

STUDY OF REPRODUCIBILITY OF SIMULATION STUDIES AND
THE APPLICABILITY OF COST CURVES IN SCHEDULING

A Thesis

Presented to

the Faculty of Graduate Studies and Research

The University of Manitoba

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in the Institute for Computer Studies

by

John M. Wren

May 1969

ABSTRACT

A simulation model of a finite capacity computer system with variable time-slice, round-robin scheduling is used to study the reproducibility of the results of simulation studies where the job-stream is selected by Monte Carlo methods. A measure of performance based upon the cost of delay, or penalty, attributed to the job is used. Job-streams containing one, two and three priority classes of jobs and of various lengths are examined. The number of jobs which must be processed in a single run and the reproducibility of the measure of performance are determined for one, two and three priority job-streams.

The applicability of penalty curves based on step functions to job scheduling is examined by simulation methods under conditions of varying system load. Three penalty curves are examined for single and multi-priority job-streams. The changes in the ratio of time in the system to the time requested by a job due to varying the penalty curves are found to be not greater than about 10%, whereas the changes due to changing the queue selection algorithm are found to be in the range of 50%.

ACKNOWLEDGEMENTS

I would like to express my sincerest thanks to my thesis supervisor, Professor T.A. Rourke for his invaluable assistance during the course of this research.

I wish to thank Professors S.R. Clark and M. Hamid for their helpful comments and criticisms and for their time spent in reading this thesis.

I would also like to thank the operations personnel at The University of Manitoba Institute for Computer Studies for the assistance and cooperation they gave to me while I was running the simulation model.

The financial assistance given during the course of this research by the National Research Council of Canada is also gratefully acknowledged.

TABLE OF CONTENTS

CHAPTER I - INTRODUCTION.	1
CHAPTER II - DESCRIPTION OF THE SIMULATION MODEL.	4
2.1 INTRODUCTION.	4
2.2 THE JOB-STREAM.	5
2.3 PERFORMANCE MEASURES.	10
2.4 OPERATION OF THE SIMULATION MODEL	14
2.5 SERVICE TO THE USER AND THE ALLOCATION OF CENTRAL PROCESSOR TIME	20
CHAPTER III - AN INVESTIGATION OF THE REPRODUCIBILITY OF THE RESULTS OF SIMULATION STUDIES OF COMPUTER SYSTEMS . . .	22
3.1 INTRODUCTION.	22
3.2 THE JOB-SCHEDULING ALGORITHM.	23
3.3 NOMENCLATURE.	24
3.4 DERIVATION OF THE CONFIDENCE INTERVAL OF P_n	24
3.5 EXPERIMENTATION TO DETERMINE THE REPRODUCIBILITY OF THE MEASURE OF PERFORMANCE	26
3.6 CONCLUSIONS	35
CHAPTER IV - AN INVESTIGATION OF THE EFFECTS OF VARYING THE PENALTY CURVES	38
4.1 INTRODUCTION.	38
4.2 THE CHOICE OF STEP FUNCTIONS.	41
4.3 THE EFFECT OF STEP FUNCTIONS ON THE ALLOCATION OF CENTRAL PROCESSOR TIME	42
4.4 THE SELECTION OF JOBS FROM THE WAITING QUEUE AND THE SERVICING OF JOBS IN THE EXECUTION LIST.	42
4.5 THE EFFECTS OF VARIATION OF PENALTY CURVES ON THE MEAN RELATIVE RESPONSE	43
4.6 THE EFFECTS OF PENALTY CURVE VARIATION UPON THE RELATIVE RESPONSE OF A PRIORITY CLASS OF JOBS.	47
4.7 THE EFFECTS OF VARIATION OF THE PENALTY CURVE IN A MULTI-PRIORITY JOB-STREAM.	51
4.8 THE EFFECTS OF CHANGING THE QUEUE SELECTION ALGORITHM	62
4.9 THE VARIANCE OF THE RELATIVE RESPONSE FOR CHANGING PENALTY CURVES	66
4.10 CONCLUSIONS	69
REFERENCES.	72
APPENDIX A - MONTE CARLO TECHNIQUES USED.	73
APPENDIX B - SAMPLE JOB-STREAM DATA	76

TABLE OF CONTENTS (continued)

APPENDIX C - DATA MAINTAINED ON JOBS PROCESSED BY THE SIMULATION MODEL.	78
APPENDIX D - SAMPLE SERIES OF AUTOCORRELATED FINAL PENALTIES.	79
APPENDIX E - SAMPLE MEAN FINAL PENALTIES AND STANDARD DEVIATIONS.	81
APPENDIX F - SAMPLE DISTRIBUTION OF RELATIVE RESPONSES.	83
APPENDIX G - COMPARISON OF THE PERFORMANCE OF TWO EXECUTION LIST SERVICING ALGORITHMS	84
APPENDIX H - THE MEANS OF THE P_n VALUES	88
APPENDIX I - THE DISTRIBUTION OF CENTRAL PROCESSOR TIME REQUESTS.	91

LIST OF TABLES

TABLE 2.1 - SWITCH TABLE CONTAINING THE DESCRIPTION OF THE NEXT CYCLES.	19
TABLE 3.1 - JOB-MIX AND OPERATIONAL PARAMETERS USED FOR THE REPRODUCIBILITY STUDIES	27
TABLE 3.2 - CUMULATIVE FREQUENCY DISTRIBUTIONS OF THE g VALUES.	29
TABLE 3.3 - VALUES OF f FOR ONE, TWO AND THREE PRIORITY STREAMS	33
TABLE 3.4 - PERCENTAGE ERROR IN P_n FOR VARIOUS VALUES OF n AND S_n	36
TABLE 4.1 - JOB-MIX PARAMETERS AND OPERATIONAL PARAMETERS USED IN THE VARIATION OF PENALTY STUDY.	44
TABLE 4.2 - RELATIVE RESPONSES FOR SIMULATION RUNS FOR $L = 1, 2, 3$ WITH $\phi = 1.5, 2.0, 3.0$	48
TABLE 4.3 - $\bar{\omega}_j$ AND ω_j' FOR A TWO PRIORITY CLASS JOB-STREAM	50
TABLE 4.4 - $\bar{\omega}_j$ AND ω_j' FOR $L = 3$ $\phi = 1.5, 2.0, 3.0$	53
TABLE 4.5 - $\bar{\omega}$, ω' AND δ FOR $L = 2, 3$, $\phi = 1.5, 2.0, 3.0$	56
TABLE 4.6 - $\bar{\omega}_j$, ω_j' FOR $L = 2$, $\phi = 1.5, 2.0, 3.0$	59
TABLE 4.7 - $\bar{\omega}_j$, ω_j' FOR $L = 3$, $\phi = 1.5, 2.0, 3.0$	61
TABLE 4.8 - $\bar{\omega}$, ω' , $\bar{\omega}_j$ AND ω_j' FOR $L = 3$, $\phi = 1.5$ and 3.0	63
TABLE 4.9 - PERCENTAGE IMPROVEMENTS IN $\bar{\omega}_j$ VALUES FOR QUEUE SELECTION BY PRIORITY OVER QUEUE SELECTION BY PENALTY	67

LIST OF FIGURES

FIGURE 2.1 - THE JOB GENERATOR.	9
FIGURE 2.2 - PENALTY CURVES	13
FIGURE 2.3 - FLOWCHART OF THE SIMULATION MODEL.	16
FIGURE 2.4 - THE EVENT CONTROL ROUTINE.	18
FIGURE 3.1 - CUMULATIVE FREQUENCY POLYGONS $L = 1$	30
FIGURE 3.2 - CUMULATIVE FREQUENCY POLYGONS $L = 2$	31
FIGURE 3.3 - CUMULATIVE FREQUENCY POLYGONS $L = 3$	32
FIGURE 3.4 - f vs. SAMPLE SIZE.	34
FIGURE 4.1 - $\bar{\omega}$ AS A FUNCTION OF ρ $L=1,2,3$ $\phi = 1.5,2.0,3.0$	46
FIGURE 4.2 - $\bar{\omega}_j$ AS A FUNCTION OF ρ FOR $L = 2$ $\phi = 1.5,2.0,3.0$	49
FIGURE 4.3 - $\bar{\omega}_j$ AS A FUNCTION OF ρ $L = 3$ $\phi = 1.5,2.0,3.0$	52
FIGURE 4.4 - $\bar{\omega}$ AS A FUNCTION OF ρ $L = 2,3$ $\phi = 1.5,2.0,3.0$	55
FIGURE 4.5 - $\bar{\omega}_j$ AS A FUNCTION OF ρ FOR $L = 2$, $\phi = 1.5,2.0,3.0$	58
FIGURE 4.6 - $\bar{\omega}_j$ AS A FUNCTION OF ρ $L = 3$ $\phi = 1.5,2.0,3.0$	60
FIGURE 4.7 - $\bar{\omega}$ AS A FUNCTION OF ρ $L = 3$ FOR THREE COMBINATIONS OF QUEUE SELECTION ALGORITHMS AND PENALTY CURVES	64
FIGURE 4.8 - ω_1^i AS A FUNCTION OF ρ FOR THE TWO QUEUE SELECTION ALGORITHMS $L = 3$	68

NOMENCLATURE

- A_i - the arrival time of job i
 A_i^* - the arrival time plus the device time of job i
 A_{ij} - the arrival time of job i of priority class j
 $\alpha_i(t)$ - the current penalty of job i at time t
 β_i - the final penalty of job i
 δ - the reliability factor of $\bar{\omega}$
 f - the reproducibility factor
 $F(u_i(t), \gamma_i)$ - the penalty function of job i at time t with priority number γ_i
 γ_i - the priority number of job i
 j - the priority class of a job
 k - the proportionality constant
 L - the number of priority classes in a simulation run
 m_i - the CPU time request of job i
 \bar{m}_j - the mean CPU time request of priority class j
 μ - the mean of the population of β values
 n, N - the number of jobs in a simulation run
 n_e - the number of execution slots
 n_j - the number of jobs of priority class j
 n_q - the number of queue slots
 n_s - the number of jobs currently in store
 $p(t)$ - the current system penalty at time t
 P_n - the mean final penalty of a sample of final penalties of size n
 q_i - the quantum of time allocated to job i
 Q_i - the sum of the quanta received by job i

NOMENCLATURE (Continued)

- r_i - the interarrival time of job i
- \bar{r}_j - the mean interarrival time for jobs of priority class j
- ρ - the CPU utility or load
- t - the current time
- t_{di} - the device time of job i
- t_m - the minimum time-slice
- t_{rr} - the round-robin cycle time
- t_s - the supervisor cycle time
- T_i - the time of completion of job i
- ΔT_i - the current duration of job i
- τ_i - the sum the device time and the CPU time received by job i
- u_i - the lack of attention of job i
- u_i^* - the augmented lack of attention of job i
- ω_i - the relative response of job i
- $\bar{\omega}$ - the mean relative response of jobs for a simulation run
- ω' - the standard deviation of the ω_i values
- $\bar{\omega}_j$ - the mean relative response of jobs of priority class j
- ω'_j - the standard deviation of the ω_i values for jobs of priority j
- $u_i(t)$ - the current lack of attention of job i at time t

CHAPTER I

INTRODUCTION

Simulation studies has been used effectively to study computer systems [1,2], but they also have certain disadvantages and problems. For example, when there is no data available to serve as guidelines, such as in the simulation of hypothetical computer systems, the determination of when the model is working properly is often difficult. As the volume of literature relating to computer systems and job-scheduling increases, it becomes easier to detect gross errors in the simulation model, but minor errors remain a difficult problem. Another difficulty, not encountered in analytical studies is that of working backwards from observable trends to meaningful analytic relationships between the input and output parameters. The solution of the latter problem is not a vital part of any simulation study but may lead to a better understanding of a system.

To make analytic studies tractable, simplifying assumptions are usually made about the situation being modelled. These assumptions include an infinite capacity machine and no separate queue of waiting jobs. By way of contrast, the simulation model used in this research is of finite capacity and maintains a waiting job queue as well as a list of executing programs.

Analytical studies also deal with the population values of the input and output parameters and thus are independent of the number of jobs. In contrast to analytic methods and deterministic simulation methods, the results of simulation studies where the job-stream is defined by Monte Carlo techniques are dependent upon the size of the job-stream. Therefore, another problem encountered in simulation studies where the job-stream is selected

by Monte Carlo techniques is the determination of the reproducibility of the results when a different pseudo-random series is used to produce the job-stream. Further it is often desirable to determine the number of jobs which should be simulated in order to produce a particular accuracy in the results of a simulation run. In order to do the latter, it is necessary that the relationship between reproducibility and the number of jobs simulated in a single run be known.

The relationship between the reproducibility of the results and the number of jobs processed through the model on each simulation run may be derived analytically when the job statistics are stochastically independent and normally distributed. However, the nature of the system being simulated may give rise to job statistics which are autocorrelated since at times when the system is temporarily heavily loaded, a group of consecutive jobs are likely to experience delays. Thus it is reasonable to expect a deviation from the relationship appropriate to a series of stochastically independent job statistics, and in this study such deviations are determined in an empirical manner, resulting in an overall semi-empirical approach.

