lambda}$. Thus we have derived a "transition" probability

$$p_{11} = 1 - e^{-\lambda} \tag{1.3}$$

where the subscripts indicate the given number of arrivals at the first instant, and the observed number at the second, respectively.

On the other hand, no arrivals will be present at the second time point if the idle source does not generate a service demand. The probability of this event gives

$$p_{10} = e^{-\lambda} \tag{1.4}$$

Now when no arrivals are present at the first time point considered, the second completion occurs one time unit after the next arrival for service. At the second point there will be one arrival present provided a second arrival occurs during the unit interval where the next arrival is in service. Therefore

$$p_{01} = 1 - e^{-\lambda} \tag{1.5}$$

and further

$$p_{00} = e^{-\lambda} \tag{1.6}$$

The number of arrivals present can be defined as the "state" of the system at the service completions. With respect to this definition of state, the system has the Markov property at these time points. This means that the known state at any such time is sufficient to determine the probability of the state at any future one of these time points.

The states of the system at successive service completions then constitute a stochastic process known as a Markov chain.[1] Because of the properties of the system we have described, one finds a set of asymptotic values, denoted π_1 and π_0, which are the equilibrium probabilities of states 1 and 0 at a service completion. These satisfy the system of equations

$$\begin{aligned} \pi_0 &= \pi_0 p_{00} + \pi_1 p_{10} \\ \pi_1 &= \pi_0 p_{01} + \pi_1 p_{11} \end{aligned} \tag{1.7}$$

Thus we find that

$$\begin{aligned} \pi_0 &= e^{-\lambda} \\ \pi_1 &= 1 - e^{-\lambda} \end{aligned} \tag{1.8}$$

With this result the probabilities P_0, P_1, and P_2 can be found. Intuitively,[2] they represent the average fraction of time that the system spends in states 0, 1, 2, respectively, throughout its history of operation. But they also are the average fraction of time the system spends in these states during an interval between successive completions, after the system has attained statistical equilibrium. At the first completion defining such an interval, the system is in state 0 or 1 with probabilities π_0 and π_1 respectively. By considering the random events during such an interval, conditional on states 0 or 1 at its beginning,

[1] Also called an "imbedded" Markov chain for systems such as treated here, because only specific points in time are included.

[2] The truth of this can be rigorously established by considering in detail the ergodic properties of the system.

one can evaluate the expected time spent in states 0, 1, and 2 before the end of the interval. Then

$$P_i = \frac{\text{Expectation of time spent in state } i \text{ between successive completions}}{\text{Expectation of total time between successive completions}} \tag{1.9}$$

for i = 0,1,2. For example, we obtain

$$P_0 = \frac{\pi_0\left(\frac{1}{2\lambda}\right)}{\pi_0\left(1+\frac{1}{2\lambda}\right)+\pi_1(1)} \tag{1.10}$$

where $1/2\lambda$ is the expectation of time spent in state 0, conditional on state 0 at the first service completion considered. But $\pi_0+\pi_1 = 1$, and so

$$P_0 = \frac{\pi_0}{2\lambda+\pi_0} = \frac{e^{-\lambda}}{2\lambda+e^{-\lambda}} \tag{1.11}$$

A similar consideration leads to

$$P_1 = \frac{2(1-e^{-\lambda})}{2\lambda+e^{-\lambda}} \tag{1.12}$$

and

$$P_2 = \frac{2\lambda-2(1-e^{-\lambda})}{2\lambda+e^{-\lambda}} \tag{1.13}$$

Knowledge of P_0, P_1, P_2, π_0 and π_1 enables one to determine a number of other statistical properties of system behavior. For example, the expected number of arrivals present for service at any point in time is (P_1+2P_2). The important point to bear in mind is that analysis consists of evaluating the behavior of a completely specified system. The

general objective of analysis is to determine the equilibrium probabilities on the possible states of the system of interest.

CHAPTER II

OPTIMIZATION IN THE CONTROL OF QUEUES

This chapter has two objectives. First, a brief review will be given of some results from the classical theory of queues. These are aimed at optimizing the control of queues and illustrate the limited results obtained from closed form analysis. Second, the general optimization model to be studied in this research is introduced. Previous research in this area is then reviewed, and the new theoretical results obtained here are summarized.

2.1 RESULTS OBTAINED FROM ANALYSIS

2.1.1 Single Queue Systems

One of the simplest systems in structure is that where arrivals form a single queue in the service system. Generally arrivals are presumed to be a Poisson process and we limit discussion to this case. Then the interval between successive arrivals has the negative exponential distribution. It is important to distinguish two cases. On the one hand, arrivals may issue from an unlimited source. With an unlimited source the mean rate of arrivals is independent of the number present for service. On the other hand, arrivals may arise from a finite source as in the simple example given in Section 1.3. Here the mean rate of arrivals diminishes as the number waiting for service in-

creases.

Different aspects of system behavior may be of interest in determining performance, but a primary concern is often the expectation of the number of arrivals in queue at an arbitrary time. The system designer might like to institute a queue control procedure which reduces this expectation below that of the first-come, first-served procedure. However, there is a rather widely accepted principle defining a class of procedures which do not exhibit such improvement. Paraphrasing Morse ([7], p. 116): all single server systems having the following properties have equal expectation of the queue length. The properties are:

1. There is a common distribution applying to the service time of all arrivals.

2. All arrivals remain in queue until served.

3. Every arrival remains in service until completed.

4. The service system is not allowed to be idle when arrivals are waiting.

This class of controls includes in particular the first-come, first-served process; random selection for service; and last-come, first-served selection. The author is not aware of any rigorous mathematical proof of this claim in the generality with which it is stated. However for the three before-mentioned queue disciplines the result has been proven by separate analyses of each case, assuming arrivals from an unlimited source. In regard to random service the analysis has been carried out

only for exponential service times and for constant service time at a single server (see Saaty, p. 243). For first-come, first-served and last-come, first-served, the analyses have been done for service times having an arbitrary distribution with finite expectation. In all cases it has also been shown that the expected time-in-system of a random arrival is unchanged.

In fact, the principle may hold under other conditions than those given. This is suggested by an analysis given recently by Kleinrock [38, 47] for the round-robin strategy. Kleinrock uses a discrete-time formulation in which a time step has duration q, the length of the quantum of service time allocated to each arrival in queue. Bernoulli arrivals are defined for the process, where at each time step there is probability λq of an arrival, and at most one arrival may occur. The service time is assumed to have a geometric distribution, the discrete-time analog of the negative exponential, and in each time step there is probability σ that service continues to the next step.

Kleinrock proves that the expected time-in-system is identical for the round-robin strategy and for the first-come, first-served procedure. Moreover he derives an expression for the conditional expectation of the time-in-system, given an arrival requires nq units of service time for completion. This result is

$$T_n = \frac{nq}{1-\rho} - \rho q - \frac{\lambda q^2 \rho}{1-\rho}\left[1 + \frac{(1-\sigma\alpha)(1-\alpha^{n-1})}{(1-\sigma)^2(1-\rho)}\right] \qquad (2.1)$$

where $\alpha = \sigma+\lambda q$

$$\rho = \frac{\lambda q}{1-\sigma}$$

Figure 2.1 is a graph of T_n for one set of values of σ, λ, and q. Note in particular that the conditional expectation for $n = q/1-\sigma$ is the same as the expected time-in-system for the first-come, first-served case. This value of n is the average service time, and this result is generally true. For smaller values of n, however, T_n is less and for larger values of n the converse is true. Thus a round-robin strategy favors arrivals having smaller service time than the average, at the expense of increasing the time-in-system of arrivals requiring more service time than the average.

Coffman and Krishnamoorthi [48] have given a more general analysis of the round-robin strategy for the case of a finite arrival source. Their model treats the geometric distribution on service time, but allows arrivals to occur as a continuous time process. Moreover the model includes a constant time for swapping jobs, included in every quantum.

Coffman and Krishnamoorthi approach the system in basically the same manner as used for the system of Section 1.3. By considering the number of arrivals present at the end of every quantum of service they establish an imbedded Markov chain. Formulae are derived for the probabilities π_i that i arrivals are waiting at the end of a quantum. The mean and variance of the number waiting is also given. However there is no derivation equivalent to Kleinrock's for T_n, the expected time-in-system for a job requiring n quanta to complete. This is more difficult to obtain for the limited source model.

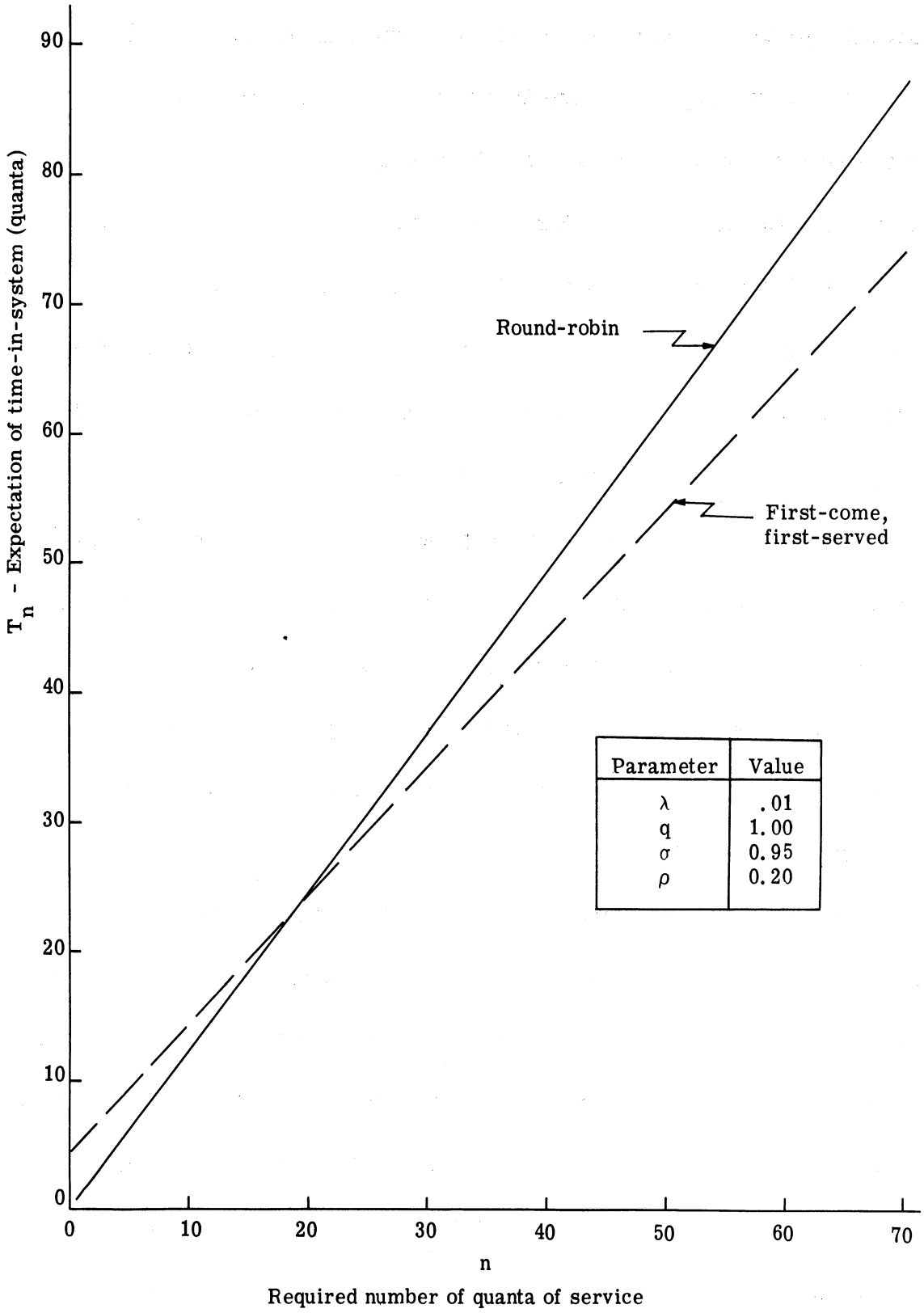


Fig. 2.1. Graph of expected time-in-system for round-robin service.

Analyses like the foregoing suggest that the system designer must turn to another class of control procedures to reduce the expected time-in-system. This leads to consideration of procedures where distinctions between arrivals are allowed on the basis of their service time. Such procedures give rise to multiple queue systems with priorities.

2.1.2 Multiple Queue Systems

One of the noteworthy general results of the theory of queues was obtained by Cobham [49] and Holley [50] in treating a priority scheduling procedure. As depicted in Fig. 2.2 they assumed Poisson arrivals from an infinite source to each of R queues, serviced by a single facility. Service is non-preemptive in that any service once begun proceeds uninterrupted until it is completed.

Priority service means that whenever the facility is vacated, the next arrival served is from queue p only if queues 1,2,...,p-1 are empty and p is not. The earliest arrival is always taken from each queue. The priority 1 queue is always served unless empty. Arrivals of each priority class have a common service time distribution which may be arbitrary. Cobham's principal result was a general formula for the mean time spent in queue by a job of any priority class. This enables a system designer to investigate the reduction of waiting time by priority assignment. Cobham also gave a similar result for the case of multiple parallel servers, and a negative exponential distribution on service time, common to all arrivals.

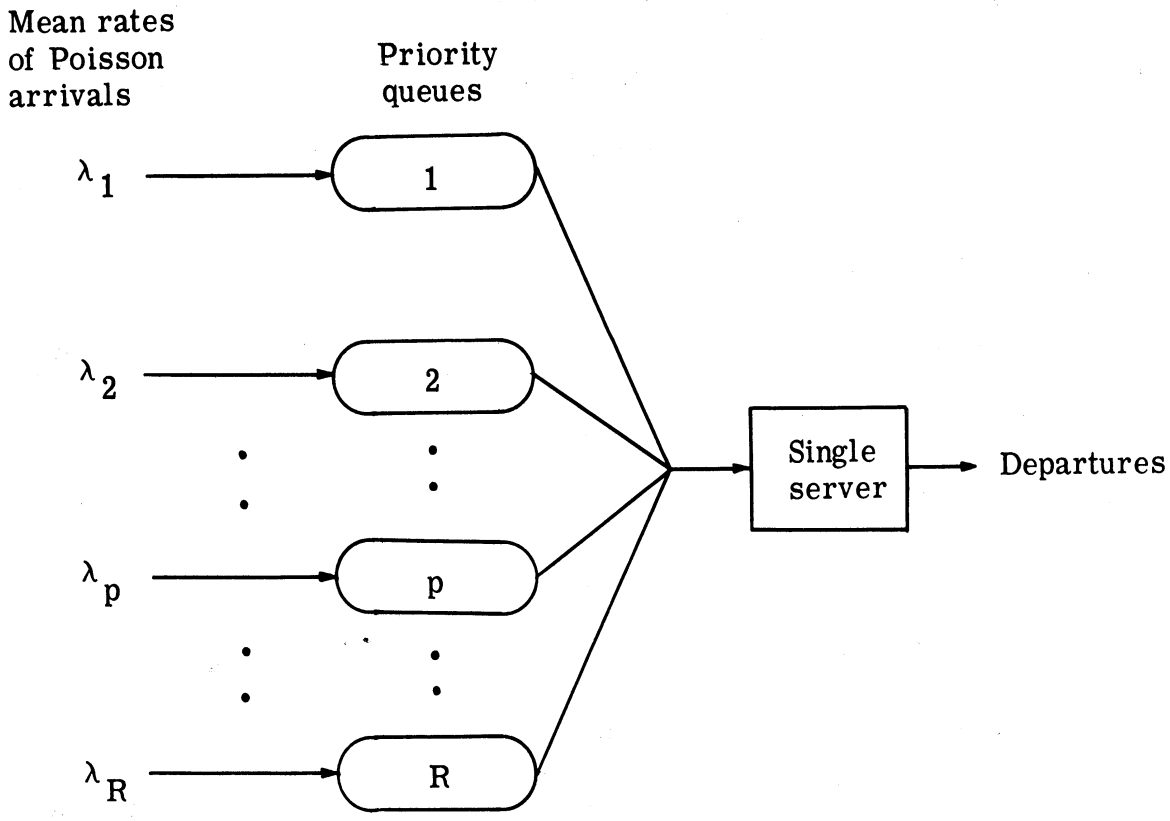


Fig. 2.2. Priority service on a single facility.

Cobham's result has been extended to a system with an infinite number of priority levels by Phipps (Saaty [8], p. 236). This makes it possible to study priority scheduling based on the service time of arrivals. Assuming the service time of every arrival is known exactly, it follows that the expected time-in-system is minimized when the arrival of shortest service time has highest priority (see [3,38,51,52]). In other words, "shortest-job-first" describes the rule for optimizing the expected time-in-system when each arrival is served to completion and service times are known exactly. This holds moreover for any distribution of service times over the population of arrivals.

Rather than minimize the expected time-in-system it may be desirable to treat a weighted expectation. Here assume that the population of arrivals can be separated into classes based on the required service time, denoted by a, and a quantity p. The latter is a "loss rate" by which the time-in-system is to be weighted to yield a loss in servicing each arrival. The expected total loss of a random arrival is then minimized by giving highest priority to the arrival in queue having the largest value of p/a. See Refs. 3, 38, 51, 52. Again it is assumed in this result that p and a are known exactly, and that a service once begun is not interrupted until completed.

Another type of priority mechanism has been treated by Jackson [53] and Kleinrock [38]. Jackson, who terms his a "dynamic priority," derives upper and lower bounds on the expected waiting time, and thus compares his control process with first-come, first-served and static priority

schemes as treated by Cobham. Only non-preemptive service is treated. Kleinrock treats a delay-dependent priority system which generalizes one aspect of Jackson's approach.

Other consideration has been given to preemptive priority scheduling under static priority assignments. Saaty [8], p. 237 ff, gives a review of such studies. Generally they are limited to requiring a negative exponential service time distribution common to all arrivals, and negligible time involved in the act of preemption.

The priority methods which have been studied have been shown to give the system designer some freedom in manipulating the expectation of waiting times. But only in special cases, as we have described, has a priority control process been proven optimal.

2.1.3 Experimental Analysis and Simulation

An immediate and correct conclusion from the preceding discussion is that the closed form analytical results of queueing theory will not be sufficient to cope with the analysis and optimization of many practical systems. Where these results are insufficient they nevertheless form valuable background from which the system designer can proceed to carry out experimental analysis and simulation of proposed system operation.

By experimental analysis we mean the process of formulating a mathematical model which can be rapidly solved numerically by a digital computer. The most convenient and powerful class of models for experimental analysis is that of Markov stochastic processes. For such models, very

rapid numerical procedures have been devised (see [54], for example), and used to explore many aspects of behavior in a queueing model [55]. Simulation, by contrast, consists of developing a descriptive functional model of the physical system in a computer program. Execution of this program allows one to observe and record the actual operation of the simulation model and thereby deduce properties of the operation of the physical system. Simulation is a useful tool because a much wider range of physical systems can be treated, but it is more expensive to use than experimental analysis.

Experimental analysis and simulation are fruitful for investigating complex control procedures, provided the permissible control processes can be restricted to a few alternatives by virtue of economic or practical reasons. Often there is a very large number of acceptable control processes, and it is not feasible to analyze and compare each of these separately. The system designer needs to supplement the available tools of analysis and simulation with an optimization technique. This technique should allow formulating a set of alternative control procedures which reasonably represents the possibilities in a physical system. The technique should also include an efficient algorithm which will systematically synthesize a control process attaining optimum performance. These objectives are largely realized by the techniques of Markov decision theory. The remainder of this chapter will describe a particular Markov decision process to be studied in the following chapters.

2.2 A MARKOVIAN DECISION PROCESS

This section defines a sequential decision process by which one can treat optimal control of a class of stochastic systems. Following the definition of the process, the remainder of the chapter will show how this decision process relates to other work and will review the contributions made in this report to the theory and application of such processes.

The wide spread application which Markov stochastic processes receive in the analysis of queues has already been pointed out. It seems quite likely that Markov decision processes will be equally versatile in the optimization of service systems. Since one cannot anticipate all problems to which the formulation may be applied, an attempt is made to keep some generality in the abstract definition. The particular model we define is more precisely described as a "discrete Semi-Markov decision process", as we will explain.

2.2.1 System Behavior

The decision process involves a system which exists at any time in one of a finite number of states. We identify each of the states by an integer index i and let N denote the total number of states. Thus i may take on values $1,2,\ldots,N$.

As time proceeds the system will experience random changes or transitions in state. A change of state may only occur at an integer value of time. For this reason time essentially proceeds in steps of one unit duration, and we say that we have a discrete-time process.

The system remains in a given state for a random number of time steps. At the end of such an interval it moves immediately to a new state, determined by a random mechanism. We allow however that the new state may be the same as the old, i.e., the system may make a transition into the same state. As used here, a "transition" includes the time interval wherein the system remains in a known state and ends with the subsequent random movement to the next state. Figure 2.3 depicts this behavior. In describing this behavior we will often say that a transition "begins in state i and ends in state j", or is "from state i to state j". For example, the first transition depicted in Fig. 2.3 begins in state 3 and ends in state 4. It need not be assumed, as shown in Fig. 2.3 that the origin of time is the beginning of a transition. The system state at the beginning of the first transition, as well as its starting time relative to the time origin may be specified by a joint probability distribution.

A sequential decision process is associated with the system behavior because a control action must be taken at the beginning of each transition. The control action is taken instantaneously upon entering the new state. The state is assumed to be observable, and the observation is used in determining the control action to be taken. For each observed state, there is a finite set of alternative control actions. For state i, this set is denoted by A_i and the total number of different actions by the integer K_i. An integer index k is used to identify each action, so that A_i consists of the set of integers $k = 1,2,\ldots,K_i$.

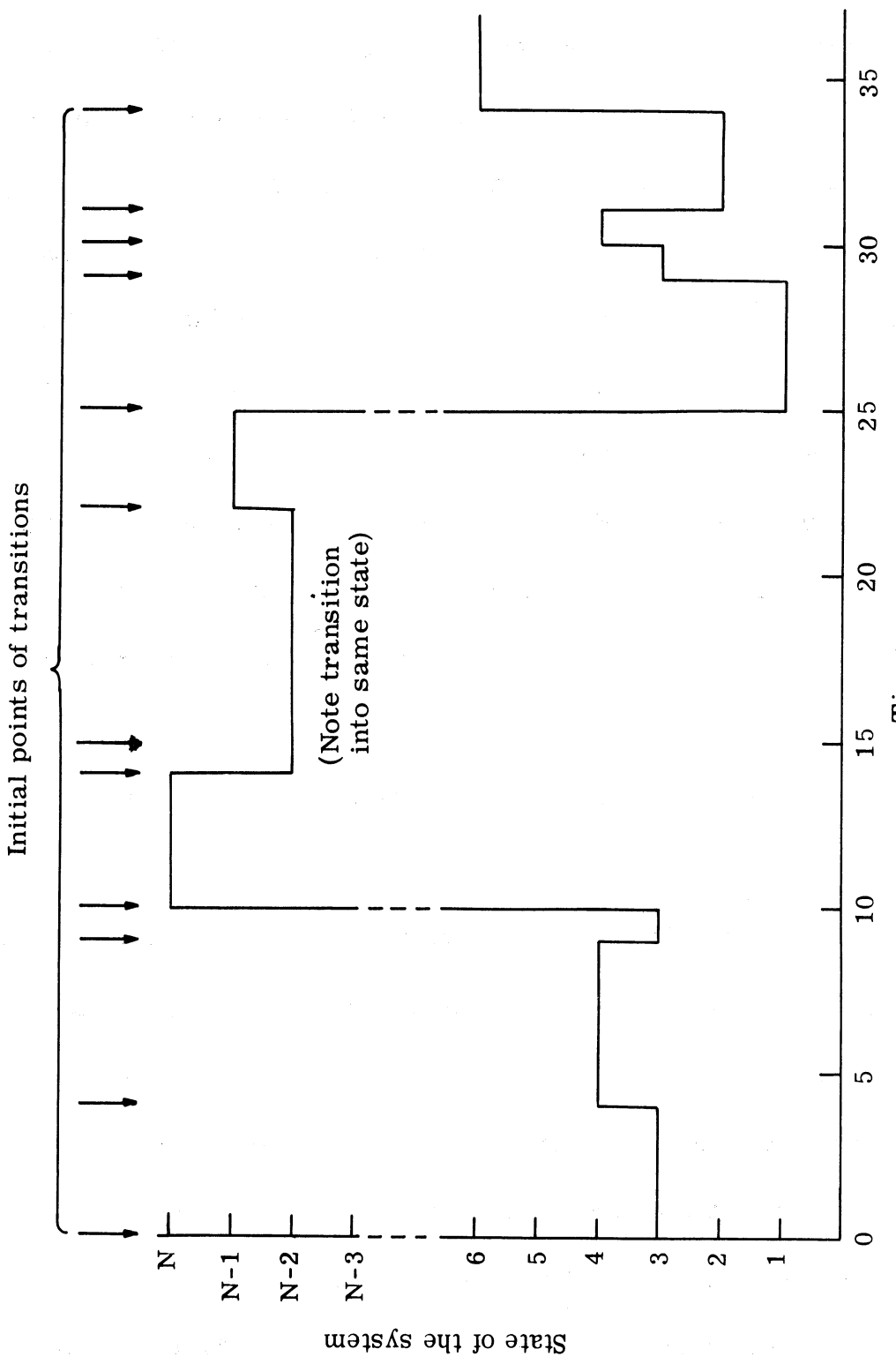


Fig. 2.3. Example of changes of state during the decision process.

The probabilistic mechanism governing each transition in state is assumed to be Markovian with respect to the state observed at the beginning of the transition and the control action taken. This means that the only historical knowledge required to describe a transition is the current state i and the action k taken upon entering i. The probabilistic behavior is also assumed to be stationary, so that given i and k, one need not know the time at which a transition begins. The probability that a transition ends with the system entering state j is dependent only on i, j, and k, and is denoted by p_{ij}^{k}. (Here k is a superscript.) Further, the probability that the transition interval is τ units long ($\tau = 1,2,3,\ldots$) depends only upon i, j, k, and τ, and is written $f_{ij}^{k}(\tau)$. Every transition must be at least one time step in duration.

The joint probability that a transition has duration τ and terminates in state j, given that it began in state i with action k, is written[1]

$$q_{ij}(\tau|k) = p_{ij}^{k} f_{ij}^{k}(\tau) \tag{2.2}$$

The following conditions must be true for the permissible values of i, j, k, and τ.

$$p_{ij}^{k} \geq 0 \tag{2.3}$$

$$\sum_{j=1}^{N} p_{ij}^{k} = 1 \tag{2.4}$$

[1]The use of k as a superscript in defining some quantities stems from the usage established by Howard [63,74] and Jewell [73]. Otherwise we follow the convention indicated in Eq. (2.2).

$$f_{ij}^{k}(\tau) \geq 0 \tag{2.5}$$

$$\sum_{\tau=1}^{\infty} f_{ij}^{k}(\tau) = 1 \tag{2.6}$$

The expectation of the duration of a transition from state i when action k has been taken is denoted ν_i^k and satisfies

$$\nu_i^k = \sum_{j=1}^{N} \sum_{\tau=1}^{\infty} \tau p_{ij}^{k} f_{ij}^{k}(\tau) \tag{2.7}$$

It will be assumed that $f_{ij}^{k}(\tau)$ is such that ν_i^k is finite for all permissible actions k and all states i. This assumption is denoted by the statement

$$\nu_i^k < \infty \quad \text{for all i, k.} \tag{2.8}$$

2.2.2 Control Policies and Returns

A control policy or strategy is the means of specifying which control action is to be taken at the beginning of each transition. Given an observed state i, one can in general select a control action by a random choice among the actions permitted for that state, the set A_i. Moreover the probability governing the random selection of a given action k may depend explicitly on time.

The control policy is especially simple however if it is stationary and deterministic. In this case a unique action is specified for each state, and it is to be taken at any time a transition begins in that state. Such a policy is simply described by a table which lists the

control action associated with each of the possible states of the system. See Fig. 2.4. One objective of this report is to determine sufficient conditions for which one need only consider such policies in attempting to optimize system performance.

In the formulation we are now defining, the performance of the system during a transition interval is measured by a "reward" or "return" received as time proceeds. The return received in each unit time step during a transition can be arbitrary, except for the following limitations. The dependence of the return on the past history of the system operation may be described knowing only the state and the action taken at the beginning of the transition. In other words, the return has the Markov property. The return may additionally depend upon the end state j of the transition and the duration τ of the transition interval. Thus the return received for the m^{th} unit time step during a transition may be denoted $r_{ij}^{k}(m|\tau)$. We assume that $r_{ij}^{k}(m|\tau) = 0$ when $m > \tau$ for all i, j, k, τ, so that a transition will contribute a return only while it is in progress.

Later in this work we are interested in the expectation of the return contributed by a transition at the m^{th} unit time step after it begins. For a transition from state i with action k, this expectation is denoted $u(m|i,k)$ and

$$u(m|i,k) = \sum_{j=1}^{N} p_{ij}^{k} \sum_{\tau=m}^{\infty} r_{ij}^{k}(m|\tau) f_{ij}^{k}(\tau) \quad (m = 1,2,3\ldots) \qquad (2.9)$$

Index i of Observed State	Index k of Action to be Taken
1	2
2	1
3	2
4	3
5	1
6	2
7	3
8	1

Fig. 2.4. Example of tabular representation of a stationary deterministic control policy.

This definition accounts for the possibility that the transition ends before m steps have elapsed, in which case no return would be received then from the transition. The expectation of the total return from a transition, called hereafter the one-transition return, is denoted as b_i^k. This is the sum of the expected returns at every unit step after the transition begins, so

$$b_i^k = \sum_{m=1}^{\infty} u(m|i,k) \qquad (2.10)$$

We will assume for convenience in obtaining the results of Chapters III and IV that the series in Eq. (2.10) is absolutely convergent, i.e.,

$$\sum_{m=1}^{\infty} |u(m|i,k)| < \infty \quad \text{for all } i \text{ and } k. \qquad (2.11)$$

An example will be given in Section 4.2, Chapter IV, which demonstrates that this is not a necessary condition for these results.

The complete specification of the decision process requires giving the values of N, K_i, p_{ij}^k, $f_{ij}^k(\tau)$ and $r_{ij}^k(m|\tau)$ for all i, j, k, τ, m. With this information one can proceed to consider optimization of the process. It will be seen later that N, K_i, p_{ij}^k, v_i^k, and b_i^k suffice for the optimization problem of principal interest in service systems.

2.2.3 Optimization

The performance of the system over a period of time is measured by the sum of the separate returns for the unit time steps in the interval.

The expectation of the total return will depend upon the initial state and time at which the process begins, and the control policy which is used.

There are several optimization problems which are pertinent to the background and application of the decision process. It is convenient now to characterize the optimal policy for each case. However the significance of these definitions can be questioned at this point on mathematical grounds. The required mathematical properties will be established in Chapters III and IV.

Let the symbol π denote any one of the set of admissible control policies. A precise specification of this set will be given in Chapter III. Assume for the present that the decision process begins at time 0, with a transition beginning in state $i(i = 1,2,\ldots N)$. Define $R_n(i,\pi)$ as the expectation of the return received at the n^{th} time step of the process with this starting condition when policy π is used. Then letting $V_T(i,\pi)$ denote the expectation of the total return of the process at time T, one has

$$V_T(i,\pi) = \sum_{n=1}^{T} R_n(i,\pi) \qquad (2.12)$$

A worthwhile optimization problem arises with the above definitions.

Finite-Time Optimization: For given i and T, find the policy π which yields a maximum value for $V_T(i,\pi)$.

If one considers an infinite interval of time $(T \to \infty)$, corresponding to indefinite operation of the system, $V_T(i,\pi)$ will not in general be

finite for any admissible policy. One cannot define optimization in the same way as the finite time case, and still have a meaningful problem. It is natural to inquire in this case whether the average expectation per unit time

$$A_T(i,\pi) = V_T(i,\pi)/T \tag{2.13}$$

remains finite as $T \to \infty$. Generally this is true for every policy.[1]

It is very tempting here to suggest defining optimization as producing a maximum value for the limit of $A_T(i,\pi)$ as $T \to \infty$. For an arbitrarily chosen policy however a limit may not exist. Rather $A_T(i,\pi)$ may oscillate within finite bounds for increasing values of T.[2] Nevertheless there will be points within the oscillation interval such that within any distance of these points there are infinitely many points of the sequence $\{A_T(i,\pi),\ T = 1,2,\ldots\}$. Consider the inferior limit[3] or lim inf of the sequence. Only a finite number of the values $A_T(i,\pi)$ are strictly less than the lim inf. It is therefore consistent to define the following problem.

Infinite-Time Optimization: Find the policy π which yields a maximum value of

$$g_\ell(i,\pi) = \liminf_{T \to \infty} \frac{V_T(i,\pi)}{T} \tag{2.14}$$

---

[1] See Chapter III, Section 3.2.4.

[2] The function $\sin \pi T/2$ $(T = 1,2,3\ldots)$ gives an example of such behavior. Its bounds are +1, -1 of course.

[3] See [56], p. 41-42.

Finally there are circumstances, notably in applications to economics and management science, where one wishes to consider infinite-time optimization but with "discounting" of the return received in the distant future. Discounting reduces the value of the return received at the n^{th} time step by a factor α^n, where $0 < \alpha < 1$. The expectation of the total return with discounting, conditional on the initial state i, is

$$D_\alpha(i,\pi) = \sum_{n=1}^{\infty} \alpha^n R_n(i,\pi) \qquad (2.15)$$

We then define the following problem.

Discounting Optimization: For given i and α, find the policy π which yields maximum value of

$$D_\alpha(i,\pi), \quad 0 < \alpha < 1.$$

The return function with discounting, $D_\alpha(i,\pi)$, can also be viewed as the generating function[1] of the sequence of unit time returns $R_n(i,\pi)$. This interpretation is more appropriate to its use in this study. Moreover by substituting $z = 1/\alpha$, the return function with discounting is seen to be the z-transform[2] of the time sequence of unit step returns. Hence, although discounting has a physical interpretation, it is also a well known mathematical approach to discrete-time problems. We will continue to use the term "discounting" because of its prevalence in the background of the Markov decision processes.

[1] See Feller [4], p. 248.

[2] See Kaplan [57], p. 375.

2.2.4 System Behavior as a Stochastic Process

This report will deal exclusively with the decision process we have just defined. There are other types of Markov decision processes to be found in the literature, which is reviewed in the next section.

When the control policy to be used has been specified, the state of the system as a function of time can be viewed as a stochastic process. When also the control policy is stationary and deterministic, the stochastic process is most easily described. Letting k_i denote the action specified for state i by the policy, the pertinent data are the transition probabilities $p_{ij}^{k_i}$ and the transition time probabilities $f_{ij}^{k_i}(\tau)$. Depending upon the random variables of interest, there are several stochastic processes associated with the system.

Consider the random variable X_m, X_m = the observed state of the system at the beginning of the m^{th} transition of the process, m = 1,2,... . The m^{th} transition may begin at any unit time step $\geq$ m-1. The sequence of random variables $\{X_m\}$ is a Markov chain with transition probabilities $p_{ij}^{k_i}$ (see Pyke, [58], Lemma 3.1, p. 1233.)

Next consider the random variable Y_n, Y_n = the observed state of the system at unit time n (n = 0,1,2,...). The sequence of random variables $\{Y_n\}$ is a Semi-Markov process ([58], Definition 3.5, p. 1234). It is fundamentally determined by the transition distribution $q_{ij}(\tau|k_i)$, see Eq. (2.2).

A Semi-Markov process is therefore a Markov chain with the added complexity which allows transitions to be of random duration. The

Markov chain $\{X_m\}$ is called the underlying Markov chain of the Semi-Markov process $\{Y_n\}$. The contrast between Y_n and X_m is illustrated in Fig. 2.5.

An additional stochastic process is obtained by considering the random variables

$U_j(T)$ = the number of times state j has been entered up to and including time T, (T = 1,2,3,...),

and the vector random variable

$$\underset{\sim}{\xi}(T) = (U_1(T), U_2(T), \ldots, U_N(T))$$

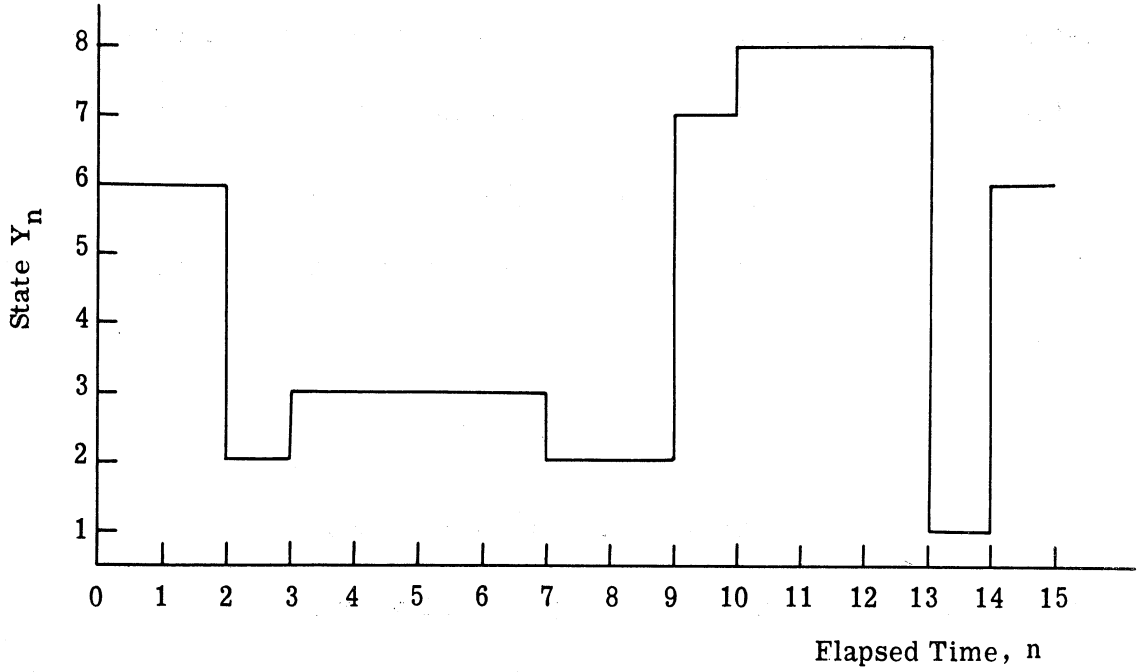
$\underset{\sim}{\xi}(T)$ is a Markov-Renewal process ([58], p. 1234). Pyke has shown that the Semi-Markov process and the Markov-Renewal process are equivalent ways of looking at the same stochastic phenomenon.

It will be fruitful at one point or another to view the decision process via each of these stochastic processes. Note that the problem treated in Section 1.3 is a Semi-Markov process, but a continuous-time rather than a discrete-time process.

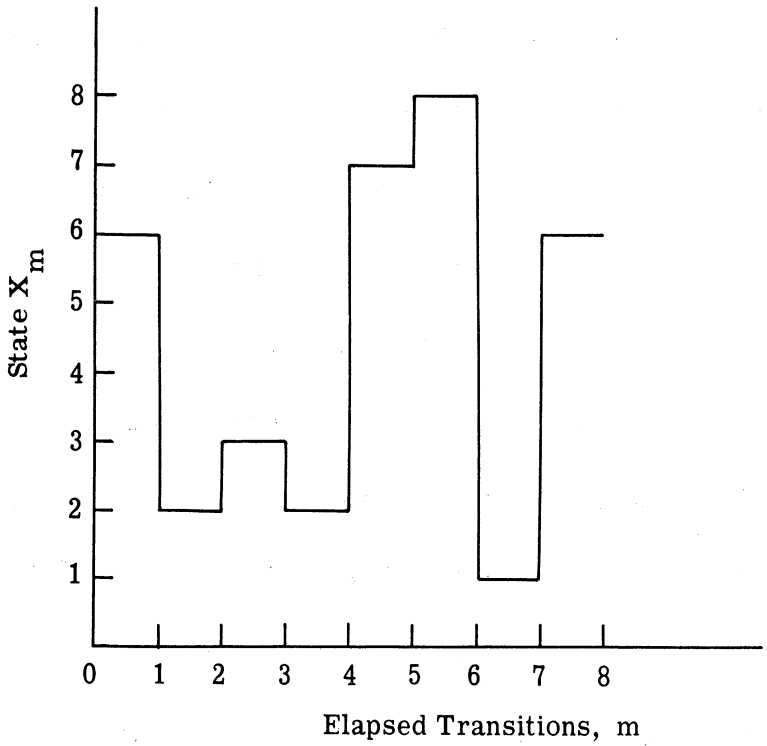
2.3 BACKGROUND ON MARKOV DECISION PROCESSES

A special case of the Semi-Markov decision process arises when all transition intervals are of constant one unit duration. Most of the past efforts have concentrated on this case.

The earliest reported treatment of such a decision process over infinite time is due to R. Bellman [59]. Considering only stationary



(a) An 8-state discrete Semi-Markov process.



(b) The associated Markov chain.

Fig. 2.5. Behavior of a Semi-Markov process and its associated Markov chain.

deterministic policies, Bellman was able to show that the expected total return has a linear asymptotic behavior for large T, i.e., for $T \to \infty$,

$$V_T(i,\pi) \to gT + v_i (i = 1,2,\ldots N)$$

Here both v_i and g depend upon the policy π, but g is independent of i. By this result

$$\lim_{T\to\infty} A_T(i,\pi) = g$$

for all i. Bellman's proof requires however that every stationary policy have a positive transition probability matrix, i.e., $p_{ij}^k > 0$ for all i, j, and k. Later Chuang [62] extended Bellman's proof to the less restrictive case where some finite power of the transition probability matrix is positive. This means that the Markov chain describing system behavior under any stationary policy is irreducible and aperiodic (see [46], p. 256). Results of this kind are important in ascertaining whether a proposed decision process can be treated with the theory.

Bellman's and Chuang's results ensure that continued iteration by the dynamic programming algorithm [60,61] will converge to yield the optimal values of g and v_i. However a great many iterations may be necessary and it may be difficult to determine when convergence has occurred.

This realization motivated R. Howard in his well known work [63]. Like Bellman, Howard considered only stationary, deterministic policies. Using the z-transform technique he established the linear asymptotic behavior for the above decision process under the most general conditions. This behavior is explicitly utilized in the computational technique now

known as the Howard algorithm for infinite time optimization. The Howard algorithm is much superior to the dynamic programming method, which Howard calls "value iteration". By contrast, Howard's method is a "policy iteration". This will be clarified in Chapter IV where Howard's method is introduced. Howard also considered a continuous-time decision process, and demonstrated the application of these processes to baseball strategies, automobile replacement, and taxi cab operation.

Blackwell [64,65] and Derman [66,67] are apparently the first to have considered infinite time optimization without an ad hoc restriction to stationary, deterministic policies. Blackwell in [65], restricting consideration to deterministic policies, proves that discounted time optimization can be achieved with a stationary policy regardless of the properties of the transition probability matrices of admissible policies. This is shown to be true for all values of the discount factor $0 < \alpha < 1$. Derman [66] proves without restriction that there exists an optimal policy which is stationary and deterministic for infinite-time optimization.

In addition to the dynamic programming algorithm and the Howard algorithm, the Simplex method of linear programming has also been suggested for solving the Markov decision process with constant transition times [66,68,69]. In connection with Manne's paper, Wagner argues that an optimal policy is deterministic [70] by virtue of the properties of a linear programming solution. Also de Ghellinck [71] has suggested an algorithm based on manipulations of the graph depicting the possible transitions in state. This does not seem to be a feasible approach for

large systems however. Finally, White [72] has proposed an iterative algorithm which uses the recursive calculation of dynamic programming, but employing the linear asymptotic behavior to accomplish a substantial improvement.

The discrete Semi-Markov decision process which was defined in Section 2.2 is a special case of the process introduced recently by W. S. Jewell [73] and Howard [74]. In the process treated by Jewell, the transition time intervals need not have integer values, but may vary continuously. Jewell and Howard, again considering only stationary deterministic policies, extend Howard's algorithm to cover this decision process where transition times are of random duration. De Cani [75] has also treated the extension of Howard's algorithm for infinite-time optimization of this process.

Although much of the past development of Markov decision processes has been motivated by operations research and management science applications, significant contributions have also come from control systems engineering. Eaton and Zadeh [76] have developed the theoretical formulation for decision processes which have constant transition times and terminate by entry to an absorbing state. Florentin [77], Astrom [78], and Feldbaum [79] have also studied such processes, the latter two treating the case where the state of the system cannot be accurately determined for decision making. De Leve [80] has shown a formulation for a decision process with a continuous state variable, and an algorithm analogous to Howard's.

The reported cases were Markov decision processes have been applied to practical problems are not extensive. Howard [63], Jewell [73], and de Cani [75] give numerical results for problems of baseball strategy, automobile replacement, and machine repair. Except for Howard's, these results are merely illustrative of the computations required. Howard treats the largest system, an automobile replacement model having 40 states. Derman [81,82] discusses the mathematical formulation for machine replacement problems, as does Manne [68] for an inventory control problem. A serious attempt to apply the method to business strategy over finite time was made by de Vries [83] using actual statistics on the machine tool industry.

The earliest suggestion that Markov decision processes would be of value in the design of service systems seems to have been made by Rutenberg [84]. Rutenberg limits the suggestion however to optimizing the economic return from systems in which the number of servers is varied, e.g. the checkout counters of a supermarket. Chuang [62] observes the general merit of the approach and, in particular, its potential value for computer scheduling.

2.4 OBJECTIVES AND RESULTS

The remainder of this report is devoted to development of the Markov decision process defined in Section 2.2 as a useful optimization technique. This development must deal with both the theory of the process and the considerations involved in applying it to a practical situa-

tion. More specific objectives are outlined below.

2.4.1 Theory

In Chapters III and IV the decision process will be considered with a view to infinite-time optimization, with and without discounting. The process will be formulated in a very general way and consideration will begin at the most basic level. Using an approach contained in a rather abstract paper by Karlin [85], the existence of an optimal policy with discounting will be demonstrated. Having this result, the dynamic programming formulation may be derived. This is used to prove that a stationary policy exists which is optimal for any initial condition. It is then shown that an optimal policy is deterministic.

In Chapter IV, the method suggested by Derman [66] will be used to prove the principal theoretical result, establishing the existence of a stationary deterministic optimal policy, without discounting. In this proof, the results obtained in Chapter III are vital. Our consideration of discounted return is largely as an avenue to obtain the principal theorem of Chapter IV. The work of Howard [74] and Jewell [73] on a computational algorithm will then be reviewed to complete the theory. Some original results on the effect of round-off errors in Howard's algorithm are also presented.

Most of the theoretical results in Chapters III and IV have not been given before for this particular decision process. The need for these results can be pointed out solely with regard to the principal

theorem. One could of course assume as other authors have that only stationary, deterministic policies need to be considered. This assumption has appeal for two reasons. First, as a practical matter one might not consider implementing non-stationary or non-deterministic policies because of their additional complexity. Second, intuition may lead one to correctly surmise that the decision process must have an equilibrium behavior, in which a stationary policy is optimal. On the other hand, so long as the possibility exists that a non-stationary or non-deterministic policy might lead to improved performance, designers may be lead to an erroneous justification for implementing such controls. Further, an ad hoc restriction to stationary deterministic policies in conducting a study is an unnecessary qualification on the results, based on our conclusion. Finally and most importantly, applying any mathematical theory while supposing facts not rigorously evident may lead to unperceived errors. The theory we present is necessary understanding for avoiding such pitfalls. Moreover the theory suggests that the principal result is more widely applicable than one might suppose on the basis of intuitive ideas. Several examples will be given to illustrate the complexity of systems covered by the theorems.

2.4.2 Application

An equally important contribution of this dissertation is to show that a Markov decision process can lead to worthwhile practical conclusions regarding queue control procedures. In Chapter V we shall

formulate a Markov decision model for multi-level queue control in a multiple-access computer. This model is a discrete Semi-Markov decision process, as we have defined it. It incorporates all queue control policies which are presently in use in actual computer systems, except the "dynamic round-robin" used by Schwartz, et al. [25]. But a large number of other stationary policies are also treated, demonstrating the capability of the formulation. The principal limitation is that only a small number of remote consoles can be feasibly studied. We shall deal with a 4 console model, which although not unreasonable at the present time (see [20,28]), will not be representative of future multiple-access systems. Still, the general conclusions we obtain seem equally valid for larger systems, based upon the intuitive justification which develops.

In order to carry out the optimization of the queue control for this computer model, a computer program has been developed. This program implements the Howard algorithm, for both discrete-time and continuous-time Semi-Markov decision processes. Although little effort was expended to achieve an exceedingly efficient program design, the solution times for optimization of the queue control model are quite competitive with the solution times for analysis [55] of a comparably large model. Often they are less. Thus the computation involved in treating Markov decision processes is reasonably modest, and shows promise for future application of the technique.

The specific results obtained for the scheduling optimization are presented in Chapter VI.

CHAPTER III

THE DECISION PROCESS WITH DISCOUNTING

The purpose of this chapter is to establish certain properties of the decision process defined in Chapter II, when the future return of the process is discounted in value. These will lead to the principal theorem on infinite-time optimization to be proven in Chapter IV.

The first substantial result of this chapter is to establish the existence of an optimal policy in the manner suggested by Karlin [85]. This is accomplished by showing that the discounted total return of the process is a continuous function on the set of admissible policies. To this end section 3.1 describes a precise characterization of the set of all policies as a metric space and as a topological product space. Section 3.2 is devoted to the formulation of the expected total discounted return as a function of an arbitrary policy. This formulation depends upon viewing the Semi-Markov process as a two-dimensional Markov process, under the assumed policy. The existence theorem is established in section 3.3.

The last part of this chapter deals with the nature of an optimal policy for discounting optimization. An important result, obtained in section 3.5, is the existence of an optimal policy which is stationary, deterministic, and uniformly optimal for any initial condition of the decision process. This theorem stems from a functional equation of the type used in dynamic programming, derived in section 3.4.

3.1 THE SET OF ADMISSIBLE POLICIES

The proof of existence of an optimal policy hinges upon a precise mathematical description of all possible control policies for the decision process. The important properties of this set are developed in the following.

In Chapter II it was stated that a control action could generally be determined at any time by a random selection among the actions permitted for the state being observed. Such random selection is governed by assigning a probability to each of the possible actions. Suppose state i is observed at the beginning of a transition and define

$$\eta(k|i) = \text{probability of selecting action k when state i is observed} \tag{3.1}$$

Because some control action must be selected and there are K_i different actions,

$$\sum_{k=1}^{K_i} \eta(k|i) = 1, \quad i = 1,2,\ldots N$$

The random selection can then be described by a vector e_i of probabilities,

$$e_i = (\eta(1|i), \quad \eta(2|i), \quad \eta(3|i),\ldots, \quad \eta(K_i|i)) \tag{3.2}$$

The state observed at the beginning of a transition could be any one of the N possible states of the system. Because the state which will occur is not known in advance, specification of the action to be taken at any particular time requires a set of N probability vectors, one for each

state. An N-tuple

$$h = (e_1, e_2, \ldots e_N) \tag{3.3}$$

can be conveniently viewed then as a single decision in a control policy. We shall continue to use the term "decision" whenever the control action is not uniquely specified, but depends upon a random selection.

The transitions in state occurring during the system operation may begin at any integer time value. A control policy must therefore specify a decision for every value of n. For any n the decision given could depend upon n, so h_n will denote the decision specified for time n. The action selection probabilities will similarly be denoted by $\eta_n(k|i)$. A control policy can then be viewed as a sequence

$$\pi = (h_0, h_1, h_2, \ldots, h_n, \ldots) \tag{3.4}$$

Now we proceed to further characterization of the set E_i of all possible vectors e_i, the set H of all possible decisions h, and the set P of all possible policies π.

3.1.1 The Sets E_i

For each i the symbol E_i will denote the set of all possible vectors e_i. Every vector e_i is a probability vector of dimension K_i. Thus E_i is a set of points in Euclidean space of K_i dimensions. With $\eta(1|i)$, $\eta(2|i)$, ..., $\eta(K_i|i)$, the 1st, 2nd,..., K_i^{th} co-ordinate variables, E_i is the set of all points such that

$$1 \geq \eta(k|i) \geq 0, \quad k = 1,2,\ldots,K_i, \tag{3.5}$$

and

$$\sum_{k=1}^{K_i} \eta(k|i) = 1, \quad i = 1,2,\ldots,N. \tag{3.6}$$

The properties of E_i of special interest arise when we consider E_i as a metric space by associating a distance between any pair of points. A convenient and familar metric function to assign on E_i is the Euclidean distance (Rudin [56], p. 27). With respect to this metric the sets E_i are uniformly bounded, and closed ([56], p. 28). By a well known theorem ([56], p. 35), E_i is therefore a compact set for every i.

The metric associated with E_i defines a class of subsets of E_i called open sets (Simmons [86], p. 59). These open sets constitute a topology on E_i, known as the usual topology on a metric space ([86], p. 92). Thus E_i can be considered a compact topological space for every i.

The concept of a continuous function defined on a compact space leads to some useful properties, and extends the familiar case where the function is real-valued and defined on an interval of the real line. A closed, bounded interval on the real line is an example of a compact set. We know that a continuous function defined on a closed, bounded interval is bounded, and attains its greatest lower bound (g.l.b.) and its least upper bound (l.u.b.) at some points in the interval. This theorem extended to a metric space is ([56], p. 77):

Theorem 3.1: Suppose f is a continuous real-valued function on a compact metric space X, and

$$M = \text{l.u.b.}_{x\epsilon X} f(x), \quad m = \text{g.l.b.}_{x\epsilon X} f(x)$$

Then there exist points x, yϵX such that f(x) = M, f(y) = m.

This theorem will be important later on.

3.1.2 The Set H

The symbol H will denote the set of all decisions which may be specified for any time during the decision process. It is the set of all ordered N-tuples

$$h = (e_1, e_2, \ldots, e_N) \tag{3.3}$$

where $e_i \epsilon E_i (i = 1,2,\ldots,N)$. By this definition H is the cartesian product of the sets $E_1, E_2, \ldots, E_N$, written

$$H = E_1 \times E_2 \times \ldots \times E_N \tag{3.7}$$

([86], p. 23). The E_i are called the component spaces of the product space H.

It is important to establish that H is also a compact space. For this purpose one can call upon the Tychonoff theorem ([87], p. 143):

Theorem 3.2 (Tychonoff): The cartesian product of a collection of compact topological spaces is compact relative to the product topology.

The Tychonoff theorem does not require that the collection of component spaces be finite. It establishes that H is a compact topological space with the product topology ([86], p. 115ff. or [87], p. 90).

3.1.3 The Set P

The policy space P is the set of all sequences

$$\pi = (h_0, h_1, h_2, \ldots, h_n, \ldots) \tag{3.4}$$

where $h_n \epsilon H$ for all n. Any such sequence is a control policy for the process, and the space P includes all conceivable policies. From the above definition one can represent P as an infinite cartesian product

$$P = H \times H \times H \ldots \tag{3.8}$$

Inasmuch as H is a compact topological space, the Tychonoff theorem can again be used to establish that P is a compact topological space with the product topology. However it is convenient in dealing with the expected total return as a function on P to treat P as a compact metric space. This can be done providing the metric is chosen so that the usual topology on P as a metric space is the same as the product topology on P.

Two theorems in Kelley [87] ensure that a suitable metric on P exists. Let $d_i(e,\hat{e})$ be the Euclidean metric for points e and $\hat{e}$ in the set E_i, and define a new metric

$$m_i(e,\hat{e}) = \min(1,d_i(e,\hat{e}))$$

Theorem 13 of Kelley ([87], p. 121) establishes that the metric space[1] (E_i,m_i) has unit diameter and the same topology as the space (E_i,d_i). Then Theorem 14 of Kelley ([87], p. 122) proves that for component spaces of unit diameter, the metric

$$d_H(h,\hat{h}) = \sum_{i=1}^{N} \frac{1}{2^i} m_H(e,\hat{e})$$

[1]This notation indicates that the metric space is comprised of the set E_i with the metric m_i.

on the product space H generates the same open sets as the product topology on H. This argument can now be repeated to establish a metric m_H such that (H,m_H) has unit diameter, and is homeomorphic[1] to (H,d_H). A similar formula to the above yields a metric d_P on P such that the usual topology is the product topology,

$$d_P(\pi,\hat{\pi}) = \sum_{n=o}^{\infty} \frac{1}{2^n} m_H(h_n,\hat{h}_n)$$

Thus P is a compact metric space.

3.2 BASIC RELATIONS AND THE OPTIMIZATION CRITERIA

The goal of this section is a formulation of the expectation of the total return of the decision process as a function of an arbitrary policy π from the set P. As indicated earlier, we let $\eta_n(k|i)$ be the generic symbol for the action selection probabilities of the decision h_n.

Although a policy must give a decision for every unit time n, a given decision h_n will have no effect on the return during actual system operation unless a transition happens to begin at time n. This is a random event, and in order to formulate the expectation of return we must consider the probability of such random events. The first part of this section deals with a pair of random variables whose joint probability distribution provides this information.

---

[1]That is the topologies of the two spaces are equivalent, see [86], p. 93.

After the formulation of the return has been achieved we will reconsider the optimization criteria introduced in Chapter II, giving a more precise statement of each, and developing some properties which make them meaningful optimization objectives.

3.2.1 The Random Variables

Our basic approach, as indicated in Chapter II, treats the total return as the sum of one-step returns received in each unit time step of system operation. We need then to express the expectation of the return received in an arbitrary unit time step after the policy π has been placed in operation. This one-step return for time n is contributed by a transition which may begin at any time $m < n$. Thus we must determine the probability of the random event that the system enters state i at time m, for all i and m. Beyond that, however, we must ascertain how this probability is affected by the sequence of decisions $h_0, h_1, \ldots, h_{m-1}$ of the policy π.

The way to begin is to relate the random event to a set of random variables which describe the time behavior of the system. Although several choices are perhaps available, we have found a suitably elegant formulation to result from using the random variables Z_m and W_m, defined as

Z_m = the state entered at the first transition beginning at time m or later,

W_m = the time until the beginning of a transition, measured forward from time m,

for $m = 0,1,2,3,\ldots$. The variable Z_m takes values in the set $i = 1,2,3,$

...N, while W_m may have value 0,1,2,... for any m. Figure 3.1 illustrates the values of the random variables for a sample period of operation of a five state system, and shows their relation to the Semi-Markov process Y_m defined in Chapter II. Note that in actual system operation the values of Z_m and W_m may not be observable at time m, for they depend upon a transition which may begin later than m. The exception occurs if a transition begins at m. Then $W_m = 0$ and $Z_m = i$ if state i is entered. The joint probability distribution of Z_m, W_m therefore provides the information needed to formulate the expected return.

The variables Z_0, W_0 demand special attention. In Chapter II we limited consideration to the case where the decision process begins in a known state i at time 0. Z_m and W_m allow us to introduce some additional generality in the form of a priori uncertainty regarding the beginning of the process. Z_0 now accounts for uncertainty in the initial state of the process. W_0 accounts for uncertainty regarding the time of the first decision of the process, relative to the time origin from which the policy π is specified. Z_0 and W_0 are specified by a joint probability distribution

$$\psi_0(j,v) = \Pr(Z_0 = j, \quad W_0 = v) \qquad (j = 1,2,...N; \quad v = 0,1,2,...)$$

which is to be given as additional data for the decision process. We assume of course that

$$0 \leq \psi_0(j,v) \leq 1 \qquad (3.9a)$$

and

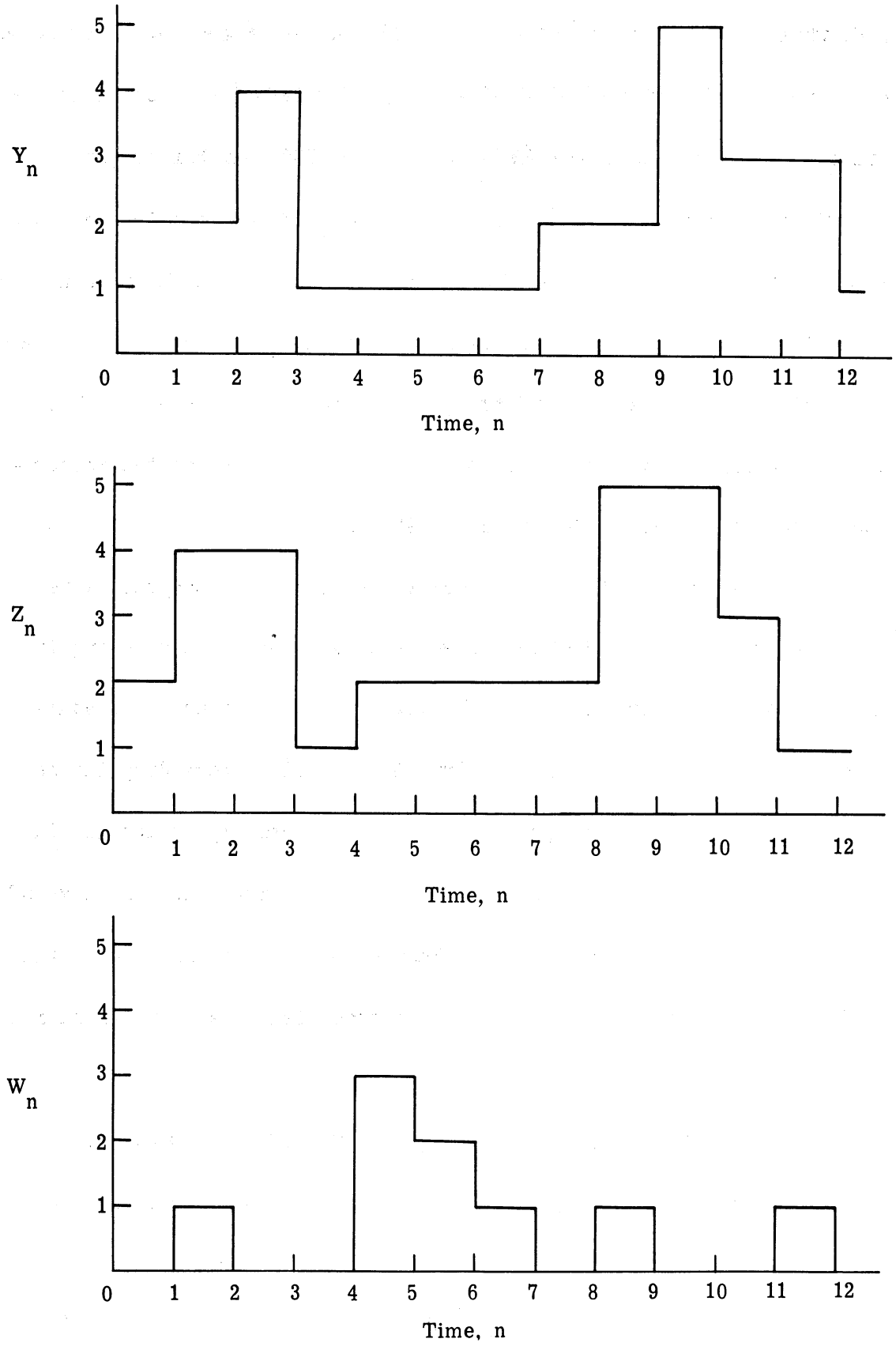


Fig. 3.1. Illustration of the random variables Y_n, Z_n, W_n in a 5-state process.

$$\sum_{j=1}^{N} \sum_{v=o}^{\infty} \psi_o(j,v) = 1 \qquad (3.9b)$$

Moreover, we assume that there is no return from the process between time 0 and the beginning time W_o.

The vector random process $\{(Z_m,W_m),\ m = 1,2,\ldots\}$ is a Markov process. For suppose that at time m it is given that $Z_m = i$, and $W_m = u$. If $u \geq 1$, then it is clear that $Z_{m+1} = i$, and $W_{m+1} = u-1$. But if $u = 0$, then

$$Pr(Z_{m+1} = j,\ W_{m+1} = v | Z_m = i,\ W_m = 0) = \sum_{k=1}^{K_i} \eta_m(k|i) q_{ij}(v+1|k)$$

Despite the fact that Z_m and W_m are not always observable, the above is an important point. This will become clear as we discuss how the joint probability distribution of these random variables is affected by the policy π.

3.2.2 The Set S

Define the following sets,

I_N = the finite set of integers 1,2,...,N,

I = the set of all integers 0,1,2,... .

The set S consists of all probability distributions defined over I_N x I. To explain further, let s denote an element of S. For every pair of values $(j,v) \in I_N$ x I, the element s assigns a value $\psi(j,v)$ such that

$$\left.\begin{aligned} & 0 \le \psi(j,v) \le 1 \\ \text{and} \quad & \\ & \sum_{v=0}^{\infty} \sum_{j=1}^{N} \psi(j,v) = 1 \end{aligned}\right\} \qquad (3.10)$$

The joint probability distribution on the random variables Z_m and W_m can therefore be taken as an element of S, for m = 0,1,2,... In particular we let s_o denote the distribution for Z_o, W_o, so that

$$s_o = \{\psi_o(j,v), \quad j = 1,2,\ldots,N; \quad v = 0,1,2,\ldots\}$$

Similarly, s_m will denote the distribution for Z_m, W_m. The set S can be metrized by associating a distance between elements s, $\hat{s}$ given by

$$d_S(s,\hat{s}) = \sum_{v=0}^{\infty} \sum_{j=1}^{N} |\psi(j,v) - \hat{\psi}(j,v)| \qquad (3.11)$$

So S is a metric space.

The probabilities comprising the distribution s_m will be denoted by[1] $\psi_m(j,v)$,

$$\psi_m(j,v) = \Pr(Z_m = j, \quad W_m = v)$$

The element s_m of S depends upon the given distribution s_o and the policy π, but we will leave this as an implicit fact temporarily for we can show that s_m is determined when s_{m-1} and h_{m-1} are given. This is a consequence of $\{(Z_m,W_m)\}$ being a Markov process. First we define the transition distribution for time n resulting from the decision h_n of the policy π,

[1] A more conventional notation perhaps is $s_m(j,v)$ rather than $\psi_m(j,v)$.

$$q_{ij}(v|h_n) = \sum_{k=1}^{K_i} \eta_n(k|i) q_{ij}(v|k) \tag{3.12}$$

Also, to shorten notation somewhat we use

$$\psi_m(j) = \psi_m(j,0) \tag{3.13}$$

Then we have the equation

$$\psi_m(j,v) = \psi_{m-1}(j,v+1) + \sum_{i=1}^{N} \psi_{m-1}(i) q_{ij}(v+1|h_{m-1}) \tag{3.14}$$

for $j = 1,2,\ldots,N$, and $v = 0,1,2,\ldots$ This equation gives the probability of the event that at time m there are v time steps until a new transition begins, and that the state entered is j. This event comes about if a transition from any state i is just beginning at time m-1, which will last v+1 time steps and move the system to state j. The probability of this is given by the summation on the right side of (3.14). But one may also have a transition already in progress at time m-1, and this produces the event in question if it lasts for v+1 steps beyond m-1 and moves the system to j. This probability is the first term on the right of (3.14).

Based on (3.14) we may formally write in operator notation,

$$s_m = f_1(h_{m-1}) s_{m-1} \tag{3.15}$$

for $m = 1,2,\ldots$, where f_1 operates on s_{m-1} to effect the transformation of (3.14). The transformation (3.15) is exactly the type defined by Karlin [85] in treating the general structure of sequential decision

models.[1] Although s_m is not observable from the process, (3.15) reveals that it can be calculated given the policy π and the initial distribution s_o.

3.2.3 Formulation of Return

The initial probability distribution s_o defines the beginning of the decision process relative to a time origin from where the policy π is specified. The policy is actually placed in operation at time W_o. We are concerned with the expectation of the total return received from the decision process at the end of T unit time steps from this origin. This depends upon s_o and π, and is denoted $V_T(s_o,\pi)$. The total return is the sum of the returns for each of the unit steps comprising the interval T, and so we are lead to consider the expectation of a one-step return. The expectation of the one-step return received at time n will be denoted $R_n(s_o,\pi)$, and we have

$$V_T(s_o,\pi) = \sum_{n=1}^{T} R_n(s_o,\pi) \qquad (3.16)$$

[1] Equation (3.15) represents a deterministic transformation, and in Karlin's framework s_m corresponds to the "state" of the system. From the viewpoint of modern control theory, it is often useful to consider a probability distribution as a "generalized state", in contrast to the observable state which is random. See R. Bellman, Adaptive Control Processes: A Guided Tour, Princeton University Press, Princeton, N.J., 1961, pp. 139-140, for further discussion. Also see H. J. Kushner, "On the Dynamical Equations of Conditional Probability Density Functions, with Applications to Optimal Stochastic Control Theory", J. Math. Analysis and Applications, 8, 332-334, 1964, for an example of this approach.

In order to formulate $R_n(s_o,\pi)$ we need an expression for the expectation of the one-step return received from a transition at the τth time step after it begins. (Note that the transition may end before τ steps, but in that event no return is contributed to the expectation.) If the transition begins at time m from state i, this expectation is $u(\tau|i,h_m)$, given by

$$u(\tau|i,h_m) = \sum_{k=1}^{K_i} \eta_m(k|i)u(\tau|i,k) \tag{3.17}$$

and $u(\tau|i,k)$ defined in Eq. (2.9). Then we have

$$R_n(s_o,\pi) = \sum_{m=o}^{n-1} \sum_{i=1}^{N} \psi_m(i)u(n-m|i,h_m) \tag{3.18}$$

for n = 1,2,3,... .

From (3.17) and (3.18) it immediately follows that

$$|R_n(s_o,\pi)| \leq \sum_{m=o}^{n} \sum_{i=1}^{N} \sum_{k=1}^{K_i} |u(m|i,k)|$$

and then

$$|R_n(s_o,\pi)| \leq \sum_{i=1}^{N} \sum_{k=1}^{K_i} \sum_{m=1}^{\infty} |u(m|i,k)|$$

The assumption stated in Eq. (2.11) now establishes that $R_n(s_o,\pi)$ is uniformly bounded for all n, s_o, and $\pi \in P$.

3.2.4 Optimization Criteria

The three optimization problems introduced in Chapter II are now precisely stated:

a. Finite-Time Optimization: Find a policy $\pi \in P$ which maximizes $V_T(s_o,\pi)$,

$$V_T(s_o,\pi) = \sum_{n=1}^{T} R_n(s_o,\pi)$$

for a given $s_o \in S$.

b. Discounting Optimization: Find a policy $\pi \in P$ which maximizes

$$D_\alpha(s_o,\pi) = \sum_{n=1}^{\infty} \alpha^n R_n(s_o,\pi) \tag{3.19}$$

for a given α, $0 < \alpha < 1$, and $s_o \in S$.

c. Infinite-Time Optimization: Find a policy $\pi \in P$ which maximizes

$$g_\ell(s_o,\pi) = \liminf_{T\to\infty} \frac{V_T(s_o,\pi)}{T} \tag{3.20}$$

for given $s_o \in S$.

As a first step to establishing the validity of these objectives, it should be proven that the objective functions[1] $V_T(s_o,\pi)$, $D_\alpha(s_o,\pi)$, $g_\ell(s_o,\pi)$ are well defined. First it is easily seen that $V_T(s_o,\pi)$ is bounded for

[1]Here we are borrowing a term from linear programming where the quantity to be maximized is usually called the "objective function."

finite T regardless of s_o and π, inasmuch as $R_n(s_o,\pi)$ is uniformly bounded.

Theorem 3.3: For every $s_o \in S$ and $\pi \in P$

$$g_\ell(s_o,\pi) = \liminf_{T\to\infty} \frac{V_T(s_o,\pi)}{T}$$

is finite.

Proof: Because $R_n(s_o,\pi)$ is uniformly bounded, the average expectation of return $V_T(s_o,\pi)/T$ is uniformly bounded. Therefore $g_\ell(s_o,\pi)$ is finite.

Theorem 3.4: The discounted return

$$D_\alpha(s_o,\pi) = \sum_{n=1}^{\infty} \alpha^n R_n(s_o,\pi)$$

exists for $0 < \alpha < 1$, any $s_o \in S$, and all $\pi \in P$.

Proof: The infinite series above must be proven convergent under the stated conditions. Let R be the least upper bound on $|R_n(s_o,\pi)|$ for given s_o and π. Then for a large integer β and $\gamma > \beta$,

$$\left| \sum_{n=\beta}^{\gamma} \alpha^n R_n(s_o,\pi) \right| \le \sum_{n=\beta}^{\infty} \alpha^n |R_n(s_o,\pi)| \le \sum_{n=\beta}^{\infty} \alpha^n R = \alpha^\beta \frac{R}{1-\alpha}$$

The last term can be made as small as desired for $0 < \alpha < 1$. By the Cauchy criterion ([56], p. 52), the series defining $D_\alpha(s_o,\pi)$ is therefore convergent.

The next requirement is to establish that an optimal control policy indeed exists for each of these problems.

3.3 EXISTENCE OF AN OPTIMAL POLICY

The question of the existence of an optimal policy essentially concerns the behavior of the objective function on the defined set P of admissible policies. An optimal policy is one which maximizes the value of the objective function. But does the objective function have a "maximum" value on the set P? Conceivably the function may behave analogously to the function $f(x) = x$ defined on the semi-open interval of the real line $0 \leq x < 1$. Here for any value x, there exists an $x' > x$ such that $f(x') > f(x)$. Hence $f(x) = x$ has no maximum value on the semi-open interval $0 \leq x < 1$. Of course, $f(x)$ does have a maximum value of unity on the closed interval $0 \leq x \leq 1$.

In order to demonstrate the existence of an optimal policy for the first two optimization problems in section 3.2.4 we will show that the objective functions $V_T(s_o,\pi)$ and $D_\alpha(s_o,\pi)$ are continuous functions on the compact metric space P. Then the existence of an optimal policy follows from Theorem 3.1. The existence of an optimal policy for infinite-time optimization is treated in Chapter IV.

Theorem 3.5: For any $s_o \in S$, s_m is a continuous function of π on the set P, for $m = 1,2,\ldots$.

Proof: In the framework of metric spaces, a function f, defined on a space P and taking values in a set S, is continuous at a point π_o if for any $\epsilon > 0$ there exists $\delta > 0$ such that $d_P(\pi,\pi_o) < \delta$ implies $d_S(f(\pi), f(\pi_o)) < \epsilon$. The function is continuous on the set P if it is continuous at every

point in the set. Alternatively, the function f is continuous at π_o if for every sequence of points $(\pi_1,\pi_2,\ldots,\pi_\nu,\ldots)$ where $\pi_\nu \in P$ for $\nu = 1,2,\ldots$ and $\lim_{\nu\to\infty} \pi_\nu = \pi_o$, we have $\lim_{\nu\to\infty} f(\pi_\nu) = f(\pi_o)$ (Simmons [86], p. 76).

Suppose we have such a sequence of policies $\{\pi_\nu\}$, and let us write $s_m(\pi_\nu)$ to denote the dependence of s_m on the policy π_ν for given $s_o \in S$. Because the metric topology of S is the topology of point wise convergence, $\lim_{\nu\to\infty} s_m(\pi_\nu) = s_m(\pi_o)$ if and only if[1]

$$\lim_{\nu\to\infty} \psi_m(j,v|s_o,\pi_\nu) = \psi_m(j,v|s_o,\pi_o) \tag{3.21}$$

for $j = 1,2,\ldots N$; $m = 1,2,\ldots$; $v = 0,1,2,\ldots$. Moreover $\lim_{\nu\to\infty} \pi_\nu = \pi_o$ if and only if

$$\lim_{\nu\to\infty} [\eta_m(k|i)]_\nu = [\eta_m(k|i)]_o \quad (n = 1,2,\ldots) \tag{3.22}$$

Here $[\eta_m(k|i)]_\nu$ denotes the action selection probability of the decision specified for time m in the policy π_ν.

Assuming (3.22) we proceed to show that (3.21) holds by induction on m. For m = 1, we have from (3.14)

$$\psi_1(j,v|s_o,\pi_\nu) = \psi_o(j,v+1) + \sum_{i=1}^{N} \psi_o(i) q_{ij}(v+1|h_o^\nu)$$

[1]The dependence of $\psi_m(j,v)$ on s_o and π is now made explicit by the notation $\psi_m(j,v|s_o,\pi)$.

where h_o^ν is the decision specified for time 0 by π_ν. The above holds also for v = 0. From (3.12) it is clear using (3.22) that

$$\lim_{\nu\to\infty} q_{ij}(\tau|h_n^\nu) = q_{ij}(\tau|h_n^o)$$

for any n. Thus

$$\lim_{\nu\to\infty} \psi_1(j,v|s_o,\pi_\nu) = \psi_o(j,v+1) + \sum_{i=1}^{N} \psi_o(i)q_{ij}(v+1|h_o^o) = \psi_1(j,v|s_o,\pi_o)$$

Assuming (3.21) holds for all values up to some arbitrary value m (the induction hypothesis), it is easily shown true for (m+1) from (3.14). We conclude that it holds for all m, and s_m is continuous at π_o. Because π_o is an arbitrary element of P, the result is true for all $\pi \in P$ and the theorem is proved.

Theorem 3.6: For any $s_o \in S$, and n = 1,2,...,$R_n(s_o,\pi)$ is a continuous function of π on the set P.

Proof: Consider a sequence of policies $\{\pi_\nu\}$ converging to an element $\pi_o \in P$. From (3.18),[1]

$$R_n(s_o,\pi_\nu) = \sum_{m=o}^{n-1} \sum_{i=1}^{N} \psi_m(i|s_o,\pi_\nu)u(n-m|i,h_m^\nu) \qquad (3.23)$$

But from (3.17) and (3.22) it is seen that

[1]The quantity $\psi_m(i|s_o,\pi_\nu)$ appearing on the right of (3.23) should be interpreted for m = 0 as $\psi_o(i)$, see (3.13).

$$\lim_{\nu\to\infty} u(n\text{-}m|i,h_m^{\nu}) = u(n\text{-}m|i,h_m^{o}).$$

Using this result and Theorem 3.5, it is seen that the summation in (3.23) is a sum of products of continuous functions of π on the set P. Hence

$$\lim_{\nu\to\infty} R_n(s_o,\pi_\nu) = R_n(s_o,\pi_o)$$

for any $s_o \in S$ and $n = 1,2,\ldots$. Since π_o is any element of P, $R_n(s_o,\pi)$ is continuous on P for any s_o.

Theorem 3.7: For any $s_o \in S$ and α such that $0 < \alpha < 1$, there exists at least one policy $\pi_T \in P$ which maximizes $V_T(s_o,\pi)$ and at least one policy $\pi_\alpha \in P$ which maximizes $D\alpha(s_o,\pi)$. An optimal policy therefore exists for the finite-time optimization problem, and for the discounting optimization problem.

Proof: Because of Theorem 3.6,

$$V_T(s_o,\pi) = \sum_{n=1}^{T} R_n(s_o,\pi)$$

is a sum of continuous functions of π, and is therefore a continuous function of π on P for any $s_o \in S$. But P has been shown in section 3.1.3 to be a compact metric space. Appealing to Theorem 3.1, there must exist some policy π_T in P for which

$$V_T(s_o,\pi_T) = \underset{\pi\in P}{\text{l.u.b.}}\ V_T(s_o,\pi).$$

This establishes one part of the theorem.

Now as we have seen there exists a uniform bound R for any s_o such that

$$|R_n(s_o,\pi)| \leq R \quad \text{for all } \pi \in P, \quad n = 1,2,\ldots .$$

But the infinite series

$$\sum_{n=1}^{\infty} \alpha^n R$$

is summable for $0 < \alpha < 1$, being a geometric series. It follows therefore that

$$\sum_{n=1}^{\infty} \alpha^n R_n(s_o,\pi)$$

converges uniformly on the set P for any s_o. (Rudin [56], p. 134). Then

$$D_\alpha(s_o,\pi) = \sum_{n=1}^{\infty} \alpha^n R_n(s_o,\pi)$$

is a uniformly convergent series of continuous functions and is therefore itself a continuous function on the set P. (Rudin, p. 136). Again, by Theorem 3.1 and the compactness of P, there exists some policy $\pi_\alpha \in P$ such that

$$D_\alpha(s_o,\pi_\alpha) = \underset{\pi \in P}{\text{l.u.b.}} \; D_\alpha(s_o,\pi)$$

for given α and $s_o \in S$.

3.4 DYNAMIC PROGRAMMING FORMULATION

Let π_α denote an optimal policy for the discounting optimization problem, and $D^*_\alpha(s_o)$ the optimal return for given α and $s_o \in S$, i.e.

$$D^*_\alpha(s_o) = D_\alpha(s_o,\pi_\alpha) \tag{3.24}$$

As a consequence of the previous section of this report, π_α maximizes $D_\alpha(s_o,\pi)$ and we can write symbolically

$$D^*_\alpha(s_o) = \max_{\pi\in P} D_\alpha(s_o,\pi) \tag{3.25}$$

We now proceed to expand (3.25) to arrive at a functional equation in terms of the first decision h_o of any policy $\pi\epsilon P$. This result will then be used in the last part of this chapter to establish that an optimal policy exists which is stationary and deterministic.

3.4.1 A Functional Equation

In the derivation it is necessary to consider a transformation of a policy π, which replaces the decision h_n at time n by the decision $h_{n+\ell}$ previously assigned for time $n+\ell$, for n = 0,1,2,... and some fixed positive integer ℓ. This left time shift of ℓ units will result in a policy denoted symbolically by $\Delta^\ell\pi$. Thus if

$$\pi = (h_o,h_1,h_2,\ldots,h_n,\ldots)$$

and

$$\pi' = (h'_o,h'_1,h'_2,\ldots,h'_n,\ldots)$$

are two policies in P, then

$$\pi' = \Delta^\ell\pi$$

implies

$$h'_n = h_{n+\ell} \quad (n = 0,1,2,\ldots) \tag{3.26}$$

The following will also involve the notion of a decision process beginning with the initial distribution $s_1 \in S$. Previously we have used $s_0 \in S$ to denote a joint probability distribution $\psi_0(j,v)$ on the indexed state Z_0 and the starting time W_0 at the beginning of the decision process. The point $s_1 \in S$ corresponds to a joint distribution determined by s_0 and decision h_0 of the policy,

$$s_1 = f_1(h_0)s_0$$

as expressed in (3.15). However s_1 can represent an initial distribution equally well as s_0. We shall use $s_0^1 = \{\psi_0^1(j,v)\}$ to denote the initial distribution in this case, i.e. $\psi_0^1(j,v) = \psi_1(j,v|s_0,\pi)$.

Theorem 3.8: For any initial state $s_0 \in S$ and any admissible policy $\pi \in P$, the expectation of return $R_n(s_0,\pi)$ for time n ($n \geq 2$) satisfies

$$R_n(s_0,\pi) = \sum_{i=1}^{N} \psi_0(i)u(n|i,h_0) + R_{n-1}(s_0^1,\Delta\pi) \tag{3.27}$$

where $s_0^1 = s_1 = f_1(h_0)s_0$ as represented in Eq. (3.15).

Proof: Equation (3.27) separates $R_n(s_0,\pi)$ into two parts. The first term on the right side is the part due to the event that the first transition begins at time 0. The second term is simply the rest, but its interpretation is important. What we claim is that if we neglect the part due to a transition beginning at time 0, the expected one-step return at time

n is the same as if we shifted the time origin one unit ahead, used s_1 as the initial distribution, and found the expected one-step return at time n-1 on the new time scale.

From (3.18) we have

$$R_n(s_o,\pi) = \sum_{i=1}^{N} \psi_o(i)u(n|i,h_o) + \sum_{i=1}^{N}\sum_{m=1}^{n-1} \psi_m(i|s_o,\pi)u(n-m|i,h_m) \qquad (3.28)$$

Comparing this to (3.27) it is clear that the second summation must be shown to be $R_{n-1}(s_o^1,\Delta\pi)$. We begin by proving that

$$\psi_m(j,v|s_o,\pi) = \psi_{m-1}(j,v|s_1,\Delta\pi) \qquad (3.29)$$

for m = 1,2,... This holds by definition for m = 1, since

$$\psi_o(j,v|s_1,\Delta\pi) = \psi_o^1(j,v) = \psi_1(j,v|s_o,\pi).$$

For other values of m, the proof is conveniently done using the properties of the operator $f_1(h_m)$ introduced in (3.15). Let us write

$$f_1(h_{m-1}) = T_m(\pi), \quad m = 2,3,...$$

and note that

$$T_m(\pi) = T_{m-1}(\Delta\pi)$$

By repeated application of the operator the dependence of s_m on s_o and π is explicit in

$$s_m(s_o,\pi) = T_m(\pi)T_{m-1}(\pi)...T_1(\pi)s_o$$

But then

$$s_m(s_o,\pi) = T_{m-1}(\Delta\pi)T_{m-2}(\Delta\pi)\ldots T_1(\Delta\pi)T_1(\pi)s_o$$

$$= T_{m-1}(\Delta\pi)T_{m-2}(\Delta\pi)\ldots T_1(\Delta\pi)s_1 = s_{m-1}(s_1,\Delta\pi)$$

which proves (3.29). In (3.28) we can then write

$$R_n(s_o,\pi) = \sum_{i=1}^{N} \psi_o(i)u(n|i,h_o)$$

$$+ \sum_{i=1}^{N}\sum_{m=1}^{n-1} \psi_{m-1}(i|s_1,\Delta\pi)u(n-m|i,h_m)$$

$$= \sum_{i=1}^{N} \psi_o(i)u(n|i,h_o)$$

$$+ \sum_{i=1}^{N}\sum_{m=o}^{n-2} \psi_m(i|s_1,\Delta\pi)u(n-1-m|i,h_m')$$

and thus, comparing this with (3.18), we see that (3.27) is valid.

The functional equation governing $D_\alpha^*(s_o)$ is established by the next theorem.

Theorem 3.9: For any $s_o \in S$ and α such that $0 < \alpha < 1$, the optimal return $D_\alpha^*(s_o)$ uniquely satisfies

$$D_\alpha^*(s_o) = \max_{h_o \in H} \left\{ \sum_{n=1}^{\infty} \alpha^n \sum_{i=1}^{N} \psi_o(i)u(n|i,h_o) + \alpha D_\alpha^*(s_o^1) \right\} \quad (3.30)$$

where $s_o^1 = s_1 = f_1(h_o)s_o$.

Proof: The relationship in (3.25) can also be expressed as

$$D_\alpha^*(s_o) = \max_{(h_o,h_1,\ldots)} \left\{ \sum_{n=1}^{\infty} \alpha^n R_n(s_o,\pi) \right\}$$

Using Theorem 3.8 this becomes

$$D_\alpha^*(s_o) = \max_{(h_o,h_1,\ldots)} \left\{ \sum_{n=1}^{\infty} \alpha^n \sum_{i=1}^{N} \psi_o(i) u(n|i,h_o) + \sum_{n=2}^{\infty} \alpha^n R_{n-1}(s_o^1,\Delta\pi) \right\} .$$

The second summation is $\alpha\, D_\alpha(s_o^1,\Delta\pi)$, so

$$D_\alpha^*(s_o) = \max_{(h_o,h_1,\ldots)} \left\{ \sum_{n=1}^{\infty} \alpha^n \sum_{i=1}^{N} \psi_o(i) u(n|i,h_o) + \alpha\, D_\alpha(s_o^1,\Delta\pi) \right\} .$$

But given s_1, $D_\alpha(s_o^1,\Delta\pi)$ depends only upon the sequence of decisions $(h_1,h_2,\ldots)$. Hence

$$D_\alpha^*(s_o) = \max_{h_o \epsilon H} \left[\sum_{n=1}^{\infty} \alpha^n \sum_{i=1}^{N} \psi_o(i) u(n|i,h_o) + \alpha \max_{h_1,h_2,\ldots} D_\alpha(s_o^1,\Delta\pi) \right] \tag{3.31}$$

or

$$D_\alpha^*(s_o) = \max_{h_o \epsilon H} \left[\sum_{n=1}^{\infty} \alpha^n \sum_{i=1}^{N} \psi_o(i) u(n|i,h_o) + \alpha\, D_\alpha^*(s_o^1) \right]$$

We have therefore shown that the optimal return satisfies (3.30). The uniqueness proof is straight forward following Karlin [85] and we omit it.

Equation (3.30) is a functional equation of the same general form as those usually encountered in the dynamic programming formulation of optimization problems [59,60]. It is not a very useful equation for computational purposes, because of the nature of the sets S and H. Nevertheless it is an extremely useful vehicle for establishing some general principles concerning optimal policies, which allow considerable simplification in describing an optimal policy. These results are the subject of the remainder of this chapter.

3.5 OPTIMALITY OF A STATIONARY, DETERMINISTIC POLICY

Proceeding from the functional Eq. (3.30) we can now establish three important properties of the discounting optimization problem. These are:

a. There exists a policy which is optimal for all $s_0 \in S$. Hence it is not necessary to know the distribution s_0 describing the beginning of the decision process.

b. There exists at least one stationary policy having the property described in a.

c. There exists a stationary deterministic optimal policy having property a.