The first part of the research described herein deals with the determination of a semi-empirical relationship between the variation of the reproducibility of simulation study results and the number of jobs simulated in a single run of a simulation model. Two benefits arise from the determination of such a relationship. The first is to be able to determine the accuracy of a result from the details of the simulation run itself, without having to perform repetitive calculations. The second benefit is to be able to determine the number of jobs required in a simulation run to produce a particular percentage accuracy in a result after performing only a pilot calculation.

The studies described in this thesis use a computer simulation model

that was designed by Dr. T.A. Rourke, implemented by Rourke and Chai and used by Chai [3] to investigate time-slicing algorithms. The model is that of a hypothetical computer system which incurs no overheads to load jobs into core, and which processes jobs which require no input or output operations during their execution. The main store is large enough to accommodate up to three jobs simultaneously and jobs unable to be accommodated in core are held in a waiting queue.

As part of the supervisor system of a time-shared computer system, there is a job-scheduling algorithm which determines which job to service next and how much time to allocate to each job in the main store. To make these decisions, some criteria or knowledge about the user's requests or jobs is necessary. It has been suggested by Greenberger [4], that penalty curves based upon the delay experienced by a job may be a useful criteria. These curves can be made to serve in a number of ways: for the allocation of central processor time, the measurement of system performance, and the assigning of discounts to the user.

When penalty curves are used for job-scheduling, the question arises as to the effects produced upon the response of the system by changes in the shape or values assigned to the penalty curve. Step functions, a simple form of cost curve also suggested by Greenberger are used in the second part of this study to investigate the effects of changing the penalty curve upon the response of the system and the distribution of these responses. It is hoped that the results obtained from scheduling by penalties of the type used can serve as an indicator of the effectiveness of penalty curves in scheduling under the particular circumstances.

CHAPTER II

DESCRIPTION OF THE SIMULATION MODEL

2.1 INTRODUCTION

The simulation model, which was used previously by Chai [3], has been designed to simulate a job scheduling algorithm operating in a computer system. The loading of jobs into the system, the removal of jobs from the system and the input-output operations of the simulated jobs are ignored. This model is the nucleus of a more comprehensive model that will simulate job loading and removal as well as input-output operations for the simulated jobs.

The model concentrates on an algorithm which is primarily concerned with the allocation of a single central processor unit (CPU) and a main store of finite capacity; in particular a maximum of three jobs may be resident and scheduled by the algorithm at any given time. The central processor is scheduled by means of time-slicing techniques.

Since a complete description of the model is given by Chai, only a general outline of the model is presented here with emphasis on those portions of the model which are important to the research under discussion. The description of the model is presented in the following three sections:

- (a) Job Generation and the Job-Stream: in which the input or job-stream parameters, job generation techniques, and the resultant job-stream are described;
- (b) Performance Measurements: in which the output of the simulation run and the criteria used to measure model performance are described;
- (c) Model Operation: in which is described how jobs are processed by the simulation model.

2.2 THE JOB STREAM

This section describes the characteristics of the job generator and of the jobs produced by it. The job generator uses Monte Carlo techniques to produce a series of n individual entities called jobs which compete for the storage and the central processor unit of the simulated computer system. This series of jobs comprises the job-stream to be processed during the simulation run. Each job in the job-stream is defined in terms of four variables. For the i^{th} job in the job-stream, the four defining variables are:

- (1) the priority number of the job denoted γ_i ;
- (2) the execution time or the central processor time requested by the job denoted by m_i ;
- (3) the arrival time of the job, A_i ;
- (4) the assumed device time of the job, t_{di} . This value is fixed, and the same for all jobs in the job stream, throughout the run.

A priority number is assigned to each job generated. This priority number designates the priority class, j , to which the job belongs and indicates the relative importance of the job. The number of priority classes is L , and the priority of job i is denoted by γ_i . Thus, if job i has a priority number, γ_i , which is equal to j , it is said to belong to priority class j . The highest priority jobs, or those jobs of the greatest relative importance are jobs belonging to priority class 1 and the jobs of the least relative importance, that is, lowest priority, are those jobs belonging to priority class L .

The execution time or central processor time requested, m_i , by job i of priority class j is taken to be normally distributed about the mean execution time for that priority class, \bar{m}_j , with a standard deviation of $\bar{m}_j/4$.

The job-stream generated is structured so that the mean of the CPU request times for each priority class is linked to the mean central processor request time of priority class 1. Linkage is by means of a proportionality constant, k , which is established at the beginning of each simulation run. For a job-stream consisting of three priority classes, that is $L = 3$,

$$\bar{m}_2 = k \bar{m}_1 , \quad \dots(2.2.1)$$

$$\bar{m}_3 = k \bar{m}_2 , \quad \dots(2.2.2)$$

and in general,

$$\bar{m}_j = k \bar{m}_{j-1} . \quad \dots(2.2.3)$$

The arrival time, A_{ij} , of the i^{th} job of priority class j is taken to be the sum of the arrival time of the last job of the priority class and the interarrival time of the job i , denoted r_{ij} , thus $A_{ij} = A_{i-1,j} + r_{ij}$. The interarrival time, r_{ij} , of a job i is determined from \bar{r}_j , the mean interarrival time of priority class j . The distribution of interarrival times is assumed to be exponential [5].

It is another property of the job stream that the mean interarrival times of the different priority classes are linked by the proportionality constant k . For a job-stream consisting of jobs from three priority classes:

$$\bar{r}_2 = k \bar{r}_1 , \quad \dots(2.2.4)$$

$$\bar{r}_3 = k \bar{r}_2 , \quad \dots(2.2.5)$$

and in general,

$$\bar{r}_j = k \bar{r}_{j-1} . \quad \dots(2.2.6)$$

The arrival time of the first job in each priority class, j , is arbitrarily determined as $A_{1j} = r_{1j}/2$.

The assumed device time, t_{di} , of job i is used to represent the time spent in transferring the job into a waiting area within the computer system and its presence avoids a singularity in some of the calculations made. The device time of a job is simulated after the arrival time of the job but

before the job is made ready for execution. Thus the earliest time that a job i of priority class j may begin execution is given by:

$$A_{ij}^* = A_{ij} + t_{di} . \quad \dots(2.2.7)$$

The total number of jobs in a given run of the simulation model is n . The mean number of jobs in any priority class, j , is determined from n , the total number of jobs, L , the number of priority classes and k , the proportionality constant. For a job-stream with three priority classes ($L = 3$):

$$n_1 = k n_2 , \quad \dots(2.2.8)$$

$$n_2 = k n_3 , \quad \dots(2.2.9)$$

where n_j is the mean number of jobs with priority number, γ , equal to j .

The total number of jobs is n , thus,

$$n = n_1 + n_2 + n_3 , \quad \dots(2.2.10)$$

or

$$n = (k^2 + k + 1) n_3 , \quad \dots(2.2.11)$$

and

$$n_1 = \frac{k^2 n}{k^2 + k + 1} , \quad \dots(2.2.12)$$

$$n_2 = \frac{k n}{k^2 + k + 1} , \quad \dots(2.2.13)$$

$$n_3 = \frac{n}{k^2 + k + 1} \quad \dots(2.2.14)$$

where k is the proportionality constant.

In summary, the job-mix parameters that are input to the simulation model at the beginning of each simulation run and serve to define the job-stream are:

n , the total number of jobs to be processed during the simulation run;

L , the number of priority classes to be generated;

\bar{r}_1 , the mean interarrival time of jobs of priority class 1;
 \bar{m}_1 , the mean central processor time to be requested by jobs of
 priority class 1;

t_{di} , the assumed device time of all jobs in the job-stream.

The normal and Poisson generating functions used to generate the CPU request times and the interarrival times, respectively, are described in Appendix A.

From the job-mix parameters, the i^{th} job of the job-stream is defined in terms of:

m_i , the central processor time requested;

A_i , the arrival time;

γ_i , the priority number;

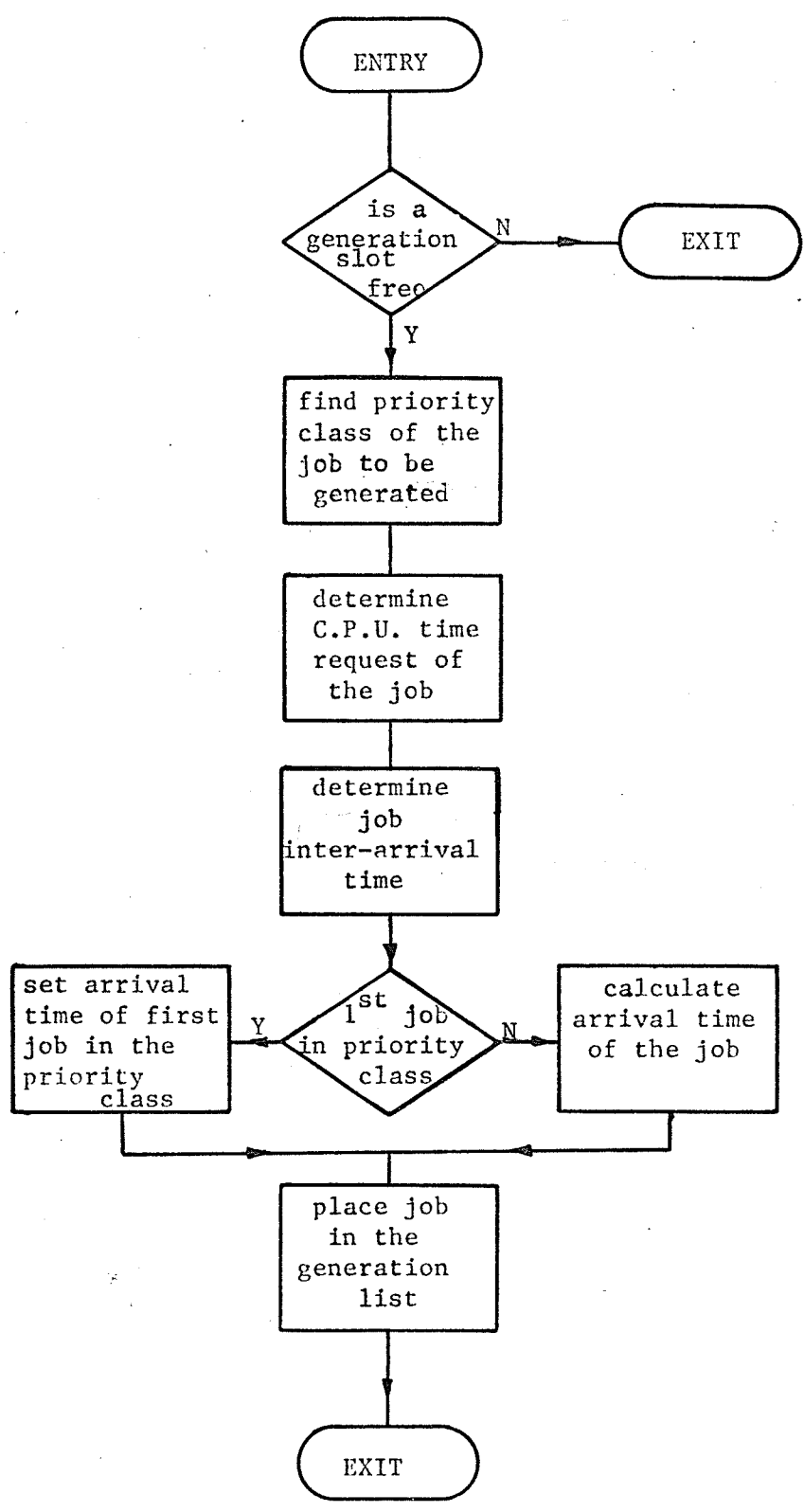
t_{di} , the assumed device time.

A sample job-stream is shown in Appendix B.

The job-stream generator is that section of the simulation model logic that uses the job-mix parameters and the normal and Poisson generating functions to generate the individual jobs that make up the job-stream. These jobs are then made available to other sections of the simulation model logic as required. A flow chart outlining the operation of the job generation is given in Figure 2.1.

Jobs are generated sequentially by priority class. A generation list is maintained with one entry defined for each priority class that is specified. Thus, if the number of priority classes defined is three, there will be three entries in the generation list. This list contains the next job available for each priority class. When the simulated time is equal to the time, A_{ij}^* , of any job in the generation list, that job is removed from the generation list and placed in either the list of jobs being allocated CPU

FIGURE 2.1 - THE JOB GENERATOR



time, (the execution list), or if the execution list is full, the waiting queue which contains those jobs requesting CPU facilities but unable to obtain them. The job generator is then invoked to fill the empty slot in the generation list.

Under certain conditions, notably when a heavy load is placed upon the simulated system, it is possible that both the execution list and the waiting queue will be filled to capacity. In these circumstances, when the simulated time is equal to that of any job in the generation list, the job is left in the generation list until a slot is available in one of the other lists.

When n jobs have been processed through the system, the simulation run is terminated.

2.3 PERFORMANCE MEASURES

In order to evaluate the performance of the model and thus the performance of the job scheduling algorithm used, it is necessary to specify some measure of performance. The measure of performance used in this research is based upon the penalty or cost of the delay [4] accumulated by a job. The use of cost of delay as a function of real-time does not permit any mechanism whereby the penalty may be reduced during the course of a job's execution; the best that can be done is to prevent the penalty from increasing too much. Therefore, the penalty of a job has been made a function of the ratio of the total time elapsed since the job arrived to the sum of its device time plus its accumulated CPU time. This ratio is termed the job's lack of attention. Since the lack of attention may increase or decrease with real-time, so may the penalty.