If $\pi = (h_0, h_1, h_2, \ldots, h_n, \ldots)$ is stationary then $h_0 = h_n$, $n = 1,2,3,\ldots$ If π is also deterministic then the action selection probabilities are

either zero or unity, i.e. for the decision h_o one has

$$\eta(k|i) = 1, \quad \text{for } k = k_i \in A_i$$

and

$$\eta(k|i) = 0 \quad \text{otherwise.}$$

Because of the above properties, an optimal policy for discounting optimization can be simply described by a table which lists the action index k_i for every state i. This was pointed out in Chapter II. Chapter IV will show that this property holds also for infinite-time optimization.

In order to establish the validity of the above three properties it is convenient to introduce the expectation of discounted return, conditional upon the policy π becoming operational with an observed state j at time v, (j = 1,2,...N; v = 0,1,2,...). (Recall from section 3.2.3 that the time variation of the policy π is specified from a time origin which is not necessarily the beginning of the actual sequence of decisions used.) This will be denoted by $D_\alpha(j,v,\pi)$ for any policy $\pi \epsilon P$,

$D_\alpha(j,v,\pi)$ = Expectation of discounted total return using policy π, conditional on the policy becoming operational with state j at time v.

$D_\alpha(j,v,\pi)$ is the value of $D_\alpha(s_o,\pi)$ when s_o assigns unit probability to the pair of values j,v. $D_\alpha(j,v,\pi)$ is defined for all j and v, and the unconditional expectation $D_\alpha(s_o,\pi)$ for arbitrary s_o can be written as

$$D_\alpha(s_o\pi) = \sum_{j=1}^{N} \sum_{v=0}^{\infty} \psi_o(j,v) D_\alpha(j,v,\pi) \qquad (3.32)$$

Because of (3.32) it is sufficient to show that the three properties hold for every pair of values j, v. For then

$$\max_{\pi \in P} D_\alpha(s_o,\pi) = \sum_{j=1}^{N} \sum_{v=o}^{\infty} \psi_o(j,v) \max_{\pi \in P} D_\alpha(j,v,\pi)$$

because there is one policy, denoted π_d, which simultaneously realizes the maximum of $D_\alpha(j,v,\pi)$ for all j, v. First we establish

Lemma: For all j,[1]

$$D^*_\alpha(j,v) = \alpha^v D^*_\alpha(j) \tag{3.33}$$

and moreover, if

$\pi(j) = \pi(j,0)$ =· an optimal policy for the case when the policy becomes operational with initial state j at time 0,

and

$\pi(j,v)$ = an optimal policy with the initial state j at time v,

it is sufficient to take

$$\Delta^v \pi(j,v) = \pi(j)$$

Proof: We first point out that[2]

$$D^*_\alpha(j,v) = \max_{\pi \in P} \sum_{n=1}^{\infty} \alpha^n R_n(j,v,\pi).$$

However

---

[1]For notational simplicity, we use $D^*_\alpha(j)$ instead of $D^*_\alpha(j,o)$ when v = 0.

[2]$R_n(j,v,\pi)$ denotes the expectation of one-step return at time n under the same initial conditions as apply to $D_\alpha(j,v,\pi)$.

$$R_n(j,v,\pi) = 0 \quad \text{for } n = 1,2,\ldots,v$$

inasmuch as no return is received until after the process begins. Hence

$$D_\alpha^*(j,v) = \max_{\pi\epsilon P} \alpha^v \sum_{n=1}^{\infty} \alpha^n R_{n+v}(j,v,\pi).$$

By repeated application of Theorem 3.8 however, we can establish

$$R_{n+v}(j,v,\pi) = R_n(j,\Delta^v\pi)$$

and therefore

$$D_\alpha^*(j,v) = \max_{\pi\epsilon P} \alpha^v \sum \alpha^n R_n(j,\Delta^v\pi).$$

However, as a result of the existence theorem (Theorem 3.7) and the fact that the decisions $h_0,h_1,\ldots,h_{v-1}$ of any policy do not enter into maximizing $\sum_{n=1}^{\infty} \alpha^n R_n(j,\Delta^v\pi)$, we can write

$$D_\alpha^*(j,v) = \alpha^v \max_{\Delta^v\pi\epsilon P} \sum_{n=1}^{\infty} \alpha^n R_n(j,\Delta^v\pi).$$

But

$$D_\alpha^*(j) = \max_{\pi\epsilon P} \sum_{n=1}^{\infty} \alpha^n R_n(j,\pi) = \max_{\Delta^v\pi\epsilon P} \sum_{n=1}^{\infty} \alpha^n R_n(j,\Delta^v\pi)$$

and so

$$D_\alpha^*(j,v) = \alpha^v D_\alpha^*(j)$$

Also this shows that the maximizing policy $\pi(j,v)$ for initial state j at time v may be taken as a v unit time delay of an optimal policy for initial state j at time 0. Any sequence of decisions $h_0,h_1,\ldots,h_{v-1}$ may make up the initial part of the policy $\pi(j,v)$ since they do not affect the return.

It is sufficient therefore to consider only the case of a process beginning at time 0 in some state j. If we establish the desired properties a., b., c. here, then (3.32) and (3.33) allow the immediate extension of the result to the general case.

Theorem 3.10: There exists a stationary policy $\pi_d \in P$ for the discounting optimization problem such that

$$D_\alpha(i,\pi_d) \geq D_\alpha(i,\pi)$$

for all i and all $\pi \epsilon P$. Thus π_d is a uniformly optimal policy for all initial states.

Proof: Let $\pi(i)$ be an optimal policy for initial state i at time 0, so that

$$D_\alpha(i,\pi(i)) \geq D_\alpha(i,\pi) \quad \text{for all } \pi \epsilon P \ .$$

Further let e_i be the vector of action selection probabilities for the initial decision (h_0) of policy $\pi(i)$, for $i = 1,2,\ldots,N$. From the functional equation or Eq. (3.31) it follows that[1]

$$D_\alpha(i,\pi(i)) = \sum_{n=1}^{\infty} \alpha^n u(n|i) + \alpha \max_{h_1,h_2,\ldots} D_\alpha(s_1,\Delta\pi)$$

[1]For simplicity we use $u(n|i)$ in place of $u(n|i,h_0)$, this being solely determined by the vector e_i of $\pi(i)$.

where s_1 is determined by the transformation of Eq. (3.15) from the vector e_i and the initial state i. For simplicity of notation we write

$$d_i = \sum_{n=1}^{\infty} \alpha^n u(n|i) \quad i = 1,2,\ldots N.$$

Now we can say that[1]

$$D_\alpha(i,\pi(i)) \leq d_i + \alpha \sum_{\ell=1}^{N} \sum_{m=o}^{\infty} q_{i\ell}(m+1) D^*_\alpha(\ell,m).$$

But $D^*_\alpha(\ell,m) = \alpha^m D_\alpha(\ell,\pi(\ell))$. So

$$D_\alpha(i,\pi(i)) \leq d_i + \sum_{\ell=1}^{N} \sum_{m=1}^{\infty} q_{i\ell}(m)\alpha^m D_\alpha(\ell,\pi(\ell))$$

We next introduce the notation

$$\sum_{m=1}^{\infty} q_{i\ell}(m)\alpha^m = \rho_{i\ell}$$

Using this we have

$$D_\alpha(i,\pi(i)) \leq d_i + \sum_{\ell=1}^{N} \rho_{i\ell} D_\alpha(\ell,\pi(\ell))$$

Now we can apply this inequality again to the term $D_\alpha(\ell,\pi(\ell))$, obtaining

[1] Again, we use $q_{i\ell}(m)$ in place of $q_{i\ell}(m|h_o)$ for simplicity.

$$D_\alpha(i,\pi(i)) \leq d_i + \sum_{\ell=1}^{N} \rho_{i\ell} d_\ell + \sum_{\ell=1}^{N} \rho_{i\ell} \sum_{k=1}^{N} \rho_{\ell k} D^*(k)$$

Writing

$$\sum_{\ell=1}^{N} \rho_{i\ell}\rho_{\ell k} = \rho_{ik}^{(2)}$$

and

$$\rho_{ik}^{(n)} = \sum_{\ell=1}^{N} \rho_{i\ell}^{(n-1)} \rho_{\ell k} \quad \text{for } n > 2$$

we have

$$D_\alpha(i,\pi(i)) \leq d_i + \sum_{\ell=1}^{N} \rho_{i\ell} d_\ell + \sum_{k=1}^{N} \rho_{ik}^{(2)} D_\alpha(k,\pi(k))$$

Continuing to iterate this expression, we obtain for large M

$$D_\alpha(i,\pi(i)) \leq d_i + \sum_{n=1}^{M-1} \sum_{\ell=1}^{N} \rho_{i\ell}\, d_\ell + \sum_{k=1}^{N} \rho_{ik}^{(M)} D_\alpha(k,\pi(k))$$

We note at this point that $0 < \rho_{ik} < 1$ and moreover

$$\sum_{k=1}^{N} \rho_{ik} < 1$$

Hence

$$\sum_{k=1}^{N} \rho_{ik}^{(M)} < \sum_{k=1}^{N} \rho_{ik}^{(M-1)} < 1 \quad \text{for all } M$$

which can be established by induction. Now

$$\sum_{k=1}^{N} \rho_{ik}^{(M)} D_\alpha(k,\pi(k)) < \sum_{k=1}^{N} \rho_{ik}^{(M)} \max_k D_\alpha(k,\pi(k))$$

but for $M \to \infty$, $\sum_{k=1}^{N} \rho_{ik}^{(M)} \to 0$ monotonically.

Then in the limit $M \to \infty$,

$$D_\alpha(i,\pi(i)) \leq d_i + \sum_{n=1}^{\infty} \sum_{\ell=1}^{N} \rho_{i\ell}^{(n)} d_\ell$$

The right side of this inequality we claim to be the return of a stationary policy $\pi_d = (h,h,\ldots)$, where $h = (e_1,e_2,\ldots,e_N)$ and the e_i are the action selection probability vectors specified for time 0 in the optimal policies $\pi(i)$. Because the inequality holds for all i, we have shown the existence of a uniformly optimal stationary policy.

The proof is now completed by showing that

$$D_\alpha(i,\pi_d) = d_i + \sum_{n=1}^{\infty} \sum_{\ell=1}^{N} \rho_{i\ell}^{(n)} d_\ell$$

for all i. We do this by deriving $D_\alpha(i,\pi_d)$ in the manner previously used and showing it yields the same expression. From earlier definitions,

$$D_\alpha(i,\pi_d) = \sum_{n=1}^{\infty} \alpha^n R_n(i,\pi_d)$$

and

$$R_n(i,\pi_d) = u(n|i) + \sum_{m=1}^{n-1} \sum_{\ell=1}^{N} \psi_{i\ell}(m) u(n-m|\ell)$$

where

$\psi_{i\ell}(m)$ = probability of entering state ℓ at time m given initial state i at time 0.

(3.34)

But by proceeding from (3.14) one can establish

$$\psi_{i\ell}(m) = q_{i\ell}(m) + \sum_{n=1}^{m-1} \sum_{j=1}^{N} \psi_{ij}(n) q_{j\ell}(m-n), \quad m > 1.$$

and so

$$D_\alpha(i,\pi_d) = \sum_{n=1}^{\infty} \alpha^n \left[u(n|i) + \sum_{m=1}^{n-1} \sum_{\ell=1}^{N} \psi_{i\ell}(m) u(n-m|\ell) \right]$$

$$= d_i + \sum_{m=1}^{\infty} \sum_{\ell=1}^{N} \psi_{i\ell}(m) \alpha^m \sum_{n=1}^{\infty} \alpha^n u(n|\ell)$$

$$= d_i + \sum_{m=1}^{\infty} \sum_{\ell=1}^{N} \psi_{i\ell}(m) \alpha^m d_\ell$$

Now one can show that

$$\sum_{n=1}^{\infty} \rho_{i\ell}^{(n)} = \sum_{m=1}^{\infty} \alpha^m \psi_{i\ell}(m)$$

and thus

$$D_\alpha(i,\pi(i)) = D_\alpha(i,\pi_d)$$

for all i.

Theorem 3.11: There exists a uniformly optimal stationary policy which is deterministic.

Proof: Confining attention now to the stationary policy π_d, there is a single decision h to be determined. The notation in the functional equation (3.30) can therefore be simplified. Because $h_n = h$ for all n, the return for the m^{th} step after a transition begins from state i can be denoted $u(m|i)$ rather than $u(m|i,h_n)$. Similarly the transition distribu-

tion $q_{ij}(\tau|h_n)$ can now be simply denoted as $q_{ij}(\tau)$. The equations for these quantities are

$$u(m|i) = \sum_{k=1}^{K_i} \eta(k|i)u(m|i,k)$$

and

$$q_{ij}(\tau) = \sum_{k=1}^{K_i} \eta(k|i)q_{ij}(\tau|k)$$

where $\eta(k|i)$ are the action selection probabilities for π_d, and the other quantities are defined in (2.2) and (2.9). Note that for each i these depend only upon $e_i = (\eta(1|i),\ldots\eta(K_i|i))$. Equation (3.30) can now be written

$$D^*_\alpha(s_o) = \sum_{i=1}^{N}\sum_{m=o}^{\infty} \Psi_o(i,m)\alpha^m \max_{e_i \epsilon E_i} \left\{ \sum_{k=1}^{K_i} \eta(k|i) \left[\sum_{n=1}^{\infty} \alpha^n u(n|i,k) + \sum_{j=1}^{N}\sum_{v=1}^{\infty} \alpha^v q_{ij}(v|k) D^*_\alpha(j) \right] \right\}$$

The bracketed quantity $[\cdot]$ has a maximum value for some $k_i \epsilon A_i$. The $\eta(k|i)$ are constrained to be probabilities. It is seen then from the properties of such a constrained maximization that it is sufficient to choose $e_i = (0,0,\ldots,0,1,0,\ldots,0)$ where for every i the unit value occurs in the k_i position, that is

$$\eta(k|i) = 1 \quad \text{for } k = k_i,$$

$$\eta(k|i) = 0 \quad \text{otherwise.}$$

The two preceding theorems allow one to describe an optimal policy in much simpler terms than one might originally anticipate. This is reason enough to seek only an optimal policy fitting this description, i.e. a stationary deterministic policy which is optimal for every initial state. Granting this, any further consideration of discounted optimization may consider the process as beginning at time 0 in an initial state i, without loss of generality. Hence forth, the subset of P consisting of all stationary deterministic policies will be called P', and any member will be called a P' policy. Our principal interest is to show in the next chapter that P' contains an optimal policy for infinite-time optimization. The proof will lean heavily on the results of Theorems 3.7, 3.10, and 3.11.

CHAPTER IV

THE INFINITE-TIME DECISION PROCESS

As we have already mentioned, the notion of discounting has a definite physical interpretation in applications to economics and management science. But discounting does not seem to be indicated in applying the decision process to the control of queues. In high volume traffic situations in modern computer and communication systems, the transitions in state and the control decisions occur very rapidly in time. Over a period of operation as short as a few hours it is reasonable to view the system as operating indefinitely, with respect to the average time interval separating successive decisions. With respect to the useful life time of the system, however, a few hours is small enough to obviate any discounting of performance throughout this interval.

The infinite-time optimization problem discussed in Chapters II and III therefore seems to be the most important decision problem in the control of queues. This chapter will extend the principal result of Chapter III to this case. With this result we shall be able to introduce the Howard algorithm for determining an optimal stationary deterministic policy. This chapter will close with some consideration of the computational aspects of the Howard algorithm, preparatory to using it for optimization of computer time-sharing in Chapters V and VI.

4.1 EXISTENCE OF AN OPTIMAL STATIONARY DETERMINISTIC POLICY

We are able to establish that finite-time optimization can be achieved with a stationary deterministic policy by considering discounting optimization when the discount factor $\alpha \to 1$ from the left, written $\alpha \to 1^-$. From Chapter III it is known that optimization with discounting can be obtained using a P' policy for $0 < \alpha < 1$. The following section develops an important theorem relating the discounted return $D_\alpha(s_o,\pi)$ as $\alpha \to 1^-$ and the asymptotic behavior of $V_T(s_o,\pi)/T$ as $T \to \infty$ for a policy $\pi \epsilon P'$. This is subsequently used in the existence theorem, Theorem 4.2, which is the principal theoretical result of this report.

4.1.1 System Behavior with a P' Policy

Under any stationary deterministic policy, i.e., any P' policy, the decision process can be treated as a Markov-Renewal stochastic process [58,88]. As Pyke [88] points out, this process is fundamentally determined by the transition distribution

$$q_{ij}(\tau) = q_{ij}(\tau|k_i), \quad i,j = 1,2,\ldots,N; \quad \tau = 1,2,\ldots,$$

where k_i is the action specified for state i by the assumed P' policy. Theorem 4.1, to be established shortly, relies on certain results obtained from the Markov-Renewal approach.

The Markov-Renewal process concerns the random variable $U_j(T)$ and its conditional expectation $M_{ij}(T)$. The precise definitions are

$U_j(T)$ = the number of times the system enters state j in the time interval [1,T], T = 1,2,3,... .

and

$M_{ij}(T)$ = the expectation of $U_j(T)$, conditional on the initial state i at time 0,

or

$$M_{ij}(T) = E(U_j(T) | Z_0=i, W_0=0)$$

If we also define the probability $\psi_{ij}(n)$,

$\psi_{ij}(n)$ = probability that the system enters state j at time n, conditional on the initial state i at time 0,

for n = 1,2,3,..., then

$$M_{ij}(T) = \sum_{n=1}^{T} \psi_{ij}(n) \qquad (4.1)$$

Note that the initial state i is not counted in $M_{ii}(T)$.

The renewal function $M_{ij}(T)$ is related to the first passage probability distribution $G_{ij}(T)$, where

$G_{ij}(T)$ = probability that the system has entered state j at least once in the interval [1,T], conditional on the initial state i at time 0.

Note that

$$G_{ij}(T) = P_r(U_j(T) > 0 | Z_0=i, W_0=0)$$

The relation between $G_{ij}(T)$ and $M_{ij}(T)$ is derived by Pyke [88] and is expressed in a convolution form following Eq. 5.13 of [88]. Written out for the discrete-time model, this result is

$$M_{ij}(T) = G_{ij}(T) + \sum_{n=1}^{T-1} G_{ij}(T-n)\, \psi_{jj}(n) \qquad (4.2)$$

The important properties of $M_{ij}(T)$ which we require concern its asymptotic behavior as $T \to \infty$. In this regard, we define

f_{ij} = probability of passage from state i to state j,

or

$$f_{ij} = \lim_{T\to\infty} G_{ij}(T) \qquad (4.3)$$

Any state j for which $f_{jj} = 1$ is called a recurrent state ([58], p. 1240). For a recurrent state the mean recurrence time ℓ_{jj} is defined by

$$\ell_{jj} = \sum_{n=1}^{\infty} n[G_{jj}(n) - G_{jj}(n-1)] = \sum_{n=1}^{\infty} n\, g_{jj}(n) \qquad (4.4)$$

Note that $g_{jj}(n)$ is the probability that the first recurrence of state j occurs at time n. Pyke has shown ([58], p. 1241) that ℓ_{jj} is finite for a finite state Markov-Renewal process if and only if j is a recurrent state and $v_i^{k_i} < \infty$ for all states i which communicate with state j. Because we have assumed $v_i^k < \infty$ for all i and k, and the decision process has a finite number N of states, ℓ_{jj} is finite for any recurrent state j.

There are two important theorems on the asymptotic behavior of $M_{ij}(T)$ which can be established from the theory of simple renewal processes [90,91]. These are developed in Appendix B. We have chosen to give them the names by which their counterparts in renewal theory are known. For Theorem 4.1 we need only the following.

Elementary Renewal Theorem

If j is a recurrent state of a Markov-Renewal process, then

$$\lim_{T\to\infty} \frac{M_{ij}(T)}{T} = \frac{f_{ij}}{\ell_{jj}}$$

for any state i. If j is non-recurrent, i.e., $f_{jj} < 1$, the limit is zero for any state i.

A proof is given in Appendix B. Using this we now establish

Theorem 4.1. Let R be the subset of the states i=1,2,3,...,N, such that $f_{ii}=1$ under a given policy $\pi\epsilon P'$. For any $s_o\epsilon S$

$$\lim_{\alpha\to 1^-} (1-\alpha)D_\alpha(s_o,\pi) = \lim_{T\to\infty} \frac{V_T(s_o,\pi)}{T}$$

$$= \sum_{i=1}^{N} \sum_{j\epsilon R} \varphi_i f_{ij} b_j^{k_j}/\ell_{jj} \qquad (4.5)$$

where

$$\varphi_i = \sum_{v=0}^{\infty} \psi_o(i,v) \qquad (4.6)$$

is the probability of the initial state i, and $b_j^{k_j}$ is the one-transition return for state j.

$$b_j^{k_j} = \sum_{m=1}^{\infty} u(m|j,k_j)$$

Proof: We will establish that $V_T(s_o,\pi)/T$ has the limiting value claimed, and use an Abelian theorem from Appendix A to complete the proof of the theorem.

To begin, one can show by specializing (3.18) that

$$\frac{V_T(s_o,\pi)}{T} = \frac{1}{T}\sum_{n=1}^{T} R_n(s_o,\pi)$$

$$= \frac{1}{T}\sum_{n=1}^{T}\sum_{i=1}^{N}\sum_{v=0}^{n-1} \psi_o(i,v)u(n-v|i,k_i)$$

$$+ \frac{1}{T}\sum_{n=2}^{T}\sum_{i=1}^{N}\sum_{v=0}^{n-2}\psi_o(i,v)\sum_{j=1}^{N}\sum_{m=1}^{n-v-1}\psi_{ij}(m)\ u(n-v-m|j,k_j)$$

Because the summations are finite, the order can be interchanged and

$$\frac{V_T(s_o,\pi)}{T} = \sum_{i=1}^{N}\frac{1}{T}\sum_{v=0}^{T-1}\psi_o(i,v)\sum_{n=1}^{T-v} u(n|i,k_i)$$

$$+\sum_{i=1}^{N}\sum_{v=0}^{T-2}\psi_o(i,v)\sum_{j=1}^{N}\frac{1}{T}\sum_{n=2}^{T-v}\sum_{m=1}^{n-1}\psi_{ij}(m)\ u(n-m|j,k_j)$$

From lemma 1 of Appendix A,

$$\lim_{T\to\infty}\sum_{v=0}^{T-1}\psi_o(i,v)\sum_{n=1}^{T-v} u(m|i,k_i) = \varphi_i b_i^{k_i}$$

which has finite value, and therefore

$$\lim_{T\to\infty}\frac{V_T(s_o,\pi)}{T} = \sum_{i=1}^{N}\lim_{T\to\infty}\sum_{v=0}^{T-2}\psi_o(i,v)\sum_{j=1}^{N}\frac{1}{T}\sum_{n=2}^{T-v}\sum_{m=1}^{n-1}\psi_{ij}(m)\ u(n-m|j,k_j)$$

Again by lemma 1 of Appendix A,

$$\lim_{T\to\infty}\sum_{v=0}^{T-2}\psi_o(i,v)\sum_{j=1}^{N}\frac{1}{T}\sum_{n=2}^{T-v}\sum_{m=1}^{n-1}\psi_{ij}(m)\ u(n-m|j,k_j) =$$

$$\sum_{v=0}^{\infty} \psi_o(i,v) \sum_{j=1}^{N} \lim_{T\to\infty} \frac{1}{T} \sum_{n=1}^{T} \sum_{m=1}^{n} \psi_{ij}(m)\, u(n+1-m|j,k_j)$$

whenever the latter limit exists. Now from lemma 2 of Appendix A and the Elementary Renewal Theorem above,

$$\lim_{T\to\infty} \frac{1}{T} \sum_{n=1}^{T} \sum_{m=1}^{n} \psi_{ij}(m)\, u(n+1-m|j,k_j) =$$

$$\lim_{T\to\infty} \frac{1}{T} \sum_{m=1}^{T} \psi_{ij}(m) \sum_{n=1}^{T+1-m} u(n|j,k_j) = \frac{f_{ij} b_j^{k_j}}{\ell_{jj}}$$

if state j is recurrent, and is zero otherwise. Therefore

$$\lim_{T\to\infty} \frac{V_T(s_o,\pi)}{T} = \sum_{i=1}^{N} \sum_{v=0}^{\infty} \psi_o(i,v) \sum_{j\in R} \frac{f_{ij} b_j^{k_j}}{\ell_{jj}} = \sum_{i=1}^{N} \varphi_i \sum_{j\in R} \frac{f_{ij} b_j^{k_j}}{\ell_{jj}}$$

Abelian Theorem II of Appendix A now establishes that

$$\lim_{\alpha\to 1^-} (1-\alpha) D_\alpha(s_o,\pi) = \lim_{T\to\infty} \frac{V_T(s_o,\pi)}{T}$$

4.1.2 Principal Existence Theorem

The principal theorem of this report is:

Theorem 4.2. There exists a stationary, deterministic policy $\pi^* \in P'$ which maximizes

$$g_\ell(s_o,\pi) = \liminf_{T\to\infty} \frac{V_T(s_o,\pi)}{T}$$

for every $s_o \in S$.

Proof. First we establish that there exists a sequence $\{\alpha_i\}$, such that $0 < \alpha_i < 1$ and $\{\alpha_i\} \to 1$, and a policy $\pi_d \in P'$ for which

$$D_{\alpha_i}(s_o,\pi) \leq D_{\alpha_i}(s_o,\pi_d)$$

for every $s_o \in S$, all $\pi \in P$, and every i. Let $\{\beta_i\}$, $0 < \beta_i < 1$ for all i, be any sequence converging to the value 1. Because $\{\beta_i\}$ is a convergent sequence it is also a Cauchy sequence (see Rudin[56], p. 46). Therefore every subsequence of $\{\beta_i\}$ converges to the same limit, 1. In Chapter III it has been shown that for every element β_i there exists an element, say π_i, contained in P' and maximizing $D_{\beta_i}(s_o,\pi)$ for every $s_o \in S$. But P' is a finite set of policies. In fact the number of policies in P' is equal to the product of the values K_i (i=1,2,...N), $\prod_{i=1}^{N} K_i$. Hence there exists at least one element of P' which occurs infinitely often in the sequence $\{\pi_i\}$. Letting π_d be such an element, $\{\alpha_i\}$ can be taken as the subsequence of $\{\beta_i\}$ such that $\pi_i = \pi_d$.

Then for the sequence $\{\alpha_i\}$ and the associated policy π_d,

$$(1-\alpha_i)D_{\alpha_i}(s_o,\pi) \leq (1-\alpha_i)D_{\alpha_i}(s_o,\pi_d)$$

This holds for any policy, i.e., any $\pi \in P$, and therefore

$$\limsup_{i \to \infty} (1-\alpha_i)D_{\alpha_i}(s_o,\pi) \leq \lim_{i \to \infty} (1-\alpha_i)D_{\alpha_i}(s_o,\pi_d)$$

From Theorem 3.4 and Theorem III of Appendix A it follows that

$$\liminf_{T \to \infty} \frac{V_T(s_o,\pi)}{T} \leq \limsup_{i \to \infty} (1-\alpha_i)D_{\alpha_i}(s_o,\pi)$$

for every $s_o \epsilon S$ and any $\pi \epsilon P$. Theorem 4.1 however has established that

$$\lim_{i \to \infty} (1-\alpha_i) D_{\alpha_i}(s_o, \pi_d) = \lim_{T \to \infty} \frac{V_T(s_o, \pi_d)}{T}$$

Hence

$$\lim_{T \to \infty} \frac{V_T(s_o, \pi_d)}{T} \geq \liminf_{T \to \infty} \frac{V_T(s_o, \pi)}{T}$$

for any $\pi \epsilon P$ and every $s_o \epsilon S$. The theorem is proven by letting $\pi^* = \pi_d$.

This theorem, with Theorem 4.1, points out that for any policy $\pi \epsilon P'$ the following limit exists,

$$g(s_o, \pi) = \lim_{T \to \infty} \frac{V_T(s_o, \pi)}{T}, \text{ for } \pi \epsilon P'. \tag{4.7}$$

The value g is referred to as the "gain" of the policy π. This term was introduced by Howard [63].

The results of Chapter III, together with the above theorem, correspond to what Derman [66] has done for the decision process with constant transition durations.

4.2 ILLUSTRATIONS OF THE PRINCIPAL THEOREM

The principal theorem has considerable merit because of its generality. No unusual restrictions are imposed on the transition probabilities p_{ij}^k, and the return functions $r_{ij}^k(m|\tau)$ can be arbitrary as long as b_i^k, the expectation of total return over one transition, is defined by an absolutely convergent series, see Eq. (2.10). Since there are no complex conditions in the theorem, the conclusion can be easily applied to models of physical processes, often by inspection.

The theorem is not trivial to prove as evidenced by the deliberations up to this point, even allowing that we have included certain steps which are obvious to the sophisticated reader. Moreover, the result is not a trivial property. To illustrate this point, we would like to consider some simple examples. The first shows that an obviously non-ergodic system falls under the theorem. The second demonstrates that systems do exist which do not possess the property of the theorem. This example has non-stationary return functions, and we show that a non-stationary policy is better than any stationary policy for the system. Finally we show in a third example that the assumption of absolute convergence associated with b_i^k is not necessary, but only sufficient for our results.

4.2.1 First Example

Consider a decision process in which there are two states for the system, i = 1,2. Let there be only one action allowed in the first state, and two alterative actions in the second state. Thus there are two P' policies corresponding to the choice of k = 1 or k = 2 in state 2. We shall refer to the former as policy 1, and the latter as policy 2. For both policies, let the transition probabilities be

$$p_{11} = 0, \qquad p_{12} = 1,$$

$$p_{12} = 1, \qquad p_{22} = 0.$$

The two policies will differ however in regard to the transition intervals and the return values. Assume that

$$f_{12}(1) = 1, \quad f_{12}(\tau) = 0 \quad \text{for } \tau > 1;$$

$$f^1_{21}(1) = 1, \quad f^1_{21}(\tau) = 0 \quad \text{for } \tau > 1;$$

$$f^2_{21}(1) = 0, \quad f^2_{21}(2) = 1, \quad f^2_{21}(\tau) = 0 \quad \text{for } \tau > 2;$$

and also assume that

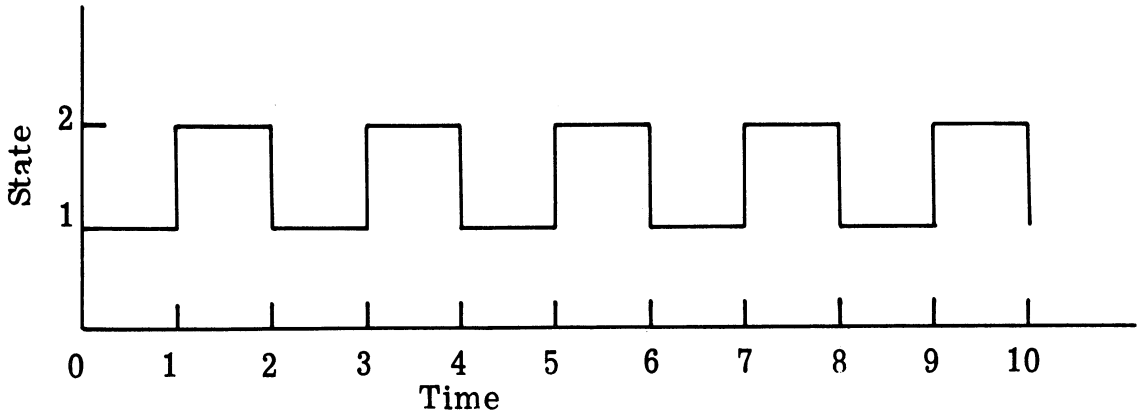
$$\begin{aligned} r_{12}(1|1) &= 0, \\ r^1_{21}(1|1) &= 1, \\ r^2_{21}(1|2) &= 0, \quad r^2_{21}(2|2) = 2 \end{aligned} \tag{4.8}$$

These are the only data values which matter because of the assumed transition probabilities, and transition time distributions.

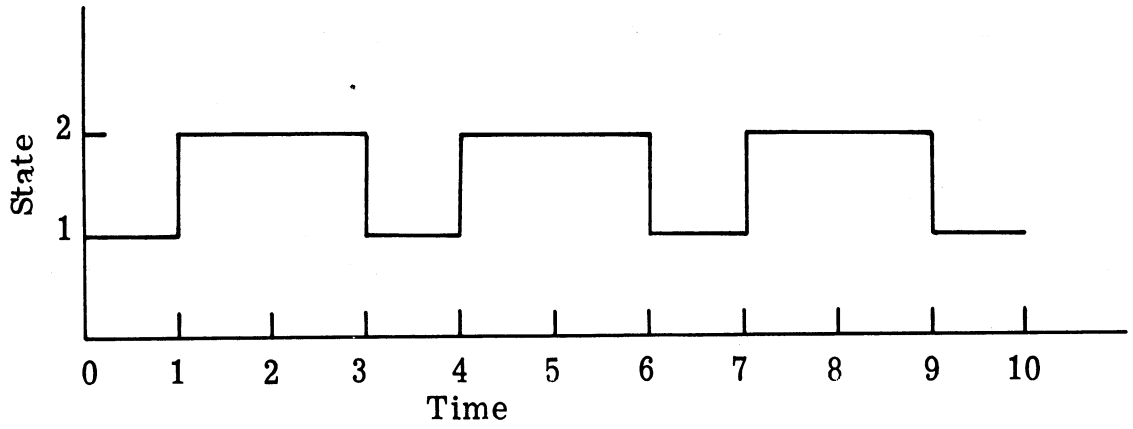
Figure 4.1 depicts the behavior of the system under the two P' policies, assuming the decision process begins in state 1. It is clear that the two systems are periodic and thus the underlying Markov chains do not have stationary solutions. The systems are not ergodic in the sense of a stochastic process. Observe that the two processes can be compared in terms of 6 unit intervals, i.e., the relative phase of the two processes is the same at times 6, 12, 18, etc., as at time 0.

The expected total return at time n assuming the process begins in state 1 at time 0 is $V^1_n(1)$ for policy 1 and $V^2_n(1)$ for policy 2. The return at time 1 is zero for both policies since $r_{12}(1|1) = 0$. But $V^1_2(1) = 1$, while $V^2_2(1) = 0$ because $r^2_{21}(1|2) = 0$. Using the notation

$$[x] = \text{integer part of } x, \quad x > 0$$



(a) Behavior under policy 1.



(b) Behavior under policy 2.

Fig. 4.1. State transitions in the first example.

we can write in general

$$V_n^1(1) = \left[\frac{n}{2}\right]$$

and

$$V_n^2(1) = 2\left[\frac{n}{3}\right]$$

Then

$$A_n^1(1) = \frac{V_n^1(1)}{n} = \frac{1}{n}\left[\frac{n}{2}\right]$$

and because $[n/2] \leq n/2$, it is clear that $0 \leq A_n^1(1) \leq 1/2$. Moreover $A_n^1(1)$ approaches $1/2$ as a limit for $n \to \infty$. This is true because

$$\left[\frac{n}{2}\right] = \frac{n}{2} - \delta_n$$

Although δ_n changes value with n, $0 \leq \delta_n < 1$ for all n, and therefore

$$\left|\frac{1}{n}\left[\frac{n}{2}\right] - \frac{1}{2}\right| = \left|\frac{1}{n}\frac{n}{2} - \frac{\delta n}{n} - \frac{1}{2}\right| < \frac{1}{n}$$

which is as small as desired for n sufficiently large. This limit value, $1/2$, is the gain g of the policy 1.

Similarly it can be seen that $0 \leq A_n^2(1) \leq 2/3$, and $A_n^2(1) \to 2/3$ as $n \to \infty$. Because policy 2 has a higher gain than policy 1 we conclude that policy 2 is optimal. From our theorem, no other admissible policy has higher gain than the value $2/3$.

It is also clear that policy 2 is optimal if the processes begin in state 2. Looking at Fig. 4.1, this situation comes about by shifting the time origin to the right one unit. Because $V_1^1(1) = V_1^2(1) = 0$, the gain of either policy has the same value as before.

4.2.2 Second Example

We now show that a decision process which does not conform with the assumptions we have made has a nonstationary policy with greater gain than any stationary policy. Consider basically the same process as the first example, but replace (4.8) with another set of returns including one which changes with time. For values of time n such that $6(m-1) < n \leq 6m$, let $r_{12}(1|1) = \frac{2}{3} + (-1)^m \frac{2}{3}$, $m = 1,2,\ldots$. Let the other return functions have the same values as in (4.8). Thus over the first 6 intervals the transitions from state 1 contribute nothing to the total return, as in the first example. Over the next 6 units, from time 6 to time 12, each transition from state 1 earns a return of 4/3.

In the first example policy 2 earned 4 return units in each 6 unit time interval, whereas policy 1 earned 3 units of return. Here policy 1 will earn $3 + 3 \cdot 4/3 = 7$ return units in any 6 unit interval in which state 1 contributes return. On the other hand, policy 2 earns only $4 + 2 \cdot 4/3 = 6\ 2/3$ return units in such intervals. Thus

$$A_n^1(1) \to \left[\frac{1}{2}(3) + \frac{1}{2}(7)\right] \Big/ 6 = 5/6 \text{ as } n \to \infty$$

$$A_n^2(1) \to \left[\frac{1}{2}(4) + \frac{1}{2}\,\frac{20}{3}\right] \Big/ 6 = 8/9 \text{ as } n \to \infty$$

and policy 2 is the preferred stationary policy. However the nonstationary policy which uses action 1 in state 2 for those time values n such that $6(m-1) < n \leq 6m$ $m = 2, 4, 6, 8, \ldots$, and action 2 in state

2 for other values of time, will have a greater gain than either policy 1 or 2. Its gain is

$$g = \left[\frac{1}{2}(4) + \frac{1}{2}(7)\right] \Big/ 6 = 11/12$$

4.2.3. Third example. The requirement that the series

$$\sum_{m=1}^{\infty} |u(m|i\ k)|$$

be convergent for all i and k, where

$$b_i^k = \sum_{m=1}^{\infty} u(m|i,k)$$

has simplified the proof of the results in Chapters III and IV. It has played an important role for example in Theorems 3.3 and 3.4, Theorem 3.7, and Theorem 4.1. Yet these theorems may be true for certain decision processes which fail to meet this requirement. We now give an example of a particular system for which Theorems 3.3 and 3.4 hold.

It is sufficient to consider the behavior of a two state system under only one stationary policy.[1] Let the transition matrix P of the underlying Markov chain be

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Let

$$f_{12}(1) = 1, f_{12}(\tau) = 0 \quad \text{for } \tau > 1$$

[1] Because there is only one action in each state we drop the k index from $u(m|i,k)$, $r_{ij}^k(m|\tau)$, etc.

and

$$f_{21}(\tau) = \mu^{\tau-1}(1-\mu) \quad (\tau = 1,2,3,\ldots)$$

where $0 < \mu < 1$. Thus the time duration of the transition from state 2 to state 1 is determined by a "coin-flipping" random mechanism, with μ the probability that the transition does not end on any one unit time step.

We additionally specify

$$r_{12}(m|\tau) = 0 \quad \text{for all } m,\tau$$

and

$$r_{21}(m|\tau) = \frac{(-1)^{m-1}}{m\mu^{m-1}} \quad (m = 1,2,3,\ldots,\tau)$$

Thus $r_{21}(m|\tau)$ oscillates with increasing magnitude as m increases.

Now one sees that[1]

$$u(m|1) = 0 \quad \text{for all } m,$$

and

$$u(m|2) = \sum_{\tau=m}^{\infty} \frac{\mu^{\tau-1}(1-\mu)(-1)^{m-1}}{m\mu^{m-1}} = \frac{(-1)^{m-1}}{m}$$

for $m = 1,2,3,\ldots$. Then the alternating series

$$b_2 = \sum_{m=1}^{\infty} u(m|2) = \sum_{m=1}^{\infty} \frac{(-1)^{m-1}}{m}$$

converges ([56] p. 62), whereas the series of absolute values

$$\sum_{m=1}^{\infty} |u(m|2)| = \sum_{m=1}^{\infty} \frac{1}{m}$$

---

[1]Because there is only one action in each state we drop the k index from $u(m|i,k)$, $r_{ij}^{k}(m|\tau)$, etc.

diverges, for it is the well-known harmonic series.

Nevertheless the discounted return $D_\alpha(s_o,\pi)$ exists for every s_o, since

$$|D_\alpha(s_o,\pi)| \leq \sum_{n=1}^{\infty} \alpha^n |R_n(s_o,\pi)| \leq \sum_{n=1}^{\infty} \alpha^n$$

and the last series converges for $0 < \alpha < 1$. Moreover

$$\lim_{T\to\infty} \frac{V_T(s_o,\pi)}{T} = \lim_{T\to\infty} \frac{1}{T} \sum_{n=1}^{T-1} \psi_{12}(n) \sum_{m=1}^{T-n} u(m|2)$$

and this limit exists.

4.3 THE CASE OF ERGODIC POLICIES

An objective of this dissertation is to demonstrate that the Semi-Markov decision process is useful for investigating systems with a large number of alternative control policies. The procedure just used in the illustrations to determine an optimal stationary policy is essentially one of direct enumeration. This is an impractical procedure for the large models of interest in applications. An optimization technique which is effective in large problems has been devised by R. Howard [74] and W. S. Jewell [73] for the general class of Semi-Markow decision processes. This includes the discrete-time process which we are considering.

The Howard algorithm for Semi-Markov decision processes has been developed for the case where all policies in the set P' are "ergodic " originally called the completely ergodic case by Howard [74]. A similar algorithm for nonergodic policies can undoubtedly be developed, as Howard has done [74] for the special case of constant transition intervals. This will not be necessary for our demonstration.

The ergodic property is determined by the underlying Markov chain of the decision process, and the associated transition time distributions. A policy in P' will be called ergodic if it produces a single closed, communicating class of states which is aperiodic and recurrent. These terms indicate concepts which are very much related to the same concepts in the theory of Markov chains.[1]

A set of states is a closed, communicating class if every state can be reached from every other state, and it is impossible to reach a state outside the class from any state within it. A closed communicating class with a finite number of states can be shown to be recurrent, i.e., every state in the class is recurrent. Since we are treating a decision process with a finite number of states, every closed communicating class of states is recurrent.

A recurrent state i is aperiodic (for the discrete-time case) if the greatest common divisor of the values n for which $\psi_{ii}(n) > 0$ is unity. Now one can see that

$$\psi_{ii}(n) = q_{ii}(n) + \sum_{j=1}^{N} \sum_{m=1}^{n-1} q_{ij}(m)\, \psi_{ji}(n-m)$$

for the system may return to state i at time n in one transition of duration n; or, it may pass at time $m < n$ to any other state and subsequently return to state i at n. It is evident from this equation that state i is aperiodic if $q_{ii}(n)$ is aperiodic, i.e., is not a lattice distribution.

---

[1] See Parzen [46], Gantmacher [95] for a complete discussion of the classification of states of a Markov chain.

This means for discrete time that the greatest common divisor of the values n for which $q_{ii}(n) > 0$ is unity (and that $q_{ii}(n) > 0$ for at least one value of $n = 1,2,3,\ldots$).

A sufficient condition that a closed communicating class be aperiodic is that there exists at least one state i in the class for which $q_{ii}(n)$ is aperiodic. For all states in the class are aperiodic if any one is, as can be seen from the corresponding argument for Markov chains, Feller [4], p. 354.

For an ergodic policy the asymptotic behavior of the expected total return $V_T(i)$ as $T\to\infty$ has an especially convenient form. This behavior is the key to Howard's algorithm. The purpose of the next theorem is to establish this behavior. In stating the theorem, we suppress the notation which indicates the dependence of the return upon the policy and the actions for each state. The symbol R denotes the recurrent class of states for the assumed ergodic policy.

Theorem 4.3. An ergodic policy is such that

$$V_T(i) \to gT + v_i$$

as $T\to\infty$, where v_i is a constant and the gain g is independent of i,

$$g = \sum_{j\in R} \frac{b_j}{\ell_{jj}} \tag{4.9}$$

Proof: It is sufficient to prove that

$$\lim_{T\to\infty} [V_{T+1}(i) - V_T(i)] = g$$

for any state i. By specializing Eq. (3.18) one can show that for any

policy in P, the one step return at time T+1 conditional on the initial state i at time 0 is

$$R_{T+1}(i) = V_{T+1}(i) - V_T(i) = u(T+1|i) + \sum_{j=1}^{N} \sum_{m=1}^{T} u(m|j)\psi_{ij}(T+1-m)$$

Then using (4.1) one sees that

$$\psi_{ij}(T+1-m) = M_{ij}(T+1-m) - M_{ij}(T-m)$$

Next using (4.2)

$$\begin{aligned} V_{T+1}(i) - V_T(i) = {} & u(T+1|i) \\ & + \sum_{j=1}^{N} \sum_{m=1}^{T} u(m|j)[G_{ij}(T+1-m) - G_{ij}(T-m)] \\ & + \sum_{j=1}^{N} \sum_{m=1}^{T} u(m|j) \sum_{t=1}^{T-m} [G_{ij}(t) - G_{ij}(t-1)]\psi_{jj}(T+1-m-t) \end{aligned} \tag{4.10}$$

Taking the limit of (4.10) as $T \to \infty$, $u(T+1|i) \to 0$ because it is a term of a convergent series. Because $\lim_{T\to\infty} G_{ij}(T) = f_{ij} \leq 1$, it is seen that

$$\lim_{T\to\infty} [G_{ij}(T+1-m) - G_{ij}(T-m)] = 0$$

Then because the series in $u(m|j)$ is absolutely convergent for all j, lemma 1 of Appendix A allows us to establish that

$$\sum_{m=1}^{T} u(m|j)\,[G_{ij}(T+1-m) - G_{ij}(T-m)] \to 0 \text{ as } T \to \infty$$

The first two terms of (4.10) therefore vanish in the limit. Again using lemma 1 of Appendix A we can now say that

$$\lim_{T\to\infty}[V_{T+1}(i)-V_T(i)] =$$

$$= \sum_{j=1}^{N} b_j \left[\lim_{T\to\infty} \sum_{t=1}^{T} [G_{ij}(t)-G_{ij}(t-1)]\psi_{jj}(T+1-t) \right]$$

whenever the limit on the right side exists.

To show the existence of the limit we must consider whether or not j is a recurrent state. If j is not recurrent, from Feller [100] we have the result (B.3) that

$$\sum_{t=1}^{\infty} \psi_{jj}(t) < \infty$$

So from lemma 1 of Appendix A

$$\lim_{T\to\infty} \sum_{t=1}^{T} [G_{ij}(t)-G_{ij}(t-1)]\psi_{jj}(T+1-t)$$

$$= \lim_{T\to\infty} [G_{ij}(t)-G_{ij}(t-1)] \sum_{t=1}^{\infty} \psi_{jj}(t) = 0.$$

Next if j is a recurrent state, the Key Renewal Theorem in Appendix B establishes that the limit has value f_{ij}/ℓ_{jj}. Therefore

$$\lim_{T\to\infty} [V_{T+1}(i)-V_T(i)] = \sum_{j\in R} \frac{b_j f_{ij}}{\ell_{jj}}$$

But $f_{ij} = 1$ for $j\in R$, since the system eventually passes into the recurrent class from any state. So we have shown that

$$\lim_{T\to\infty} [V_{T+1}(i)-V_T(i)] = g = \sum_{j\in R} \frac{b_j}{\ell_{jj}}$$

Note therefore that the gain of an ergodic policy does not depend upon transient states, those not contained in R.

Although both Howard [74] and Jewell [73] recognize and use the truth of Theorem 4.3, neither have provided the proof in their papers. The values v_i have been termed "relative values" by Howard. They can only be determined to within an additive constant, as a consequence of the next theorem.

Theorem 4.4. For an ergodic policy the gain g and the relative values $v_i (i = 1,2,\ldots N)$ are given by the system of equations

$$v_i + g v_i = b_i + \sum_{j=1}^{N} p_{ij} v_j \tag{4.11}$$

Proof: Defining

$$\sigma_i(T) = \sum_{m=1}^{T} u(m|i)$$

and again suppressing the index k denoting the action, the finite time return $V_T(i)$ is given by

$$V_T(i) = \sigma_i(T) + \sum_{j=1}^{N} \sum_{m=1}^{T-1} p_{ij} f_{ij}(m) V_{T-m}(j) \tag{4.12}$$

Then subtracting gT from both sides,

$$V_T(i) - gT = \sigma_i(T) + \sum_{j=1}^{N} \sum_{m=1}^{T-1} p_{ij} f_{ij}(m) [V_{T-m}(j) - g(T-m)]$$

$$- \sum_{j=1}^{N} \sum_{m=1}^{T-1} g\, p_{ij} f_{ij}(m) m - gT \sum_{j=1}^{N} p_{ij} \sum_{m=T}^{\infty} f_{ij}(m) \tag{4.13}$$

Consider the limit of both sides as $T \to \infty$. From Theorem 4.3 we know that

$$V_T(i) - gT \to v_i$$

and clearly

$$\sum_{j=1}^{N} \sum_{m=1}^{T-1} p_{ij} f_{ij}(m) m \to \nu_i$$

$$\sigma_i(T) \to b_i$$

Because

$$0 \leq T \sum_{j=1}^{N} p_{ij} \sum_{m=T}^{\infty} f_{ij}(m) \leq \sum_{j=1}^{N} p_{ij} \sum_{m=T}^{\infty} m\, f_{ij}(m)$$

and the right side of this inequality approaches zero, the right most term of (4.13) approaches zero as $T \to \infty$. From Theorem 4.3 and the properties of $f_{ij}(m)$ it follows that

$$\sum_{m=1}^{T-1} f_{ij}(m)[V_{T-m}(j) - g(T-m)] \to v_j \qquad \text{as } T \to \infty.$$

Thus from (4.12) we find

$$v_i = b_i + \sum_{j=1}^{N} p_{ij} v_j - g\nu_i$$

as claimed.

One can see that (4.11) remains valid if any constant is added to both sides of every one of the N equations. In particular if we add $(-v_N)$ the equations remain valid. Hence forth then we can consider $v_N = 0$ if this is convenient. With this convention, (4.11) represents N equations in N unknowns, g, v_1, $v_2, \ldots, v_{N-1}$. Thus it can be solved

to give the gain g of any P' policy in the completely ergodic case. An optimal P' policy is one having a maximum value of g.

4.4 THE HOWARD ALGORITHM

The Howard algorithm is a systematic and effective computing procedure for finding a P' policy having maximum gain g. The algorithm consists of two fundamental operations. The "Value-Determination Operation" consists of solving (4.11) for a given P' policy to find its gain g and relative values v_i. The "Policy-Improvement Operation" allows one to use the relative values v_i of a policy in a systematic way to determine a new policy of increased gain. Execution of the algorithm consists of cycling through these operations until the policy on two successive passes through the Policy-Improvement Operation is the same. The computation can begin in either of the two phases.

Figure 4.2 diagrams the calculation of the Howard algorithm. In the Policy-Improvement Operation one evaluates a set of test quantities

$$\delta_i(k) = \frac{1}{\nu_i^k} \left[b_i^k + \sum p_{ij}^k \, v_j - v_i \right] \tag{4.14}$$

for each i and $k = 1,2,\ldots K_i$. The values v_i are those determined for the previous policy considered, or can be taken as zero if the computation is being started. The new policy consists of the set of action indices $k_i (i = 1,2,\ldots,N)$ such that

$$\delta_i(k_i) = \max_k \delta_i(k) \tag{4.15}$$

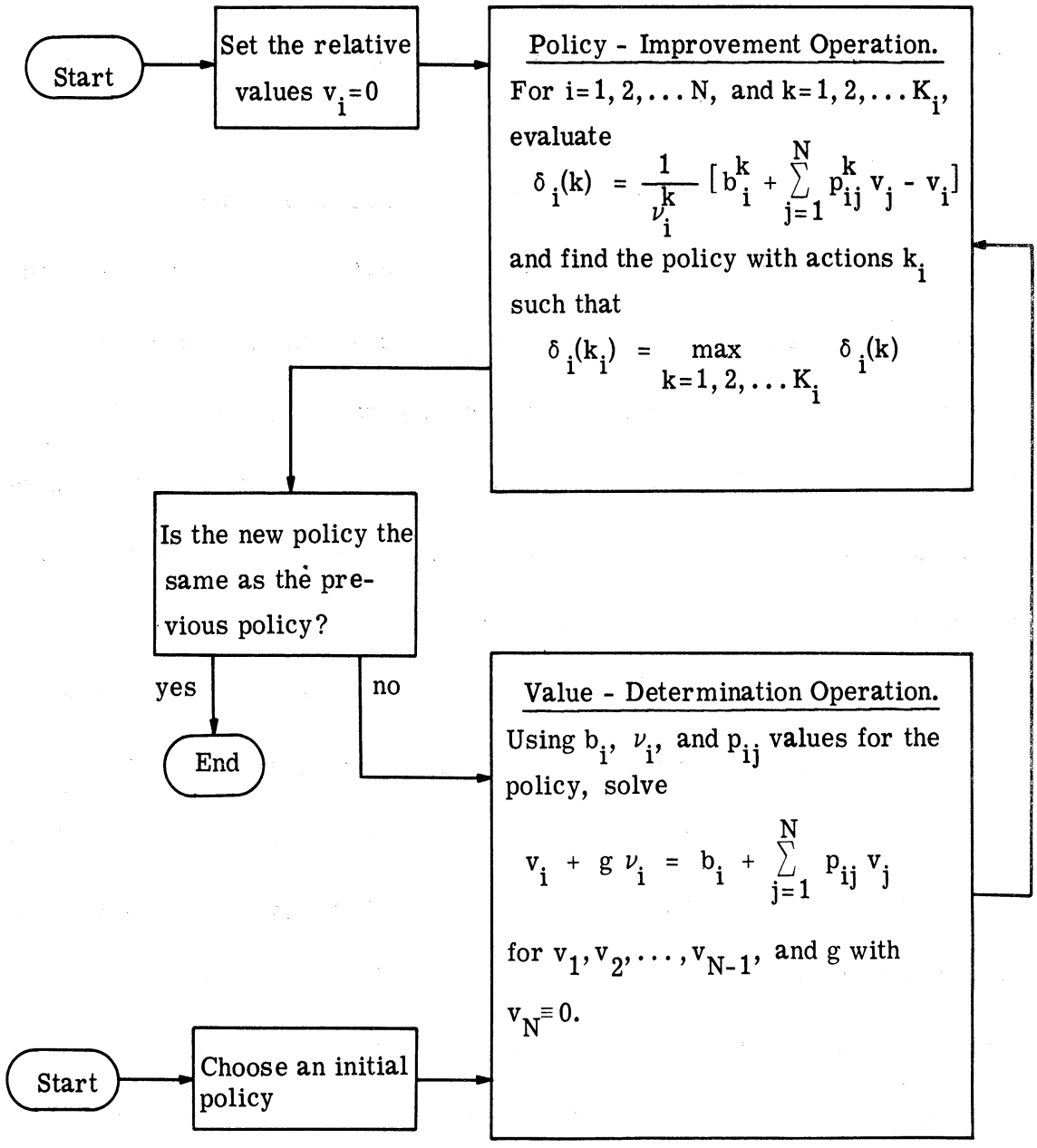


Fig. 4.2. The computation of the Howard algorithm.

Should there be several values of k satisfying (4.15), any one can be used. If this occurs at the termination of the algorithm then the optimal policy is not unique. All optimal P' policies are discovered by listing for each i all the k values maximizing (4.14) at conclusion of the calculation.

Howard has given in [74] the theorem which guarantees that the algorithm will always converge on a policy having maximum gain g. Convergence is assured regardless of whether one starts in the Value-Determination or Policy-Improvement, and regardless of the initial values used or the initial policy chosen. In this proof Howard assumes that the relative values v_i and the gain g are determined with perfect accuracy in the Value-Determination Operation.

Semi-Markov decision models of physical systems can be expected to have a large number of states. This will require use of a digital computer to carry out the computation of the Howard algorithm. Appendix D describes briefly the program which has been developed for this research. Because of the finite nature of computer arithmetic, it is inevitable that errors will arise in the course of computation. In order to have confidence in the Howard algorithm, some thought must be given to the effect of such errors.

4.5 COMPUTATIONAL CONSIDERATIONS

The effect of computational errors on the Howard algorithm will be considered first. Satisfactorily disposing of this question, we then

turn attention to the actions which are found by the algorithm for the transient states. Theorem 4.3 assures one that such actions do not contribute to determining the gain g of the decision process. The Howard algorithm however determines the actions for transient states so as to maximize their relative values. Since the relative values do not enter in the optimization criterion, these actions are irrelevant for an optimal policy.

4.5.1 Errors

The Value Determination Operation, as implemented in the program described in Appendix D, produces a set of numbers, say $x_1, x_2, \ldots, x_N$, where

$$x_i \simeq v_i + g \qquad (i=1,2,\ldots,N-1)$$

and $x_N \simeq g$. The computation is only approximately correct because of computational errors which inevitably arise. We consider that

$$\begin{aligned} x_i + \epsilon_i &= v_i + g \qquad (i=1,2,\ldots,N-1) \\ x_N + \epsilon_N &= g \end{aligned} \tag{4.16}$$

so that the values ϵ_i $(i=1,2,\ldots,N)$ are the errors of the computation. The values x_i are rational numbers with 8 significant digits (for the program of Appendix D). The ϵ_i are unrestricted in value, but are presumably not too large, say

$$|\epsilon_i| < 10^{-4} |v_i + g|$$

In this case the error in x_i would occur beyond the fourth significant digit. Such an assumption will not play a major role in the following however

The principal concern is whether the Howard algorithm will always converge to an optimal policy even if errors are present. The answer, strictly speaking, is no. Two difficulties appear to arise. First, the algorithm may converge to a non-optimal or sub-optimal policy. We can show in this case however that the gain of the policy will differ from the optimum within the same order of magnitude as the computational errors. Second, the algorithm may fail to converge at all. This would be evidenced by a continuing iteration among policies whose gains are very close in value. In the following paragraphs we explain why these problems can come about, and show how to cope with them in practice.

The argument proceeds via the same consideration used by Howard to prove the theorem on convergence without errors. Suppose that the Value Determination Operation (VDO) has been carried out for some policy, called policy A. Consider what the Policy Improvement Operation (PIO) indicates for some other policy, say policy B, based on the results of the VDO for A. Excluding errors, the difference in the test quantities of state i for the two policies is[1]

$$\xi_i = \frac{1}{\nu_i^B}\{b_i^B - v_i^A + \sum_{j=1}^{N} p_{ij}^B v_j^A\} - \frac{1}{\nu_i^A}\{b_i^A - v_i^A + \sum_{j=1}^{N} p_{ij}^A v_j^A\} \qquad (4.17)$$

[1]Superscripts A and B refer to the values for policies A and B.

It can be seen, as Howard has indicated, that the difference in the gains of the two policies is

$$g^B - g^A = \sum_{i \in R_B} \frac{1}{\ell_{ii}^B} \xi_i \qquad (4.18)$$

where R_B is the recurrent class of states for policy B. Thus $g^B > g^A$ if and only if $\xi_i \geq 0$ for all i, with $\xi_i > 0$ for at least one state $i \in R_B$. When there are negligible errors in the VDO, the PIO will declare policy B to be an "improved" policy if these conditions on ξ_i hold. Thus the PIO can discover any policy B for which $g^B > g^A$, assuming negligible errors.

Now consider the effect of errors. In our implementation of the Howard algorithm, the test quantities are

$$\delta_i^A = \frac{1}{\nu_i^A} \{b_i^A - x_i^A + \sum_{j=1}^{N} p_{ij}^A x_j^A\}$$

and

$$\delta_i^B = \frac{1}{\nu_i^B} \{b_i^B - x_i^A + \sum_{j=1}^{N} p_{ij}^B x_j^A\}$$

where x_i^A, i=1,2,...N, are the values computed in the VDO for A. The values b_i^A, b_i^B, ν_i^A, ν_i^B, p_{ij}^A, p_{ij}^B can be assumed to have negligible error since they are given data. Moreover any round-off error in computing δ_i^A, δ_i^B using the x_i values can be treated as negligible compared to errors in the VDO. Then using (4.16) in the two previous equations, one has

$$\delta_i^A = \frac{1}{\nu_i^A}\{b_i^A - v_i^A + \epsilon_i^A + \sum_{j=1}^{N} p_{ij}^A v_j^A - \sum_{j=1}^{N} p_{ij}^A \epsilon_j^A\}$$

and

$$\delta_i^B = \frac{1}{\nu_i^B}\{b_i^B - v_i^A + \epsilon_i^A + \sum_{j=1}^{N} p_{ij}^B v_j^A - \sum_{j=1}^{N} p_{ij}^B \epsilon_j^A\}$$

Now we have from (4.17) and (4.18) that

$$g^B - g^A = \sum_{i\epsilon R_B} \frac{1}{\ell_{ii}^B}\{\delta_i^B - \delta_i^A - \epsilon_i^A\left[\frac{1}{\nu_i^B} - \frac{1}{\nu_i^A}\right] + \frac{1}{\nu_i^B}\sum_{j=1}^{N} p_{ij}^B \epsilon_j^A - \frac{1}{\nu_i^A}\sum_{j=1}^{N} p_{ij}^A \epsilon_j^A\} \qquad (4.19)$$

Assuming that B is any stationary policy, the result we would like at this point is an upper bound on the difference $g^B - g^A$ in terms of δ_i^B, δ_i^A, and ϵ_j^A. Since δ_i^B and δ_i^A can be observed during the computation, such a result would provide a means of evaluating whether some policy B is better than A. To accomplish this, we first use a limit theorem for Semi-Markov processes ([92] or [94]) relating ℓ_{ii}^B to the equilibrium probability π_i^B of state i in the underlying Markov chain. This is

$$\frac{1}{\ell_{ii}^B} = \frac{\pi_i^B}{\sum_{j\epsilon R_B} \pi_j^B \nu_j^B}$$

and from this it follows that

$$\sum_{i\epsilon R_B} \frac{1}{\ell_{ii}^B} \leq 1$$

With this and the fact that ν_i^A, $\nu_i^B \geq 1$ we can state that

$$g^B - g^A \leq \sum_{i \epsilon R_B} \frac{1}{\ell_{ii}^B} (\delta_i^B - \delta_i^A) + \frac{3 \max |\epsilon_i^A|}{\min_{i,k} \nu_i^k}$$

The set R_B may not be easily determined by inspection of an arbitrary policy and we would like to circumvent its use. Let R(A,B) be the subset of system states for which the actions of policies A and B are different. Then $\delta_i^B - \delta_i^A = 0$ for all i not contained in R(A,B), so

$$\sum_{i \epsilon R_B} \frac{1}{\ell_{ii}^B} (\delta_i^B - \delta_i^A) \leq \max_{i \epsilon R_B \cap R(A,B)} (\delta_i^B - \delta_i^A) \sum_{i \epsilon R_B \cap R(A,B)} \frac{1}{\ell_{ii}^B}$$

$$\leq \gamma \max_{i \epsilon R(A,B)} (\delta_i^B - \delta_i^A)$$

where

$$\gamma = \sum_{i \epsilon R_B \cap R(A,B)} \frac{1}{\ell_{ii}^B}$$

is positive and less than unity. So we finally have

$$g^B - g^A \leq \gamma \max_{i \epsilon R(A,B)} (\delta_i^B - \delta_i^A) + 3 \frac{\max |\epsilon_i^A|}{\min_{i,k} \nu_i^k} \qquad (4.20)$$

Now if the algorithm has converged to policy A, we have $\delta_i^B \leq \delta_i^A$ for all i and any policy B different from A. As seen from (4.19) however, the errors ϵ_i^A may conceivably be such that in fact $g^B > g^A$ for some policy B. Equation (4.20) ensures that g^B cannot exceed g^A by more than the value of the right most term. Moreover if $\max_{i \epsilon R(A,B)} (\delta_i^B - \delta_i^A)$ is a

sufficiently large negative number for all policies other than A, we are safe in concluding that policy A is indeed optimal. In practice the values of $\delta_i^B - \delta_i^A$ can be used to resolve this question.

The possibility that errors will lead to an unending iteration can be avoided by terminating the computation whenever the computed gain x_N does not increase on every iteration. This also prevents the additional iterations brought about by transient states (see the next section). A drawback of this modified procedure occurs in the possible event that the algorithm passes through two policies of equal gain before reaching the optimal policy. Termination of the computation would be premature. This event can be ascertained in practice by inspecting the new policy obtained on the last iteration. Should the new policy have the same actions in the recurrent states as the old policy, the new one is optimal. If not, then a premature termination has occurred and the computation should be begun again with some other initial policy. If a policy with larger gain exists, it will be discovered after one or more such attempts.

In conclusion, it is important to keep computational errors reasonably small in order for the Howard algorithm to be successful in practice. Any detrimental effect of small errors is easily surmounted by observing the sequence of policies and test quantities obtained in the computation.

4.5.2 The Actions for the Transient States

Suppose that there is no difference in the test quantities for policies A and B for all states in the recurrent class R_B. From (4.18), the gains of the two policies must therefore be equal. There may however be a positive difference in the test quantities for some state i not in the recurrent class, i.e., a transient state. The Howard algorithm would still suggest policy B as better than policy A. But for what reason?

The answer given by Howard in [74] is that the relative value v_i^B is greater than v_i^A. Thus when further increase in the gain is impossible, the Howard algorithm determines the actions for the transient states so as to maximize the relative values.

To see this it is sufficient to consider the recurrent class of states as consisting of a single absorbing state, say state N. Then $p_{Nj} = 0$ for $j \neq N$ and $g = \frac{b_N}{v_N}$, assuming as usual that $v_N = 0$. For $i \neq N$, consider (4.11) in the form

$$v_i = \left(\frac{b_i}{v_i} - g\right) v_i + \sum_{j=1}^{N-1} p_{ij}\, v_j \qquad (i=1,2,\ldots,N-1)$$

or in matrix form

$$\underline{v} = \underline{e} + P^* \underline{v}$$

where $\underline{v}$ is a column vector of the v_i, $\underline{e}$ is a column vector of terms $(b_i/v_i - g)v_i$, and P^* is an (N-1) x (N-1) matrix obtained from the transition probability matrix P by deleting the $N^{\underline{th}}$ row and column.

From this equation

$$\underline{v} = (I-P^*)^{-1}\,\underline{e}$$

The inverse $(I-P^*)^{-1}$ is guaranteed to exist and

$$(I-P^*)^{-1} = I + P^* + (P^*)^2 + \dots . \qquad (4.21)$$

see [96, pp. 22 and 46]. In fact (4.21) shows that the terms in the inverse matrix $(I-P^*)^{-1}$ are the expected number of times the system enters state j (a transient state) in an infinite number of transitions, having started in state i. Thus denoting the latter quantity by m_{ij},

$$v_i = \sum_{j=1}^{N-1} m_{ij} \left(\frac{b_j}{\nu_j} - g\right) \nu_j \qquad (4.22)$$

Now (4.22) is equivalent to the set of Equations (4.11). So upon realizing that (4.18) arising from the equation

$$(v_i^B - v_i^A) + (g^B - g^A)\nu_i^B = \nu_i^B \xi_i + \sum_{j=1}^{N-1} p_{ij}^B (v_j^B - v_j^A)$$

is a solution to (4.11), we have from (4.22),

$$v_i^B - v_i^A = \sum_{j=1}^{N} m_{ij}^B \left[\xi_i - (g^B - g^A)\right] \nu_i^B$$

But if $g^B - g^A = 0$ and $\xi_i \geq 0$ with $\xi_i > 0$ for at least one i, this reveals that

$$v_i^B - v_i^A > 0$$

as Howard has pointed out.

With the confidence and understanding derived from these theoretical results, we can now proceed to use the Semi-Markov decision process formulation and the solutions obtained via the Howard algorithm for the study of control policies for queues.

CHAPTER V

A DECISION MODEL FOR A MULTIPLE-ACCESS COMPUTER

The introductory discussion of multiple-access computer systems given in Chapter I has shown that proper control of queues is important in obtaining most effective performance from such on-line systems. At the same time, the classical results of queueing theory provide very meager aid to understanding the control of queues in existing systems, since their complexity forbids a closed form analysis in the traditional approach. Those analyses which have been produced have depended on very special assumptions which limit their applicability. Considerable progress could be made through the numerical solution of complex queueing models, or through simulations, but no really appropriate studies of this kind have been publicized. The large number of alternative models and queue control procedures for computer time-sharing, which limit the generality of any single analysis, may be part of the explanation for this.

It is therefore a timely and worthwhile effort to apply the discrete Semi-Markov decision theory which has been developed to the queueing control problem of multiple-access computers. This chapter and the following one will serve to demonstrate that a rather general model can be developed for existing systems, and that a number of useful conclusions are obtained. The present chapter will concentrate on the formulation and justification of the model. The restrictions imposed on the physical systems represented by the model are explained first. The next chapter

will present the conclusions obtained from solutions of the decision via the Howard algorithm.

5.1 THE PHYSICAL SYSTEM

The process of developing a mathematical model which accurately represents a physical system is generally quite difficult. Ideally, the development is well supported by experimental data on subsystem behavior and interactions. Usually however the need for analysis and quantitative system studies antedates the final design specification and prototype construction, whereupon such information could be obtained. The system engineer must conduct preliminary studies relying on various hypotheses to construct appropriate models.

Bearing this in mind, we can proceed to describe the important physical characteristics of a hypothetical time-shared computer system whose behavior is quite reasonably represented by a discrete Semi-Markov decision process. The model for this system captures to a significant extent the important and fundamental properties of time-shared computer operation today. No simulation or analysis to our knowledge has dealt with a more realistic or more general model of these systems. The decision model, as we shall see, includes the important queue control policies currently in use in actual systems as well as a great number of more complex procedures. The simplifying assumptions we will make are motivated in most instances by a desire to promote a clear understanding of the relation between the decision model and the physical system, not by a limitation of the technique.

5.1.1 Hardware Structure

The computer system consists of a single arithmetic processor, a communications interface with the telephone lines to remote users, a magnetic drum serving as the secondary memory, and a magnetic core main memory unit.

The drum memory is a large storage capacity unit, and we assume its revolution time to be 50 ms. This value does not enter into modeling the system, but does enter in the physical interpretation of results for the model. Because of its fixed revolution time, the drum provides a convenient source for clocking information for the system. We will assume that the system derives interval timing information from a drum interrupt signal occurring on each revolution of the drum. Such signals could be obtained by detecting a field of bits recorded on a special clock track. This assumption will lead to the discrete-time character of the decision model for the system.

The data organization of the drum memory is an important consideration in arriving at an appropriate model. This is so because some interval of non-productive time is required to find any program on the drum, bring it into the main memory, and prepare it to begin execution. The data organization influences the duration and variability of this interval. The drum memory is organized into a number of recording fields. Each field consists of a band of adjacent tracks running around the circumference of the drum. All fields have their origin at the same circumferential point. Each field consists of the same number of fixed

length records. A data or program file on drum memory is comprised of some number of records. It will be assumed that the successive records of any file are stored sequentially in one field, and if the file requires more than one field, the remaining records are stored sequentially in one or more adjacent fields. The process of retrieving a file will consist of reading the records of the first field occupied by the file, until the initial record of the file is located, then transmitting the file record-by-record and serially-by-field to the main memory. Files which occupy one field or less can therefore be retrieved in one drum revolution. The origin of the drum fields is displaced from the clock field by one-half revolution. This gives the processor sufficient time to initiate a drum read/write operation after receiving a clock interrupt, if desired. See Fig. 5.1.

The main memory is taken to be of moderate size, allowing say 8000 words for use by user programs, the remainder of the memory devoted to Executive Control Program usage. This rather limited user memory space forbids having multi-programming, in which more than one user program could occupy main memory at one time. Consequently, only one user program occupies main memory at any time and it can be permitted to access any part of the user memory area. The entire area (8000 words) must therefore be unloaded when the user is removed from main memory. An important class of data files on the drum consists of these "core images" for user programs which have been unloaded from main memory.

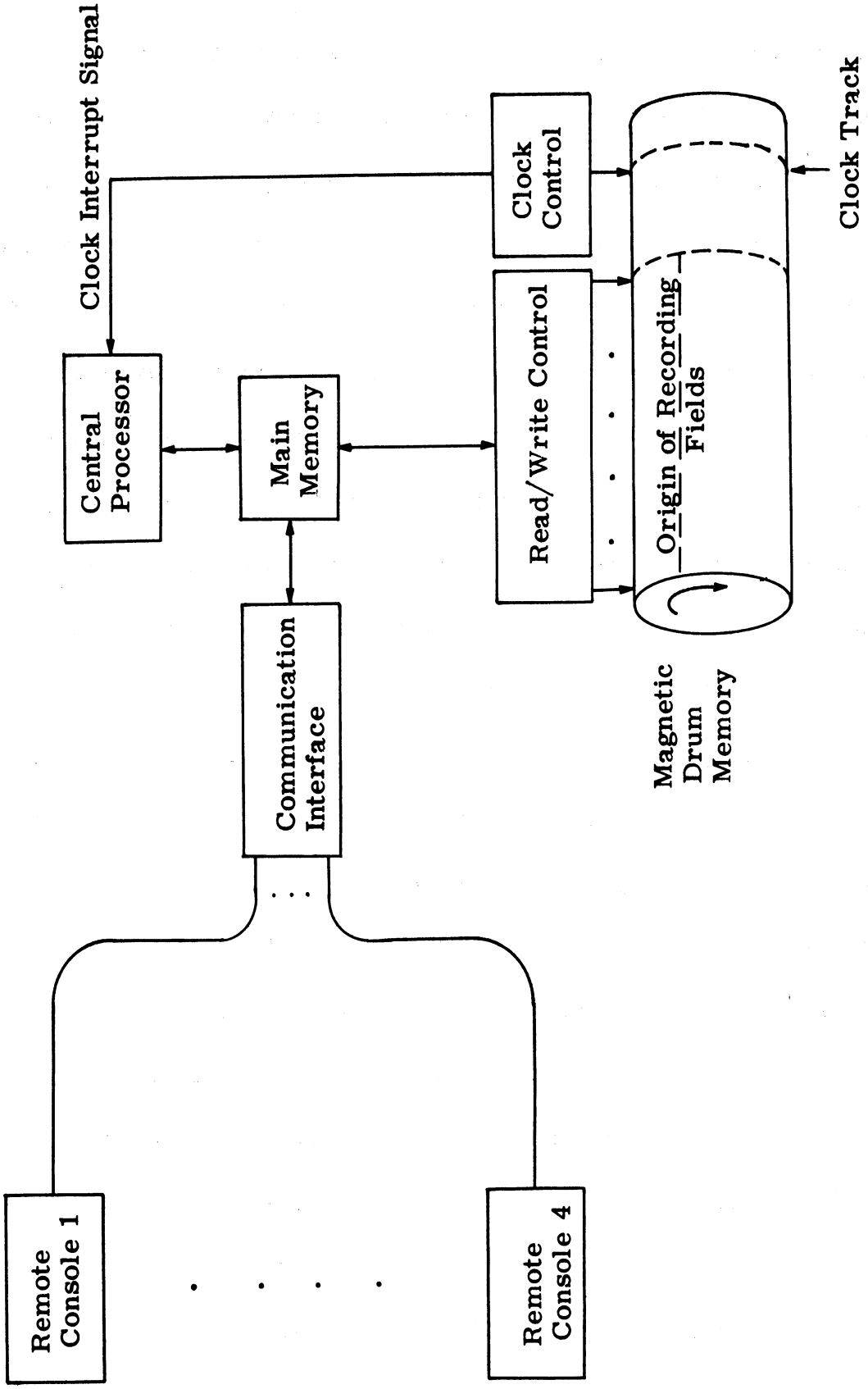


Fig. 5.1. Organization of the system to be studied.

Unloading of a user could be done in one drum revolution, assuming that the drum is capable of storing a user core image in one recording field around the drum periphery. This is certainly possible with present day drum memories. If the user memory area is larger than the drum field capacity, several revolutions may be required to unload a user program. We shall assume at all times that the user memory area is an integer multiple of the drum field capacity.

The portion of main memory devoted to the Executive Control Program is to be large enough that the ECP and its associated tables can permanently reside there. No control function should require retrieving a data file from the secondary memory before it can be carried out. The ECP area shall include the subroutines needed for scheduling and control of queues, for communicating with the drum memory and properly loading the programs needed to carry out a user command. The command programs themselves are all to be retained in the drum memory.

As shown in Fig. 5.1, we shall only treat a system with four remote consoles actively in use at one time. This is done to simplify the task of specifying the data for the decision model.

5.1.2 Software Structure

A good understanding of the behavior of a multiple-access computer system requires a familiarity with the structure of programs or "software," in addition to that of hardware. Two principal classes of programs are involved, the Executive Control Program, and the user programs. In regard

to user programs, interest centers on the programming conventions to which each user must conform in order that his programs will operate on the system.

One programming convention which we shall introduce in the system being modeled is that no user program can directly issue commands to the ECP which call for the execution of another program. This means that any programs or subroutines required must be loaded as part of a user program, or else brought into action by command from the user console. This is not an especially desirable convention for it limits the flexibility of usage of the system. On the other hand, it makes the ECP somewhat simpler in design.

Another convention imposed allows user communication with the secondary memory only through a command program initiated from the user console. Moreover, the largest admissible data transfer is equal to a drum field, which can be accomplished in one drum revolution.

The above conventions ensure that all commands entering the system come from a user console. The latter convention also ensures that the minimum time interval in which a program would be allowed to run is sufficient to accomplish any data transfer, without interruption and loss of data.

An additional convention we will require is that each console may have only one program or command in process at any time. This is generally the case in existing systems.

The Executive Control Program must perform essentially the same functions discussed in Chapter I. Because the drum memory is used via a command program, the diagram of Fig. 1.2 can be simplified, eliminating the I/O Queue from the ECP. (The I/C Queue becomes a part of the Job List, which is the central queue.) Because the drum interrupt only counts off the passage of a 50 ms interval, the ECP must include a time-keeping function not emphasized in Fig. 1.2. In order to measure any prescribed interval, we presume the ECP uses a one-word register, called the clock register, which the ECP decrements by one upon receipt of each drum interrupt. If the clock register is initially set to any number, it subsequently will reach zero value when that number times 50 ms has elapsed. Figure 5.2 indicates this function, and gives more detail than Fig. 1.2 on ECP functions for the system we shall model.

A user program or command program is removed from further ECP consideration for several reasons: an error, such as arithmetic overflow; successful termination of execution; or, an excessive amount of output to the console. The latter points out another assumed convention. Presumably the output buffer assigned to the user is adequate for normal communication of several type written lines in a short interval of time. If the user program generates output equal to the buffer capacity in less time than the communication processor can transmit it to the console, the ECP will remove the program. The user must reactivate the program by command from the console to obtain further output. In a

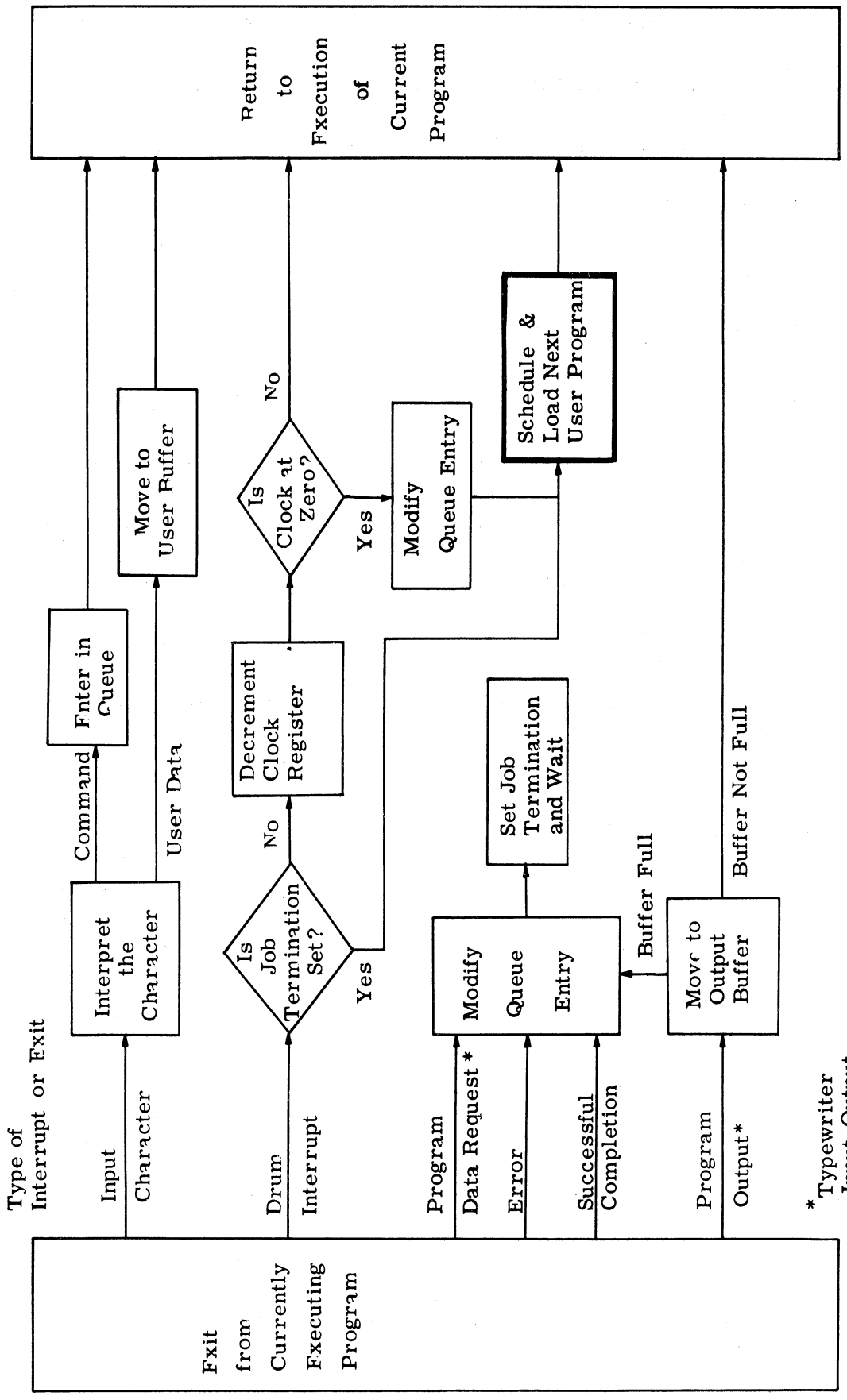


Fig. 5.2. Executive control functions in the system being modeled.

physical system, this convention would serve a useful purpose in discouraging users from excessive typing, and as a safety measure against program errors which result in excessive output. For purposes of modeling, it simply eliminates the possibility that jobs will exist in queue which require further execution but cannot be given it, even if desired.

The decision model we shall formulate shortly serves to investigate the queue control actions involved in scheduling the execution of jobs. This ECP function is emphasized in Fig. 5.2. Each scheduling decision or queue control action requires the selection of a single job for execution, and the allocation of a maximum interval of time in which the job may execute. We now describe the general framework in which this selection and allocation is to be made in the physical system.

5.1.3 General Control Process

The physical constraints resulting from the assumed structure of hardware and software, and additional practical constraints which will be discussed now, establish the general form of the queue control process for the system.

Because there is a single arithmetic processor, the computer system appears as a single server service system. There is a finite source of arrivals, because only four consoles are allowed to be in use at one time and each may have but one job in process. Since only one job may be held in memory, changing from execution of one user program to another involves a swapping times for saving one core image and loading the other.

An additional time may be needed if the second program is a newly received command. This will be called the "set-up time" S. It is the computing time required to establish proper relocation and linking of the subroutines comprising the program.

At any time that jobs are present for service,[1] the ECP has a very basic choice of either executing one of the jobs, or simply remaining idle, waiting for a new command to arrive. Conceivably the latter course could be useful if it was imperative to give immediate attention to newly received commands, and there was no way to interrupt a job in execution. The time-keeping function of the ECP, which controls the maximum time allocated to a job on any execution pass, always provides a way to break into any job. We assume that proper allocation of this time interval will in practice be sufficient to eliminate any need for the action of leaving the machine idle.

Every queue control action therefore consists of selecting a job to be executed and allocating a maximum interval of time in which the job may run before possibly being swapped for another. The choice involved in each action should conceivably be based on the totality of information obtainable about the jobs present. Such information could consist of the number of jobs, the list of subroutines comprising each program, the total program size of each in words, the execution time each has obtained, the console originating each command, a user estimate

[1] That is, waiting in queue or being executed.

on how much execution time is needed, and many other data. The applicable and useful data is sharply diminished by practical considerations and the physical constraints already imposed.

As a general principle, the execution time required to complete a command is not accurately known before hand. So it appears reasonable for simplicity to consider the execution time of a job as a random quantity, with a probability distribution obtained by observation of the entire population of user jobs submitted. Because the system of interest here has a somewhat limited user area in main memory, there will not be a wide variation in total user program size. Thus any correlation between program size and execution time will not be as distinctive and useful as in a larger memory system. We shall assume then that the simplest and most relevant item of information on a single job which bears upon the queue control actions is the execution time which the job has received at any point in time. Neither the name or class of a program or the originating user enters the scheduling process. This situation is sometimes referred to as the "context-free" case.

The most general scheme possible for scheduling is therefore as follows. Each job in queue has associated with it the value of execution time which it has received. Whenever a job is placed in execution it is allocated a maximum interval for execution. There is no restriction on the interval which the ECP may allocate, other than it being an integral number of 50 ms steps. At the expiration of that interval, the ECP examines the queue again. The ECP then selects another job to be executed

or elects to continue executing the same job. Upon swapping, if any, the ECP determines another value for the execution interval and turns control of the computer over to the user job for that interval.

We shall simplify and modify this framework in several ways. These restrictions are consistent with design simplicity for the ECP and present practice, but also are convenient for modeling the system. We shall limit the permitted values for the allocated execution interval to three. The first value, denoted e_2, is to be assigned to every new job receiving its first pass of execution. The second value, denoted as a difference (e_3-e_2), is allocated to any job entering a second execution pass, having previously received an interval of duration e_2. The third value, denoted by q, is assigned to every job which has previously received execution for a total time e_3 or greater.

These time allocations result in establishing a multi-level queueing structure. Each level corresponds to a particular value of accomplished execution time. The successive values of accomplished execution are 0 (new arrivals), e_2, e_3, $e_3 + q$, $e_3 + 2q, \ldots$, etc. As a job receives time on the processor it passes from one level to the next lower one, or is completed and leaves the system, as shown in Fig. 5.3.

We shall further assume that when there are two or more jobs at the same level, for either levels 1 or 2, the earliest arrival is to be chosen for execution. For example, if there is a job at level 2 and another job is just completing its first pass of execution, the former

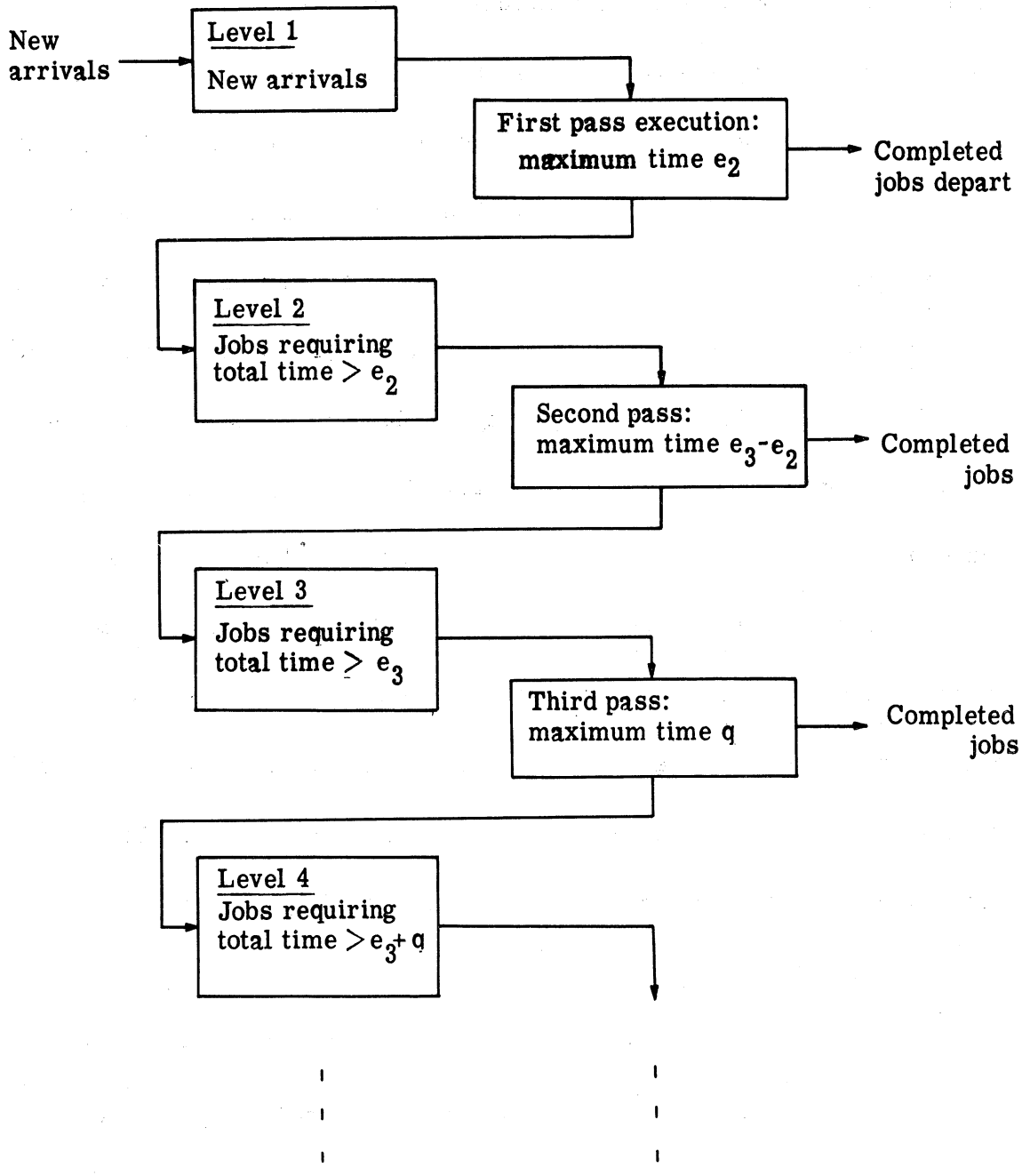


Fig. 5.3. Multi-level queueing resulting from restricted execution time allocation.

would be preferred to the latter for next execution, since both jobs are then at the same level. This will involve swapping these jobs, which could otherwise be avoided by continuing execution of the latter. A loss of productive time thus occurs, but it will be a small loss when swapping is done rapidly, as it will be with a drum memory. Maintaining order by time of arrival among equivalent jobs seems to be a good service policy from an individual user's view, and is followed in most systems. Thus the small additional loss in time to achieve this seems worthwhile.