The duration of any job i at a time t during its execution phase is taken to be the total time elapsed between t and the job's arrival time, A_i , and is given by:

$$\Delta T_i = t - A_i \quad \dots(2.3.1)$$

The time devoted to a job i , τ_i , is defined as the sum of its device

time plus the current sum Q_i of the quanta received by the job. Thus,

$$\tau_i = Q_i + t_{d_i} . \quad \dots(2.3.2)$$

The lack of attention of any job i at a time t is given by:

$$u_i(t) = \frac{\Delta T_i}{\tau_i} . \quad \dots(2.3.3)$$

Both t and Q_i increase with real time but not necessarily by the same amount. Thus $u_i(t)$ may increase or decrease with the passage of real time.

Immediately after loading job i and before any CPU time is given to job i , that is $\tau_i = t_{d_i}$ and $t = A_i^*$, the lack of attention is:

$$u_i(t) = \frac{A_i^* - A_i}{t_{d_i}} = 1 \quad \dots(2.3.4)$$

The earliest possible time that a job i may begin execution is A_i^* .

The minimum time a job will be on the execution list is the job's requested CPU time, m_i . Thus the earliest time, T_i , that job i may complete processing is given by:

$$T_i = A_i^* + m_i , \quad \dots(2.3.5)$$

and at job completion time, the duration of job i is given by:

$$\Delta T_i = m_i + t_{d_i} . \quad \dots(2.3.6)$$

Similarly, the minimum time devoted to a job to process it to completion is given by:

$$\tau_i = m_i + t_{d_i} \quad \dots(2.3.7)$$

and thus the minimum final lack of attention for a job i is:

$$u_i = \frac{\Delta T_{i_{\min}}}{\tau_{i_{\min}}} = \frac{m_i + t_{d_i}}{m_i + t_{d_i}} = 1 . \quad \dots(2.3.8)$$

Since the lack of attention increases or decreases with real time, the penalties as functions of lack of attention permit the use of the penalty curves directly in the scheduling algorithm. In the study by Chai and in this research, the penalty α_i , of any job i is taken to be a function of its lack of attention, $u_i(t)$, and its priority γ_i .

The penalty functions selected for use in this research are shown in Figure 2.2. The curves are arbitrary in that they do not correspond to the cost of delay in a real environment. The curves, however are realistic in the sense that they are of the type that one would expect to encounter in a real environment. Curve (a) would appear to be applicable to high priority jobs requiring fast servicing since the penalty values increase rapidly for low values of u . Similarly, for low priority jobs, curve (c) would be suitable since the penalty values do not rise rapidly until the value of u is quite large.

Each priority class of jobs has associated with it a penalty curve, the ordinate of which reflects the attention required by the job. The current penalty of any job i , in the execution phase at time t is given by:

$$\alpha_i(t) = F(u_i(t), \gamma_i), \quad \dots(2.3.6)$$

where $u_i(t)$ is the current lack of attention and γ_i is the job's priority number. The total operating penalty of the system, $p(t)$, at time t is taken as the sum of the current penalties of all jobs in the execution list at time t and is given by:

$$p(t) = \sum_{i=1}^{n_s} \alpha_i(t) = \sum_{i=1}^{n_s} F(u_i(t), \gamma_i), \quad \dots(2.3.7)$$

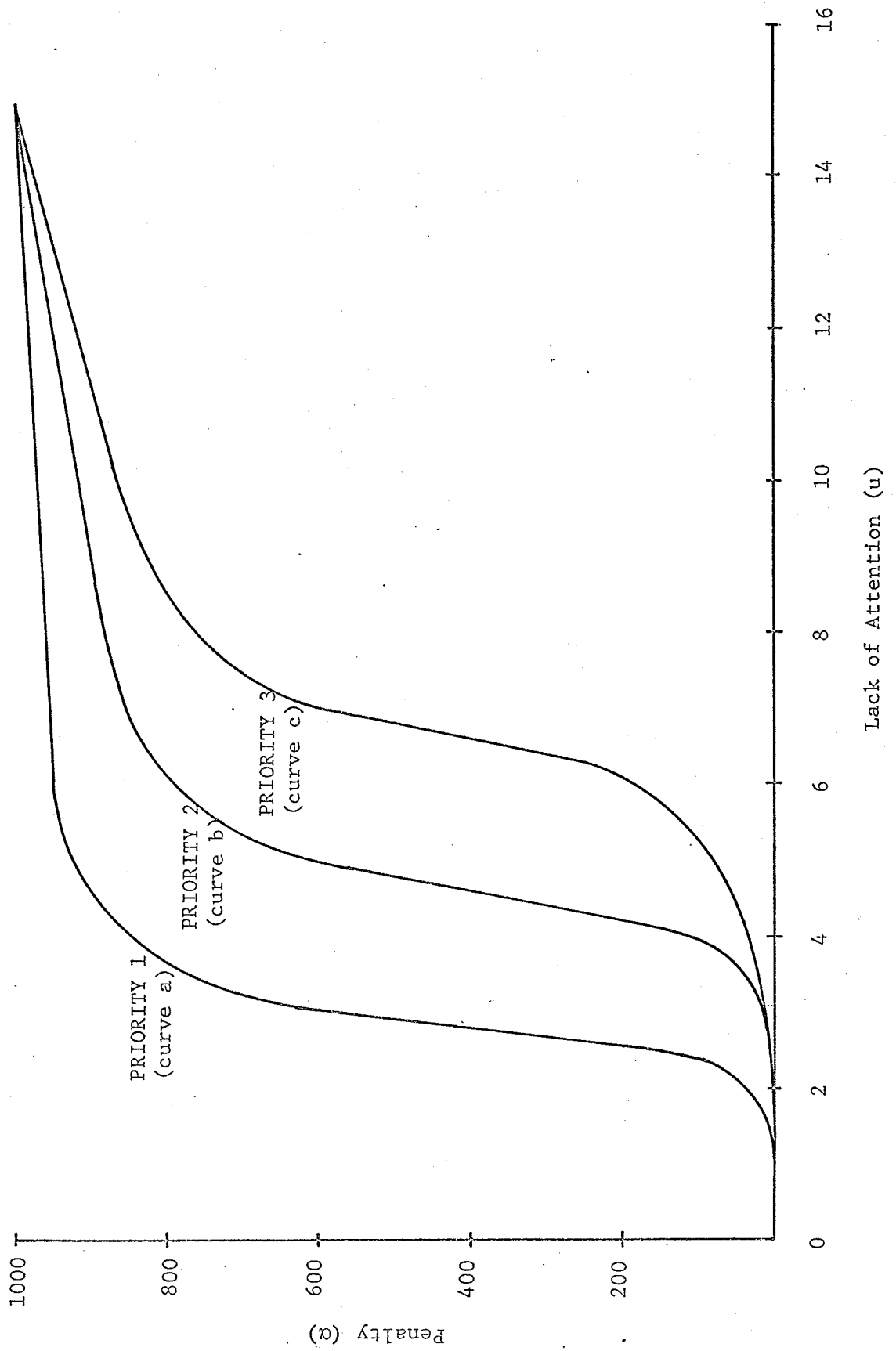
where n_s is the number of jobs in the execution list.

The criteria used for model evaluation is the mean final penalty, P_n , and the n jobs processed through the system during a simulation run. Thus, if β_i is the final penalty of any job i , then the mean final penalty for n jobs is given by:

$$P_n = \frac{1}{n} \sum_{i=1}^n \beta_i. \quad \dots(2.3.8)$$

The minimum final lack of attention that any job i may have has been shown to be equal to one. The functions $F(u_i, \gamma_i)$ for all γ are defined so that $F(1, \gamma_i)$ is equal to one. Thus, if the final penalties of all the n jobs in a simulation run are equal to one, then $P_n = 1$. A mean final penalty of one will be obtained only if every job in the simulation run is processed to completion

FIGURE 2.2 - PENALTY CURVES



without interruption, immediately upon entry into the system and there are no system overheads. P_n equal to one represents optimal service to the users and higher values of P_n reflect poorer service.

The mean final penalty, P_n , is one measure of performance of the simulation model. The calculation of P_n for different algorithms, using the same job-stream allows a simple comparison to be made between the different algorithms. A single penalty curve for all priority classes, which is linearly related to the lack of attention, u , gives rise to a measure of performance, P_n , which is very similar to the performance measure used by Fife [6].

2.4 OPERATION OF THE SIMULATION MODEL

The simulation model processes the variable job-stream produced by the job-stream generator described in the previous section. This section describes how the jobs are processed. Three lists are maintained by the simulation model: the generation list, the execution list and the waiting queue list.

The generation list is a list of the next jobs by priority class and is filled by the job generator. When the simulated time is equal to the arrival time, A_i^* , for job i on the generation list of priority class j , that job is transferred to one of the two other lists.

The execution list has n_e slots ($n_e \leq 3$) and contains those jobs which are being executed, that is, receiving CPU time. No consideration is made of a job's size when it is assigned a slot in the execution list. Thus a job will be placed in the first slot that is available in the execution list. The model operates without roll-in, roll-out, so that once a job has been assigned a slot in the execution list, it will remain there until it has received CPU time equal to that requested by the job. The job is then removed and the execution list slot made available to another job. The simulated system is of finite capacity in the sense that there can be a maximum of three jobs receiving CPU time.

The third list is the waiting queue which has n_q slots. Job are placed in this list when the simulated time has reached their arrival time but there is no slot available for the job in the execution list. As jobs in the execution list are completed and the execution slots are freed, the waiting queue is scanned for jobs awaiting execution. If such a job is found, it is transferred to the execution list from the waiting queue, and the waiting queue slot is then available for another job which cannot be accepted onto the execution list. The transfer of a job from the waiting queue to the execution list is considered to be accomplished with no overhead cost in terms of time, that is, it is instantaneous.

The series of transfers from the generation list to the execution list or the waiting queue, and from the waiting queue to the execution list continue until n jobs have been processed to completion. When the n jobs have been processed through the system, the simulation run is terminated. The generation list, the execution list and the waiting queue are not drained. Each of the 3 lists has attached to it a number of sub-lists containing the variables which describe each job and the data generated by the processing of the job.

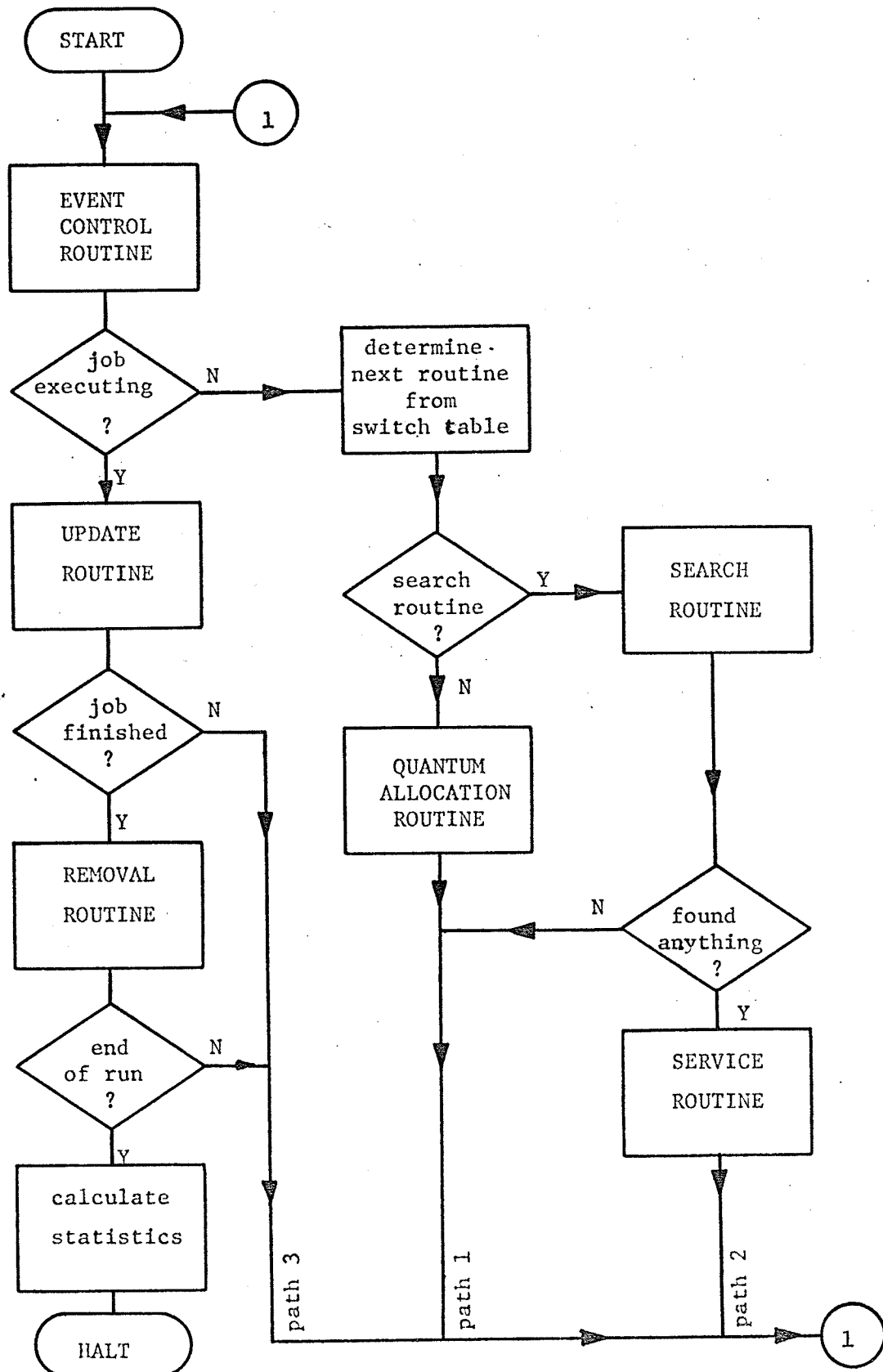
Appendix C describes the data maintained in each list for each job.