Finally for additional simplicity we shall consolidate the levels corresponding to e_3, $e_3 + q$, $e_3 + 2q, \ldots$, into a single third level queue which is handled in a round-robin fashion. Whenever a job from this queue completes an execution interval of duration q, it is moved to the end of the queue. Jobs reaching the third level by execution from level 2 are placed at the end of the third level queue, which is consistent with a head-of-the-line selection on each level. The three queue levels which now constitute the framework for the scheduling process will be denoted by Q_1, Q_2, and Q_3.

Figure 5.4 illustrates the general control process. Note that an execution pass on only one job can be in progress at any time, and therefore each scheduling decision corresponds to the choice of the queue level which will be served. The Semi-Markov decision process which we will now formulate will allow us to investigate how this choice should optimally depend on the number of jobs present in each queue.

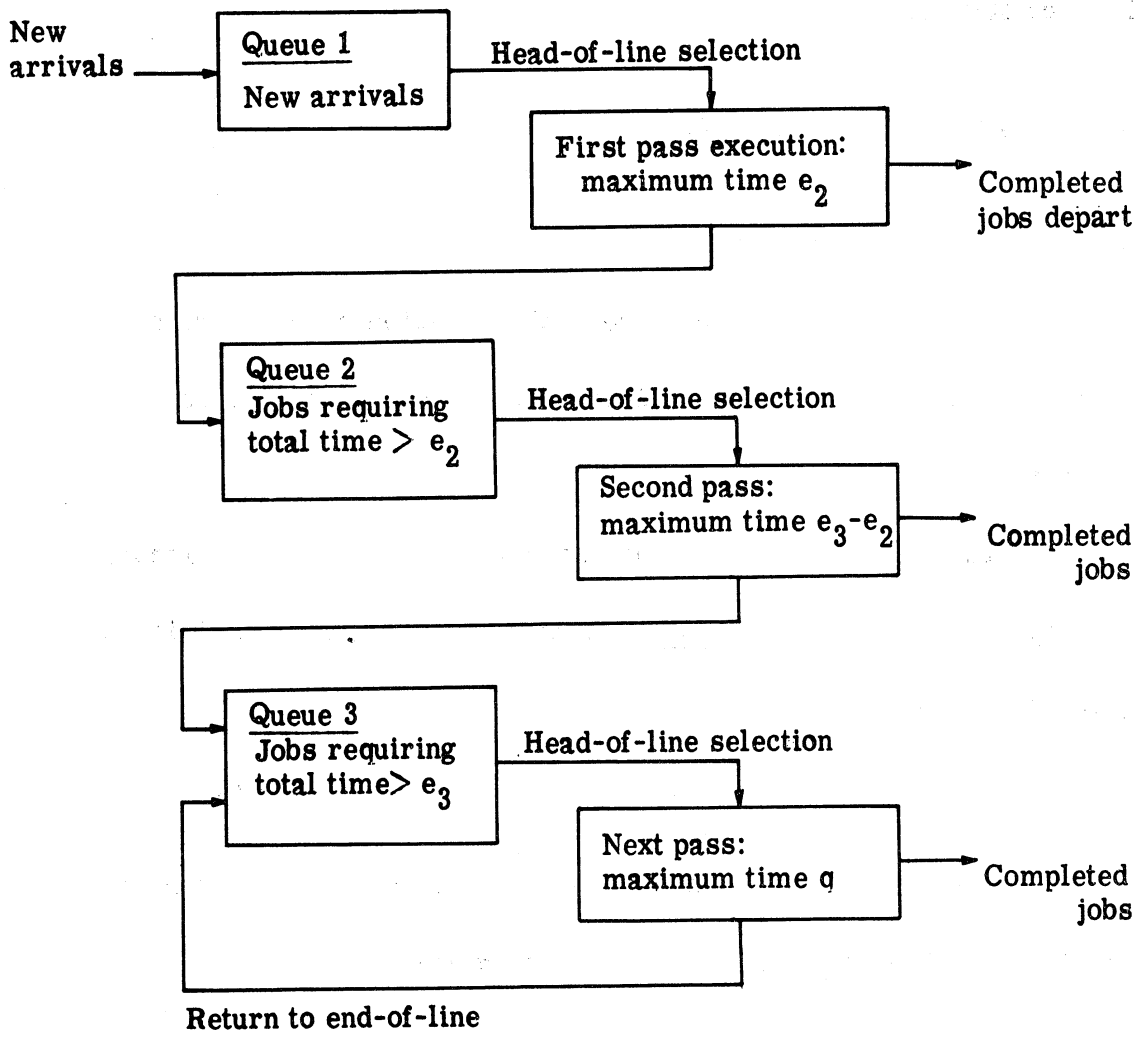


Fig. 5.4. Three level queue with round-robin service on last level.

We point out that the freedom to assign the choice arbitrarily allows us to include priority policies, as used in existing systems such as Project MAC [18].

5.2 FORMULATION OF THE MODEL

5.2.1 Simplifying Assumptions

A formulation of the queue control operation as a discrete Semi-Markov decision process is obtained on the basis of the following assumptions.

1. Each command arriving at the computer requires an execution time t_e which is an independent random variable with probability distribution function $F(T)$,

$$F(T) = \Pr(t_e \leq T), \quad T > 0.$$

2. The distribution function $F(T)$ has a negative exponential tail for $T > e_3$. That is, for some constants σ and μ

$$F(T) = 1 - \sigma e^{-\mu T}, \quad T > e_3 \qquad (5.1)$$

with $\mu > 0$, and $0 < \sigma \leq 1$.

3. The time interval between completion of any user's command and receipt of the next command from the user is called the arrival time. It is assumed to be an independent random variable with a negative exponential distribution. Denoting this interval by t_a, we have

$$\Pr(t_a \leq T) = 1 - e^{-\lambda T}, \quad T > 0 \qquad (5.2)$$

and $T_u = 1/\lambda$ is the expectation of t_a.

4. The swapping time s and the set-up time S are of constant value, and an integral number of 50 ms steps.

5. The event that an arriving command requires a set-up time is an independent random event with probability θ, $0 \leq \theta \leq 1$.

6. The ECP actions occurring in the interval between successive scheduling decisions are accomplished in negligibly short time.

Assumptions 1, 3, and 5 are necessary in order to achieve a formulation of the physical system within the theory we have developed. The independence of the random variables, and (5.2) are the key parts of these assumptions. We can offer no concrete justification for these. They are examples of the assumptions which the system engineer is forced to make, the validity of which can only be verified by examining statistics of actual system operation. The need for statistical observations of time-shared computer operation is coming to be realized by the management of existing systems, and some observations have been reported [19, 48, 97]. No reported data bears on assumptions 1, 3, and 5 however. As long as we are seeking to establish broad notions of system behavior, these appear to be reasonable approximations. The use of (5.2) can easily be relaxed by use of a different model, but this brings up a new theoretical area which is beyond our present effort. More will be said on this point in Chapter VI.

Quite reasonable justification can be given for assumption 2 on the basis of some recent observations [98, 99] on the IBM 7090 batch-processing system at The University of Michigan Computing Center. Figure 5.5 shows

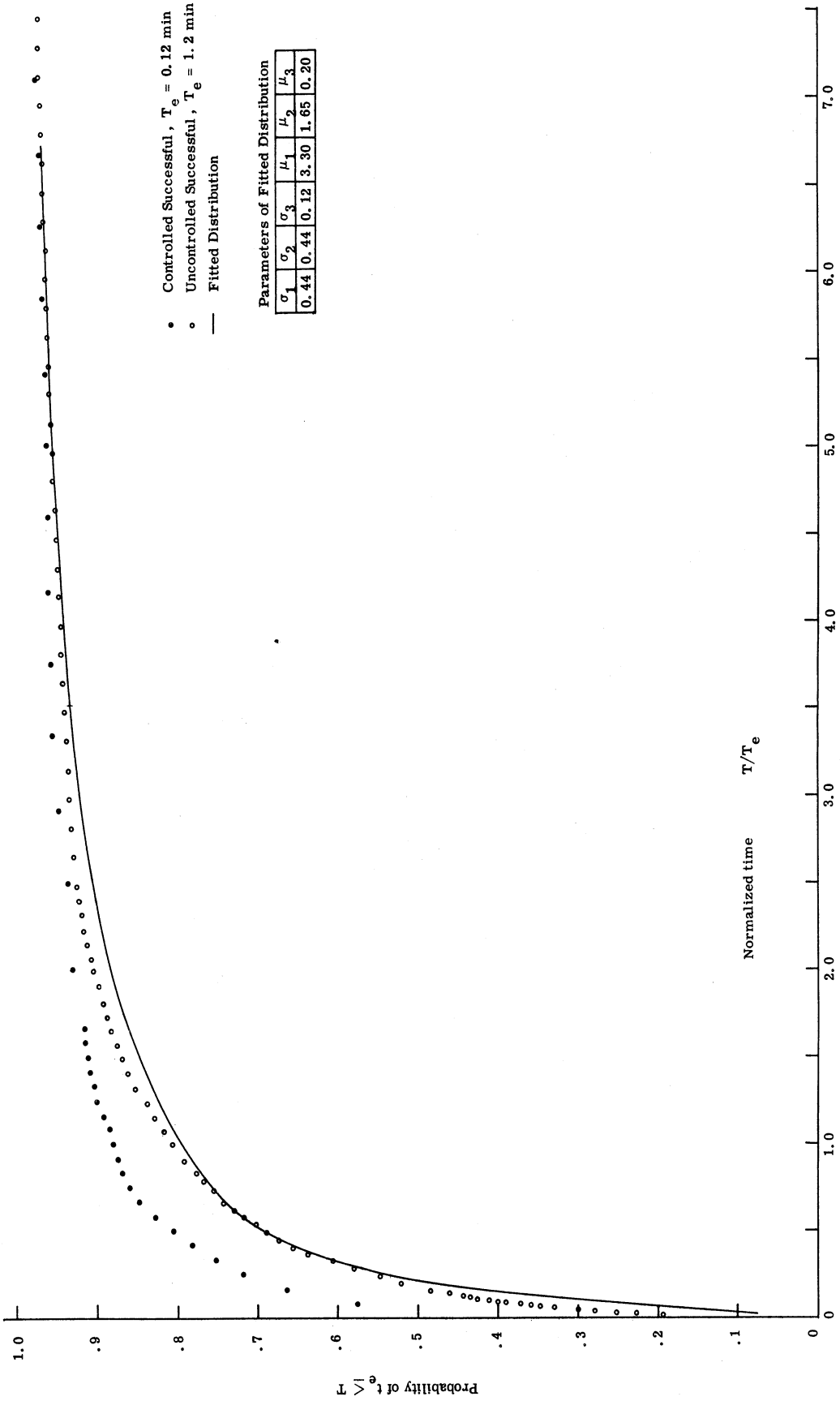


Fig. 5.5. Observed distribution of execution time at The University of Michigan Computing Center.

the probability distribution function on execution time for two classes of users of this system. The classes are designated "Controlled," which consists of all student programming for scheduled courses, and "Uncontrolled," which includes faculty, sponsored research, and thesis users. Only programs which executed without recognized errors or resort to memory dumps are included, hence the term "Successful" is applied to these programs. The distribution functions are plotted on a time scale T which has been normalized to the mean execution time, T_e. For Controlled users the observed mean was 0.12 minutes, while for Uncontrolled users the observed value was 1.21 minutes. If is clear that for $T/T_e > 2.5$, the tails of the two distributions are much alike.

Figure 5.5 also shows a rather good fit to the observed data by means of a distribution known as a three-phase hyper-exponential ([7], p. 39ff), described by

$$F(T) = 1-\sigma_1 e^{-\mu_1 T}-\sigma_2 e^{-\mu_2 T}-\sigma_3 e^{-\mu_3 T} \qquad (5.3)$$

with $\sigma_1 + \sigma_2 + \sigma_3 = 1$, and $0 \leq \sigma_i \leq 1$, i=1,2,3. One can see that for $\mu_1 > \mu_2 > \mu_3$ and T sufficiently large, (5.3) gives

$$F(T) \approx 1 - \sigma_3 e^{-\mu_3 T}$$

which is the behavior required by (5.1). Figure 5.6 shows the tail of the observed data for Uncontrolled users on a semi-logarithmic plot, verifying that the observed execution time distribution does indeed have an exponential tail.

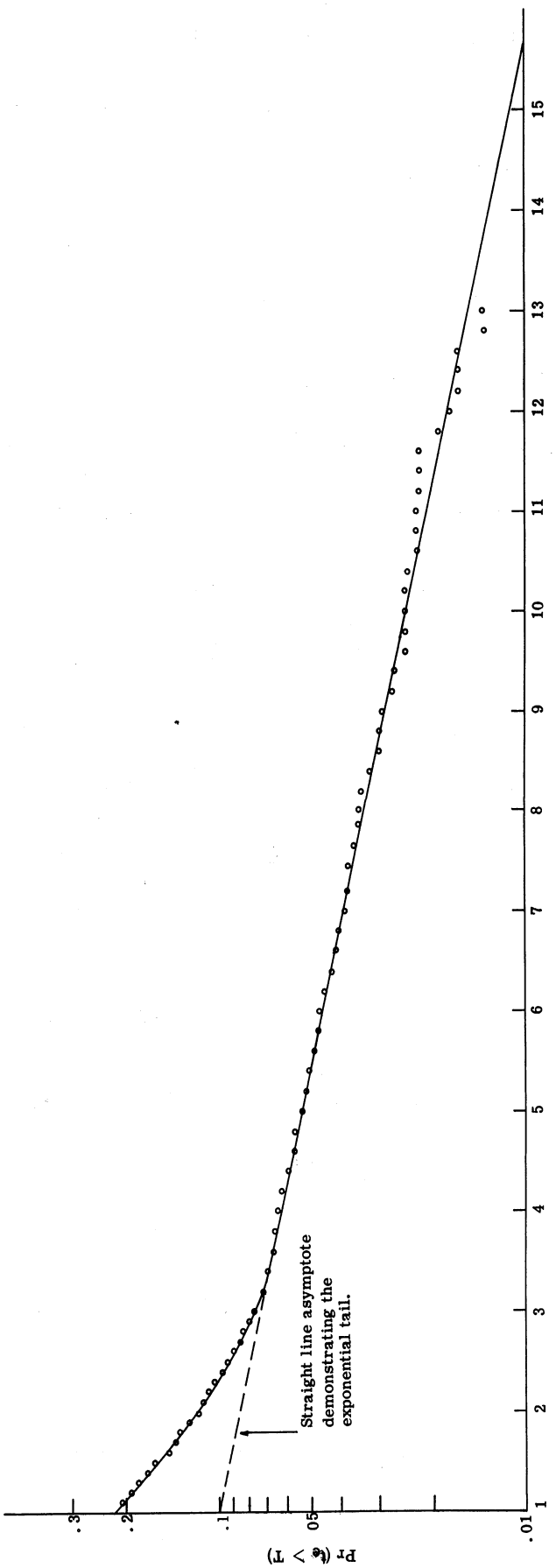


Fig. 5.6. Observed probability distribution of execution time for uncontrolled users of The University of Michigan Computing Center.

Of course, time-shared computer programming is different than batch processing, due largely to the convenience of direct communication. Two facts suggest that the result of Figs. 5.5 and 5.6 can be carried over. First, Fig. 5.5 indicates that two classes of batch processing users, engaged in fundamentally different activities, have similar execution time distributions despite a wide disparity in the mean values. Second, the time-sharing system we are modeling retains the fundamental structure of a main program and associated subroutines, loaded together at execution, which characterizes a batch processor. We shall therefore use execution time distributions of the class described by (5.3) in investigating the multiple-access computer. The mean execution time however will be taken to be on the order of a few seconds, which one expects for users taking advantage of the man-machine interaction possible with the system.

Assumptions 4 and 6 could be avoided by a more elaborate model than we will investigate. Instead of 4 we could assign probability distributions to the swap time s and the set-up time S. These would have to be justified on some grounds which account for the data organization of the secondary memory and the operation of the loader subroutine of the ECP. The size of the user memory area, the integral relation between this and the drum field capacity, and the fact that the ECP operation is in effect synchronized to the drum through the drum interrupt, seem sufficient to warrant 4 as a good approximation.

Assumption 6 is easily justified since from Fig. 5.2 the primary ECP actions which occur are timekeeping and console input interpretation. For the latter, the ECP simply examines each character received and makes an entry in a queue when a recognizable command is completely received. Considering the great speed of a modern processor, the comparatively slow speed at which the fastest typist can strike keys, and that a number of characters are usually required to make up a complete command, the time needed is negligible. For example, at a speed of 60 words per minute a typist takes one second to type 5 characters. Allowing as much as 100 memory cycles (2μs each) to process each character, the ECP input interpretation time is 0.1% of the time to type them. The timekeeping time in the same interval should be even less.

5.2.2 Definition of States and Actions

The state of the system for modeling as a decision process is described by four state variables. Three of these describe the jobs which may be waiting in queue for execution,

n_1 = the number of jobs in Q_1

n_2 = the number of jobs in Q_2

n_3 = the number of jobs in Q_3

The programs corresponding to jobs in Q_1, Q_2, or Q_3 are located on the drum memory. In addition, there may at any time be a job in the main memory which is being executed, or on the other hand, the memory may be empty due to the completion of a job. It is necessary to record this

information in the state in order to properly account for swapping time. The fourth state variable ℓ has values as follows,

ℓ = 0 if the user memory area is unoccupied,

ℓ = 1 if user memory contains a Q_1 job,

ℓ = 2 if user memory contains a Q_2 job,

ℓ = 3 if user memory contains a Q_3 job.

When we use the expressions "a Q_1 job," "a Q_2 job," "a Q_3 job," we mean a job which has obtained 0, e_2, or e_3 or more units of execution time, respectively. If ℓ = 1, 2, or 3, the job occupying the user memory area presumably does not appear in Q_1, Q_2, or Q_3, and hence is not counted in n_1, n_2, or n_3. (In the physical system, the list of job entries comprising Q_1, Q_2, and Q_3 would include the job, but with a suitable mark indicating it was in execution.)

The execution of a job begins at the completion of the swapping and set-up time, if any, needed to bring the program into the user memory area. Execution continues until the drum interrupt which signals the end of the maximum time allocation, or until an interrupt which terminates the job (see Fig. 5.2). The next scheduling action occurs at the drum interrupt marking the end of execution, or the first one thereafter if execution is terminated during a clock cycle. The scheduling action commences with an examination of the queue and a decision as to the queue to be serviced next. Then any necessary swapping and set-up is carried out. The one-half drum revolution between the drum interrupt which initiates the scheduling action and the origin of the drum fields should

be sufficient to accomplish all table searching and computing needed in the scheduling decision. The important point is that the decision is based on the state of the system at the end of an execution interval. Now the state variable ℓ can only have the value 1 at the beginning of an execution interval. This value is therefore excluded from the possible states observed at scheduling control points.

Table 5.1 lists the states of the system for formulation of the decision process. The description of the state by the values of the state variables is given in addition to an arbitrarily assigned integer index i. The value of N, the number of states, is 75. The action indices k = 1,2,3 correspond to the decisions to execute a Q_1 job, a Q_2 job, or a Q_3 job respectively, except for i = 1 where the system is empty, and the only possible action is to wait until a command arrives. The action k = 1 in state 1 corresponds to this course.

For each action, Table 5.1 also lists the indices for the possible end states of a transition from each state. A transition in state, as observed in scheduling, takes place over the interval between successive drum interrupts which start the scheduling control action. The system state throughout the interval is taken to be the state which the system had at the first interrupt. The state is considered as changing instantaneously at the second interrupt to a value determined by the random events which have transpired during the interval. Among these random events are arrivals of new commands from the user consoles. Each arrival increases the value of n_1 by one. Another random event is the completion

TABLE 5.1

DESCRIPTION OF POSSIBLE TRANSITIONS FOR EACH ALLOWED ACTION

State Index i	State Variable Description n_1	n_2	n_3	ℓ	Possible End State Indices for a Transition from State i: Action k=1	Action k=2	Action k=3
1	0	0	0	0	1,2,7,19,41		
2	1	0	0	0	1,2,5,7,10,19,22,44		
3	0	1	0	0		1,2,6,7,11,19,23,45	
4	0	0	1	0			1,2,6,7,11,19,23,45
5	0	0	0	2		1,2,6,7,11,19,23,45	
6	0	0	0	3			1,2,6,7,11,19,23,45
7	2	0	0	0	2,7,10,19,22,44		
8	1	1	0	0	3,8,14,20,26,48	2,7,11,19,23,45	
9	1	0	1	0	4,9,17,21,29,51		2,7,11,19,23,45
10	1	0	0	2	3,8,14,20,26,48	2,7,11,19,23,45	
11	1	0	0	3	4,9,17,21,29,51		2,7,11,19,23,45
12	0	2	0	0		3,8,15,20,27,49	
13	0	1	1	0		4,9,18,21,30,52	3,8,15,20,27,49
14	0	1	0	2		3,8,15,20,27,49	
15	0	1	0	3		4,9,18,21,30,52	3,8,15,20,27,49
16	0	0	2	0			4,9,18,21,30,52
17	0	0	1	2		4,9,18,21,30,52	3,8,15,20,27,49
18	0	0	1	3			4,9,18,21,30,52
19	3	0	0	0	7,19,22,44		
20	2	1	0	0	8,20,26,48	7,19,23,45	

TABLE 5.1 (Continued)

i	n_1	n_2	n_3	ℓ	k=1	k=2	k=3
21	2	0	1	0	9,21,29,51		7,19,23,45
22	2	0	0	2	8,20,26,48	7,19,23,45	
23	2	0	0	3	9,21,29,51		7,19,23,45
24	1	2	0	0	12,24,33,55	8,20,27,49	
25	1	1	1	0	13,25,36,58	9,21,30,52	8,20,27,49
26	1	1	0	2	12,24,33,55	8,20,27,49	
27	1	1	0	3	13,25,36,58	9,21,30,52	8,20,27,49
28	1	0	2	0	16,28,39,61		9,21,30,52
29	1	0	1	2	13,25,36,58	9,21,30,52	8,20,27,49
30	1	0	1	3	16,28,39,61		9,21,30,52
31	0	3	0	0		12,24,34,56	
32	0	2	1	0		13,25,37,59	12,24,34,56
33	0	2	0	2		12,24,34,56	
34	0	2	0	3		13,25,37,59	12,24,34,56
35	0	1	2	0		16,28,40,62	13,25,37,59
36	0	1	1	2		13,25,37,59	12,24,34,56
37	0	1	1	3		16,28,40,62	13,25,37,59
38	0	0	3	0			16,28,40,62
39	0	0	2	2		16,28,40,62	13,25,37,59
40	0	0	2	3			16,28,40,62
41	4	0	0	0	19,44		
42	3	1	0	0	20,48	19,45	
43	3	0	1	0	21,51		19,45
44	3	0	0	2	20,48	19,45	
45	3	0	0	3	21,51		19,45
46	2	2	0	0	24,55	20,49	
47	2	1	1	0	25,58	21,52	20,49

TABLE 5.1 (Concluded)

i	n_1	n_2	n_3	ℓ	k=1	k=2	k=3
48	2	1	0	2	24,55	20,49	
49	2	1	0	3	25,58	21,52	20,49
50	2	0	2	0	28,61		21,52
51	2	0	1	2	25,58	21,52	20,49
52	2	0	1	3	28,61		21,52
53	1	3	0	0	31,65	24,56	
54	1	2	1	0	32,68	25,59	24,56
55	1	2	0	2	31,65	24,56	
56	1	2	0	3	32,68	25,59	24,56
57	1	1	2	0	35,71	28,62	25,59
58	1	1	1	2	32,68	25,59	24,56
59	1	1	1	3	35,71	28,62	25,59
60	1	0	3	0	38,74		28,62
61	1	0	2	2	35,71	28,62	25,59
62	1	0	2	3	38,74		28,62
63	0	4	0	0		31,66	
64	0	3	1	0		32,69	31,66
65	0	3	0	2		31,66	
66	0	3	0	3		32,69	31,66
67	0	2	2	0		35,72	32,69
68	0	2	1	2		32,69	31,66
69	0	2	1	3		35,72	32,69
70	0	1	3	0		38,75	35,72
71	0	1	2	2		35,72	32,69
72	0	1	2	3		38,75	35,72
73	0	0	4	0			38,75
74	0	0	3	2		38,75	35,72
75	0	0	3	3			38,75

of the job being executed. This event changes the value of ℓ to zero. For every state except state 1 there will be a job in execution during the transition interval. The third random event is the non-completion of the job being executed. This will also result in a change in value of the state variable ℓ.

In order to see how the state transitions come about, let us consider some of those given in Table 5.1. For example, consider state i = 2. The only action possible in this state is to execute the Q_1 job present. Under this action the transition interval will be at least s units long, for this is the minimum time required to load the job program into the user memory area. The interval will usually be longer due to the possibility of set-up time, and the execution time of the job. Suppose the job does not terminate itself during the assigned maximum execution interval e_2. Excluding set-up time the transition interval will have duration $e_2 + s$, and the value of the state variable ℓ will be 2 at the end of the interval. Because there may be arrivals from any of the three other users, the value of n_1 may be 0 (no arrivals), 1, 2, or 3 at the end of the transition interval. Writing the state variable description as a vector (n_1, n_2, n_3, ℓ), the possible end states are (0, 0, 0, 2), (1, 0, 0, 2), (2, 0, 0, 2), and (3, 0, 0, 2) if the Q_1 job is not terminated during its execution. This accounts for the indices 5, 10, 22 and 44 listed in Table 5.1. If we suppose that the job does terminate during its first execution pass, ℓ will have zero

value at the end of the transition interval. Accounting for the possible number of arrivals, the possible end states are (0, 0, 0, 0), (1, 0, 0, 0), (2, 0, 0, 0), or (3, 0, 0, 0), whose indices are given in Table 5.1. The state (4, 0, 0, 0) is not possible, for this requires that a new command arrive for the user whose job just terminated. Although there may be a non-zero time interval between the termination time and the succeeding drum interrupt where the scheduling decision is made, it will be no greater than 50 ms and we assume that the arrival time does not commence until the end of this interval.

Now consider the state i = 1. Here the computer remains waiting until a command arrives. The scheduling action does not begin until the next drum interrupt after the arrival. Between the first arrival and the drum interrupt the other three consoles could each conceivably complete typing a command, and thus up to four arrivals could occur in the transition interval. The states (1, 0, 0, 0), (2, 0, 0, 0), (3, 0, 0, 0), and (4, 0, 0, 0) are possible end states.

The state (0, 0, 0, 0) is also listed as an end state for itself in Table 5.1. This is the case if we assume the ECP examines the queue at every drum interrupt when the system is empty. We may however assign zero probability to this transition in our solutions, and simply consider the next decision to occur after there has been at least one arrival. This excludes many trivial decisions, those having only one possible result, from the decision process.

It should also be noted that we have in effect defined an imbedded decision process. For with a suitable definition of how the state variable ℓ changes during swapping and set-up, we can consider (n_1, n_2, n_3, ℓ) as the state of the system at any time of system operation. Then an arrival produces an instantaneous change of state at the time of arrival. For the imbedded decision process this change is not noted until the drum interrupt at which the scheduling action occurs. Figure 5.7 shows a sample sequence of transitions for the system and the imbedded decision process, clarifying this relationship. In this figure, swapping takes two time steps.

5.2.3 Derivation of Formulae

In order to have complete freedom to explore whatever effects prove to be interesting in optimization, we must avoid a rigid assignment of values to the system parameters. Table 5.2 is a complete list of these parameters assuming the execution time to be modeled by the class satisfying (5.3). If we are to vary these parameters at will, a set of general formulae must be derived for the transition probabilities p_{ij}^k, the expected transition times ν_i^k, and the expected one-transition returns b_i^k. As seen from the Howard algorithm, only this data is required. This is fortunate, for one can sometimes determine ν_i^k and b_i^k by inspection, while it is tedious to ascertain $f_{ij}^k(\tau)$ and $r_{ij}^k(m|\tau)$ for every i, j, k, m, and τ.

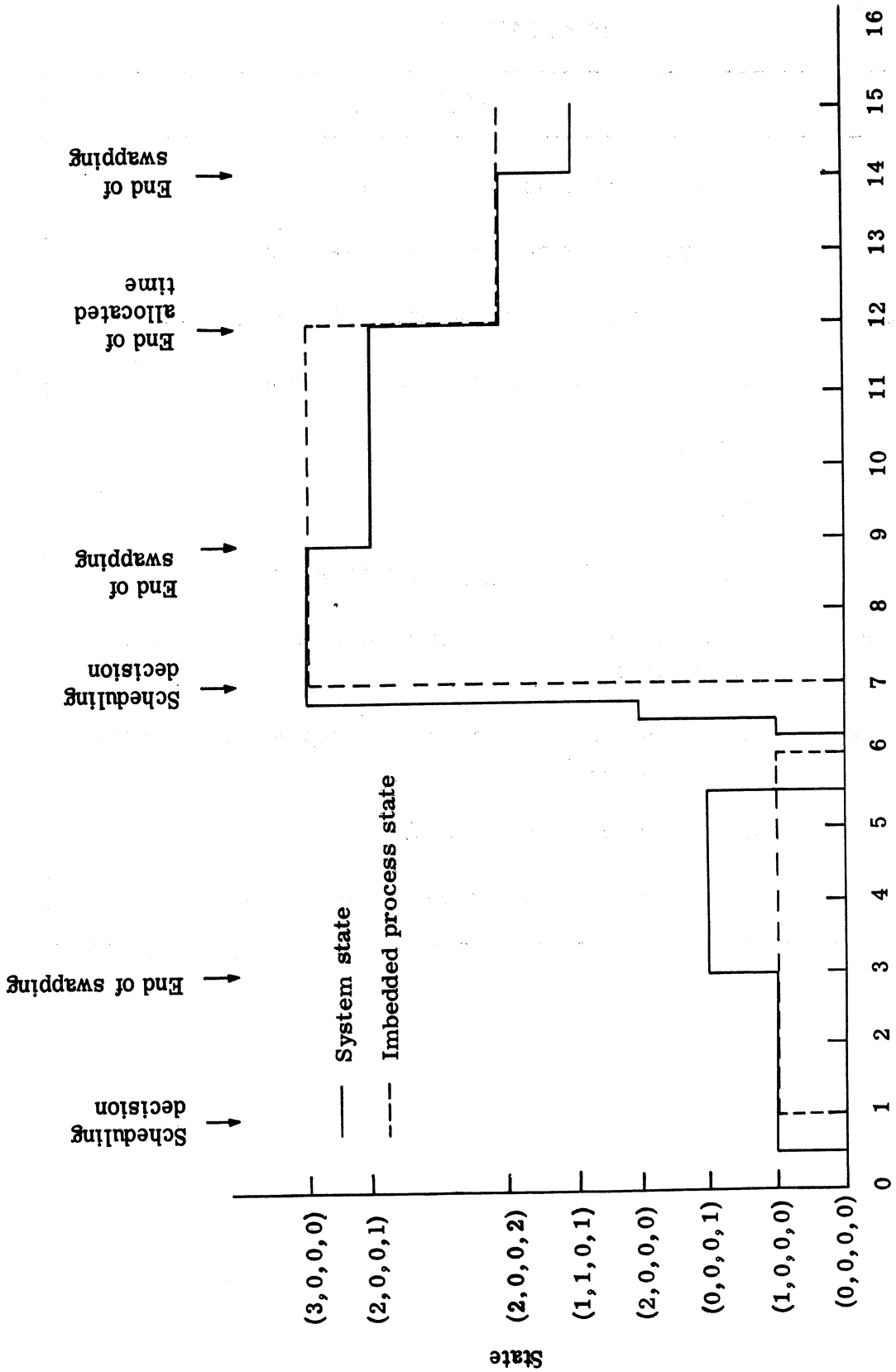


Fig. 5.7. Comparison of system transitions and transitions of the imbedded decision process.

TABLE 5.2

SYSTEM PARAMETERS

Symbol	Definition
λ	Reciprocal of mean arrival time
e_2	Accomplished execution time of Q_2 jobs
e_3	Minimum accomplished execution time of Q_3 jobs
q	Maximum execution interval for Q_3 jobs
s	Swap time
S	Set-up time
Θ	Probability that a Q_1 job requires set-up
$\sigma_1, \sigma_2, \sigma_3$	Probabilities in hyper-exponential execution time distribution
μ_1, μ_2, μ_3	Reciprocal mean values in hyper-exponential execution time distribution.
T_u	Mean user reaction time, $1/\lambda$
T_e	Mean execution time, $\sigma_1/\mu_1 + \sigma_2/\mu_2 + \sigma_3/\mu_3$

There is little to be gained by proceeding here to derive each of the many formulae required. Appendix C lists the results of the derivations. We will illustrate here the derivation of p_{ij}^k and v_i^k. The next section, where we introduce the optimization criteria of interest for multiple-access computers, will deal with the derivation of b_i^k.

Assume an arbitrary execution time distribution F(T) having a negative exponential tail as required in (5.1) and suppose the time scale T is normalized to the drum revolution time, 50 ms. Let

$$f_n = F(n) - F(n-1) \quad , \quad n = 1,2,3,... \tag{5.4}$$

This is the probability that a job terminates in the $n^{\underline{th}}$ step of its execution. Note that F(0) = 0. Assume that the parameters in Table 5.2 are also normalized to the drum revolution time, where appropriate.

We have already indicated that we set $p_{11}^1 = 0$. Consider p_{12}^1. For every transition from state 1 there must occur at least one arrival. Any transition from state 1 ends with the unit step in which the first arrival occurs. Then p_{12}^1 is the probability of only one arrival in a unit step, conditional on there being at least one arrival. The unconditional probability of only one arrival from four users is $4\, e^{-3\lambda}(1 - e^{-\lambda})$ and the probability of at least one arrival is $1 - e^{-4\lambda}$. Therefore

$$p_{12}^1 = 4\, e^{-3\lambda}(1 - e^{-\lambda})/(1 - e^{-4\lambda}) \tag{5.5}$$

Consider now the transitions from state 2. The probability that the transition interval has duration (n + s) and there is no set-up is $(1 - \theta)f_n$, for $n \leq e_2$. The probability that the duration is (n + s + S) and there is a set-up required is $\theta\, f_n$. The transition from state 2 to

state 1 requires no arrivals during the transition interval, and termination of the job. Thus

$$p_{21}^{1} = \sum_{n=1}^{e_2} (1 - \theta)\, e^{-3\lambda(n+s)} f_n + \sum_{n=1}^{e_2} \theta\, e^{-3\lambda(n+s+S)} f_n$$

or

$$p_{21}^{1} = e^{-3\lambda s}(1 - \theta + \theta\, e^{-3\lambda S}) \sum_{n=1}^{e_2} e^{-3\lambda n} f_n \qquad (5.6)$$

The transition from state 2 to state 5 requires that no arrivals occur, and the job remains uncompleted at the end of the allocated execution time. The transition interval in that event has duration $(e_2 + s)$ with probability $(1 - F(e_2))\,(1 - \theta)$. The duration is $(e_2 + s + S)$ with probability $\theta(1 - F(e_2))$. Then

$$p_{25}^{1} = (1 - \theta)\, e^{-3\lambda(e_2+s)}(1 - F(e_2)) + \theta\, e^{-3\lambda(e_2 + s + S)}(1 - F(e_2)) \qquad (5.7)$$

Now consider transitions from state i = 75. Because the system is fully occupied with a job from each console, no arrivals will occur. The only action possible is to serve a Q_3 job. Because we have assumed round-robin service, the job occupying the main memory must be swapped for the one at the head of Q_3. The exponential tail on F(T) for $T > e_3$ provides us with the memoryless property of the negative exponential distribution. So we need not know how long the next job has been in Q_3. The probability

that the transition interval has duration (n + s) is $f_{n+e_3}/(1 - F(e_3))$ for $n \leq q$. Note that this is the conditional distribution on the remaining execution time of the job. A transition from state 75 to state 38 occurs if the next job is terminated during the q units of execution which it will be allowed. Thus

$$p^3_{75,38} = \sum_{n=1}^{q} f_{n+e_3}/(1 - F(e_3)) \tag{5.8}$$

The formulae for the mean transition times are comparatively easy to derive. Consider state i = 1. Here we have a constant probability $(1 - e^{-4\lambda})$ that the transition interval ends with any unit step. The expected number of time steps in the transition is therefore

$$\nu^1_1 = 1/(1 - e^{-4\lambda}) \tag{5.9}$$

Next consider state 2. The transition interval includes swapping and set-up time with an expected value $(s + \Theta S)$. The execution interval has duration e_2 with probability $(1 - F(e_2))$, the probability that the job does not end during the assigned period. If it does end at or before e_2 the expected interval of execution is determined from f_n. Then

$$\nu^1_2 = s + \Theta S + \sum_{n=1}^{e_2} n\, f_n + e_2\,(1 - F(e_2)) \tag{5.10}$$

For state 75 the action k = 3 involves no set-up time and therefore

$$\begin{aligned} \nu^3_{75} = {} & s + \sum_{n=1}^{q} n\, f_n + e_3/(1 - F(e_3)) \\ & + q\,(1 - F(e_3 + q))/(1 - F(e_3)) \end{aligned} \tag{5.11}$$

With these illustrations as introduction, the reader can turn to Appendix C for the complete derivation of formulae.

5.3 OPTIMIZATION CRITERIA

The decision model which has been formulated now requires a quantitative statement of the appropriate optimization criteria for computer time-sharing. Although the model is well defined and rigorous within the limits of the assumptions used, the choice of an optimization criterion brings in more of what may be called intuition and personal judgment. In large measure this is because the performance of a multiple-access computer must be judged in terms of user reactions to the delays which they encounter in using the system. A large amount of experimentation and experience with time-shared machines will undoubtedly be needed before there is a precise and widely accepted standard of performance from a human stand point. At this early stage of development we are content to explore two criteria which seem to cover the first-order user reaction to delay. These will illustrate the flexibility of the formulation and provide depth to the study of the multiple-access computer.

5.3.1 Minimization of Response Time

The response time to a user command is the time interval between the reception of the complete command by the computer and the completion of typed output from the program to the user console. The user will normally wait for all output from a previous command before proceeding to develop a new request. Hence the response time may limit the rate

at which the user inputs commands and receives results.

The response time must always be as long as the execution time required for a command. Any added time results from conflicting user demands and can perhaps be reduced by proper queue control. The more closely response time approaches the execution time, the less chance there is for aggravating and noticeable delays to impede the user.

This suggests the definition of a return for any command which is equal to the negative of the response time for the command. Maximizing the return is then equivalent to minimizing the response time. But the computer system is intended to operate over a long period of time and so the system return should include the response time for all commands processed. A straight forward approach is to measure the system return by the negative sum of response times for all commands processed in an interval of time. Moreover, because we have a stochastic system, the optimization will be with respect to the expectation of the system return.

In the physical system we are modeling, response time is very close to the interval between the arrival and the termination of a job, i.e., the time-in-system of the job. Any difference would stem from the brief time to type out the characters generated just before the job termination, which we will neglect. These considerations therefore lead to the definition

$$V_T(i) = -\left[\begin{array}{l}\text{expectation of the total time-in-system of commands} \\ \text{arriving in } [0,T], \text{ conditional on the initial state } i\end{array}\right] \tag{5.12}$$

We will consider the infinite time optimization problem with this return function, and we will refer to this as "minimization of response time." The manner in which the one-transition returns b_i^k are to be determined to accomplish this will be treated after we introduce another optimization criterion.

5.3.2 Minimization of Weighted Response Time

The user can never receive a response time less than the execution time he has requested by his command. Thus users who request extensive computation must be content with longer response times. It seems reasonable then that such users will not be troubled as much by a given amount of delay as the user who has requested a trivial computation. An example of a trivial computation is the acceptance of a typed input line from a user constructing a computer program on-line. Here the user could be ready immediately to type the next line, and any significant delay would be an aggravation.

The previous criterion gives equal weight to the time-in-system for all jobs, regardless of their required execution time. Thus it does not recognize that users with longer execution time can tolerate proportionately more delay than users with trivial computations. One way to include this effect in the optimization criterion is to weight the time-in-system of any job by an amount which decreases as the required execution time increases.

Figure 5.8 illustrates a family of weighting curves which can serve for an exploratory study. We will not attempt to justify any of these as the most desirable or reasonable of all possible curves. They have several attributes however which one would expect for an appropriate curve. The curves are flat for small execution time, recognizing that users with trivial computation have equal need for rapid response. The weight falls smoothly as execution time increases, reflecting a greater user tolerance for delay. For large values of execution time, the minimum possible response time will be large enough to exceed any user's tolerance for delay. Thus the weight curve should become flat again to reflect an equal intolerance for delay.

The weight functions plotted in Fig. 5.8 have a convenient functional form $W(t_e)$ in terms of the execution time t_e.

$$W(t_e) = \alpha + (1 - \alpha) \sum_{m=0}^{k-1} \frac{(k\beta\, t_e)^m}{m!} e^{-k\beta t_e} \qquad (5.13)$$

The parameters α, β, k are given on the figure. With the weight value taken as unity for trivial computation times, the effect of weighting is to scale the response time of any job to an equivalent value for a trivial job. Table 5.3 gives the set of response time values equivalent to one second response for a trivial job according to curve 1 of Fig. 5.8.

It should be clear that minimization of response time is achieved if we merely set $W(t_e) = 1$ for all t_e. The assignment of the one-transition returns b_i^k is determined by the same general process, which we will now develop.

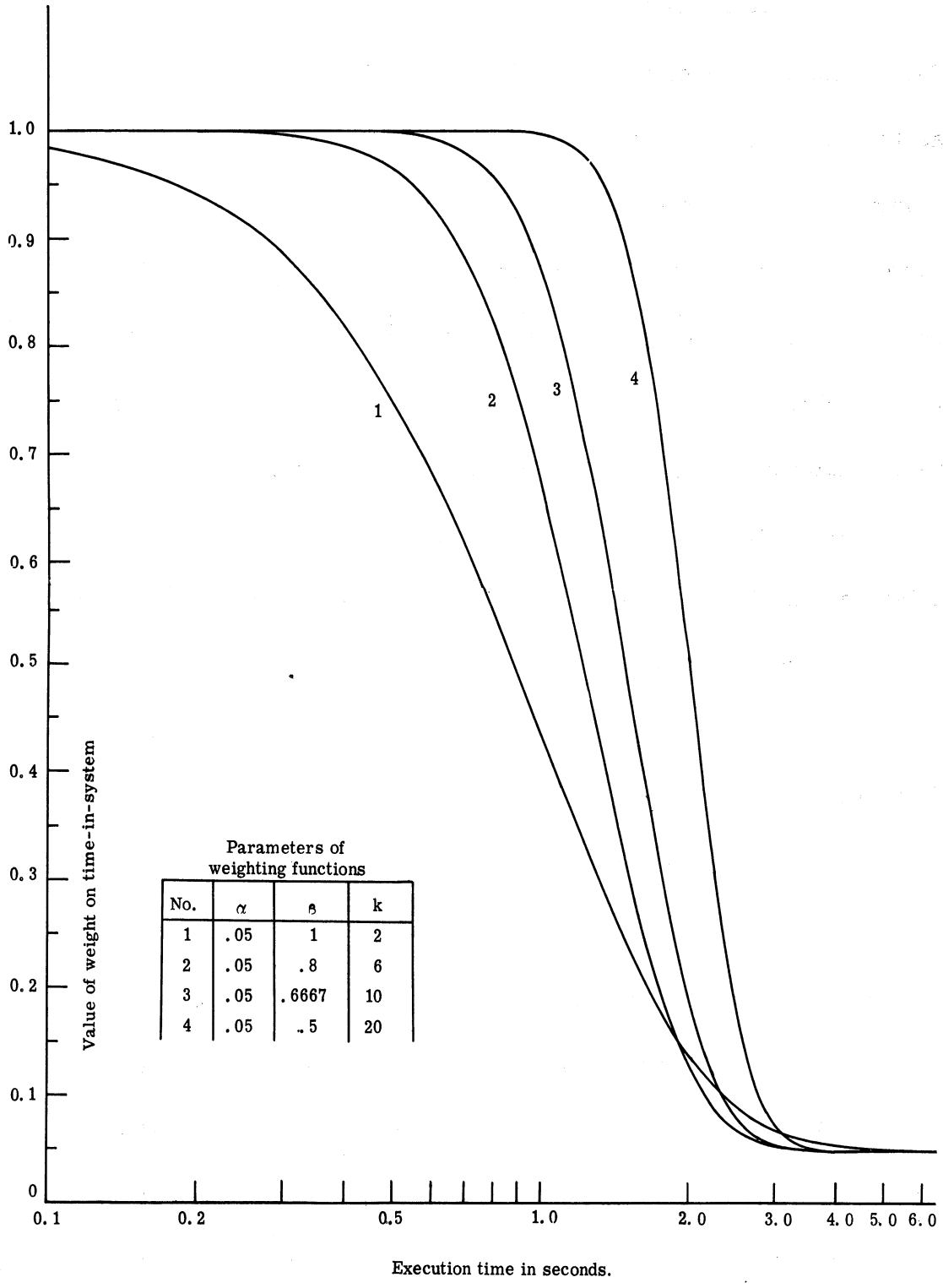


Fig. 5.8. Weighting functions for time-in-system.

TABLE 5.3

EQUAL WEIGHT RESPONSE TIMES FOR WEIGHT CURVE 1

Execution Time	Response Time
Trivial	1 sec.
0.5 secs.	1.34 secs.
1.0 secs.	2.30 secs.
2.0 secs.	7.30 secs.
3.0 secs.	15.0 secs.
20 secs.	20 secs.

5.3.3 Assignment of One-Transition Returns

We assume that we are given a function $W(t_e)$ according to which the time-in-system of a job requiring execution time t_e is to be weighted. The system return function is defined to be

$$V_T(i) = -\left\{\begin{array}{l}\text{expectation of the sum of the weighted time-in-}\\ \text{system of commands arriving in } [0,T], \text{ conditional}\\ \text{on initial state } i\end{array}\right\}$$

The Howard algorithm requires only the values of the one-transition returns b_i^k. It is convenient to decompose b_i^k into a sum of partial returns occurring over a transition interval,

$$b_i^k = \alpha_i^k + \xi_i^k + \delta_i^k - \gamma_i^k \nu_i^k \qquad (5.14)$$

where

α_i^k = the partial return attributable to new arrivals during the transition interval,

ξ_i^k = the partial return attributable to the execution time of the job executed during the transition interval,

δ_i^k = the partial return attributable to the swapping and set-up time of the job executed during the transition interval,

γ_i^k = the total weight per unit of time-in-system for jobs in system at the beginning of the transition interval, excluding the job placed into execution.

The evaluation of these terms will be easier than specification of the $r_{ij}^k(m|\tau)$ values. The term $\gamma_i^k \nu_i^k$ in (5.14) gives the expectation of weighted time-in-system over one transition for the jobs which will certainly remain in system throughout the transition. The job which

is executed during the interval may terminate and its associated return, consisting of $\xi_i^k + \delta_i^k$, is evaluated separately. The term α_i^k will have to account for the possible time and number of arrivals during the transition interval.

The execution time of a job is not observable until the instant of its termination. While the job is in the system prior to termination, each unit of time it spends must be weighted according to the expectation of $W(t_e)$ with respect to a conditional probability distribution on t_e. The condition involved is a given value of accomplished execution time. If the job is in Q_1 the appropriate weight per unit of time is

$$w_1 = \int_0^\infty W(t_e)\, d\, F(t_e) \tag{5.15}$$

Similarly the weight per unit of time for jobs in Q_2 and Q_3 are respectively

$$w_2 = \frac{\int_{e_2}^\infty W(t_e)\, d\, F(t_e)}{1 - F(e_2)} \tag{5.16}$$

and

$$w_3 = \frac{\int_{e_3}^\infty W(t_e)\, d\, F(t_e)}{1 - F(e_3)} \tag{5.17}$$

provided we require that e_3 be large enough that $W(t_e)$ is constant for $t_e \geq e_3$. This will be a restriction on the admissible values for e_3.

The weighting values w_1, w_2, and w_3 allow us to determine γ_i^k. For any state i in Table 5.1 described by the vector (n_1, n_2, n_3, ℓ) except $i = 1$,

$$\gamma_i^k = n_1 w_1 + n_2 w_2 + n_3 w_3 + w_\ell - w_k \qquad (5.18)$$

for permitted values of k only and with the understanding that $w_0 = 0$. For $i = 1$, $\gamma_1^1 = 0$.

The weight to be applied to the swapping and set-up time is also based upon the accomplished execution time of a job when it is again placed into execution. Because the set-up time may only occur when a job is first put into execution, and occurs independently of the required execution time

$$\delta_i^1 = -w_1(s + \theta S) \qquad (5.19)$$

for all i for which k = 1 is a permitted action, except i = 1. We have $\delta_1^1 = 0$. For $k \neq 1$ and i having $\ell = k$ and $n_k = 0$, $\delta_i^k = 0$; otherwise

$$\delta_i^k = -w_k s, \quad k = 2, 3 \qquad (5.20)$$

using only the values of k permitted in a given state i = 2,...N.

The return associated with the execution time of a job may be accumulated in either of two ways during system operation. On the one hand one can accumulate the return on each execution pass received at the time the execution pass is completed. On the other hand, one can accumulate the return associated with all the required execution time at the time the job is first executed. This accumulates all the return at once, ahead of the time actually received. The two ways are equally acceptable as long as we only consider infinite-time optimization. The latter is more convenient, resulting in

$$\xi_i^{\,1} = -\int_0^{\infty} t_e \, W(t_e) \, d\, F\,(t_e) \qquad (5.21)$$

for all i in which k = 1 is a permitted action, except i = 1. Again, $\xi_1^{\,1} = 0$, and moreover, $\xi_i^{\,2} = \xi_i^{\,3} = 0$ for all i in which k = 2 and k = 3 are permitted actions.

Evaluation of the term $\alpha_i^{\,k}$ requires some fairly extensive considerations. First let us define

X_i = number of jobs present in the system at the beginning of a transition from state i (5.22)

This does not depend upon the action k taken at the beginning of the transition for this model. The number of possible arrivals during a transition from state i is $(4-X_i)$, since each console may have only one job in system. Now $\alpha_i^{\,k}$ is composed of the sum of separate returns for each of the possible arrivals. But because each user generates a job independently of other users or the state of the system, the return for each of the possible arrivals is the same. Therefore we can express $\alpha_i^{\,k}$ as

$$\alpha_i^{\,k} = -w_1 \, (4 - X_i) \, T_i^{\,k} \qquad (5.23)$$

defining $T_i^{\,k}$ as

$T_i^{\,k}$ = expectation of the time interval between occurrence of an arrival from one user and the end of the transition interval.

The time interval is to be taken as zero if the arrival does not occur during the transition interval.

We will only illustrate the derivation of T_i^k, referring the reader to Appendix C for complete details. Consider the state i = 1. The transition interval ends at the first unit time step in which at least one arrival occurs. For T_1^1 we need the expectation of the interval between the occurrence of an arrival and the end of a unit time step, conditional on there being at least one arrival from four users in the unit step. This is

$$T_1^1 = \frac{1 - \int_0^1 \lambda\, t\, e^{-\lambda t}\, d\,t - e^{-\lambda}}{1 - e^{-4\lambda}}$$

$$= \frac{1}{1 - e^{-4\lambda}} - \frac{1}{\lambda}\left(\frac{1 - e^{-\lambda}}{1 - e^{-4\lambda}}\right) \qquad (5.24)$$

Now consider any other state and let

$$f_i^k(\tau) = \sum_{j=1}^{N} p_{ij}^k\, f_{ij}^k(\tau) \qquad \tau=1,2,\ldots \qquad (5.25)$$

be the unconditional probability distribution on the duration of the transition. Then

$$T_i^k = \sum_{\tau=1}^{\infty} f_i^k(\tau)\left[\tau - \int_0^{\tau} \lambda\, x\, e^{-\lambda x}\, dx - \tau e^{-\lambda\tau}\right]$$

because the arrival is independent of the transition duration. It is easily shown from this that

$$T_i^k = \nu_i^k - \frac{1}{\lambda}\sum_{\tau=1}^{\infty}(1 - e^{-\lambda\tau})\, f_i^k(\tau) \qquad (5.26)$$

The summation in (5.26) is simply the probability that the arrival occurs in the transition interval. It follows directly that for state i = 2

$$T_2^1 = \nu_2^1 - \frac{1}{\lambda}\Big\{1 - (1-\Theta)\, e^{-\lambda s} - \Theta\, e^{-\lambda(s+S)} - (1-\Theta)\, e^{-\lambda s}\sum_{n=1}^{e_2} e^{-\lambda n} f_n - \Theta\, e^{-\lambda(s+S)}\sum_{n=1}^{e_2} e^{-\lambda n} f_n - (1-\Theta)(1-F(e_2))e^{-\lambda(s+e_2)} - \Theta\,(1-F(e_2))e^{-\lambda(s+e_2+S)}\Big\}$$

Similar expressions are described in Appendix C for other states and actions.

5.4 METHOD OF SOLUTION

Finally we establish in this section some relations between the model and the theory of Chapters III and IV which provides the method of solution.

5.4.1 Application of the Principal Theorem

The only condition which must be investigated to ensure that the principal theorem, Theorem 4.2, is applicable is absolute convergence of the series

$$\sum_{m=1}^{\infty} u(m|i,k)$$

for every i and k.

In the discussion following Eq. (5.14), it is clear that the one-transition return b_i^k results from the time jobs spend in system over a transition

interval. According to our definition of return for either criterion, a negative return results whenever there is at least one job in the system. Otherwise the return is zero. Although an explicit expression has not been derived for $u(m|i,k)$, it is clearly zero or has negative value for all m, i, k. Because b_i^k exists, the series must be absolutely convergent.

For the infinite-time optimization problem it is therefore sufficient to treat only stationary deterministic (P') policies in seeking an optimal queue control policy.

5.4.2 Ergodic Policies

For the computer model we have developed there is but one recurrent aperiodic class of states under any P' policy. This is justified by study of Table 5.1, showing the possible transitions. It is possible with any action to pass in one transition from any state i, except i=1, to another state j such that $X_j < X_i$ (see Eq. (5.22)). Hence it is possible to eventually reach state 1 under any P' policy. Moreover state 1 communicates with state 2, and state 2 communicates with itself. Further, state 2 is aperiodic for state 2 can pass into itself in one transition with possible durations s, s+1,...,s+S+e_2 and these values have no common divisor greater than unity. Because state 2 is reachable from every other state, every other state is either contained in an aperiodic recurrent class including state 2, or is a transient state. By Theorem 4.3, section 4.3, $V_T(i)$ has a linear asymptotic behavior as $T \to \infty$, and

the gain g is the expectation of return in any unit step after a sufficiently long time of operation has elapsed.

5.4.3 Interpretations of the Gain g

The last statement above means that the system has a "steady state" or "equilibrium" behavior, as the term is used in queueing theory. We wish now to further explore the implications of this fact, to assist in interpreting the results in Chapter VI. The considerations to follow chiefly concern the physical interpretation of the value of g, as various one-transition returns are imposed on the model, including others besides those we have already described for the optimization criteria.

Consider first the minimum response time criterion, Section 5.3.1. Let $\{Y(t), t \geq 0\}$ be the continuous-time random process where Y(t) is the number of jobs present for service at time t. The system receives a return $-Y(t)\,\Delta t$ for the time increment $(t, T + \Delta t)$ for Δt arbitrarily small. The gain g is the expectation of the integral of -Y(t) over the unit step T for any value of T sufficiently large,

$$g = E\left\{-\int_{T-1}^{T} Y(t)\,dt\right\}, \quad T \to \infty$$

A more convenient interpretation of g can be obtained however.

Suppose we were to choose a single time point t sufficiently large to ensure steady state behavior. The chosen point falls somewhere within a unit step, and for that step the expected return is g. Define L as the expectation of Y(t),

$$L = E(Y(t)) \quad t \to \infty$$

The ergodic behavior which leads to Theorem 4.3 ensures that

$$g = -L. \tag{5.27}$$

A rigorous, albeit lengthy proof of this can be given, but we will consider this as intuitively clear.

Next suppose that one unit of return is accumulated upon termination of any job, i.e., when a job leaves the system. Then b_i^k is equal to the probability of termination of the job executed during a transition from state i with action k. The gain g in this case is the probability of a job termination in any unit time step in the steady state. We shall use the symbol Φ to denote this probability.

For future reference, we should point out that optimization with this definition of return maximizes the expectation of the number of jobs completed in a unit step, i.e., the rate of job completions.

For either of the optimization criteria we have proposed, the return of the decision process is the sum of the returns for all jobs processed by the computer system. In the formulation of the decision process the return for each job is viewed as being accumulated continuously with the passing of time spent in the system. In regard to the gain g of the process however, we can as well consider that all the return for a job is accumulated instantaneously upon its termination. Let us take this view, and let B denote the expectation of the total return for a single job terminated in any unit time step in the steady state. The

gain g, since it is the expectation of the return in any unit step, is in this case

$$g = \Phi B + (1 - \Phi)(0) = \Phi B \qquad (5.28)$$

This is evident inasmuch as Φ is the probability of a job termination, and no return is received from a job until it terminates.

Equation (5.28) has important consequences. For the minimum response time criterion, it means that[1]

$$L = \Phi T_R \qquad (5.29)$$

where

T_R = expectation of the time-in-system (response time) for a job

A better physical picture of the performance of an optimal policy can be had from T_R than from L, since it is response time that directly affects user reactions. However an even more informative way of describing response time performance under a given policy can be found using (5.28).

The notion that users requesting extensive computation can tolerate longer response time naturally suggests that one consider the expectation of response time conditional upon given execution time for a job. The question arises as to how this can be determined from the model. Suppose we could determine

W_i = expectation of time spent waiting in Q_i by a job which reaches Q_i level, including swapping and set-up time where applicable (i=1,2,3).

[1] This result is representative of the ergodic properties typically presented in classical queueing studies, see Saaty [8], p. 116, and Morse [7], p. 75.

Let $T_R(t_e)$ denote the conditional expectation of response time. Then

for $0 < t_e \leq e_2$:

$$T_R(t_e) = W_1 + t_e \tag{5.30}$$

for $e_2 < t_e \leq e_3$:

$$T_R(t_e) = W_1 + W_2 + t_e \tag{5.31}$$

We are unable to determine the conditional expectation of waiting time in Q_3 for any t_e in the range $t_e > e_3$. However W_3 is the waiting time on the less restrictive condition, $t_e > e_3$. From this we can determine the expectation of response time, conditional only on $t_e > e_3$, and this is

$$T_R(t_e \geq e_3) = W_1 + W_2 + W_3 + E(t_e | t_e > e_3) \tag{5.32}$$

With the required exponential tail on $F(T)$, the execution time distribution, Eq. (5.1) gives

$$E(t_e | t_e > e_3) = e_3 + \frac{1}{\mu}$$

where μ is the parameter describing the tail.

The values $W_i (i=1,2,3)$ are easily determined for a given policy by making three separate passes through the Value Determination Operation of the Howard algorithm. The required return on each pass associates unit value to every unit of time a job spends in Q_i, i=1,2,3 respectively. Swapping and set-up time are also included when applicable. Let g_i be the gain which is found for the decision process on these three runs (i=1,2,3). From (5.28) it follows that

$$g_1 = \Phi W_1$$

The portion of all jobs which reach Q_2 level is

$$p_2 = 1 - F(e_2) = P_r(t_e > e_2)$$

Therefore

$$g_2 = \Phi p_2 W_2$$

and similarly

$$g_3 = \Phi p_3 W_3$$

with

$$p_3 = 1 - F(e_3) = P_r(t_e > e_3)$$

Thus upon computing g_1, g_2, and g_3 the values W_1, and W_2, and W_3 are easily found. The conditional expectation of response time as determined using (5.30), (5.31), and (5.32), will play an important part in interpreting the results in Chapter VI.

5.4.4 Restrictions on Solutions

The solutions we will obtain will involve only the hyper-exponential distribution (5.3) for the execution time. The basic model will accommodate a wider class, however, subject only to the limitation (5.1). For the hyper-exponential distribution it is necessary to choose e_3 reasonably large in order to ensure that (5.1) accurately represents the behavior for $t_e > e_3$. An adequately large value is given by

$$e_3 \geq \frac{1}{\mu_2 - \mu_3} \log_e \frac{1000\ \sigma_2}{\sigma_3} \qquad (5.33)$$

There is a basic conflict in having to choose e_3 large, because many

time-shared systems use rather small time allocation (~ 0.5 sec) which we are unable to study due to this restriction. In order to permit smaller values of e_3, we have chosen to study three particular distributions shown in Fig. 5.9. One of these, number 1, is the exponential distribution for which any value of e_3 is permissible, as long as it is an integral number of 50 ms steps. The others are two-phase hyperexponential, obtained from (5.3) by simply letting $\mu_3 = \mu_2$. Then (5.33) is replaced by

$$e_3 \geq \frac{1}{\mu_1 - \mu_2} \log_e \frac{1000\ \sigma_1}{\sigma_2} \qquad (5.34)$$

The parameters shown on Fig. 5.9 are for a mean value $T_e = 1$ sec. We will have occasion to consider other values of T_e, obtained by changing μ_1 and μ_2.

We have already noted that e_3 must be chosen large enough that the weight function $W(t_e)$ is constant for $t_e \geq e_3$. With the weight curves of Fig. 5.8 this may not be less restrictive than the result of (5.34).

5.4.5 Admissible Policies

Because there are 42 states allowing two alternative actions, and 12 states allowing three alternative actions, the set P' consists of $2^{42} \cdot 3^{12}$ different policies. This number is well over a billion. Among the number there are six priority policies of special interest. One of these is the first-come, first-served policy which arises by assigning first priority to Queue 3, priority 2 to Queue 2, and priority 3 to Queue 1. In terms of the tabular presentation described in Fig. 2.4,

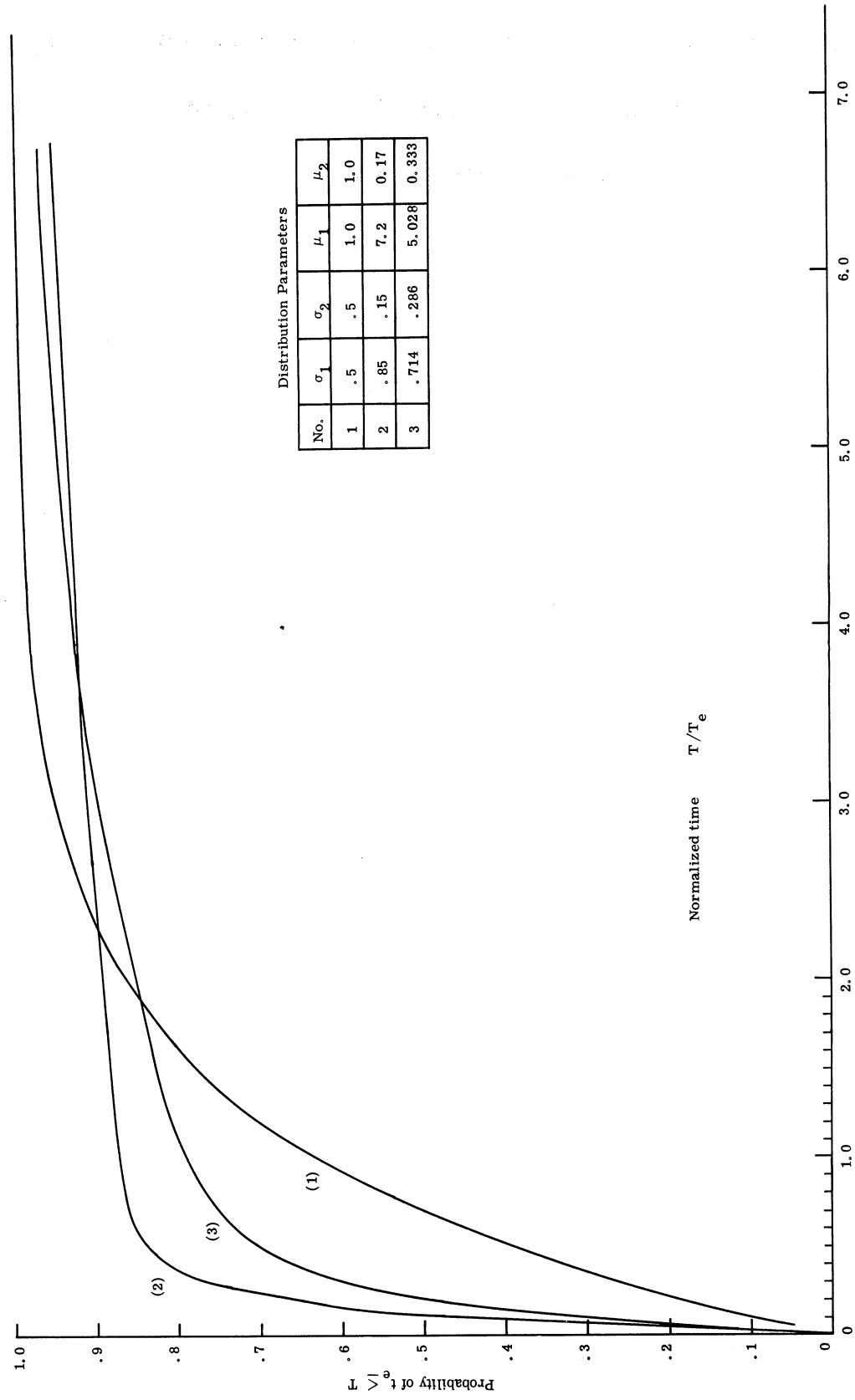


Fig. 5.9. Three distributions on execution time.

this policy is realized by choosing k=3 in every state having $n_3 > 0$ or $\ell = 3$, choosing k = 2 in any other state for which $n_2 > 0$ or $\ell = 2$, and k = 1 in the remaining states. The other priority policies are obtained by permuting the priority ordering among the three queues.

CHAPTER VI

OPTIMUM CONTROL OF QUEUES IN THE MULTIPLE-ACCESS COMPUTER

The formulation of the decision model for the multiple-access computer has required a substantial effort. This stems from the generality and realism which has been incorporated in the model. In compensation, a variety of results can be obtained from optimization solutions computed using the Howard algorithm. A collection of such numerical results is given in this chapter. These solutions do not exhaust the potentiality of the model for the study of the queue control function in multiple-access computers. They are sufficient however to provide better understanding of the proper design of queue control policies. Moreover they illustrate the versatility of the Semi-Markov decision process for the general study of such problems. We will first describe the various conclusions reached relative to the criteria introduced in Chapter V. Then we will comment on the application of the results to physical systems, and possible extensions of the model.

6.1 MINIMIZATION OF RESPONSE TIME

6.1.1 Negative Exponential Distribution

A natural starting point for the study of the computer system is to assume a negative exponential distribution on execution time, as shown in curve (1), Fig. 5.9. Under a first-come, first-served scheduling

policy,[1] the gain g of the system can be approximately determined from the solution of a classical continuous-time queueing model, namely the machine repair model [10]. This closed form analysis serves to check the accuracy of the formulae for the decision model. The comparison of the classical and the computed results also points out the degree to which the discrete-time model approximates the continuous-time model.

In order to carry out the comparison, an interpretation is needed for the gain g in terms of the performance measures obtainable from the classical analysis. As discussed in Chapter V, section 5.4, g = -L, with L = expected number of jobs in the system at any time point. The quantity L is easily obtained from formulae given in [10] for the continuous-time model.

Table 6.1 lists a number of situations in which the numerical computation for the discrete-time model has been compared to the closed form continuous-time solution.[2] The discrete-time model involves additional idle time for the server (i.e. the processor) above that of the continuous-time model. This is because a new service is not begun on the discrete-time system until the end of a unit step, even though a service completion may terminate at any time during the unit step. Hence it is expected that L will usually have a larger value for the discrete-time system for a given set of system parameter values. The close cor-

[1] See section 5.4.5.

[2] See Table 5.2, section 5.2.3 for parameter definitions.

TABLE 6.1

COMPARISON OF THEORETICAL GAIN OF CONTINUOUS-TIME MODEL AND COMPUTED GAIN OF DISCRETE-TIME MODEL: FIRST-COME, FIRST-SERVED POLICY; DISTRIBUTION 1[a]

Parameter Values					Values of -g = L	
T_u (sec.)	T_e (sec.)	S (sec.)	θ	S (sec.)	Continuous-time	Discrete-time
5	1	.15	0	-	1.11608	1.13093
5	1	.50	0	-	1.40244	1.41543
30	10	.15	0.6	0.5	1.66925	1.67088
30	1	.15	0	-	0.16255	0.16585
30	4	.15	0.7	1	0.76472	0.76763
30	5	.15	0.3	10	1.31688	1.31908
20	.05	.05	1.0	100	3.80020	3.79997

[a]See Figure 5.9.

respondence of values gives confidence that the correct formulae have been implemented in the computer algorithm, and that the discrete-time model is a very good approximation to the continuous-time model for the first-come, first-served policy.

The next question is, what is the optimal policy for minimizing the response time? A fairly good heuristic argument can be advanced that the optimal policy is first-come, first-served for any set of system parameter values with a negative exponential distribution on execution time. The argument begins with the intuitive notion that in order to minimize the expected response time the system should maximize the average rate of job completions. This means that the control policy should always select for execution the job having the smallest expectation of remaining service time, i.e. the sum of remaining execution time, swapping, and set-up time. But a characteristic of the negative exponential distribution, as seen in section 1.3.1, is that the expectation of the remaining execution time is the same regardless of how much execution a job has already had. Then because swapping and set-up time are added to determine the service time of a new arrival, the remaining service time of a job being executed always has smaller expectation than that of any arriving job. So once started, a job should be run to termination, and the first-come, first-served policy is optimal.

A substantial proof of this, stemming from a closed form analysis, can be given for the case of a single server priority queueing system

with an infinite source of Poisson arrivals, see [3,52]. But this proof cannot be extended to the case at hand. The decision model however provides the means to rigorously establish this for a given set of parameter values. By investigating a large number of cases, a firm basis can be established for claiming it as generally true. This has in fact been done for the cases listed in Table 6.1. In all of these the computed solution of the decision model revealed that the first-come, first-served policy was an optimal policy for scheduling. Hence we venture the

Conclusion: For any values of system parameters with a negative exponential distribution on execution time, a first-come, first-served policy is optimal for minimizing the expected response time to any command.

6.1.2 Non-Exponential Distributions

The optimal policy for an exponential distribution does not depend on the assigned values of e_2, e_3, and q. Such is definitely not the case for any other distribution. There is in fact a two-fold optimization problem involved in determining the optimum values of these parameters and the optimum policy associated with this assignment. We can only accomplish the first part of this dual problem by a trial and error search among the possible values for e_2, e_3, and q. Thus a number of optimization solutions will be needed.

a. To begin, let us ignore the influence of set-up time by setting $\theta = 0$. The intuition used for an exponential distribution still seems valid, suggesting that one should always be executing the job having

smallest expectation of remaining service time. With distributions such as (2) and (3) of Fig. 5.9, the expected remaining execution time increases as a job is executed. Figure 6.1 illustrates this increase. Hence it appears that the optimal policy will give preference to execution of a Q_1 job, i.e. a new arrival, before execution of a Q_2 or a Q_3 job. This statement assumes that swapping time is reasonably small compared to mean execution time, as it will be in the system being considered here for, say, $T_e > 0.5$ seconds.

This intuition is in fact borne out by solutions which have been obtained for distributions (2) and (3) of Fig. 5.9. Tables 6.2 to 6.4 list the pertinent data obtained from some of the optimization solutions. In all cases the optimal policy has a priority structure. Whenever a scheduling decision occurs, the selection of a queue from which the next job is taken is according to the priority ranking. Priority 1 indicates first preference, and we use for example the notation "1-3-2 PR" to describe a policy in which levels Q_1, Q_2, Q_3 are ranked 1, 3, and 2 respectively. As expected, the Q_1 level is preferred in the optimal policies obtained.

To whatever extent it is possible, it is desirable to extract from the data a general concept governing the optimal time allocation and the optimal policy. Inasmuch as each solution is valid only for its particular parameter values, this cannot be accomplished for every conceivable set of parameter values. But a number of solutions, as given in Tables 6.2 to 6.4, should be sufficient to achieve this for parameter values

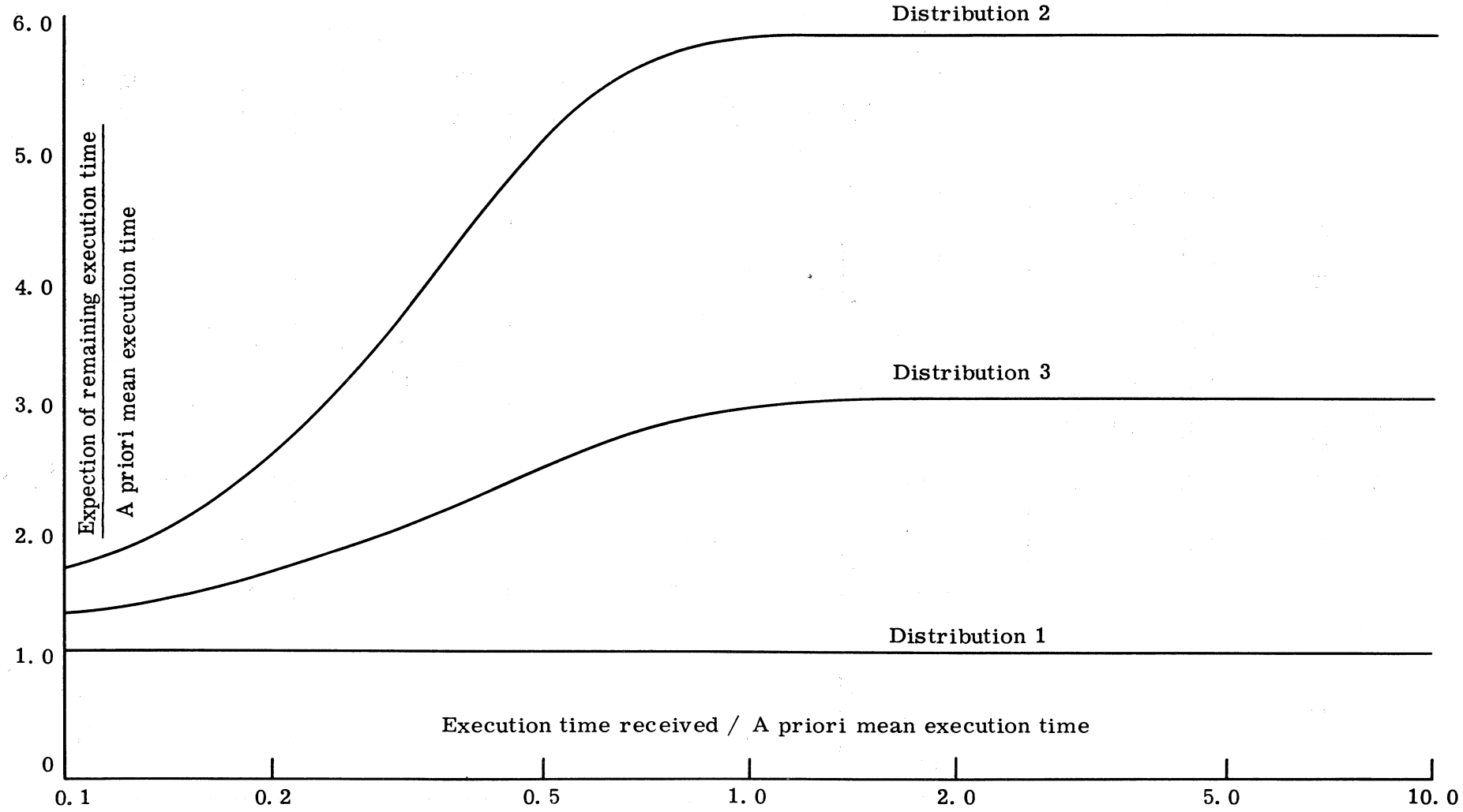


Fig. 6.1. Increase of remaining execution time with execution time received.

TABLE 6.2

MINIMIZATION OF RESPONSE TIME WITH DISTRIBUTION 2

Fixed Parameter Values				
T_u (sec.)	T_e (sec.)	s (sec.)	θ	S (sec.)
30	4	0.15	0	-

Variable Parameters			Optimal Value of - g	Optimal Policy
e_2 (sec.)	e_3 (sec.)	q (sec.)		
0.6	6.0	0.05	.81307	1-3-2 PR
2.0	6.0	0.05	.68447	1-3-2 PR
3.5	6.0	0.05	.66912	1-3-2 PR
4.0	6.0	0.05	.66713	1-3-2 PR
4.25	6.0	0.05	.66902	1-3-2 PR
3.5	6.0	0.15	.66994	1-3-2 PR
4.0	6.0	0.15	.66969	1-3-2 PR
4.25	6.0	0.15	.66985	1-3-2 PR
2.0	6.0	0.50	.68782	1-3-2 PR
4.0	6.0	0.50	.67254	1-3-2 PR
4.25	7.5	0.05	.67254	1-3-2 PR

TABLE 6.3

MINIMIZATION OF RESPONSE TIME WITH DISTRIBUTION 2

Fixed Parameter Values				
T_u (sec.)	T_e (sec.)	s (sec.)	θ	S (sec.)
20	1	0.15	0	-

Variable Parameters			Optimal Value of -g	Optimal Policy
e_2 (sec.)	e_3 (sec.)	q (sec.)		
0.75	1.50	.05	.26065	1-3-2 PR
1.0	1.50	.05	.26032	1-3-2 PR
1.25	1.50	.05	.26065	1-3-2 PR
1.0	1.50	.15	.26098	1-3-2 PR
1.0	2.50	.05	.26243	1-3-2 PR

TABLE 6.4

MINIMIZATION OF RESPONSE TIME WITH DISTRIBUTION 3

Fixed Parameter Values				
T_u (sec)	T_e (sec)	s (sec)	θ	S (sec)
30	4	0.15	0	-

Variable Parameters			Optimal Value of -g	Optimal Policy
e_2 (sec)	e_3 (sec)	q (sec)		
3.75	6.0	.05	.66952	1-3-2 PR
4.0	6.0	.05	.66928	1-3-2 PR
4.25	6.0	.05	.66930	1-3-2 PR
4.0	6.0	.15	.66992	1-3-2 PR
4.25	7.5	.05	.67551	1-3-2 PR

which are physically appropriate for contemporary time-shared computers. In Tables 6.2 to 6.4, two appropriate values of the mean user reaction time, T_u, and the mean execution time, T_e, are represented.[1] The two distributions, as seen from Figs. 5.9 and 5.5, seem appropriate also.

For all cases given in Tables 6.2 to 6.4, the optimal policy is 1-3-2 priority. In order to consider the time allocation, let us observe the values of the gain g given. The minimum value of e_3 permitted in the model for either distribution is 1.5 T_e. With this value of e_3, it appears that g is maximum when $e_2 = T_e$ and q is as small as permitted, namely one time step or one clock interval. The gain is apparently more negative for larger values of q. Moreover if e_3 is increased, the gain also decreases. This suggests that the optimum time allocation is

e_2 = T_e,

e_3 = minimum permitted value for the model,

q = minimum value permitted physically, i.e. one clock interval.

There is ample intuitive justification for this result. We see that with a 1-3-2 priority policy there is never more than one job at the Q_3 level. Hence the above choice of q does not involve swapping between Q_3 jobs. Rather it allows the system to rapidly preempt a Q_3 job to begin execution of a new arrival at the Q_1 level. This is consistent with always serving the job of smallest expected remaining service time. A Q_3

[1] The reasonableness of these values is verified by the first extensive data to come out of MIT's Project MAC, contained in the Ph.D. thesis by Allan Scherr, "An Analysis of Time-Shared Computer Systems", June, 1965.

job therefore continues to receive execution as long as there are no Q_1 jobs present. This is expected based on Fig. 6.1. There it is seen that for either distribution the expected remaining execution time is essentially constant after a job has received execution time totaling T_e. This means that jobs which have received T_e should thereafter be processed one at a time to completion assuming no jobs present which have received execution time less than T_e. Indeed, the 1-3-2 priority policy and optimal time allocation accomplish this precisely.

We may further conjecture that it would be beneficial to preempt execution of a Q_2 job to begin execution of a new arival. This would lead to a queueing structure with effectively only two priority levels. A lower level job would be preempted by an arrival, and the new arrival executed for a maximum time T_e before being placed in the lower queue. This corresponds to the time allocation

$$e_2 = T_e,$$

$$e_3 = T_e + \text{one clock interval},$$

$$q = \text{one clock interval}.$$

Such an assignment is not permitted however if our model using the two-phase hyperexponential distribution is to retain the Markov property, see Eq. (5.34). Hence we are not able to study the system with $e_3 < 1.5\ T_e$ for this distribution. There is some reason to believe that a further improvement in gain could be obtained were this not the case.

b. The assumption $\theta = 0$ is valid for a physical system in which command programs are not relocatable in memory, or where the central

processor hardware can carry out dynamic relocation [28,29]. The former is not often the case and systems capable of the latter include embellishments not present in our model. So we now consider the effect of the swap time s and the set-up time S, aiming to discover how these values would alter the result found above. A given value of set-up time will have its most prominent effect when $\theta = 1$, so let us assume this value. Then every command requires set-up before commencing execution. The set-up therefore increases the expectation of remaining service time for new arrivals. We would anticipate that the optimal policy remains the same (1-3-2 priority) for small S and changes as S approaches the expected remaining execution time for jobs drawn from the exponential tail of F(T). Table 6.5 gives a set of solutions which verify this supposition.

A value of S in the neighborhood of 1 second or less seems appropriate for the computer system we are modeling. There is little point then in considering T_e greater than 1 second, for the optimum system will not be different than the case for $\theta = 0$. Table 6.5 shows that even with S = 1 second, T_e = 1 second, the optimum system remains basically as before.

Table 6.5 however shows that the optimal policy undergoes a change for S greater than 1 second. For S = 5 seconds, a first-come, first-served policy (denoted FCFS) is optimal, and hence the time allocation has no effect on the system gain. (The last entry in the table illus-

TABLE 6.5

MINIMIZATION OF RESPONSE TIME: DISTRIBUTION 2

Fixed Parameters				
e_3 (sec)	T_u (sec)	T_e (sec)	s (sec)	θ
1.5	20	1	0.15	1.0

Variable Parameters			Optimal Value of -g	Optimal Policy
S (sec)	e_2 (sec)	q (sec)		
1	.50	.05	.48932	1-3-2 PR
1	1.0	.05	.48526	1-3-2 PR
1	1.45	.05	.48628	1-3-2 PR
1	.50	.50	.49113	1-3-2 PR
3	.85	.05	.96092	1-3-2 PR except k = 3 in state (1,2,0,3)
3	1.25	.05	.96043	1-3-2 PR except k = 3 in state (1,2,0,3)
3	1.45	.05	.96052	1-3-2 PR except k = 3 in state (1,2,0,3)
3	1.45	.50	.96054	1-3-2 PR except k = 3 in state (1,2,0,3)
3	1.45	1.50	.96149	1-3-2 PR
5	.85	.05	1.40186	FCFS
5	1.45	.05	1.40186	FCFS
5	1.0	2.5	1.40234	FCFS

trates the extent of round-off errors in the SEMSIN[1] computation. The value of gain should theoretically be identical for every time allocation.)

Table 6.5 also illustrates a phenomenon requiring explanation. The optimal policy found for S = 3 seconds has the optimal control action depending upon the number of jobs in the queues rather than on simply the presence of jobs. Thus k = 3 is the optimal action for state (1,2,0,3) but not say for state (2,1,0,3). A principal advantage of a Semi-Markov decision model is that it allows one to discover when an optimal queueing system has this structure. However in this case we can offer no intuitive explanation for what the optimization procedure has claimed to be an "optimal" policy. The possibility arises that computational errors have in fact led to a sub-optimal policy. Investigating further with Eq. (4.20) in mind, and denoting the "optimal" policy by A, we have found that

$$\max_{\pi_B \in P'} \max_{i \in R(A,B)} [\delta_i^B - \delta_i^A] = -.002$$

at termination of the iteration, for any set of parameters given in Table 6.5 with S = 3 second. Moreover, the computation for the "optimal" policy gives

$$\max_i |x_i^A| = 3.5 \times 10^3,$$

and so if we assume that

$$|\epsilon_i^A| < 10^{-5} |x_i^A|$$

[1]See Appendix D.

we can use Eq. (4.20) to estimate the gain g^B of any other policy B. With the minimum mean transition time equal to 4 time steps, (4.20) yields

$$g^B - g^A < - .002\gamma + 3(.035)/4 = .028 - .002\gamma$$

It is quite conceivable then that the "optimal" policy is in fact sub-optimal, and this is the explanation we propose. We presume that the optimal policy is 1-3-2 priority, as before.

The swapping time s will have an identical effect upon the optimal policy as the set-up time S. For small s, the optimal policy is 1-3-2 priority. As s approaches the expected remaining execution time on the tail of F(T), it brings about a significant extension of the expected service time for a Q_1 job in comparison to the expected remaining time for a job being executed. Hence the optimal policy should eventually change to FCFS, as occurs for large S. Such large values of s are not appropriate however for a drum secondary memory. We are led finally to this conclusion:

Conclusion: For execution time distributions F(T) shown as (2) and (3) of Fig. 5.9, and values of swap time s and set-up time S which are no longer than the mean execution time T_e, an optimal scheduling policy for minimizing the mean response time to any command is described by:

a. A 1-3-2 priority ranking for selecting jobs from queues 1, 2, and 3 respectively.

b. A maximum execution time allocation of T_e, for a job from Q_1; 0.5 T_e, the minimum permitted value in the model, for a job from Q_2;

one clock interval, for a job from Q_3.

6.1.3 Equivalent Criterion

The intuitive explanations put forth in the preceding paragraphs to justify the result of the computational procedure have relied on the idea that minimization of mean response time is equivalent to maximization of the mean rate of job completions. No rigorous proof of this was given and indeed it may be extremely difficult to do so. The computation itself however can provide additional confidence in the validity of this notion. For the model allows us to define b_i^k so as to collect a unit of return at completion of every job. Maximizing g for this definition of return maximizes the mean rate of completions. By using the optimization algorithm one can show that the optimal policy and the optimum time allocation are the same as previously obtained. This has been done, and Table 6.6 gives the value Φ which is the mean rate of job completions, or equivalently, the probability of a job completion in any unit step.

TABLE 6.6

VALUES OF Φ, MEAN RATE OF JOB COMPLETIONS
FOR THE OPTIMAL SYSTEM

T_u (sec)	T_e (sec)	Dist.	s (sec)	θ	S (sec)	Φ	Optimal Policy
20	4	2	0.15	0	-	.007411	1-3-2 PR
20	1	2	0.15	0	-	.009339	1-3-2 PR
20	1	2	0.15	1	1	.008777	1-3-2 PR

6.1.4 Comparison of Time-Sharing Policies

The discussion of section 5.4, Chapter V, has indicated how we may proceed to compare the optimal scheduling control policy we have found to the time-sharing policies now being used in actual systems. The comparison is in terms of the conditional expectation of response time to a command, given the required execution time. This can be accomplished for all execution time values t_e in the range $0 < t_e \leq e_3$. We can also compare the expectation of response time, on the condition that $t_e > e_3$.