A general flow chart of the simulation model is presented in Figure 2.3. The model can be seen to be made up of the following routines which are described below.

(1) The event control routine contains the job generator, logic for the selection and disposition of jobs into either the execution list or the waiting queue and advances the time to the time of the next event which is the lesser of:

- (a) the arrival time of the earliest job which can be accepted onto the execution list;
- (b) the time when a time-slice is completed or the time when a job has

FIGURE 2.3 - FLOWCHART OF THE SIMULATION MODEL



completed processing before the end of its allotted time-slice;

- (c) the current time if no job is executing and if a useful operation could be performed on the next supervisor cycle.

The event control routine also handles the supervisor cycle time and adds it to the current time. The supervisor cycle time is the overhead cost due to the supervisor and represents the time spent by the supervisor in the selection of jobs, the placing of jobs in the waiting queue, the allocation of time-slices and for general housekeeping operations. Chai considered that a supervisor cycle time of 0.01 seconds, representing the execution time of about 3000 instructions on an IBM 360/65 was a realistic figure and this has been adopted in this research. A flow chart of the event control routine is presented in Figure 2.4.

(2) The switching routine determines which of the two primary routines, the quantum-allocation routine or the search routine, is to be executed next. The selection made is dependent on which of the primary routines was executed most recently and which one of the three exit paths was used. The switch table used to determine which routine to enter next is presented in Table 2.1. Vacant entries occur because certain combinations of last routine and exit path are not possible. For example, if the previous cycle involved the search and service routines (path 2, Figure 2.3) then upon leaving the event control routine, the next cycle would involve path 3 which includes the update routine. Therefore, it is not possible to have 'search' as the most recent routine and path 2 as the most recent path taken.

(3) The quantum-allocation routine determines the amount of central processor time to be allocated to each job in the execution list for the next round-robin cycle.

(4) The search routine locates a job in the execution list with an outstanding

FIGURE 2.4 - THE EVENT CONTROL ROUTINE

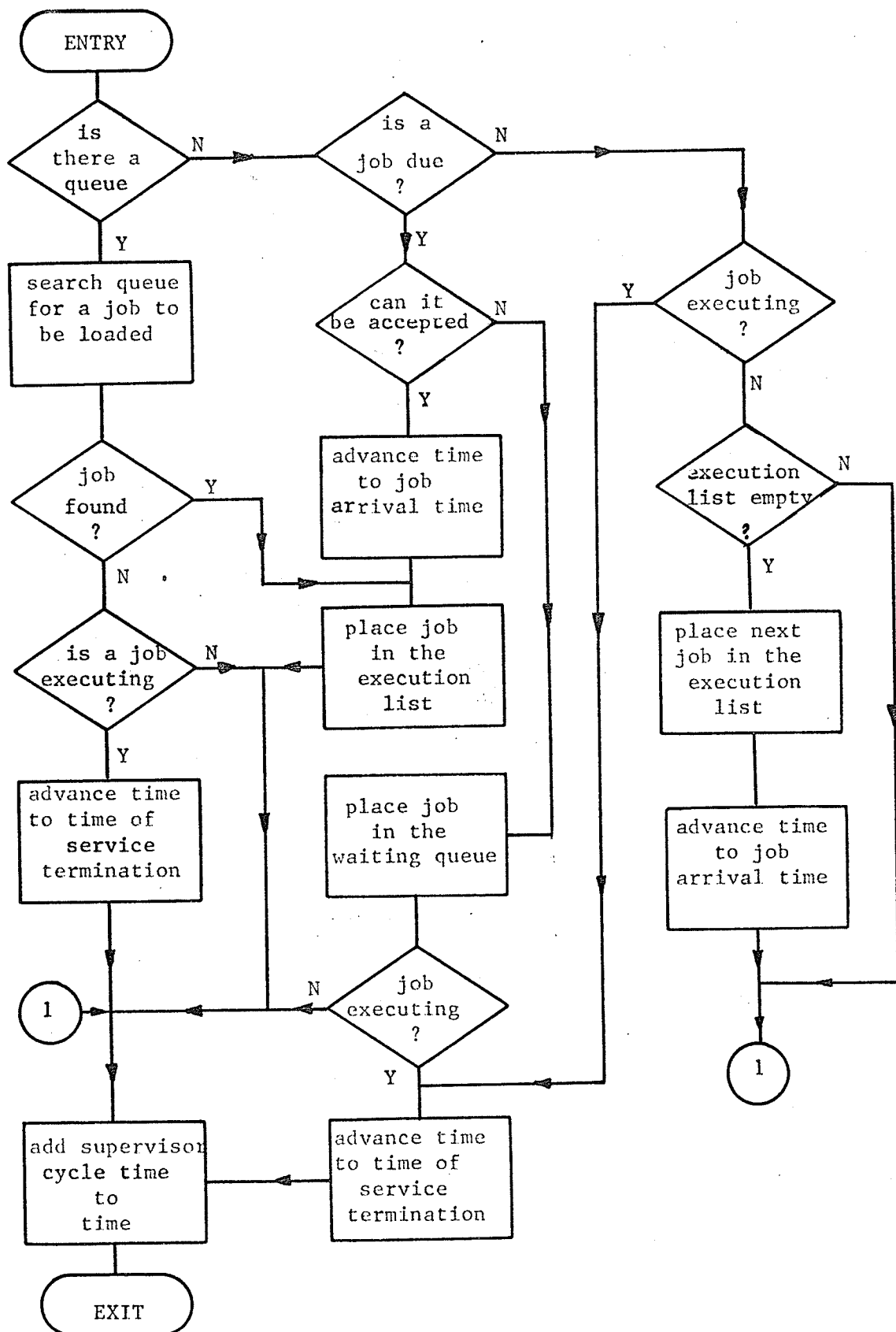


TABLE 2.1

SWITCH TABLE CONTAINING THE DESCRIPTION OF THE NEXT CYCLES

LAST PRIMARY ROUTINE EXECUTED	PATH NUMBER FOR THE PREVIOUS CYCLE		
	1	2	3
SEARCH	QUANTUM ALLOCATION	-	SEARCH
QUANTUM ALLOCATION	SEARCH	-	-

time-slice, and if it finds one, marks it for servicing by the service routine.

(5) The service routine initiates the time-slice of the job located by the search routine.

(6) The update routine records the current accumulated CPU time of a job after it has received all or part of its allocated time-slice, and if the job is finished, marks it for removal.

(7) The removal routine removes the job marked by the update routine and records the time of completion, and the final penalty attributed to a job.

2.5 SERVICE TO THE USER AND THE ALLOCATION OF CENTRAL PROCESSOR TIME

The quantum-allocation routine allocates a quantum of central processor time to each job in the execution list on a round-robin basis by means of time-slicing techniques [7]. The round-robin time, t_{rr} , is defined such that each job in the execution list receives some central processor time every t_{rr} seconds, approximately. The sum of the allotted quanta is normally equal to t_{rr} , however, supervisor overheads may be incurred during a round-robin cycle so that the actual cycle time is longer than the intended cycle time. Also, jobs may reach completion before the end of their allocated time-slice so that the sum of the time-slices received is less than the intended round-robin time.

The quantum-allocation routine is entered before the start of every round-robin cycle and calculates time-slices (quanta) for all jobs in the execution list according to a variable time-slice algorithm. For the purpose of the quantum allocation routine u_i , α_i and $p(t)$ have been redefined as u_i^* , α_i^* and $p^*(t)$ where

$$u_i^* = \frac{\Delta T_i + 1}{\tau_i} \quad \dots(2.5.1)$$

$$\alpha_i^*(t) = F(u_i^*(t), \gamma_i) \quad \dots(2.5.2)$$

and

$$p^*(t) = \sum_{i=1}^{n_s} \alpha_i^*(t) \quad \dots(2.5.3)$$

The effect of introducing the constant 1 in the numerator of equation (2.5.1) will be to give a larger time-slice to each job at the beginning of its execution phase.

The time-slice, q_i , allocated to job i in the execution list at time t is given by:

$$q_i = \frac{\alpha_i^*(t)}{p^*(t)} t_{rr} \quad \dots(2.5.4)$$

When the time-slice q_i of job i is less than some predetermined minimum value, then q_i is set equal to the minimum value. The minimum time-slice, t_m , is used to guard against the possibility of the assigned time-slice being similar to, or even less than, the overhead incurred by the supervisor program in administering the time-slice [7]. In this research, the minimum time-slice is taken to be one one-hundredth of the round-robin time, that is, $t_m = 0.01 t_{rr}$.

In summary, the performance measure used is the mean final penalty, given by:

$$P_n = \frac{1}{n} \sum_{i=1}^n \beta_i,$$

where β_i is the final penalty of job i .

The operational parameters of the simulation model used to control model operation are:

- n_e , the number of execution list slots;
- n_q , the number of slots in the waiting queue;
- t_{rr} , the round-robin cycle time;
- t_m , the minimum time-slice.

CHAPTER III

AN INVESTIGATION OF THE REPRODUCIBILITY OF THE RESULTS OF SIMULATION STUDIES OF COMPUTER SYSTEMS[†]

3.1 INTRODUCTION

Simulation studies of the behaviour of computer systems of moderate complexity have certain advantages over analytic methods. For example, the scheduling of jobs in a time-sharing computer system with non-linear penalty curves was considered by Greenberger [4] to be better approached by simulation methods rather than analytically, since the expected-value arguments used in analytic methods did not hold for non-linear cost curves. However, simulation studies have their own limitations and drawbacks [8]. These include the time and cost of developing and testing the simulation model, the actual machine time needed to produce reliable statistics, the reproducibility of the results when different pseudo-random series are used to produce the job-stream, and the determination of meaningful empirical relationships between the variables. For the latter problem it is particularly important that the reproducibility of simulation results be known. Also, since it is possible in simulation studies to select the degree of reproducibility required in the results of the simulation run, it is advantageous to know the number of jobs which must be simulated to produce the level of accuracy selected as being tolerable for a particular simulation run. To achieve this, it is necessary to know the relationship between reproducibility and the number of jobs in the simulation run.

The measure of performance, P_n , selected in this research has been described in Chapter II; this chapter describes the investigations made

[†]The work carried out for this chapter is to be published separately [9] and the description is taken partly from this material.

as to the reproducibility of this measure of performance and the relationship between P_n and n , the number of jobs in the job-stream. Repeated simulation runs for various sizes of job-stream under identical conditions but using different random series as input to the Monte Carlo type job generator, are used to study the reproducibility of P_n and its relationship to n .

3.2 THE JOB SCHEDULING ALGORITHM

The job scheduling algorithm adopted for the reproducibility study is a combination of queue selection and quantum allocation logic which determine which job is to be transferred next to the execution list and what portion of the round-robin cycle time is to be allocated to each processing job.

The maximum number of jobs permitted in the execution list and the waiting queue is three and twenty, respectively. Because there is likely to be more than one job in the execution list, it is necessary that an algorithm be defined for selecting which job to service next. The algorithm used for the selection of jobs for servicing in the execution list was, round-robin service to jobs in the order in which they are located in the execution list. Thus when scanning the execution list in the direction of increasing slot numbers, the first job found with a quantum of time assigned to it would be given control of the processing unit. The quantum of time assigned to each job in the execution list was proportional to the job's current penalty and determined by means of equation (2.5.4). Jobs in the waiting queue were selected for transfer to the execution list on the basis of first-come-first-served. Thus for two jobs in slots i and j of the waiting queue, the job in slot i would be selected for transfer to the execution list if its arrival time were less than the arrival time of the job in slot j . In the event of

there being two identical earliest arrival times, the job with the lowest queue slot number would be selected for transfer, although this eventuality is extremely rare since the arrival times are recorded as real numbers with approximately seven digit precision.

3.3 NOMENCLATURE

The nomenclature presented here is used in the subsequent discussion of reproducibility and simulation run size, and is as follows:

- $\sigma[P_n]$, the population standard deviation of P_n ;
 $\sigma[\beta]$, the population standard deviation of the β values;
 $\sigma[S_n]$, the population standard deviation of the S_n values;
 S_n , the standard deviation of β_i given by,
- $$S_n = \left\{ \sum_{i=1}^n (\beta_i - P_n)^2 / (n-1) \right\}^{1/2};$$
- n , the sample size;
 β_i , the i^{th} final penalty;
 P_n , the mean final penalty for a sample of final penalties of size n ;
 μ , the population mean value of the β_i values.