There are two basic policies now in use to which we wish to compare the optimal system. First is the round-robin procedure. This corresponds in our model to a 2-1-3 priority policy with e_2 having any value less than e_3, and with $q = e_3$. (In contrast to the round-robin procedure discussed in Chapter I, the above places new arrivals at the head of the round-robin queue.) Because of the restrictions of the model we can only study "quantum" values of at least $1.5\ T_e$.

The second basic policy is a 1-2-3 priority policy, corresponding basically to the multi-level priority procedure as used at Project MAC [18] and Lincoln Lab [21]. The time allocation espoused by the developers of these systems involves increasing the maximum execution time allocation as the priority ranking of a queue decreases. This is distinctly different from what we have found. A time allocation of

$$e_2 = 0.5\ T_e$$

$$e_3 = 1.5\ T_e$$

$$q = 2\ T_e$$

will provide execution intervals increasing by a factor of 2 with each decreasing level, as used at MAC. This will be called priority policy A. But we will also investigate the allocation

$$e_2 = 0.5\ T_e$$

$$e_3 = 1.5\ T_e$$

$$q = e_2$$

referred to as priority policy B. This time assignment allows a Q_1 job to supercede a Q_3 job only after the latter has received as much execution time as the former may take. This notion has also been advanced by time-sharing pioneers as preventing the preemption by high priority jobs from interfering too much with the advance of lower priority jobs.

Figures 6.2 and 6.3 give the conditional response time for these three scheduling control systems and the optimal system we have found. A noteworthy observation is that the optimal system gives faster response to the "trivial computation", i.e. $t_e \to 0^+$, than any of the others. By the same token, the expected response time for very long computations ($t_e > e_3$) is larger for the optimal system except in comparison to priority policy B. In every case, a user requesting a long computation can look forward to a significantly long response time.

6.2 MINIMUM WEIGHTED RESPONSE TIME

The intuitive reasoning which justifies the optimal policy computed for the preceding criterion should hold even more strongly when weighting is applied to response time, as shown in Fig. 5.8. Because jobs taking

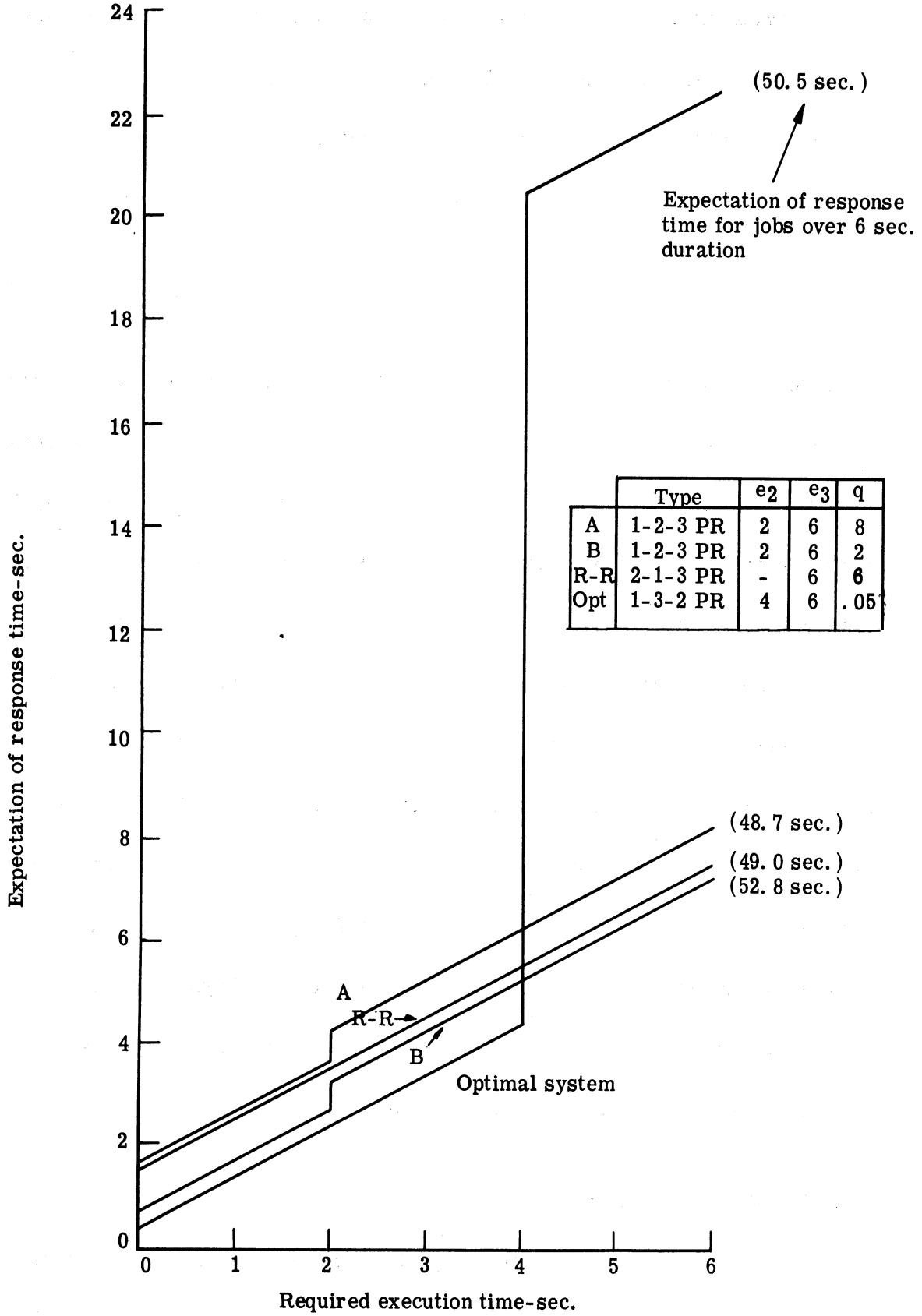


Fig. 6.2. Conditional expectation of response time for distribution 2, T_u = 20 sec, T_e = 4 sec, θ = 0, s = 0.15 sec.

Control Policy Data

	Type	e_2	e_3	q
A	1-2-3 PR	0.5	1.5	2.0
B	1-2-3 PR	0.5	1.5	0.5
R-R	2-1-3 PR	-	1.5	1.5
Opt	1-3-2 PR	1.0	1.5	.05

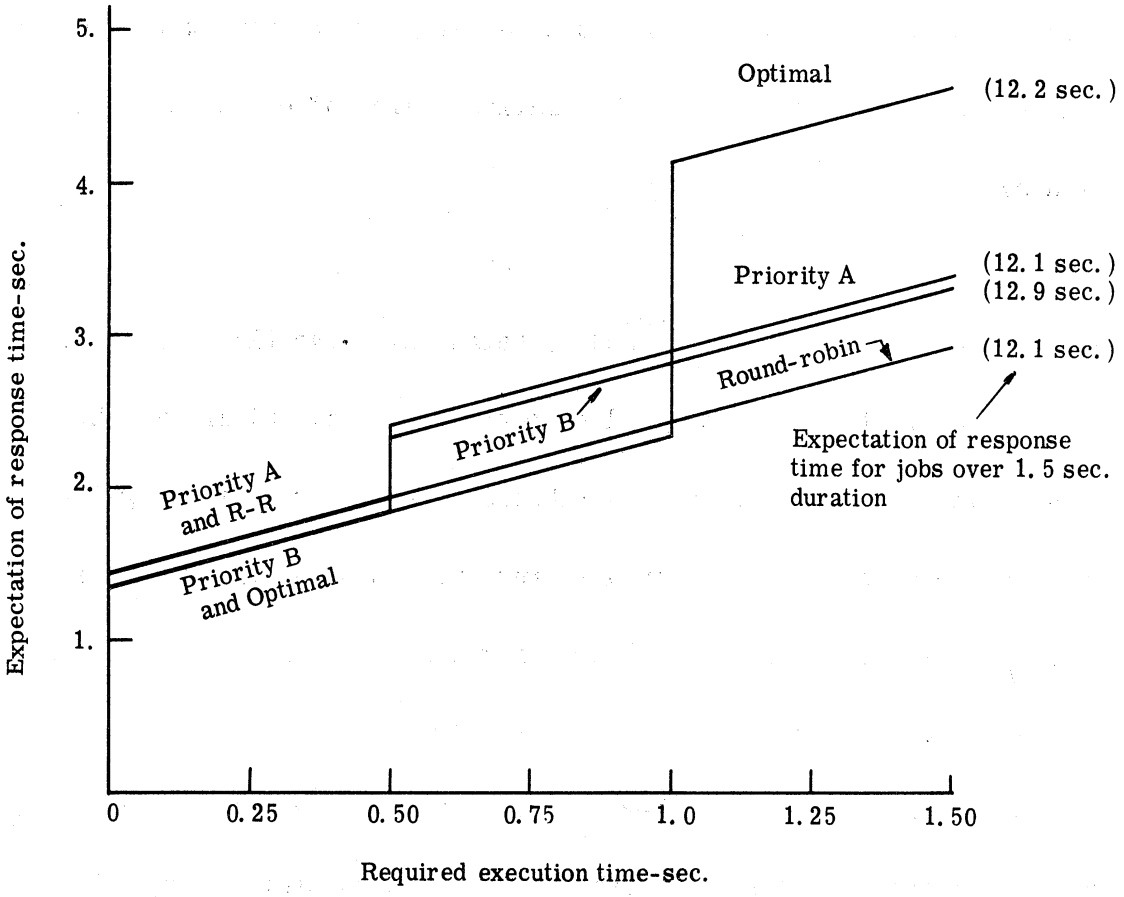


Fig. 6.3. Conditional expectation of response time for distribution 2, T_u = 20 sec, T_e = 1 sec, θ = 1, S = 1 sec, s = 0.15 sec.

longer to execute receive less weight, it is even more imperative to execute shorter jobs whenever possible. One expects that this may lead to a smaller value of e_2 than found in the previous case.

The value of e_3 should again be the smallest value permitted by the model. With this criterion however the minimum permitted value is determined by both the weighting curve and the hyperexponential execution time distributions. By contrast, only the latter was pertinent with the minimum response time criterion. The smallest value of e_3 to be studied is given by

$$e_3 = \max (4 \text{ seconds}, 1.5\ T_e)$$

The value 4 seconds is established by weighting curve (1) of Fig. 5.8.

Tables 6.7 to 6.10 present data pertinent to minimizing weighted response time. Among the three distributions under study, the negative exponential produced a distinctly different result under the previous criterion. This is not so here, for similar results hold for distributions (1) and (2) and can be expected to hold also for (3).

The optimal policy is 1-3-2 priority in almost all cases, as it was under the previous criterion. Table 6.7 indicates that for T_e = 1 sec, the optimal value of e_2 is slightly larger than T_e, where as for T_e = 4 sec, the converse is true. This is partially, if not completely, explained by the fact that in the first instance e_3 is considerably larger compared to T_e than in the second instance. The optimum value of q is one clock interval, as before, in all but one of the interesting cases.

TABLE 6.7

MINIMIZATION OF WEIGHTED RESPONSE TIME: WEIGHTING CURVE 1, DISTRIBUTION 2

Fixed Parameters		
T_u (sec)	s (sec)	θ
30	0.15	0

Variable Parameters				Optimal Value of -g	Optimal Policy
T_e (sec)	e_2 (sec)	e_3 (sec)	q (sec)		
1	1	4	.05	.04656	1-3-2 PR
1	1.5	4	.05	.04585	1-3-2 PR
1	2.5	4	.05	.04616	1-3-2 PR
1	1.5	4	.15	.04609	1-3-2 PR
1	1.5	4	.50	.04683	1-2-3 PR
4	2	6	.05	.08062	1-3-2 PR
4	3	6	.05	.07822	1-3-2 PR
4	4	6	.05	.07885	1-3-2 PR
4	3	6	.15	.07909	1-3-2 PR
4	3	6	.50	.08208	1-3-2 PR

TABLE 6.8

MINIMIZATION OF WEIGHTED RESPONSE TIME: DISTRIBUTION 2

Fixed Parameters				
T_u (sec)	T_e (sec)	s (sec)	e_3 (sec)	θ
20	4	0.15	6	0

Variable Parameters		Optimal Value of -g	Optimal Policy
e_2 (sec)	q (sec)		
Weight Curve 2			
2	.05	.13690	1-3-2 PR
3	.05	.13314	1-3-2 PR
4	.05	.13439	1-3-2 PR
3	.15	.13502	1-3-2 PR
Weight Curve 4			
2.5	.05	.15751	1-3-2 PR
3	.05	.15572	1-3-2 PR
4	.05	.15710	1-3-2 PR
3	.15	.15784	1-3-2 PR
3	.50	.16518	1-3-2 PR

TABLE 6.9

MINIMIZATION OF WEIGHTED RESPONSE TIME: DISTRIBUTION 1

Fixed Parameters		
T_e (sec)	s (sec)	θ
4	0.15	0

Variable Parameters				Optimal Value of -g	Optimal Policy
T_u (sec)	e_2 (sec)	e_3 (sec)	q (sec)		
Weight Curve 1					
20	2	6	.05	.09837	1-3-2 PR
20	3	6	.05	.09639	1-3-2 PR
20	4	6	.05	.09899	1-3-2 PR
20	3	6	.15	.09657	1-3-2 PR
20	3	6	.50	.09701	1-2-3 PR except k = 3 in (0,1,0,3) and (0,2,0,3)
20	1.5	4	.05	.09613	1-3-2 PR
20	2.5	4	.05	.09024	1-3-2 PR
20	3.75	4	.05	.09453	1-3-2 PR
20	2.5	4	.15	.09053	1-3-2 PR
20	2.5	4	.50	.09155	1-3-2 PR
30	2	6	.05	.06322	1-3-2 PR
30	3	6	.05	.06226	1-3-2 PR
30	4	6	.05	.06364	1-3-2 PR
30	3	6	.15	.06236	1-3-2 PR
30	3	6	.50	.06268	1-3-2 PR
Weight Curve 4					
20	1.5	6	.05	.15919	1-3-2 PR
20	3.0	6	.05	.14913	1-3-2 PR
20	3.5	6	.05	.15132	1-3-2 PR
20	3	6	.15	.14876	1-2-3 PR
20	3	6	.50	.14944	1-2-3 PR

TABLE 6.10

MINIMIZATION OF WEIGHTED RESPONSE TIME: WEIGHTING CURVE 1, DISTRIBUTION 2

Fixed Parameters				
T_u (sec)	T_e (sec)	e_3 (sec)	s (sec)	θ
30	1	4	0.15	1.0

Variable Parameters			Optimal Value of -g	Optimal Policy
S (sec)	e_2 (sec)	q (sec)		
1	0.85	.05	.15924	1-2-3 PR
1	1.0	.05	.15913	1-2-3 PR
1	1.5	.05	.15909	1-3-2 PR
1	2.0	.05	.15949	1-3-2 PR
1	1.5	.15	.15919	1-2-3 PR
1	1.0	.50	.15990	1-2-3 PR
3	0.85	.05	.38763	1-2-3 PR except k = 3 in (0,0,3,2)
3	1.0	.05	.38768	1-2-3 PR except k = 3 in (0,0,3,2) and (0,0,2,2)
3	1.5	.05	.38856	1-2-3 PR except k = 3 in (0,0,3,2), (0,0,2,2), (0,0,1,2) and (0,1,1,0)
3	1.0	.50	.39147	1-2-3 PR

Tables 6.9 and 6.10 exhibit again the kind of behavior recorded in Table 6.5. That is, for certain parameter values the optimal policy is in process of changing from one priority assignment to another, and the algorithm converges on the suboptimal policy which represents a mix of the control actions from each. This is especially clear in Table 6.10 for S = 3 sec. As e_2 increases, the exceptional states become more numerous. One expects that for larger e_2, the algorithm will produce 1-3-2 priority as optimal. Interestingly, it appears that for S = 3 sec the optimal policy is 1-2-3 PR rather than FCFS as found under the previous criterion. It is expected however that as S becomes much larger the increased service time for Q_1 jobs will offset the greater weighting they receive, and FCFS will again become optimal.

Once again it is desirable to extract a general conclusion regarding the optimal system from Tables 6.7 to 6.10. This is not as straightforward as before, considering for example the exceptional case observed in Table 6.9 for weighting curve 4. Taking account of the relative lack of sensitivity of the system gain to departures from optimality, we feel justified in stating this

Conclusion: For the execution time distributions in Fig. 5.9 and the weighting curves in Fig. 5.8, and for swap time and set-up time values smaller than T_e, an excellent suboptimal policy for minimizing weighted response time is given by

a. A 1-3-2 priority ranking for selecting jobs from Q_1, Q_2, Q_3 respectively.

b. A maximum execution time allocation for Q_1 jobs approximately equal to T_e; for Q_2 jobs, the smallest value permitted by the model here; for Q_3 jobs, one clock interval.

As before, we anticipate that further improvement in performance relative to this criterion is possible for smaller values of e_3, which cannot be studied through this Markov model.

Table 6.11 gives a comparison of the system gain for the optimal scheduling system, and the three other time-sharing policies described in section 6.1.4.

6.3 SENSITIVITY OF SOLUTIONS

A very encouraging outcome of the optimization solutions is evidence that the optimal system is relatively insensitive to changes in the parameter values. One must admit that parameter values are not often known precisely, and may also change slowly with time during the normal evolution of an operational system. Our investigation gives a good basis for confidence in the practical implementation of an optimal system. At any moment the system might not be truly optimal due to small perturbations of parameter values, but an excellent suboptimal policy could apparently be maintained with rather infrequent adjustment.

The data presented show that fairly broad changes in T_u, T_e, s, θ, and S are tolerable. Moreover there is ample evidence that the optimal scheduling systems for distributions (2) and (3) are identical under

TABLE 6.11

COMPARISON OF SYSTEM GAIN g

Parameter	Case 1 Value	Case 2 Value
T_u	20 sec	20 sec
T_e	4 sec	1 sec
s	0.15 sec	0.15 sec
θ	0	1 sec
S	-	1 sec
F(T)	(2)	(2)

Policy	e_2 (sec)	e_3 (sec)	q (sec)	Parameters Values	Criterion	g
Round-robin	-	6	6	Case 1	MRT[a]	-1.1044
Priority A	2	6	8	Case 1	MRT	-1.1109
Priority B	2	6	2	Case 1	MRT	-1.0780
Optimum	4	6	.05	Case 1	MRT	-1.0341
Round-robin	-	6	6	Case 1	MWRT	- .21703
Priority A	2	6	8	Case 1	MWRT	- .22561
Priority B	2	6	2	Case 1	MWRT	- .14790
Optimum	4	6	.05	Case 1	MWRT	- .09639
Round-robin	-	1.5	1.5	Case 2	MRT	- .49525
Priority A	0.5	1.5	2.0	Case 2	MRT	- .49565
Priority B	0.5	1.5	0.5	Case 2	MRT	- .49967
Optimum	1	1.5	.05	Case 2	MRT	- .48526
Round-robin	-	4.0	4.0	Case 2	MWRT	- .27086
Priority A[b]	1.25	4.0	5.0	Case 2	MWRT	- .26051
Priority B	1.25	4.0	1.25	Case 2	MWRT	- .24588
Optimum	1.5	4.0	.05	Case 2	MWRT	- .24070

[a] MRT signifies minimum response time criterion, MWRT denotes minimum weighted response time as per curve 1 of Fig. 5.8.

[b] Because e_3 is constrained here to be larger than T_e, we take e_2 larger than indicated in section 6.1.4 for priority policies A and B.

either of the criteria we have proposed. These are significantly different distributions, but more closely represent what one can expect in time-sharing systems than does the exponential distribution. For this reason we discount the importance of the result obtained for the exponential distribution in minimization of response time. Thus we conclude that the optimal system is fairly insensitive to the form of F(T) within the limits of what one expects for actual systems.

Most significantly, the optimal scheduling control systems are nearly identical for both criteria, including the three weighting curves given in Fig. 5.8. To be sure, the restriction which the Markov model imposes on the choice of e_3 accounts for this to some degree. Nevertheless, the similarity of results is good evidence that the system designer need not be overly concerned with achieving a precise specification of the relative weight associated with response time as a function of required execution time. Inasmuch as largely qualitative considerations would be used to obtain a specification, this is a valuable result in our approach.

6.4 APPLICATION AND EXTENSION OF RESULTS

In order to come finally to a scheduling control system suitable for actual practice, we should eliminate the restrictions of the model to whatever extent the data justifies. Taking advantage of the lack of sensitivity of the solutions and using a rather conservative conjecture on the improvement possible through reduction of e_3, we can suggest a near optimal system suitable for either criterion. This is a two priority

queue system as previously described in section 6.1.2. New commands are entered in the first priority queue, and a new command preempts a lower priority job at the end of one clock interval. A first priority job is executed for maximum time T_e and then placed at the end of the lower priority queue. In the absence of higher priority jobs, the lower priority queue is served on a round-robin basis for a "quantum" of execution time which is larger than T_e, say $2T_e$.

The last feature is not suggested by the optimization results from the model. Although the model was designed to include round-robin service on the last queue, the 1-3-2 priority policies found actually circumvent this. Yet this seems necessary on practical grounds, to ensure that extensive execution time is not devoted to commands which are in error.

The above system is very similar to the scheduling procedures currently in use in time-shared computers. The striking difference is the emphasis upon rapid preemption of lower priority jobs. Existing systems permit preemption only after a significant delay, and this degrades the response time obtained for trivial computations.

Indeed, the predominance of priority policies in our results suggests that existing multi-level priority systems have the basic structure needed for an optimal system under either criterion we have studied. The critical element requiring further analysis is the optimum time allocation.

As already mentioned, there is good reason to believe the optimal system could be improved by studying smaller values of e_3 than allowed by the model including hyperexponential distributions. If interest remains only on two or three queue systems, this cannot be accommodated by the Markov modeling approach. By passing to systems with more than three levels however one can obtain whatever advantage there is to be gained. The limiting consideration is the number of states in the extended model, and the attendant difficulty in deriving formulae for transition probabilities, etc. Table 6.12 illustrates how the number of states increases with the number of queue levels and the number of consoles. Of course, inasmuch as our results have simplified the systems which need be considered, one could also use a simulation model to study smaller e_3 values. A largely different Semi-Markov decision model will be needed to handle a large number of consoles (50 or more), as systems will have in a few years.

TABLE 6.12

NUMBER OF STATES FOR THE COMPUTER MODEL

Number of Consoles	Number of Queues	Number of States
4	3	75
4	4	175
6	4	462

Another short coming of the model which requires further study concerns the distribution of the user reaction time, i.e. the arrival time t_a. We have assumed a negative exponential with mean T_u. However the distribution observed in practice is likely to be more accurately modeled by a hyperexponential or Erlang distribution. In order to make these practicable we can use the method of exponential phases ([7], p. 19). Using say a two-phase Erlang, the state of the system could be described by variables,

a = number of consoles in second phase of arrival time,

n_1, n_2, n_3 = number of jobs in Q_1, Q_2, Q_3, respectively,

ℓ = a variable indicating the status of main memory.

Now, however, the value of a is <u>not</u> observable for decision making. Thus we enter a new theoretical area, decision processes without complete observability of the system state. Extensive theoretical work will be needed to determine how such processes may be optimized.

CHAPTER VII

CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

The efforts in this research have ranged over the spectrum of activities involved in the application of Semi-Markov sequential control processes to systems with queues. These include theoretical developments, computational methods, and modeling and optimization of physical systems. The results in each of these areas are promising and recommend this technique for use by system engineers concerned with stochastic service systems.

The theoretical work has dealt with a discrete-time Semi-Markov decision process, and the optimization of the return from this process operating over infinite time. Beginning from an appropriate definition of optimization, we have established under quite general conditions that there exists a particularly simple optimal control policy. Such a policy is described as stationary and deterministic, in that a unique control action is associated with observation of each of the system states, for all time. Moreover this policy is optimal for any initial condition on the beginning of the process. This original result has important implications of a practical nature, for it simplifies both the computation of the optimal policy and its implementation in a physical system.

The computational method exploited in this study is the algorithm devised by Howard and Jewell, limited to ergodic policies. Our experience

with this procedure has brought to light the effects of computational errors upon it, previously not described by its originators. Without resorting to special computing techniques to increase precision, the algorithm has been quite successfully applied to a 75 state model for a multiple-access or time-shared computer. Occasionally, evidence has been found of improper convergence, but the most unfavorable result of this has been to mask the structure of the true optimal policy. More importantly, computational errors seem to practically eliminate the discovery of equivalent optimal policies.

Although the potential applicability of Markovian decision processes to stochastic service systems has been recognized, our effort in this direction is, we believe, the first serious attempt to exploit such models. The results are largely encouraging. Using the discrete-time Semi-Markov control process, we have formulated a model for the scheduling control function in time-shared computers. It is the most realistic stochastic model, other than simulation models, which has been proposed for these systems. Two alternative optimization criteria have been proposed for the model, both concerned with the response time given a user's command by the time-shared computer. The effectiveness of the Howard algorithm has allowed us to economically determine optimal scheduling systems over an appropriate range of system parameter values.

Two important conclusions have been reached relevant to current time-sharing practice. Although the Semi-Markov decision model includes a

wide class of scheduling policies, it has been found that priority policies are predominant in the computed optimal policies. Hence existing multiple priority scheduling algorithms have the basic structure needed for an optimal system. Further study is needed regarding the maximum execution time allocation for each priority level. In this respect our results also show a need to have almost immèdiate preemption of a low priority job upon the arrival of a high priority job. In existing systems, preemption can occur only after a significant delay.

The computer model described is limited to three queue scheduling systems. Jobs are placed in the last queue after receiving total execution time significantly larger than T_e, the mean execution time of the population of jobs. Subject to these limitations, our results lead us to propose a two priority level scheduling system. A new arrival would be given first priority and would preempt any second priority job at the end of one clock interval. First priority jobs would receive maximum execution time equal to T_e before being relegated to the second priority queue. Round-robin service would be given to the lower queue, with a quantum duration much larger than T_e.

While seeking a comparison of the computed optimal scheduling system with existing time-sharing procedures, we have discovered another important advantage of the decision process formulation. This involves the freedom of using alternative return functions for the model with a specified control policy. The Howard algorithm provides an analysis of per-

formance relative to the given returns, and thus one can study various aspects of performance which are difficult to determine in any other approach. With this capability we have compared the expectation of response time for alternative policies, conditional upon the execution time of a command. This presents a much more informative picture of the performance observed by the user.

This research has shown that very practical results concerning stochastic service systems can be obtained through Markovian sequential decision processes. Future research in both the theoretical and applied areas is likely to be profitable. Attention should be given to decision processes where additional constraints are imposed on the set of admissible policies. As an example we suggest decision processes where the system state is not completely observable, so that the same control action must be used in a subset of system states. Another valuable effort would deal with systematizing the formulation of transition probabilities and such data for complex models. This formulation has been somewhat burdensome in the model undertaken here. Finally, we would emphasize that future research ought to be well motivated by realistic models for practical systems. Future developments in time-shared computer organization are bound to be a gold mine in this respect for the system engineer.

APPENDIX A

SUPPLEMENTARY THEOREMS

Several theorems used in Chapter IV are presented here. The first of these is the well-known Abel's theorem, a proof of which is given by Rudin ([56], p. 160). The second theorem stems from a corollary 1a of Widder ([89], p. 182) for the Laplace-Stieltjes transform.

Abelian Theorem I: Let $f(\alpha) = \sum_{n=0}^{\infty} a_n \alpha^n$ be convergent for $0 \leq \alpha < 1$. If

$$\sum_{n=0}^{T} a_n \to A \quad \text{as } T \to \infty,$$

then

$$f(\alpha) \to A \quad \text{as } \alpha \to 1^-.$$

Abelian Theorem II: Let $f(\alpha) = \sum_{n=0}^{\infty} a_n \alpha^n$ be convergent for $0 \leq \alpha < 1$. If

$$\frac{1}{T} \sum_{n=0}^{T} a_n \to A \quad \text{as } T \to \infty$$

then

$$(1-\alpha) f(\alpha) \to A \quad \text{as } \alpha \to 1^-.$$

Theorem III: Let a_n, $n = 0,1,2,\ldots$, be a uniformly bounded sequence, so that

$$f(\alpha) = \sum_{n=0}^{\infty} a_n \alpha^n$$

is absolutely and uniformly convergent for $0 \leq \alpha < 1$. Then

$$\liminf_{\alpha \to 1^-} (1-\alpha) f(\alpha) \geq \liminf_{T \to \infty} \frac{1}{T} \sum_{n=0}^{T} a_n$$

Proof: Let a be the uniform bound on the a_n, so that $|a_n| < a$. The stated convergence properties of $f(\alpha)$ follow immediately from theorems in Rudin ([56], p. 52, p. 62, p. 134). Moreover both sides of the above inequality are finite.

Let $S_T = \sum_{n=0}^{T} a_n$, and observe that $a_T = S_T - S_{T-1}$, for $T = 1,2,\ldots$. Then

$$f(\alpha) = a_0 + \sum_{n=1}^{\infty} (S_n - S_{n-1}) \alpha^n$$

$$= (1-\alpha) \sum_{n=0}^{\infty} S_n \alpha^n$$

from the absolute convergence of $f(\alpha)$. So

$$(1-\alpha) f(\alpha) = (1-\alpha)^2 \sum_{n=0}^{\infty} S_n \alpha^n$$

Consider a fixed value of $T \geq 1$. One can write

$$(1-\alpha) f(\alpha) = (1-\alpha)^2 \sum_{n=0}^{T-1} S_n \alpha^n + (1-\alpha)^2 \sum_{n=T}^{\infty} S_n \alpha^n$$

$$= (1-\alpha)^2 \sum_{n=0}^{T-1} S_n \alpha^n + (1-\alpha)^2 \sum_{n=T}^{\infty} \frac{S_n}{n} n \alpha^n$$

The term S_n/n can be positive or negative for any $n \geq T$. In any case however

$$\frac{S_n}{n} \geq \underset{n \geq T}{\text{g.l.b.}} \frac{S_n}{n}$$

where g.l.b. denotes the greatest lower bound. The existence of this bound follows from the boundedness of the a_n. Therefore

$$(1-\alpha)f(\alpha) \geq (1-\alpha)^2 \sum_{n=0}^{T-1} S_n \alpha^n + (1-\alpha)^2 \left[\underset{n \geq T}{\text{g.l.b.}} \frac{S_n}{n}\right] \sum_{n=T}^{\infty} n\alpha^n$$

and evaluating the last summation

$$(1-\alpha)f(\alpha) \geq (1-\alpha)^2 \sum_{n=0}^{T-1} S_n \alpha^n + \underset{n \geq T}{\text{g.l.b.}} \frac{S_n}{n} \left[T\alpha^T(1-\alpha) + \alpha^{T+1}\right] \quad (A.1)$$

Now for $\alpha \to 1^-$ we have

$$\liminf_{\alpha \to 1^-} (1-\alpha)f(\alpha) \geq \underset{n \geq T}{\text{g.l.b.}} \frac{S_n}{n}$$

for any T. Upon letting $T \to \infty$ we have

$$\liminf_{\alpha \to 1^-} (1-\alpha)f(\alpha) \geq \liminf_{T \to \infty} \frac{1}{T} \sum_{n=0}^{T} a_n$$

Lemma 1: Let $\{a_n\}$, $\{b_n\}$ be two sequences, n = 0,1,2,3,... Suppose that

a. $\sum_{n=0}^{\infty} |a_n|$ is convergent

b. $\lim_{n \to \infty} b_n = b$

Then

$$\lim_{T\to\infty} \sum_{n=0}^{T} a_n b_{T-n} = b \sum_{n=0}^{\infty} a_n$$

Proof: See Smith [91], p. 14.

Lemma 2: Let $\{a_n\}$, $\{b_n\}$ be two sequences such that

a. $\lim_{n\to\infty} b_n = b$

b. $\lim_{n\to\infty} \frac{1}{T} \sum_{n=0}^{T} a_n = a$

c. $\lim_{T\to\infty} \frac{1}{T} \sum_{n=0}^{T} |a_n| = A$

Then

$$\lim_{T\to\infty} \frac{1}{T} \sum_{n=0}^{T} a_n b_{T-n} = ab$$

Proof: One can write

$$\left| \frac{1}{T} \sum_{n=0}^{T} a_n b_{T-n} - ab \right| = \left| \frac{1}{T} \sum_{n=0}^{T} a_n (b_{T-n}-b) + \frac{1}{T} \sum_{n=0}^{T} b(a_n-a) \right|$$

and thus

$$\left| \frac{1}{T} \sum_{n=0}^{T} a_n b_{T-n} - ab \right| \leq \left| \frac{1}{T} \sum_{n=0}^{T} a_n (b_{T-n}-b) \right| + |b| \left| \frac{1}{T} \sum_{n=0}^{T} (a_n-a) \right|$$

For a given $\epsilon > 0$, let N be such that

$$|b_n-b| < \frac{\epsilon}{A} \quad \text{for } n \geq N,$$

and let M denote the least upper bound on $|b_n-b|$ for any n. Then we have

$$\left|\frac{1}{T}\sum_{n=0}^{T} a_n b_{T-n} - ab\right| < \frac{\epsilon}{A}\frac{1}{T}\sum_{n=0}^{T-N}|a_n| + M\frac{1}{T}\sum_{n=T-N+1}^{T}|a_n| + |b|\left|\sum_{n=0}^{T}(a_n-a)\right|$$

Now T can be chosen so large that

$$\left|\frac{1}{T}\sum_{n=0}^{T} a_n b_{T-n} - ab\right| < \epsilon$$

and the lemma is proved.

APPENDIX B

ASYMPTOTIC PROPERTIES OF A MARKOV-RENEWAL PROCESS

In a Markov-Renewal process the principal interest centers on the random variable $U_j(T)$ and its expectation $M_{ij}(T)$. These have been defined as

$U_j(T)$ = the number of times the system enters state j in the interval [1,T], where T = 1,2,3, ...

$M_{ij}(T)$ = the expectation of $U_j(T)$ given the process begins in state i at time 0.

or

$$M_{ij}(T) = E[U_j(T) \mid Z_0 = i,\ W_0 = 0]$$

Two important theorems concerning the asymptotic behavior of $M_{ij}(T)$ as $T \to \infty$ are given here. These are established from the theory of simple renewal processes [90,91,92]. A Markov-Renewal process contains an imbedded simple renewal process for every recurrent state. Additional references are [93,94,100].

Elementary Renewal Theorem: Let j be a recurrent state of a Markov-Renewal process so that $f_{jj} = 1$. Let ℓ_{jj} denote the mean recurrence time of state j. For any state i

$$\lim_{T\to\infty} \frac{M_{ij}(T)}{T} = \frac{f_{ij}}{\ell_{jj}}$$

If j is non-recurrent, i.e., $f_{jj} < 1$, the limit above is zero.

Proof: The proof of this theorem follows from Eq. (4.2),

$$M_{ij}(T) = G_{ij}(T) + \sum_{n=1}^{T-1} G_{ij}(T-n)\psi_{jj}(n) \tag{4.2}$$

and some results of Feller [100] on simple renewal processes. Feller has shown that for a state j for which $f_{jj} = 1$,

$$\lim_{n\to\infty} \psi_{jj}(n) = \frac{1}{\ell_{jj}} \tag{B.1}$$

provided that $G_{jj}(T)$ is not a lattice distribution, i.e., is not periodic. If $G_{jj}(T)$ is periodic then there exists an integer $\overline{\omega} > 1$ such that $g_{jj}(n) > 0$ only for values of n which are integer multiples of $\overline{\omega}$. If $G_{jj}(T)$ is periodic

$$\lim_{n\to\infty} \frac{\psi_{jj}(n\overline{\omega})}{\overline{\omega}} = \frac{1}{\ell_{jj}} \tag{B.2}$$

If j is not recurrent, Feller establishes that

$$\sum_{n=1}^{\infty} \psi_{jj}(n) < \infty \tag{B.3}$$

From Eq. (4.2) and the fact that $G_{ij}(T)$ is bounded,

$$\lim_{T\to\infty} \frac{M_{ij}(T)}{T} = \lim_{T\to\infty} \frac{1}{T} \sum_{n=1}^{T-1} G_{ij}(T-n)\psi_{jj}(n)$$

It follows then from Eqs. (B.1), (B.2) and Lemma 2 of Appendix A that if j is recurrent

$$\lim_{T\to\infty} \frac{M_{ij}(T)}{T} = \frac{f_{ij}}{\ell_{jj}}$$

On the other hand, if j is not recurrent it follows from Eq. (B.3) that the limit is zero.

A recurrent state of a Markov-Renewal process is called aperiodic if the greatest common divisor of the values n for which $\psi_{jj}(n) > 0$ is unity. The following theorem is also established from Feller's result, in the manner suggested in the discussion of Smith's paper [90]. An alternative proof is given in [91].

Key Renewal Theorem: Let j be an aperiodic recurrent state of a Markov-Renewal process. Let a_n be a non-negative sequence with

$$\sum_{n=1}^{\infty} a_n = a$$

Then

$$\lim_{T\to\infty} \sum_{n=1}^{T-1} a(T-n)\psi_{jj}(n) = \frac{a}{\ell_{jj}}$$

Proof: This follows immediately from (B.1) and Lemma 1 of Appendix A.

APPENDIX C

FORMULAE FOR THE COMPUTER MODEL

The data which are required for the decision process are the values of p_{ij}^k, ν_i^k, and b_i^k for every i, j, and k. An examination of Table 5.1, which lists all admissible transitions, shows that there are 475 transition probabilities to be specified. In addition, there are 141 values each for the mean transition time ν_i^k and the one-transition return b_i^k. It will be tedious both for the reader and the author to explain the derivation of the formula which yields each of these values as a function of the system parameters of Table 5.2. We shall rely on the illustrations given in Chapter V to convey the method. Here we will merely list the results of the derivations.

There are a number of terms common to many of the formulae. In order to condense the presentation we list these terms in Table C.1 as individual components G(m) of a vector G.

It is convenient to also treat the values p_{ij}^k and ν_i^k as elements of vectors P and T respectively. The vector P will be constructed by successively inserting the values of p_{ij}^k for the admissible values of j and k, in ascending order by i from i = 1 to i = 75. The first value of P is p_{11}^1. Behind p_{11}^1, we place the values p_{1j}^1 in ascending order by j. Because there are no other actions than k = 1 for state i = 1, the next element of P is p_{21}^1. Behind this are placed p_{22}^1, p_{25}^1,...,$p_{2,44}^1$. Con-

TABLE C.1

ELEMENTS OF THE VECTOR G

m	G(m)
1	$e^{-\lambda s}$
2	$e^{-2\lambda s}$
3	$e^{-3\lambda s}$
4	$e^{-\lambda S}$
5	$e^{-2\lambda S}$
6	$e^{-3\lambda S}$
7	$1 - \theta$
8	$1 - \theta + \theta\, e^{-\lambda S}$
9	$1 - \theta + \theta\, e^{-2\lambda S}$
10	$1 - \theta + \theta\, e^{-3\lambda S}$
11	$\sum_{n=1}^{e_2} e^{-\lambda n} f_n$
12	$\sum_{n=1}^{e_2} e^{-2\lambda n} f_n$
13	$\sum_{n=1}^{e_2} e^{-3\lambda n} f_n$
14	$\sum_{n=1}^{e_2} f_n = F(e_2)$

TABLE C.1 (Continued)

m	G(m)
15	$\sum_{n=1}^{\infty} f_{n+e_2} = 1 - F(e_2)$
16	$e^{-\lambda e_2}$
17	$e^{-2\lambda e_2}$
18	$e^{-3\lambda e_2}$
19	$e^{-\lambda(e_2 + s)}$
20	$e^{-2\lambda(e_2 + s)}$
21	$e^{-3\lambda(e_2 + s)}$
22	$\sum_{n=1}^{e_3-e_2} e^{-\lambda n} f_{n+e_2} \Big/ \sum_{n=1}^{\infty} f_{n+e_2}$
23	$\sum_{n=1}^{e_3-e_2} e^{-2\lambda n} f_{n+e_2} \Big/ \sum_{n=1}^{\infty} f_{n+e_2}$
24	$\sum_{n=1}^{e_3-e_2} e^{-3\lambda n} f_{n+e_2} \Big/ \sum_{n=1}^{\infty} f_{n+e_2}$
25	$\sum_{n=1}^{e_3-e_2} f_{n+e_2} \Big/ \sum_{n=1}^{\infty} f_{n+e_2}$
26	$\sum_{n=1}^{\infty} f_{n+e_3} \Big/ \sum_{n=1}^{\infty} f_{n+e_2}$
27	$e^{-\lambda(e_3 - e_2)}$

TABLE C.1 (Continued)

m	G(m)
28	$e^{-2\lambda(e_3 - e_2)}$
29	$e^{-3\lambda(e_3 - e_2)}$
30	$e^{-\lambda(e_3 - e_2 + s)}$
31	$e^{-2\lambda(e_3 - e_2 + s)}$
32	$e^{-3\lambda(e_3 - e_2 + s)}$
33	$\sum_{n=1}^{q} e^{-\lambda n} f_{n+e_3} \Big/ \sum_{n=1}^{\infty} f_{n+e_3}$
34	$\sum_{n=1}^{q} e^{-2\lambda n} f_{n+e_3} \Big/ \sum_{n=1}^{\infty} f_{n+e_3}$
35	$\sum_{n=1}^{q} e^{-3\lambda n} f_{n+e_3} \Big/ \sum_{n=1}^{\infty} f_{n+e_3}$
36	$\sum_{n=1}^{q} f_{n+e_3} \Big/ \sum_{n=1}^{\infty} f_{n+e_3}$
37	$\sum_{n=q+1}^{\infty} f_{n+e_3} \Big/ \sum_{n=1}^{\infty} f_{n+e_3}$
38	$e^{-\lambda q}$
39	$e^{-2\lambda q}$
40	$e^{-3\lambda q}$

TABLE C.1 (Concluded)

m	G(m)
41	$e^{-\lambda(s+q)}$
42	$e^{-2\lambda(s+q)}$
43	$e^{-3\lambda(s+q)}$
44	$1/(1-e^{-4\lambda})$
45	$s + \theta S + e_2 \sum_{n=1}^{\infty} f_{n+e_2} + \sum_{n=1}^{e_2} n f_n$
46	$s + \left\{(e_3 - e_2) \sum_{n=1}^{\infty} f_{n+e_3} + \sum_{n=1}^{e_3-e_2} n f_{n+e_2}\right\} \Big/ \sum_{n=1}^{\infty} f_{n+e_2}$
47	$s + \left\{\sum_{n=1}^{q} n f_{n+e_3} + q \sum_{n=1}^{\infty} f_{n+e_3+q}\right\} \Big/ \sum_{n=1}^{\infty} f_{n+e_3}$
48	G(46) - s
49	G(47) - s

tinuing in this manner, we eventually will insert $p^1_{8,48}$ behind $p^1_{8,26}$. Now state i = 8 also has k = 2 as a permitted action. So $p^2_{8,2}$ goes behind $p^1_{8,48}$ and this is followed by $p^2_{8,7}$. For a given i, all the elements p^k_{ij} are inserted in P before proceeding to place elements for the state i + 1. Figure C.1 illustrates the structure of P.

The vector T is structured in essentially the same manner as P. Here however there is only one element for given i and k. The first element of T is ν^1_1. Succeeding elements of the vector are obtained by going to the next higher value of i and inserting ν^k_i for each of the admissible values of k. See Fig. C.2.

The formulae for the computer model will be presented in terms of the elements of P, T, and G. In order to make it convenient to determine the appropriate entry for given values of i, j, and k, we have listed in Table C.2 the first of the entries used for a given state i. Thus if we are interested in the formulae for state i = 20, we see from Table C.2 that the first P entry pertaining to state 20 is P(164). Referring to Table 5.1 and recalling the structure of P, we find that P(164) is $p^1_{20,8}$. Counting the entries in the row i = 20 of Table 5.1, we see that $p^2_{20,45}$ is given by P(171).

Tables C.3 and C.4 give the formulae for all entries of the P and T vectors.

The formulation of the decision process for the computer model is completed by the formulae for the one-transition returns, b^k_i. A nearly complete discussion of these formulae has been given in Chapter V.

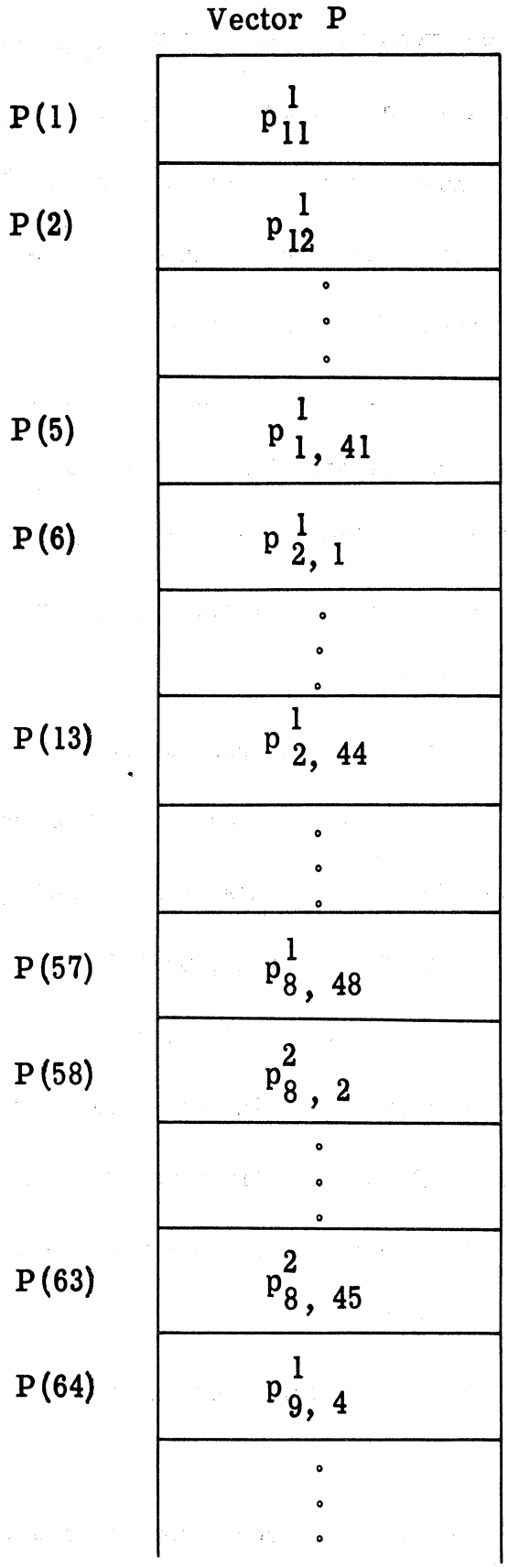


Fig. C.1. Structure of the vector P of transition probabilities p^k_{ij}.

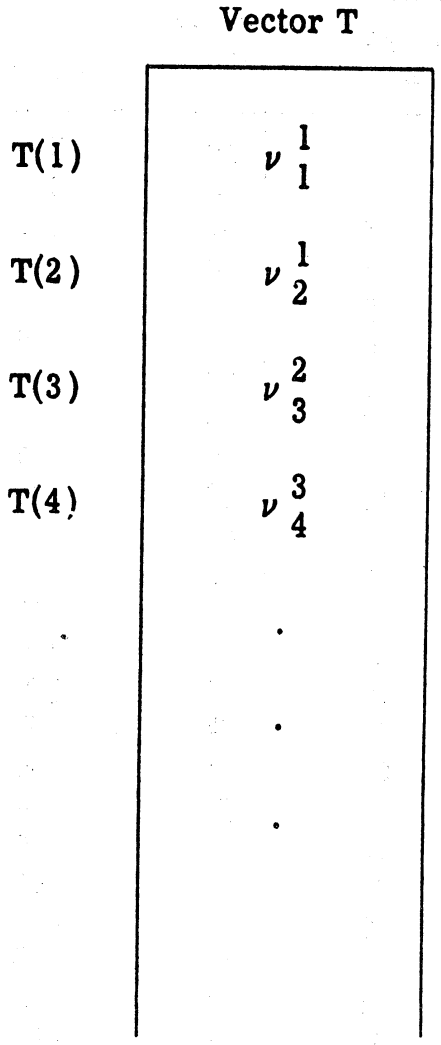


Fig. C.2. Structure of the vector T of mean transition times ν_i^k.

TABLE C.2

CORRESPONDENCE OF STATE INDICES AND VECTOR ELEMENTS

State index i	First P element for i	First T element for i	State index i	First P element for i	First T element for i
1	1	1	39	316	65
2	6	2	40	324	67
3	14	3	41	328	68
4	22	4	42	330	69
5	30	5	43	334	71
6	38	6	44	338	73
7	46	7	45	342	75
8	52	8	46	346	77
9	64	10	47	350	79
10	76	12	48	356	82
11	88	14	49	360	84
12	100	16	50	366	87
13	106	17	51	370	89
14	118	19	52	376	92
15	124	20	53	380	94
16	136	22	54	384	96
17	142	23	55	390	99
18	154	25	56	394	101
19	160	26	57	400	104
20	164	27	58	406	107
21	172	29	59	412	110
22	180	31	60	418	113
23	188	33	61	422	115
24	196	35	62	428	118
25	204	37	63	432	120
26	216	40	64	434	121
27	224	42	65	438	123
28	236	45	66	440	124
29	244	47	67	444	126
30	256	50	68	448	128
31	264	52	69	452	130
32	268	53	70	456	132
33	276	55	71	460	134
34	280	56	72	464	136
35	288	58	73	468	138
36	296	60	74	470	139
37	304	62	75	474	141
38	312	64			

TABLE C.3

ELEMENTS OF THE VECTOR P OF TRANSITION PROBABILITIES

m	P(m)
1	0
2	$4\,e^{-3\lambda}(1-e^{-\lambda})/(1-e^{-4\lambda})$
3	$6\,e^{-2\lambda}(1-e^{-\lambda})^2/(1-e^{-4\lambda})$
4	$4\,e^{-\lambda}(1-e^{-\lambda})^3/(1-e^{-4\lambda})$
5	$(1-e^{-\lambda})^4/(1-e^{-4\lambda})$
6	G(3) G(10) G(13)
7	3(G(2) G(9) G(12) - P(6))
8	G(10) G(15) G(21)
9	3(G(1) G(8) G(11)) - 2 P(7) - 3P(6)
10	3(G(9) G(20) - G(10) G(21)) G(15)
11	G(14) - P(6) - P(7) - P(9)
12	3G(8) G(15) G(19) - 2P(10) - 3P(8)
13	G(15) - P(8) - P(10) - P(12)
14	G(3) G(24)
15	3(G(2) G(23) - P(14))
16	G(26) G(32)
17	3G(1) G(22) - 3P(14) - 2P(15)
18	3(G(31) - G(32)) G(26)
19	G(25) - P(14) - P(15) - P(17)
20	3G(26) G(30) - 3P(16) - 2P(18)
21	G(26) - P(16) - P(18) - P(20)
22	G(3) G(35)
23	3(G(2) G(34) - P(22))
24	G(37) G(43)
25	3G(1) G(33) - 3P(22) - 2P(23)
26	3(G(42) - G(43)) G(37)
27	G(36) - P(22) - P(23) - P(25)
28	3G(37) G(41) - 3P(24) - 2P(26)
29	G(37) - P(24) - P(26) - P(28)
30	G(24)
31	3(G(23) - G(24))
32	G(26) G(29)
33	3G(22) - 3P(30) - 2P(31)
34	3(G(28) - G(29)) G(26)
35	G(25) - P(30) - P(31) - P(33)
36	3G(26) G(27) - 3P(32) - 2P(34)
37	G(26) - P(32) - P(34) - P(36)
38	G(35)

TABLE C.3 (Continued)

m	P(m)
39	3(G(34) - G(35))
40	G(37) G(40)
41	3G(33) - 3P(38) -2P(39)
42	3(G(39) - G(40)) G(37)
43	G(36) - P(38) - P(39) - P(41)
44	3G(37) G(38) - 3P(40) - 2P(42)
45	G(37) - P(40) - P(42) - P(44)
46	G(2) G(9) G(12)
47	2(G(1) G(8) G(11) - P(46))
48	G(9) G(15) G(20)
49	G(14) - P(46) - P(47)
50	2(G(8) G(19) - G(9) G(20)) G(15)
51	G(15) - P(48) - P(50)
58	G(2) G(23)
59	2(G(1) G(22) - P(58))
60	G(26) G(31)
61	G(25) - P(58) - P(59)
62	2(G(26) G(30) - P(60))
63	G(26) - P(60) - P(62)
70	G(2) G(34)
71	2(G(1) G(33) - P(70))
72	G(37) G(42)
73	G(36) - P(70) - P(71)
74	2(G(41) - G(42)) G(37)
75	G(37) - P(72) - P(74)
82	G(23)
83	2(G(22) - G(23))
84	G(26) G(28)
85	G(25) - P(82) - P(83)
86	2(G(26) G(27) - P(84))
87	G(26) - P(84) - P(86)
94	G(34)
95	2(G(33) - G(34))
96	G(37) G(39)
97	G(36) - P(94) - P(95)
98	2(G(38) - G(39)) G(37)
99	G(37) - P(96) - P(98)

TABLE C.3 (Continued)

m	P(m)
160	G(1) G(8) G(11)
161	G(14) - P(160)
162	G(8) G(15) G(19)
163	G(15) - P(162)
168	G(1) G(22)
169	G(25) - P(168)
170	G(26) G(30)
171	G(26) - P(170)
176	G(1) G(33)
177	G(36) - P(176)
178	G(37) G(41)
179	G(37) - P(178)
184	G(22)
185	G(25) - P(184)
186	G(26) G(27)
187	G(26) - P(186)
192	G(33)
193	G(36) - P(192)
194	G(37) G(38)
195	G(37) - P(194)
328	G(14)
329	1 - P(328)
332	G(25)
333	1 - P(332)
336	G(36)
337	1 - G(36)
52-57 64-69 76-81 88-93	P(46) to P(51) resp.

TABLE C.3 (Continued)

m	P(m)
100-105, 106-111, 118-123, 124-129	P(58) to P(63) resp.
130-135	P(94) to P(99) resp.
142-147	P(82) to P(87) resp.
112-117, 136-141, 148-153, 154-159	P(70) to P(75) resp.
164-167, 172-175, 180-183, 188-191, 196-199, 204-207, 216-219, 224-227, 236-239, 244-247, 256-259	P(160) to P(163) resp.
200-203, 208-211, 220-223, 228-231, 264-267, 268-271, 276-279, 280-283, 288-291, 296-299, 304-307	P(168) to P(171) resp.
212-215, 240-243, 252-255, 260-263, 272-275, 292-295, 300-303, 308-311, 312-315, 320-323, 324-327	P(176) to P(179) resp.
248-251, 316-319	P(184) to P(187) resp.
232-235, 284-287	P(192) to P(195) resp.
330-1, 334-5, 338-9, 342-3, 346-7, 350-1, 360-1, 366-7, 370-1, 376-7, 380-1, 384-5, 390-1, 394-5, 400-1, 406-7, 412-3, 418-9, 422-3, 428-9	P(328), P(329) resp.

TABLE C.3 (Concluded)

m	P(m)
340-1, 348-9, 352-3, 358-9, 362-3, 372-3, 382-3, 386-7, 392-3, 396-7, 402-3, 408-9, 414-5, 424-5, 432-3, 434-5, 438-9, 440-1, 444-5, 448-9, 452-3, 456-7, 460-1, 464-5, 470-1	P(332), P(333) resp.
344-5, 354-5, 364-5, 368-9, 374-5, 378-9, 388-9, 398-9, 404-5, 410-1, 416-7, 420-1, 426-7, 430-1, 436-7, 442-3, 446-7, 450-1, 454-5, 458-9, 462-3, 466-7, 468-9, 472-3, 474-5	P(336), P(337) resp.

TABLE C.4

ELEMENTS OF THE VECTOR T OF MEAN TRANSITION TIMES

m	T(m)
1	G(44)
2, 7, 8, 10, 12, 14, 26, 27, 29, 31, 33, 35, 37, 40, 42, 45, 47, 50, 68, 69, 71, 73, 75, 77, 79, 82, 84, 87, 89, 92, 94, 96, 99, 101, 104, 107, 110, 113, 115, 118	G(45)
3, 9, 16, 17, 19, 20, 28, 36, 38, 41, 43, 52, 53, 55, 56, 58, 60, 62, 70, 78, 80, 83, 85, 95, 97, 100, 102, 105, 108, 111, 120, 121, 123, 124, 126, 128, 130, 132, 134, 136	G(46)
4, 11, 18, 22, 24, 25, 30, 39, 46, 49, 51, 54, 59, 61, 63, 64, 66, 67, 72, 81, 88, 91, 93, 98, 106, 109, 112, 114, 117, 119, 122, 127, 129, 131, 133, 135, 137, 138, 140, 141	G(47)
5, 13, 23, 32, 48, 65, 74, 90, 116, 139	G(48)
6, 15, 21, 34, 44, 57, 76, 86, 103, 125	G(49)

as indicated there, it is convenient to decompose b_i^k into several terms

$$b_i^k = \alpha_i^k + \delta_i^k + \xi_i^k - \gamma_i^k \nu_i^k$$

Except for α_i^k, the values of the remaining terms for any i and k are given in Chapter V, Section 5.3.1. The term α_i^k is the part of the one-transition return which arises from new arrivals, and accounts for the time-in-system between the arrival time and the end of the transition interval. A general expression for α_i^k is

$$\alpha_i^k = - w_1(4-X_i)T_i^k$$

where X_i is the total number of jobs in system at the beginning of the transition, and T_i^k is the expected time-in-system for a new arrival during the transition interval. In Table C.5 we have listed the values of T_i^k as elements of a vector A. The vector A has the same structure as the vector T. Hence the correspondence of the state index and the vector elements indicated in Table C.2 for T applies as well to A.

TABLE C.5

ELEMENTS OF THE VECTOR A, MEAN TIME-IN-SYSTEM
OF A NEW ARRIVAL DURING A TRANSITION

m	A(m)
1	$4(\lambda + (e^{-\lambda} - 1))/\lambda(1 - e^{-4\lambda})$
2, 7, 8, 10, 12, 14, 26, 27, 29, 31, 33, 35, 37, 40, 42, 45, 47, 50	$\{1 - G(8)\,(G(11) + G(15)\,G(16))\,G(1)\}/\lambda$
3, 9, 16, 17, 19, 20, 28, 36, 38, 41, 43, 52, 53, 55, 56, 58, 60, 62	$\{1 - (G(22) + G(26)\,G(27))\,G(1)\}/\lambda$
5, 13, 23, 32, 48, 65	$\{1 - (G(22) + G(26)\,G(27))\}/\lambda$
4, 11, 18, 22, 24, 25, 30, 39, 46, 49, 51, 54, 59, 61, 63, 64, 66, 67	$\{1 - (G(33) + G(37)\,G(38))G(1)\}/\lambda$
6, 15, 21, 34, 44, 57	$\{1 - (G(33) + G(37)\,G(38))\}/\lambda$
All other values from m = 1 to m = 141	0.

APPENDIX D

THE SEMI-MARKOV SYSTEM INVESTIGATOR (SEMSIN)

D.1 GENERAL DESCRIPTION

This appendix gives a general description of the computer program which has been developed in this research. The program has two major functions:

a. To compute the input data for the Howard algorithm from the formulae of the computer model (see Appendix C and Section 5.23, Chapter V).

b. To carry out the computation of the Howard algorithm (see Section 4.4, Chapter IV).

Because the program is applicable to both discrete-time and continuous-time Semi-Markov decision processes, and because it has utility both for optimization and analysis, we feel it desirable to apply a name suggesting its general applicability. Hence we call the program a Semi-Markov System Investigator, and use the acronym SEMSIN for briefer reference.

The program has been written in the MAD language [31] for the IBM 7090 computer system at The University of Michigan. The various functions to be performed are organized into separate routines, with a MAIN routine being central. Figure D.1 shows the structure of this organization. We will shortly describe the specific function of each routine. For the moment we merely mention that MODEL, XTIME, BSET, and NUSET per-

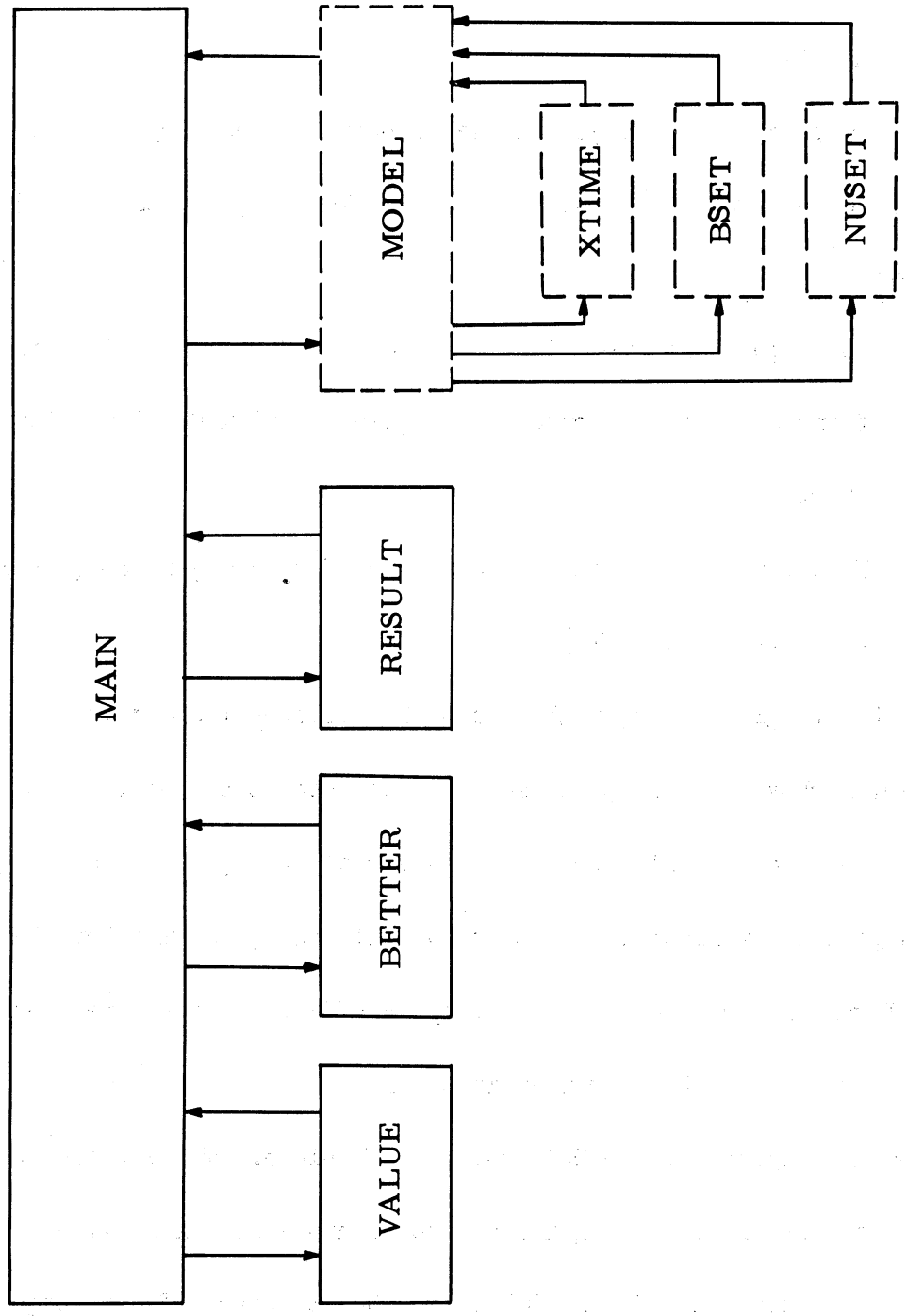


Fig. D.1. Organization of SEMSIN functions.

form the major function a. above, while MAIN, VALUE, BETTER, and RESULT carry out the Howard algorithm. In particular, VALUE carries out the Value Determination Operation, and BETTER carries out the Policy Improvement Operation.

Aside from the computation which must obviously be included in the program, we have found the following features to be of considerable value.

1. An accurate accounting for each solution of the total computation time expended, and that expended in VALUE, BETTER, and MODEL.
2. The capability to specify either a single pass through VALUE, i.e., analysis, or the iterative computation required for optimization.
3. The capability to read (as input) an initial policy to start the computation, to use a standard (first-come, first-served) initial policy, or to continue using a previously specified policy.
4. A print-out of the gain and the control policy determined on each iteration of the algorithm.
5. Termination of the algorithm when the policy has been unchanged on two successive iterations, or when the gain has not increased. (See Sections 4.5.1 and 4.5.2, Chapter IV.)
6. Functional separation of the Howard algorithm and the evaluation of the formulae for the model. The former is implemented ex-

clusively in MAIN, BETTER, VALUE, and RESULT. All other subroutines are concerned only with the particular model.

7. Separation of the execution time distribution F(T) from the most substantial parts of formula evaluation, the routines MODEL, BSET, and NUSET. F(T) enters directly only in XTIME, thus facilitating modifications of F(T).
8. The capability to print values of the test quantities for each state and each action, if desired.
9. A print-out on each iteration of the maximum difference over all states between the test quantity for the current policy and that of any other policy.

After describing each routine briefly, we will discuss the performance of the program in solving the optimization problem for the computer model.

D.2 SUBROUTINES

D.2.1 MODEL: The subroutine MODEL computes the transition probabilities p_{ij}^{k}. It also checks their accuracy by testing for each i and k whether

$$\sum_{j=1}^{N} p_{ij}^{k} = 1$$

and

$$0 \leq p_{ij}^{k} \leq 1 \quad \text{for all } j.$$

The calculation involved is the evaluation of the formulae given in Table C.3, Appendix C. The numerical values determined are stored in a vector as discussed in Appendix C. Before evaluating the formulae MODEL must first evaluate the subsidiary quantities contained in the array G, see Appendix C. Part of this evaluation is done by XTIME. MODEL also sets up the initial policy for the computation.

D.2.2 XTIME: This subroutine computes the elements of the vector G which depend upon the execution time distribution F(T). It also computes the partial return ξ_i^k (see Appendix C and Section 5.3.1 of Chapter V). This partial return depends only upon F(T).

D.2.3 NUSET: This routine computes the values of ν_i^k for every i and k, and stores them in a vector T as described in Appendix C.

D.2.4 BSET: This routine computes the values of b_i^k and stores them in a vector B having the same structure as T.

D.2.5 MAIN: The main part of SEMSIN, called MAIN, sequences the entire computation and tests for completion of the computation. Figure D.2 is a simplified block diagram of the sequence.

D.2.6 VALUE: The routine VALUE carries out the Value Determination Operation of the Howard algorithm, solving the system of equations

$$x_i + x_N \nu_i^k = b_i^k + \sum_{j=1}^{N} p_{ij}^k x_j \qquad (i = 1,2,\ldots,N-1)$$

$$x_N \nu_N^k = b_N^k + \sum_{j=1}^{N} p_{Nj}^k x_j$$

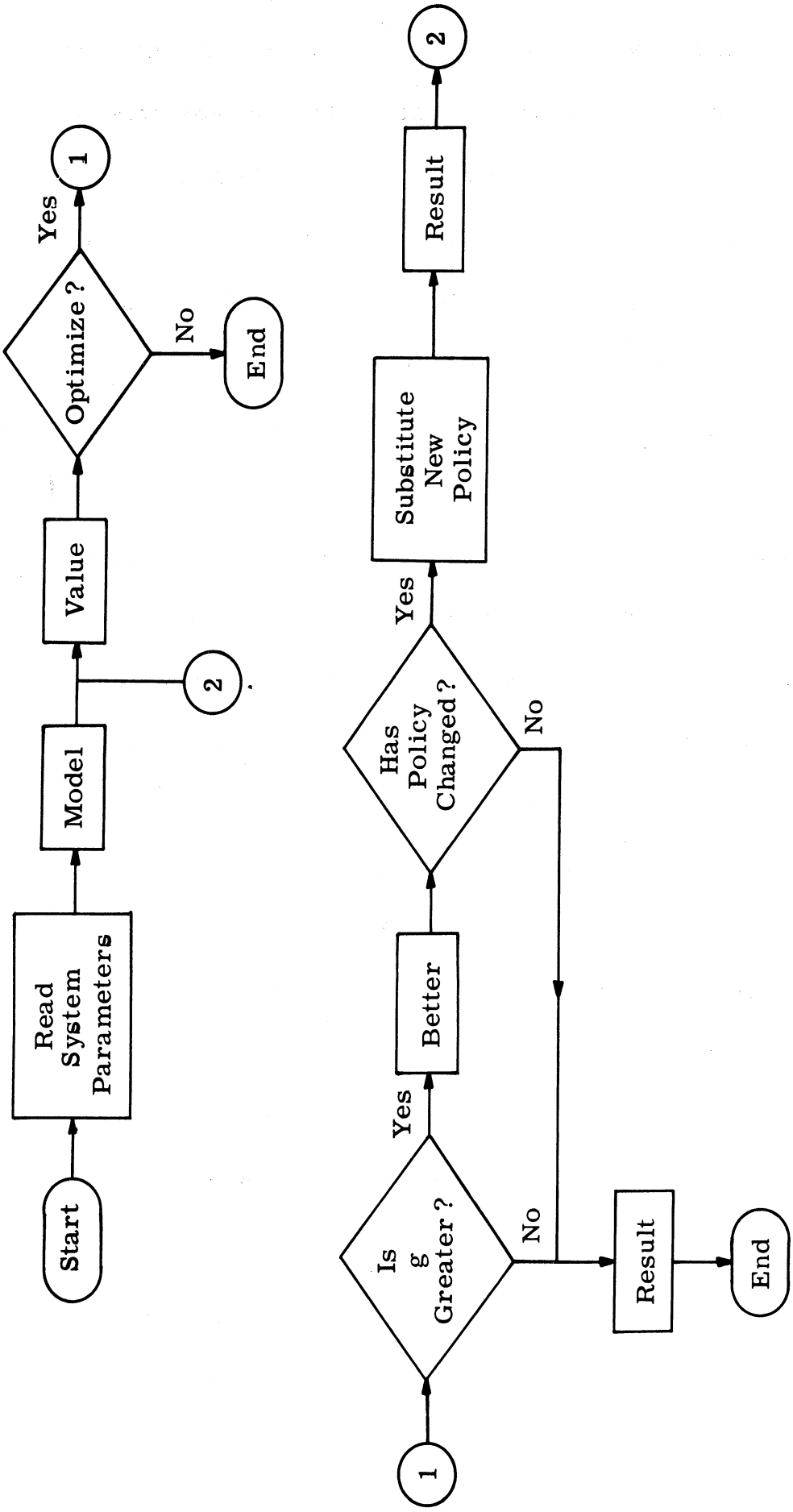


Fig. D.2. Simplified diagram of MAIN sequence.

where $x_i = v_i + g$, within the limits of computational errors. By assumption, $v_N = 0$, so that $x_N = g$. The solution of this system is performed by a library subroutine (SLE.) available at the UM Computer Center. The method is a sophisticated form of Gaussian elimination.

D.2.7 BETTER: This routine performs the Policy Improvement Operation of the Howard algorithm.

D.2.8 RESULT: This routine serves a utility function, converting from the internal representation of the policy to produce a printed tabular representation which is more readily interpreted.

D.3 PERFORMANCE

The performance of SEMSIN in obtaining solutions of the computer model is sufficiently good to recommend the Howard algorithm as a practical and economical procedure for large models. The important aspects of its performance as observed in this research are summarized in Table D.1. The size of the program, including storage reserved for the data arrays but excluding auxiliary routines obtained from the UM library, is 19,000 computer words.

TABLE D.1

OBSERVED SEMSIN PERFORMANCE ON THE COMPUTER MODEL

Item	Observation
Number of iteration for optimization	From 2 to 5, depending upon initial policy.
Time for formula evaluation	0.7 sec. approx.
Time per iteration	17 sec.
Time in VDO per iteration	15.8 sec.
Effect of computational errors	Tolerable. In almost all runs reported in Chap. VI it was clear that an optimal policy was found. No non-terminating iteration was revealed.

LIST OF REFERENCES

1. J. Riordan, Stochastic Service Systems, John Wiley and Sons, Inc., New York, 1962.

2. E. Brockmeyer, H. L. Halstrom, and A. Jensen, "The Life and Works of A. K Erlang," Trans. of the Danish Academy of Technical Sciences, No. 2, 1948.

3. D. R. Cox and W. L. Smith, Queues, Methuen and Co., Ltd., London, and John Wiley and Sons, Inc., New York, 1961.

4. W. Feller, An Introduction to Probability Theory and Its Applications, John Wiley and Sons, Inc., New York, 1957.

5. T. C. Fry, Probability and Its Engineering Uses, D. Van Nostrand Co., Inc., Princeton, N. J., 1928.

6. D. G. Kendall, "Some Problems in the Theory of Queues," J. Royal Statistical Soc., B, 13, 151-173 and 184-185, 1951.

7. P. M. Morse, Queues, Inventories and Maintenance, John Wiley and Sons, Inc., New York, 1958.

8. T. L. Saaty, Elements of Queueing Theory, McGraw-Hill, New York, 1961.

9. R. Syski, Introduction to Congestion Theory in Telephone Systems, Oliver and Boyd, London, 1960.

10. L. Takacs, Introduction to the Theory of Queues, Oxford Univ. Press, New York, 1962.

11. J. Licklider, "Man-Computer Symbiosis," IRE Trans. on Human Factors, 1, 4-11, 1960.

12. J. Licklider and W. E. Clark, "On-Line Man-Computer Communication," Proc. Spring Joint Computer Conf., National Press, Palo Alto, Calif., 1962, pp. 113-128.

13. G. J. Culler and R. W. Huff, "Solution of Nonlinear Integral Equations Using On-Line Computer Control," Ibid., pp. 129-138.

14. R. R. Everett, C. A. Zraket, and H. D. Bennington, "SAGE - A Data Processing System for Air Defense," Proc. Eastern Joint Computer Conf., Washington, D. C., 1957, pp. 148-155.

LIST OF REFERENCES - Continued

15. R. A. McAvoy, "A Central Computer Installation as a Part of an Air Line Reservations Systems," Proc. Fourth Annual Computer Applications Symposium, Armour Research Foundation, Chicago, 1957.

16. R. W. Hamming, "Frontiers in Computer Technology," Proc. Fifth Annual Computer Applications Symposium, Armour Research Foundation, Chicago, 1958, pp. 7-17.

17. F. J. Corbató, M. Merwin-Daggett, R. C. Daley, "An Experimental Time-Sharing System" Proc. Spring Joint Computer Conf., National Press, Palo Alto, Calif., 1962, pp. 335-344.

18. F. J. Corbató, et al., The Compatible Time-Sharing System,: A Programmer's Guide, The M.I.T. Press, Cambridge, Mass., 1963.

19. R. M. Fano, "The MAC system: the computer utility approach," IEEE Spectrum, 2, 1, 56-64, 1965.

20. S. Boilen, et al., "A Time-Sharing Debugging System for a Small Computer," Proc. Spring Joint Computer Conf., Sparton Books, Inc., Baltimore, Md., 1963, pp. 51-57.

21. J. B. Dennis, "A Multiuser Computation Facility for Education and Research," Comm. of the Association for Computing Machinery, 7, 9, 521-529, 1964.

22. T. M. Dunn and J. H. Morrissey, "Remote Computing - An Experimental System,"(Part I), Proc. Spring Joint Computer Conf., Sparton Books, Inc., Baltimore, Md., 1964, pp. 413-423.

23. J. M. Keller, E. C. Strum, and G. H. Yang, "Remote Computing - An Experimental System,"(Part II), Ibid., pp. 425-443.

24. H. A. Kinslow, "The Time-Sharing Monitor System," Proc. Fall Joint Computer Conf., Sparton Books, Inc., Baltimore, Md., 1964, pp. 443-454.

25. J. Schwartz, E. Coffman, and C. Weissman, "A General Purpose Time-Sharing System," Proc. Spring Joint Computer Conf., Sparton Books, Inc., Baltimore, Md., 1964, pp. 397-411.

26. J. C. Shaw, "JOSS: A Designer's View of An Experimental On-Line Computing System," Proc. Fall Joint Computer Conf., Sparton Books, Inc., Baltimore, Md., 1964, pp. 455-464.

LIST OF REFERENCES - Continued

27. Computation Center Staff, "The Dartmouth Time-Sharing System: A Brief Description," Dartmouth College, Hanover, N. H., October 1964.

28. J. B. Dennis, "Segmentation and the Design of Multiprogrammed Computer Systems," IEEE Convention Record, Session 66, 1965.

29. B. Arden, B. Galler, T. O'Brien and F. Westervelt, "Program and Addressing Structure in a Time-Sharing Environment," Computing Center, The University of Michigan, Ann Arbor, Michigan, April 1965, (submitted for publication).

30. J. E. Dresher and C. A. Zito, "The IBM 7741- A Communications-Oriented Computer," IEEE Convention Record, Pt. 5, 216-224, 1964.

31. B. Arden, B. Galler, and R. Graham, "MAD: Michigan Algorithmic Decoder," Computer Center, The University of Michigan, Ann Arbor, Michigan, 1964.

32. A. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," IBM J. Research and Development, 211-229, July, 1959.

33. F. J. Corbató, Lectures at the Engineering Summer Conferences on "Automatic Programming" and "Digital Computers in Real Time," The University of Michigan, June, 1964.

34. R. I. Wilkinson, "Theories for Toll Traffic Engineering in the U.S.A.," Bell System Tech. J., 35, 421-514, 1956.

35. J. Otterman, "Grade of Service of Direct Traffic Mixed with Store-and Forward Traffic," Bell System Tech. J., 41, 1415-1438, July, 1962.

36. D. G. Haenschke, "Analysis of Delay in Mathematical Switching Models for Data Systems," Bell System Tech. J., 42, 709-736, May, 1963.

37. A. B. Fontaine, "Queueing Characteristics of a Telephone Data Transmission System with Feedback," IEEE Trans. Communications and Electronics, No. 68, 449-455, Sept., 1963.

38. L. Kleinrock, Communication Nets: Stochastic Message Flow and Delay, Lincoln Laboratory Publications, McGraw-Hill Book Co., 1964.

39. A. Weingarten, "Storage Requirements for a Message Switching Computer," IEEE Trans. Communication Systems, CS-12, 191-195, 1964.

LIST OF REFERENCES - Continued

40. P. E. Boudreau and M. Kac, "Analysis of a Basic Queueing Problem Arising in Computer Systems," IBM J. Research and Development, 132-140, April, 1961.

41. P. E. Boudreau, J. S. Griffing, and M. Kac, "A Discrete Queueing Problem with Variable Service Times," IBM J. Research and Development, 6, 407-418, October, 1962.

42. R. L. Boyell, "Analysis of Time-Sharing of Digital Computers," J. Soc. Industrial and Applied Mathematics, 8, 102-124, March, 1960.

43. I. Flores, "Derivation of a Waiting Time Factor for a Multiple-Bank Memory," Journal of the ACM, 11, 3, 265-282, July, 1964.

44. B. B. Tasini and S. Winograd, "Multiple Input-Output Links in Computer Systems," IBM J. Research and Development, 6, 306-328, July, 1962.

45. S. A. Korff, Electron and Nuclear Counters, D. Van Nostrand Co., Inc., New York, 1955.

46. E. Parzen, Stochastic Processes, Holden-Day, Inc., San Francisco, 1962.

47. L. Kleinrock, "Analysis of a Time-Shared Processor," Naval Research Logistics Quarterly, 11, 1, 59-73, March 1964.

48. E. G. Coffman, Jr., and B. Krishnamoorthi, Preliminary Analyses of Time-Shared Computer Operation, Professional Paper SP-1719, System Development Corporation, Santa Monica, Calif., 19 August 1964.

49. A. Cobham, "Priority Assignment in Waiting Line Problems," Operations Research, 2, 1, 70-76, 1954.

50. J. Holley, "Waiting Line Subject to Priorities," Operations Research, 2, 3, 341-343, 1954.

51. R. W. Conway and W. L. Maxwell, "Network Dispatching by the Shortest-Operation Discipline," Operations Research, 10, 1, 51-73, 1962.

52. D. W. Fife, "Scheduling with Random Arrivals and Linear Loss Functions," Management Science, 11, 3, 429-437, 1965.

53. J. R. Jackson, "Some Problems in Queueing with Dynamic Priorities," Naval Research Logistics Quarterly, 7, 3, 235-249, 1960.

LIST OF REFERENCES - Continued

54. V. L. Wallace and R. S. Rosenberg, Recursive Queue Analyzer, Cooley Electronics Laboratory, The University of Michigan, Ann Arbor, Michigan, to be published.

55. D. W. Fife and R. S. Rosenberg, "Queueing in a Memory-Shared Computer," Proc. Ninteenth National Conf. of the ACM, Philadelphia, August, 1964.

56. W. Rudin, Principles of Mathematical Analysis, McGraw-Hill Book Co., New York, 1953.

57. W. Kaplan, Operational Methods for Linear Systems, Addison-Wesley Publishing Co., Inc., Reading, Mass., 1962.

58. R. Pyke, "Markov Renewal Processes: Definitions and Preliminary Properties," Annals of Math. Statistics, 32, 1231-1242, 1961.

59. R. Bellman, "A Markovian Decision Process," J. Mathematics and Mechanics, 6, 679-684, Sept. 1957.

60. R. Bellman, Dynamic Programming, Princeton Univ. Press, Princeton, N. J., 1957.

61. R. Bellman and S. Dreyfus, Applied Dynamic Programming, Princeton Univ. Press, Princeton, N. J., 1962.

62. K. Chuang, Optimal Markovian Queueing Systems, Cooley Electronics Laboratory Technical Report No. 153, The University of Michigan, Ann Arbor, June, 1963.

63. R. Howard, Dynamic Programming and Markov Processes, The M.I.T. Press, Cambridge, Mass., 1960.

64. D. Blackwell, "On the Functional Equation of Dynamic Programming," J. Math. Analysis and Applications, 2, 273-276, 1961.

65. D. Blackwell, "Discrete Dynamic Programming," Annals of Math. Statistics, 33, 719-726, 1962.

66. C. Derman, "On Sequential Decisions and Markov Chains," Management Science, 9, 16-24, 1962.

67. C. Derman, "Stable Sequential Control Rules and Markov Chains," J. Math. Analysis and Applications, 6, 257-265, 1963.

LIST OF REFERENCES - Continued

68. A. Manne, "Linear Programming and Sequential Decisions," Management Science, 6, 259-267, April,1960.

69. P. Wolfe and G. Dantzig, "Linear Programming in a Markov Chain," Operations Research, 10, 702-710, 1962.

70. H. M. Wagner, "On the Optimality of Pure Strategies," Management Science, 6, 268-269, 1960.

71. G. de Ghellinck, "Application de la Theorie des Graphes Matrices de Markov et Programmes Dynamiques," Cahiers Centre d'Etudes de Recherche Operationnelle, 3, 1, 5-35, 1961.

72. D. J. White, "Dynamic Programming, Markov Chains, and the Method of Successive Approximations," J. Math. Analysis and Applications, 6, 373-376, 1963.

73. W. S. Jewell, "Markov-Renewal Programming," Operations Research, 11, 938-971, 1963 (in two parts).

74. R. Howard, Semi-Markovian Control Systems, Tech. Report No. 3, Contract Nonr-1841 (87), Operations Research Center, Mass. Institute of Technology, Cambridge, December 1963.

75. J. De Cani, "A Dynamic Programming Algorithm for Embedded Markov Chains When the Planning Horizon is at Infinity," Management Science, 716-733, 1964.

76. J. Eaton and L. Zadeh, "Optimal Pursuit Strategies in Discrete-State Probabilistic Systems," A.S.M.E. Transactions, Journal of Basic Engineering, 23-29, March 1962.

77. J. J. Florentin, "Optimal Control of Continuous Time, Markov Stochastic Processes," Electronics and Control, First Series, Vol. X, June 1961.

78. K. J. Astrom, "Optimal Control of Markov Processes with Incomplete State Information," J. Math. Analysis and Applications, 10, 174-205, 1965.

79. A. A. Feldbaum, "On the Optimal Control of Markov Objects," Automatika i Telemekhanika, 23, 8, 993-1007, August 1962. (Translation published February 1963).

80. G. de Leve, "Stochasticke ∞-staps beslissings-problemen," Statistica Neerlandica, 16, 433-448, 1962.

LIST OF REFERENCES - Continued

81. C. Derman, "On Optimal Replacement Rules When Changes of State are Markovian," Chapt. 9, Mathematical Optimization Techniques, Report R-396-PR, The Rand Corp., Santa Monica, Calif., April 1963.

82. C. Derman, "Optimal Replacement and Maintenance Under Markovian Deterioration with Probability Bounds on Failure," Management Science, 9, 478-481, April 1963.

83. M. G. De Vries, A Dynamic Model for Product Strategy Selection, Industrial Development Research Program, The University of Michigan, Ann Arbor, August 1963.

84. Y. H. Rutenberg, Sequential Decision Models, Contract Nonr-1141(08), Operations Research Group, Case Institute of Technology, Cleveland, April 1961.

85. S. Karlin, "The Structure of Dynamic Programming Models," Naval Research Logistics Quarterly, 2, 285-294, 1955.

86. G. F. Simmons, Introduction to Topology and Modern Analysis, McGraw-Hill, New York, 1963.

87. J. Kelley, General Topology, D. Van Nostrand Co., Inc., Princeton, N. J., 1955.

88. R. Pyke, "Markov-Renewal Processes with Finitely Many States," Annals of Math. Statistics, 32, 1243-1259, 1961.

89. D. Widder, The Laplace Transform, Princeton Univ. Press, Princeton, N. J., 1941.

90. W. L. Smith, "Renewal Theory and Its Ramifications," J. Royal Statistical Soc., B, 20, 243-302 (with discussion), 1958.

91. W. L. Smith, "Asymptotic Renewal Theorems," Proc. Royal Soc. Edinburgh, 64A, 9-48, 1954.

92. A. J. Fabens, "The Solution of Queueing and Inventory Models by Semi-Markov Processes," J. Royal Statistical Soc., B, 23, 113-127, 1961.

93. S. Karlin and A. J. Fabens, "Generalized Renewal Functions and Stationary Inventory Models," J. Math. Analysis and Applications, 5, 461-487, 1962.

LIST OF REFERENCES - Concluded

94. R. Barlow, "Applications of Semi-Markov Processes to Counter Problems," Studies in Applied Probability and Management Science, Arrow, Karlin, and Scarf (Editors), Stanford University Press, 1962.

95. F. R. Gantmacher, Applications of the Theory of Matrices, Interscience Publishers, Inc., New York, 1959.

96. J. Kemeny and J. Snell, Finite Markov Chains, D. Van Nostrand Co., Inc., Princeton, N. J., 1960.

97. R. M. Brown, "An Experimental Study of an On-Line Man-Computer System," IEEE Trans. Electronic Computers, EC-14, 82-85, 1965.

98. R. Rosin, "Determining a Computing Center Environment," Comm. of the ACM, 8, 463-468, 1965.

99. E. Walter and V. Wallace, Further Analysis of a Computing Center Environment, Cooley Electronics Laboratory Technical Report, The University of Michigan, Ann Arbor, Michigan (to be published).

100. W. Feller, "Fluctuation Theory of Recurrent Events," Trans. Am. Math. Soc., 67, 98-119, 1949.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R&D
(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Cooley Electronics Laboratory The University of Michigan Ann Arbor, Michigan	Unclassified
	2b. GROUP

3. REPORT TITLE

THE OPTIMAL CONTROL OF QUEUES, WITH APPLICATIONS TO COMPUTER SYSTEMS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Technical Report

5. AUTHOR(S) (Last name, first name, initial)

Fife, Dennis W.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
October 1965	289	100

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AF 30(602)-3553	06868-1-T
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	CEL TR-170

10. AVAILABILITY/LIMITATION NOTICES

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	United States Air Force Rome Air Development Center Rome, New York

13. ABSTRACT

An important element of a system with queues is the procedure for controlling the use of system resources in processing the random demands for service. Many systems admit a wide class of procedures, and the system engineer requires an optimization technique to isolate an optimal policy within the admissible class. This research pursues the theoretical development and application of Markov sequential control processes for this purpose.

Major effort is devoted to a discrete-time Semi-Markov decision process, and the optimization of performance return from this process over infinite time. A timely example for the application of the technique is the scheduling control problem of multiple-access or time-shared computers. A Semi-Markov model is developed for a computer serving four remote consoles with three queueing levels for user jobs. The computational method devised by Howard and Jewell is used to determine optimal policies.

Results for the computer model show that optimal policies are predominantly priority policies. Also, a need is indicated for more rapid preemption of low priority jobs than used in existing time-shared systems. The mean response time for given execution time is compared for the optimal system and the control procedures now being used.

The practicality of the optimization results establish that this new technique can profitably be used in the design of stochastic service systems.

DD FORM 1 JAN 64 1473

Unclassified
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Queues Computers Markov processes Control Optimization						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.