3.4 DERIVATION OF THE CONFIDENCE INTERVAL OF P_n

Repeated runs of the simulation model varying only the initial random number gives rise to a series of values of P_n and S_n . Fluctuations in P_n and in S_n are related to $\sigma[P_n]$ and $\sigma[S_n]$ which are in turn related to $\sigma[\beta]$ and n .

For a normal distribution of β values, the population standard deviation, $\sigma[P_n]$, for samples of size n is related to the population standard deviation of the β values, $\sigma[\beta]$ by the equation [10]:

$$\sigma[P_n] = \sigma[\beta] / \sqrt{n} \quad \dots(3.4.1)$$

and $\sigma[S_n]$ is related to $\sigma[\beta]$ according to the equation [10]:

$$\sigma[S_n] = \left\{ 1 - \frac{1}{8n} - \frac{25}{128n^2} + \dots \right\} \frac{\sigma[\beta]}{\sqrt{2n}} \quad \dots(3.4.2)$$

When n is greater than about 10, the equation reduces to:

$$\sigma[S_n] = \sigma[\beta]/\sqrt{2n} \quad \dots(3.4.3)$$

and the population of S_n values is approximately normally distributed [9].

The range about S_n within which $\sigma[\beta]$ lies with 95% certainty, is given by:

$$\sigma[\beta] = S_n \pm 2 \sigma[S_n] \quad \dots(3.4.4)$$

and from equation (3.4.3):

$$\sigma[\beta] = S_n \pm 2 \sigma[\beta]/\sqrt{2n} \quad \dots(3.4.5)$$

hence the range about S_n within which $\sigma[\beta]$ lies within 95% certainty is given by:

$$\sigma[\beta] = S_n (1 \pm \sqrt{2}/\sqrt{n})^{-1} \quad \dots(3.4.6)$$

With slightly more than 95% certainty then,

$$\sigma[\beta] < S_n (1 - \sqrt{2}/\sqrt{n})^{-1} \quad \dots(3.4.7)$$

Substituting in equation (3.4.1) for $\sigma[\beta]$ gives:

$$\sigma[P_n] < S_n (1 - \sqrt{2}/\sqrt{n})^{-1}/\sqrt{n} \quad \dots(3.4.8)$$

For a normal distribution, the range within which μ lies with approximately 95% certainty is given by:

$$\mu = P_n \pm 2 \sigma[P_n] \quad \dots(3.4.9)$$

Substitution of the expression obtained in equation (3.4.8) for $\sigma[P_n]$ into equation (3.4.9) gives the 95% range of μ in terms of P_n , S_n and n as,

$$\mu = P_n \pm 2S_n (\sqrt{n} - \sqrt{2})^{-1} \quad \dots(3.4.10)$$

An assumption made for the derivation of equation (3.4.10) is that the β values used to obtain P_n and S_n are stochastically independent and do not display autocorrelation.

The assumption of stochastic independence of the β values is not correct under certain conditions. These conditions arise during the operation

of the simulation model when temporary heavy loading is imposed by the stochastic selection of jobs. Heavy loading of the simulated system can cause penalty values to become large and a group of consecutive penalties may have higher than average values, resulting in a larger mean final penalty and standard deviation. Appendix D shows a sample of final penalties which exhibit this behaviour. The result of this behaviour is that the penalty values are not stochastically independent, normal variables and variations from equation (3.4.10) are to be expected. Appendix E shows a sample distribution of mean final penalties.

The deviation from normality may be expressed empirically in terms of a factor f where:

$$\mu = P_n \pm f \{2S_n (\sqrt{n} - \sqrt{2})^{-1}\}. \quad \dots(3.4.11)$$

In order to estimate the factor f for a particular job size, it is convenient to define a property g , of each estimate of P_n and S_n such that

$$g = \left| \mu - P_n \right| (\sqrt{n} - \sqrt{2}) / 2S_n. \quad \dots(3.4.12)$$

3.5 EXPERIMENTATION TO DETERMINE THE REPRODUCIBILITY OF THE MEASURE OF PERFORMANCE

In order to examine the reproducibility of P_n , one hundred and twenty simulation runs were performed for each of the n values, 50, 100, 200, 400, 600, 800, 1200 and 1600. Also, the runs were replicated for one, two and three priority job streams ($L = 1, 2, 3$). The job-mix parameters used in each run were identical (apart from n and L) and only the starting random number was varied from one run to the next. The fixed job-mix parameters and simulation model operational parameters are shown in Table 3.1. The total number of jobs simulated for each of the different L values ($L = 1, 2, 3$) was 594,000. The average final penalty, taken over the 594,000 jobs simulated for each stream was taken as, μ , the population mean value of β . The values

TABLE 3.1
 THE JOB-MIX AND OPERATIONAL PARAMETERS USED FOR
 THE REPRODUCIBILITY STUDIES

JOB-MIX PARAMETERS:

Proportionality Constant	k	=	4.0
Mean CPU Request	\bar{m}_1	=	1.0 seconds
Mean Interarrival Time	\bar{r}_1	=	6.0 seconds

OPERATIONAL PARAMETERS:

Size of the Execution List	n_e	=	3 jobs
Size of the Waiting Queue	n_q	=	20 jobs
Round-Robin Cycle Time	t_{rr}	=	6.0 seconds
Minimum Time-Slice	t_m	=	0.06 seconds
Supervisor Cycle Time	t_s	=	0.01 seconds
Constant Device Time	t_d	=	1.0 seconds

of μ were found to be 4.65, 48.35 and 134.73 for L equal to 1, 2 and 3, respectively. Values of g were calculated for each run using the μ values specified above and the cumulative frequency distributions were determined. Table 3.2 illustrates the cumulative frequency distributions for each value of μ and for L = 1, 2 and 3. Figures 3.1, 3.2 and 3.3 show the variation of the cumulative frequency of g for the various combinations of n and L.

Values of f for each combination of n and L were determined by taking the smallest values such that 95% of the g values were enclosed in the interval $0 \leq g \leq f$, for each distribution. These values are tabulated in Table 3.3. Since f is determined from the cumulative frequency distribution of g which in turn is made up of data grouped into intervals of width .2, it is unlikely that the error in f is less than the width of this interval. Figure 3.4 shows f for L = 1, 2 and 3, plotted against the sample size n. The deviation of the majority of the plotted points from the drawn curves is less than .2 in absolute value. One exception to this is at the point N = 1200, L = 1 where the deviation from the curve is .5, but it is perhaps noteworthy that an f value of 1.8 corresponds to a confidence of 92.5% for this set of calculations.

Figure 3.4 shows that the factor, f, is a function of n when n is small ($n < 200$) and that f increases rapidly for values of n less than 100. This increase in f may be due to the fact that the assumptions underlying equation (3.4.11) are no longer valid, that is the series of P values are not normally distributed and, when temporary heavy loading occurs, exhibit autocorrelation. It is to be expected that the equation will break down when n is less than 10 since equation (3.4.3) is no longer a good approximation.

When n is greater than 200, f is virtually constant, but dependent on the number of priority classes in the job-stream. The values of this

TABLE 3.2
 CUMULATIVE FREQUENCY DISTRIBUTIONS OF THE g VALUES

CLASS LIMIT	SAMPLE SIZE																					
	L = 1					L = 2					L = 3											
	50	100	200	400	600	50	100	200	400	600	50	100	200	400	600							
0.205	25	20	17	38	23	31	27	36	32	28	24	29	27	20	22	25	13	16	15	15	14	14
0.405	53	41	36	56	43	53	50	62	61	54	43	57	58	48	51	52	30	32	26	32	32	26
0.605	68	65	52	73	62	64	71	77	74	79	70	74	78	70	80	65	52	51	54	44	41	41
0.805	83	84	71	86	84	81	86	92	90	89	87	90	90	88	94	75	75	66	62	53	53	55
1.005	91	93	89	94	91	91	95	104	99	95	98	101	100	96	105	85	85	80	73	67	68	68
1.205	97	101	95	96	99	100	104	107	102	104	107	108	109	104	112	88	94	91	82	80	75	77
1.405	101	105	106	103	107	104	109	110	103	107	112	110	113	115	113	95	101	96	89	84	84	87
1.605	106	108	110	107	110	110	111	112	106	108	117	116	116	117	117	102	105	102	94	99	95	95
1.805	108	111	115	113	113	117	111	114	110	110	120	116	117	119	120	103	111	109	98	106	100	105
2.005	110	112	117	116	115	118	112	115	111	113	117	119	119	119	120	105	112	111	103	111	107	109
2.205	111	114	118	116	117	119	113	116	111	114	118	120	120	120		107	112	115	108	115	112	112
2.405	112	114	119	118	117	119	117	118	112	117	119					107	116	117	114	120	114	116
2.605	114	114	119	119	117	120	118	118	112	117	120					110	117	117	116	116	116	115
2.805	115	116	119	119	119	119	118	118	113	117						111	118	118	118	118	118	118
3.005	116	119	119	119	119	119	120	119	113	117						112	119	118	119	119	119	119
3.205	120	119	119	119	120	120			119	117						113	120	119	120	120	119	119
3.405		119	119	120		120			120	117						113		119			119	119
3.605		119	119						115	118						114		120			119	119
3.805		119	119						116	118						115					119	119
∞		120	120						120	120						120					120	120

FIGURE 3.1 - CUMULATIVE FREQUENCY POLYGON L = 1

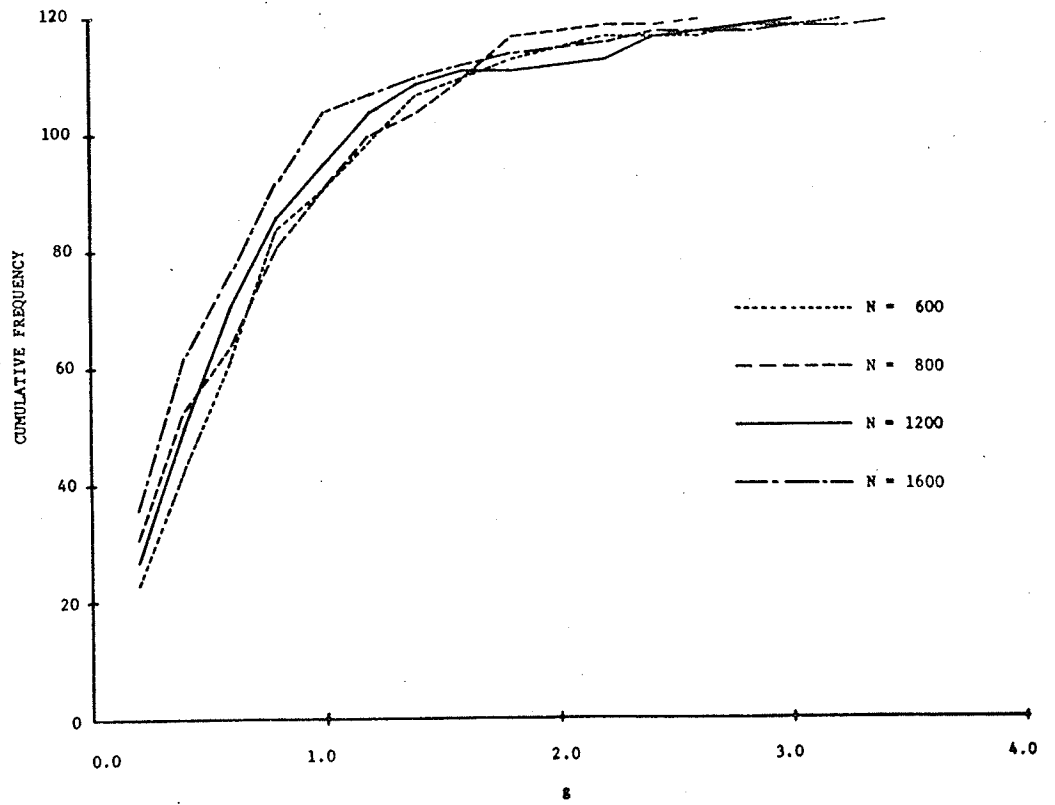
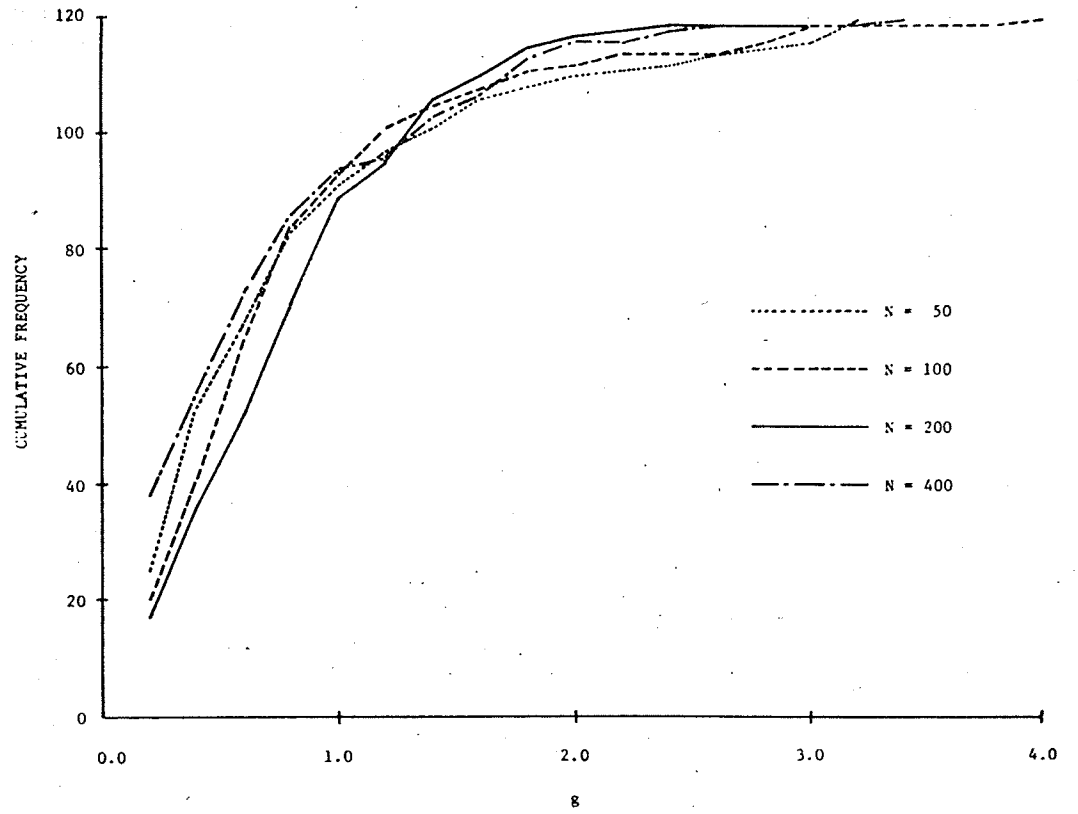


FIGURE 3.2 - CUMULATIVE FREQUENCY POLYGON L = 2

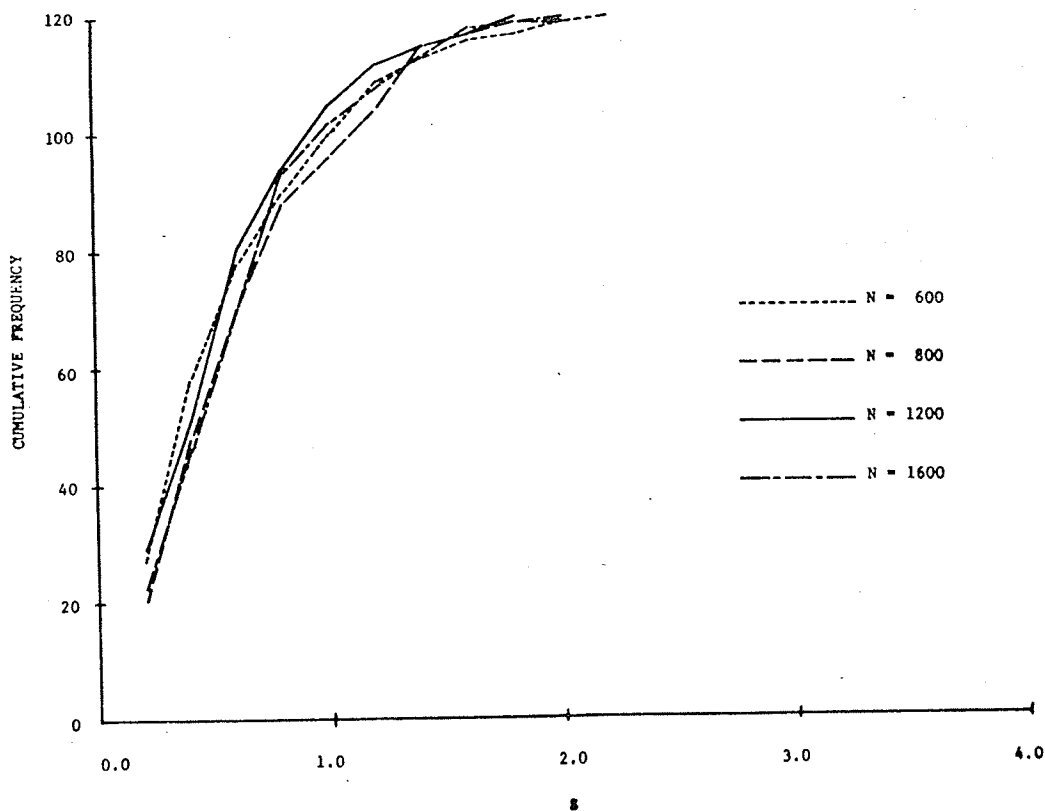
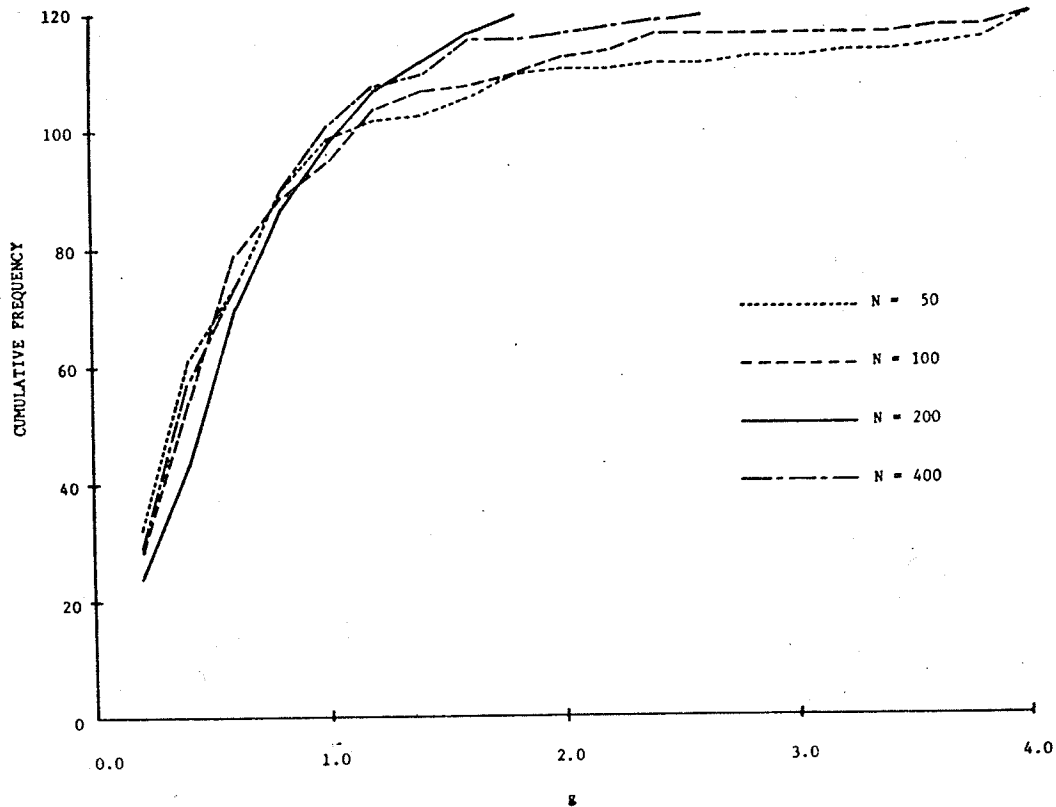


FIGURE 3.3 - CUMULATIVE FREQUENCY POLYGON L = 3

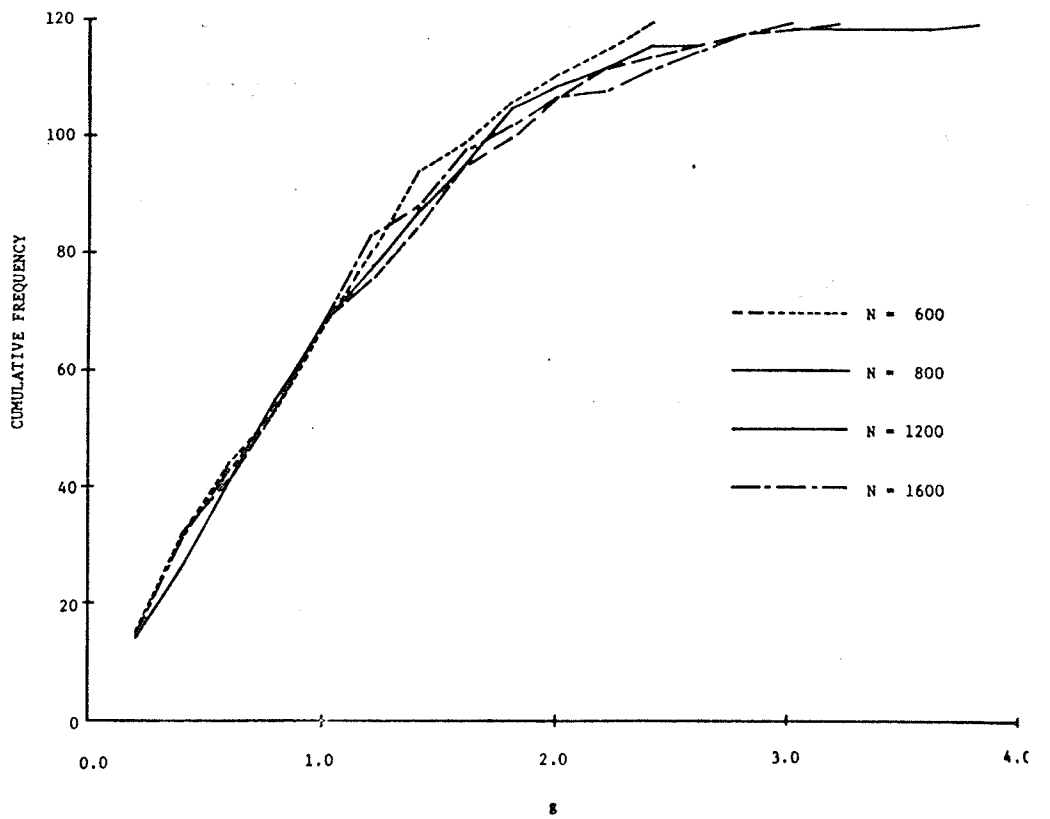
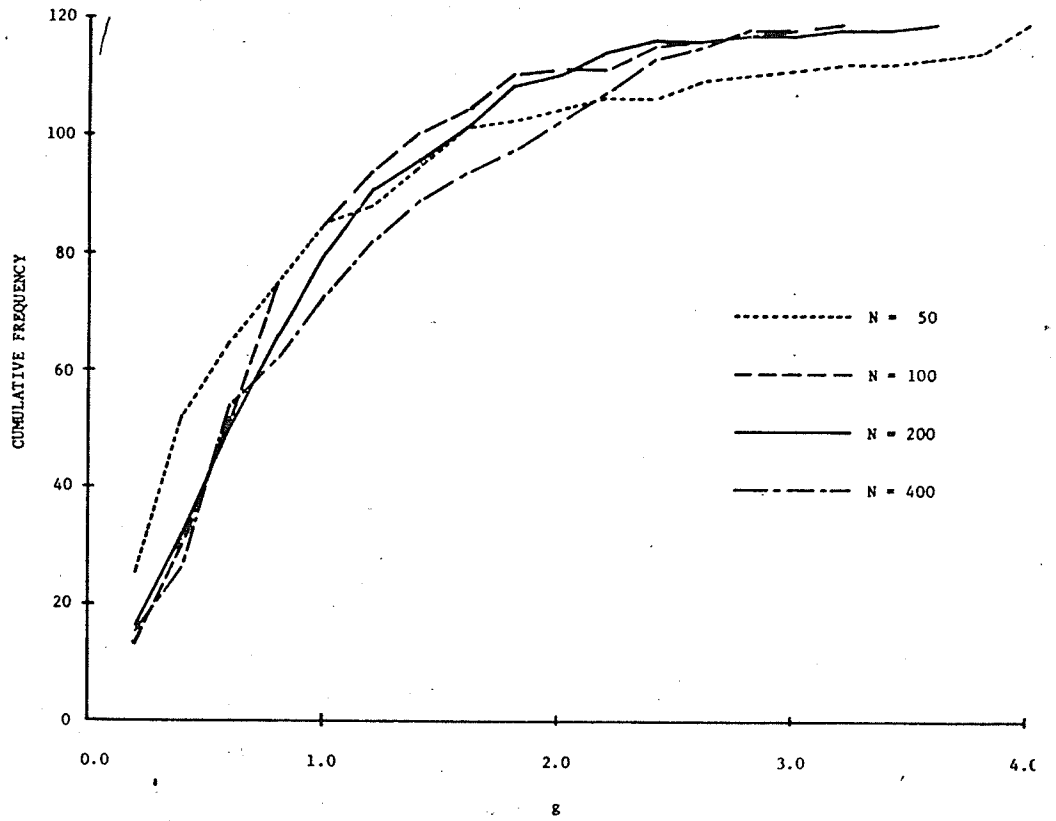
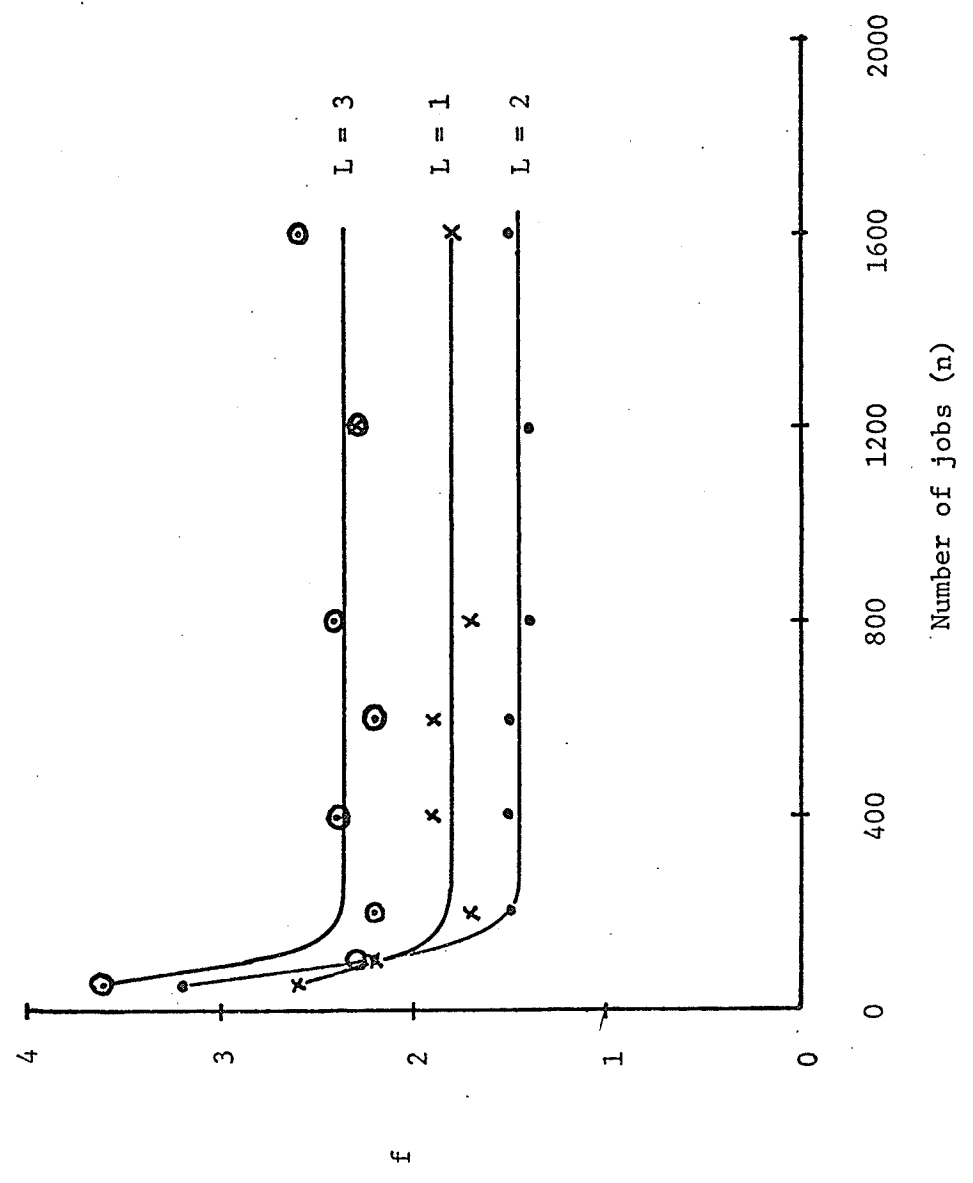


TABLE 3.3
VALUES OF f FOR ONE, TWO AND THREE PRIORITY STREAMS

n	f		
	L = 1	L = 2	L = 3
50	2.6	3.2	3.6
100	2.2	2.2	2.3
200	1.7	1.5	2.2
400	1.9	1.5	2.4
600	1.9	1.5	2.2
800	1.7	1.4	2.4
1200	2.3	1.4	2.3
1600	1.8	1.5	2.6

FIGURE 3.4 - f v.s. SAMPLE SIZE



constant are 1.8, 1.5 and 2.4 for $L = 1, 2$ and 3 , respectively.

Simulation studies are therefore more feasible for n values greater than 200 and the accuracy of P_n is greater for job-streams with two priority classes, although not significantly.

Equation (3.4.11), using the appropriate value of f , may be used to determine the reproducibility of P_n obtainable from a single simulation run. When P_n and S_n are substituted in equation (3.4.11) the range within which μ lies with 95% certainty is obtained.

In order to determine the value of μ necessary to obtain a certain percentage accuracy in P_n , it is necessary that S_n be determined approximately by means of either a pilot simulation run or by other methods, if such are available. For the simulation model used in this research, a simulation run for a job-stream of 1000 jobs ($n = 1000$), takes approximately 5 seconds of execution time using an IBM 360/65 computer.

Table 3.4 shows the percentage error in P_n for a number of different values of n and S_n . The values of S_n are related to the values of P_n , for simplicity by the equation,

$$S_n = c P_n, \quad \dots(3.5.1)$$

whereupon the percentage error in P_n is given by,

$$\pm 200 f c / (\sqrt{n} - \sqrt{2}) . \quad \dots(3.5.2)$$

3.6 CONCLUSIONS

The reproducibility of P_n , the measure of performance, has been determined for a number of values of n by calculating the range around P_n within which the population mean, μ , lies with 95% certainty. The range was assumed to be related to that of a normal distribution by the inclusion of a multiplying factor, f , which was determined by means of repeated simulation runs. The factor f , has been shown to be a function of n , the number of jobs

TABLE 3.4
 PERCENTAGE ERROR IN P_n FOR VARIOUS VALUES OF n AND S_n

n	(S_n/P_n)											
	L = 1				L = 2				L = 3			
	0.2	0.5	1.0	1.5	0.2	0.5	1.0	1.5	0.2	0.5	1.0	1.5
200	5.7	14.1	28.3	42.4	4.7	11.8	23.6	35.6	7.5	18.9	37.7	56.5
400	3.9	9.7	19.4	29.1	3.2	8.1	16.1	24.2	5.2	12.9	25.8	38.7
800	2.7	6.7	13.4	20.1	2.2	5.6	11.1	16.7	3.6	8.9	17.9	26.8
1600	1.9	4.7	9.3	14.0	1.6	3.9	7.8	11.7	2.5	6.2	12.4	18.7
3200	1.3	3.3	6.5	9.8	1.1	2.7	5.4	8.2	1.7	4.4	8.7	13.1
6400	0.9	2.3	4.6	6.9	0.8	1.9	3.8	5.7	1.2	3.1	6.1	9.2
12800	0.6	1.6	3.2	4.8	0.5	1.3	2.7	4.0	0.8	2.1	4.3	6.4
25600	0.4	1.1	2.3	3.4	0.4	0.9	1.9	2.8	0.6	1.5	3.0	4.5
51200	0.3	0.8	1.6	2.4	0.3	0.7	1.3	2.0	0.4	1.1	2.1	3.2

in the job-stream, when n is small ($n < 200$). When n is greater than 200, f is virtually constant. The evaluation of f for job-streams composed of jobs from one, two and three priority classes has shown that the constant value of f when n is greater than 200, is dependent on the number of priority classes in the job-stream. The values of f determined in the constant range were found to be 1.8, 1.5, 2.4 for one, two and three priority job-streams, respectively. The values of f that were determined give rise to equations which can be used to determine the reproducibility of P_n , when S_n and n are known. In order to determine the value of n necessary for a particular percentage accuracy in P_n , it is necessary to first determine S_n by means of a pilot simulation run or by *a priori* methods.

The ratio of S_n , the standard deviation, to P_n , the mean final penalty (Appendix E), is about 2.6, 3.0 and 2.0 for $L = 1, 2$ and 3 , respectively. Thus the values of n necessary to give a reproducibility of $\pm 5\%$ would be 40,000, 40,000 and 36,000 for $L = 1, 2$ and 3 , respectively.

In the following chapter, the job-stream is set at $n = 6400$. This size of job-stream provides an accuracy of about $\pm 12\%$ for $L = 1, 2, 3$. This value was chosen since the number of jobs required to provide $\pm 5\%$ accuracy (40,000 $L = 1, 2$, 36,000 $L = 3$) was deemed unacceptable because of the extra machine time required.

CHAPTER IV

AN INVESTIGATION OF THE EFFECTS OF VARYING THE PENALTY CURVES

4.1 INTRODUCTION

Penalty or cost curves may be used in three related areas within a computer system. They may be used to convey the costs incurred by users as a result of delays in service, to measure the performance of the system as the mean cost of delay and to schedule jobs within the system in order to attempt to reduce the mean cost of delay. In the research described in Chapter III, penalty curves were used in part of the scheduling algorithm (quantum-allocation), but their main use was as a measure of system performance.

For a given system environment, the cost curves are usually fixed and the mean penalty, P_n , may be used to measure system performance in a variety of situations brought about by changes in job loading, scheduling algorithms, etc.

It is reasonable to enquire whether the use of penalty curves in job scheduling has a significant effect upon the pattern of relative responses, the ω_i 's, that is the ratio of time spent in the system to the time requested from the system. If penalty curves are effective, the distribution of relative responses should be sensitive to changes made in the penalty curves. In order to study whether the use of penalty curves in scheduling is effective, it is necessary to vary the shape of the penalty curves, and under these particular circumstances, the measure of performance, P_n , is no longer significant. The more basic quantity here is, $\bar{\omega}$, the mean of the relative responses.

The penalty curves used in Chapter III were arbitrary, but one could reasonably expect the users of an actual computer system to assign similar costs to their delays. As an alternative to penalty curves of the

type shown in Figure 2.2, it has been suggested by Greenberger [4] that one could postulate a desirable response time or deadline for each request with the deadline being assigned on the basis of the importance of the problem, nature of use or some combination of such factors. When the response time is less than or equal to the deadline value, the cost of delay or penalty is minimal, when the response time is greater than the deadline the maximum cost of delay is reached. This approach leads to simple cost curves which are step functions of the form:

$$\begin{aligned} \alpha_i &= \alpha_{\min} & u_i &\leq \phi, \\ \alpha_i &= \alpha_{\max} & u_i &> \phi, \end{aligned}$$

where α_i is the penalty attributed to job i , α_{\min} and α_{\max} are the minimum and maximum penalties, u_i is the lack of attention and ϕ is the deadline or step point.

Presupposing that penalty curves are effective in scheduling, it may be that the design of the curve necessary to give a particular distribution of relative responses may be contrary to the thought that they should reflect the costs of delay. However, if this situation were to arise, both scheduling and costing could be achieved by the use of two sets of curves.

This chapter describes the investigations made as to the applicability to job scheduling algorithms of cost curves which are step functions and to the effect of movement of the step point or deadline.

The final penalty attributed to a job and the mean final penalty of a simulation run are functions of the penalty curves used and will vary as the penalty curve is changed. It is for this reason that the mean final penalty, P_n , the measure of performance used in the reproducibility investigations described in Chapter III, is not used in this study. However, it is assumed that the relationship determined between the measure of performance

and its standard deviation (equation 3.4.11) is valid for $\bar{\omega}$ and ω' (the standard deviation of the ω_i values), rather than P_n and S_n .

The relative response of a job, ω_i , is not a function of the penalty curve used, but depends only on the CPU time requested by a job and the time interval which the job spends in the system. For a job i , the relative response, ω_i , is a measure of how quickly the system responds to the job's request for servicing and is given by:

$$\omega_i = \frac{T_i - A_i}{m_i + t_{di}} = \frac{\Delta T_i}{\tau_i}, \quad \dots(4.1.1)$$

where T_i is the time of completion, A_i is the arrival time, m_i is the CPU time requested and t_{di} is the assumed device time for job i . Comparison of equation (4.1.1) to equation (2.3.3) shows that ω_i is equal to u_i , the lack of attention value of job i as it finishes.

The mean relative response for a simulation run, denoted $\bar{\omega}$, is given by:

$$\bar{\omega} = \frac{1}{n} \sum_{i=1}^n \omega_i \quad \dots(4.1.2)$$

where n is the number of jobs in the simulation run. The mean relative response for a priority class j in a multi-priority job-stream is denoted $\bar{\omega}_j$ where j is the priority class and is given by:

$$\bar{\omega}_j = \frac{1}{n_j} \sum_{i=1}^n \omega_i, \quad \dots(4.1.3)$$

($Y_i=j$)

and n_j is the number of jobs of priority class j in the simulation run.

The effects of the variation of penalty curves upon the job scheduling algorithm might be observed in the $\bar{\omega}$, $\bar{\omega}_j$ values and their distributions, and because they are independent of the penalty curves, they are used as a measure of model performance when the penalty curves are being varied. The measure of performance used by Fife [6] is very similar to $\bar{\omega}$. From the

defining equation of ω_i , (4.1.1), it can be seen that since τ_i is constant, ω_i can increase only if ΔT_i increases. However, a large value of ΔT_i indicates that the job has been delayed within the system and thus ω_i may be considered as an inverse measure of performance with large values of ω_i , $\bar{\omega}$ or $\bar{\omega}_j$ indicating poorer service than small values.

In order to examine the effects upon $\bar{\omega}$ of varying the penalty curves, a number of simulation runs were made using various combinations of penalty curves over a range of job loadings and for one, two and three priority class job-streams ($L = 1, 2, 3$). The effects of changing the penalty curves on the $\bar{\omega}_j$ values were also examined.

4.2 THE CHOICE OF STEP FUNCTIONS

In previous simulation runs it was found that the majority of jobs had at completion, lack of attention values that were less than three and the lower limit was known to be one. Because the performance measure selected in this chapter, the relative response, is equal to the lack of attention at job completion, the step functions were selected to lie in the interval (1.0 to 3.0) in which the majority of values were known to occur. The step points, denoted ϕ , that were selected were:

$$\phi = 1.5, 2.0, 3.0 .$$

For use in the job scheduling algorithm, a penalty curve is defined in terms of the selected step point such that for job i at time t :

$$\alpha_i(t) = 1 \quad u_i(t) \leq \phi, \quad \dots(4.2.1)$$

$$\text{or} \quad \alpha_i(t) = 1000 \quad u_i(t) > \phi, \quad \dots(4.2.2)$$

where $u_i(t)$ is the current lack of attention of job i and ϕ is the designated step point.

4.3 THE EFFECTS OF STEP FUNCTIONS ON THE ALLOCATION OF CENTRAL PROCESSOR TIME

One effect of the use of penalty curves that are step functions is that with a variable time-slicing algorithm, jobs whose current lack of attention, u , is greater than the step point can receive a very large portion of the round-robin time. For example, if three jobs, i , j and k are in the execution list with u values of $u_i = 1$, $u_j = 1$ and $u_k = 1000$, the round-robin time, t_{rr} , will be allocated as follows:

$$q_i = \frac{1}{1002} t_{rr},$$

$$q_j = \frac{1}{1002} t_{rr},$$

and
$$q_k = \frac{1000}{1002} t_{rr} .$$

In this example, job k will receive about 99.8% of the round-robin time and, since the quanta q_i and q_j are less than the minimum time-slice, these jobs will receive quanta equal to one one-hundredth of the round-robin cycle time.

4.4 THE SELECTION OF JOBS FROM THE WAITING QUEUE AND THE SERVICING OF JOBS IN THE EXECUTION LIST

The penalty curves as defined were used by the job scheduling algorithm to determine the order of job servicing in the execution list, the order of job selection from the waiting queue and by the quantum-allocation routine to determine the size of the quanta allocated to processing jobs. In Chapter III, a separate penalty curve was defined for each priority class and the penalty of a job was a function of its priority class as well as its current lack of attention. In this examination only one penalty curve was defined for a given simulation run. Thus in the event of a multi-priority job-stream, the same penalty curve was used by each priority class. Time-slices were allocated to the processing jobs according to the variable time-slice algorithm outlined in Section 2.5.

Jobs in the waiting queue were selected for transfer to the execution list on the basis of the highest penalty, (first-come-first-served by penalty) the penalty of a job in the queue being calculated in the manner outlined in Section 4.2. In the case of two or more jobs with identical penalties, the job with the earliest arrival time would be selected for transfer to the execution list. If two jobs had identical arrival times, the job with the lowest queue slot number would be selected for transfer to the execution list. The probability of the latter event occurring was very small since the arrival times were recorded with approximately 7 decimal digits of precision. Since the penalty curves were the same for each priority class, the queue selection algorithm performs as a first-come-first-served, selection algorithm.

Jobs in the execution list were serviced in the order of decreasing penalty, that is the job with the highest penalty and CPU time available was executed first. In the case of equal penalties, the first job located with execution time available, in a scan of the execution list in the direction of increasing slot numbers, would be serviced first.

4.5 THE EFFECTS OF VARIATION OF PENALTY CURVES ON THE MEAN RELATIVE RESPONSE

In order to investigate the effects of changes in ϕ , the step point, a number of simulation runs were made using one, two and three priority class job-streams ($L = 1, 2, 3$) and varying the mean CPU request time, \bar{m}_1 . All other job-mix and operational parameters were held constant. Table 4.1 shows the job-stream and operational parameters used for $L = 1, 2$ and 3 . The number of jobs processed through the model was 6400 on each simulation run. This number of jobs was selected as being a reasonable compromise between accuracy of results (approximately 40,000 jobs for $\pm 5\%$) and machine run time. For L equal to one, two and three, the results are assumed to be

TABLE 4.1
 JOB-MIX AND OPERATIONAL PARAMETERS USED
 IN THE VARIATION OF PENALTY STUDY

JOB-MIX PARAMETERS:

Proportionality Constant	k = 4.0
Number of Jobs	n = 6400
Starting Random Number	= 9875211
Mean Interarrival Time	$\bar{r}_1 = 20$ seconds

OPERATIONAL PARAMETERS:

Size of the Execution List	n _e = 3 jobs
Size of the Waiting Queue	n _q = 20 jobs
Round-Robin Cycle Time	t _{rr} = 6.0 seconds
Minimum Time-Slice	t _m = 0.06 seconds
Supervisor Cycle Time	t _s = 0.01 seconds
Constant Device Time	t _d = 1.00 seconds

accurate within $\pm 12\%$. The starting random number was 9875211 for all simulation runs. The relative response for each job in a simulation run was determined and from these the average relative response of a simulation run, $\bar{\omega}$ and its standard deviation, ω' was calculated.

The ratio of \bar{m}_1 to \bar{r}_1 is the mean proportion of the CPU capacity demanded by priority class one jobs. Also,

$$\frac{\bar{m}_{j+1}}{\bar{r}_{j+1}} = \frac{\bar{m}_j}{\bar{r}_j} \quad \dots(4.5.1)$$

Thus the proportion of the CPU capacity demanded by priority class j is the same as that of priority class 1 and only \bar{m}_1 , \bar{r}_1 and L , the number of priority classes need be specified to determine the overall system loading factor, ρ . The system load, ρ , a more fundamental measure than the ratio of \bar{m}_1 to \bar{r}_1 is independent of the job-stream and is given by:

$$\rho = \frac{\bar{m}_1}{\bar{r}_1} L \quad \dots(4.5.2)$$

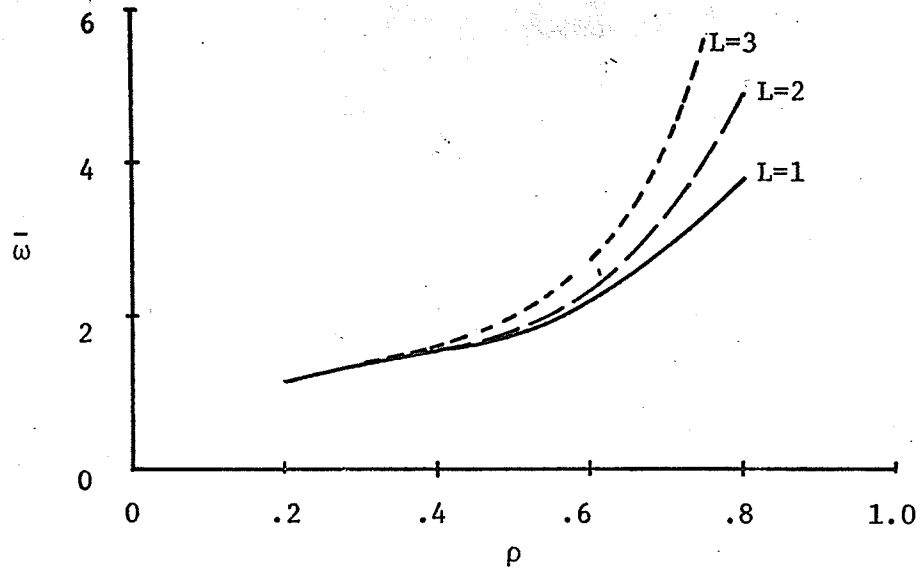
Since the ratio \bar{m}_1/\bar{r}_1 changed as the mean CPU request time, \bar{m}_1 , was changed, the ratios were converted to ρ values so as to provide a common ground for comparison of the $\bar{\omega}$ values.

For each value of L used, four values of ρ were selected. They were $\rho = .2, .4, .6$ and $.8$ for $L = 1$ and 2 , and $\rho = .30, .45, .60$ and $.75$ for $L = 3$. For each ρ value and for a given value of L , a simulation run was performed for $\phi = 1.5, 2.0$ and 3.0 . Thus a total of thirty-six simulation runs were made, from which the $\bar{\omega}$ and $\bar{\omega}_j$ values were calculated.

Figure 4.1(a) shows the mean relative response, $\bar{\omega}$, as a function of ρ for $L = 1, 2$ and 3 and $\phi = 1.5$. Figures 4.1(b) and 4.1(c) are graphs of the differences for $L = 1, 2$ and 3 from $\phi = 1.5$ for $\phi = 2.0$ and 3.0 , respectively. As is to be expected, the mean relative response increases in all

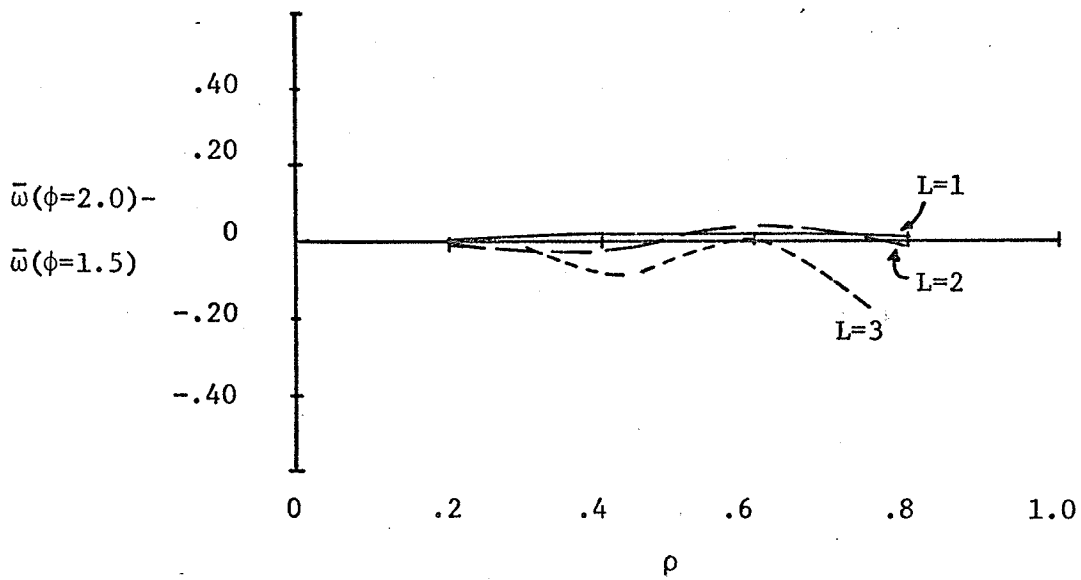
FIGURE 4.1

$\bar{\omega}$ as a function of ρ $L = 1, 2, 3$ $\phi = 1.5, 2.0, 3.0$



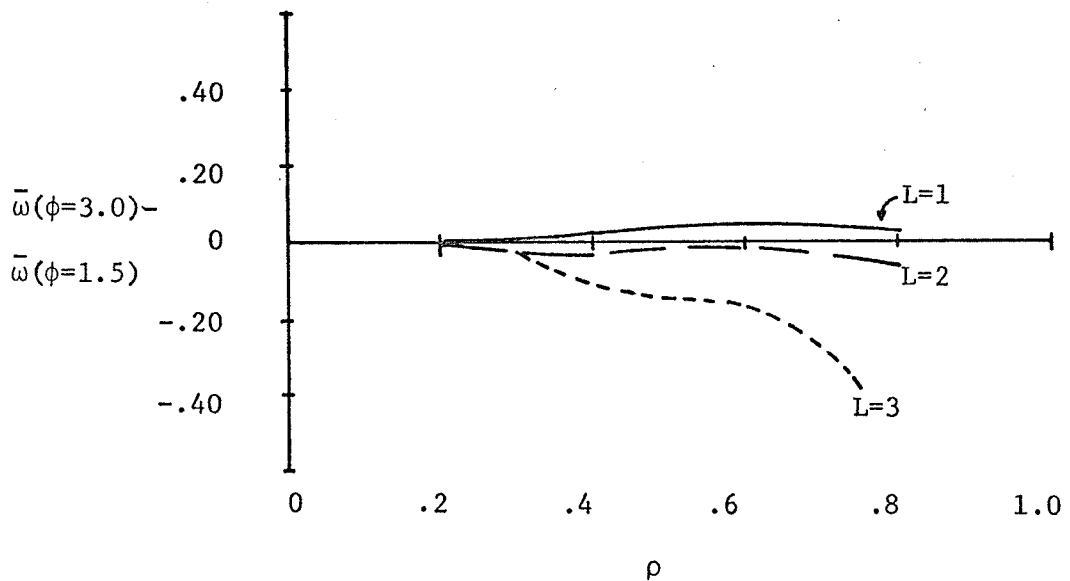
$\phi = 1.5$

(4.1(a))



$\phi = 2.0$

(4.1(b))



$\phi = 3.0$

(4.1(c))

cases as the system load increases [7,11]. For a one priority class job-stream, the relative response increases as ϕ increases, with the increase being larger for heavy system loadings. For $L = 2$, system performance is virtually independent of the penalty curve used. When the number of priority classes L , was three, the system performance improved as ϕ moved away from 1.5, with the improvement being greatest (7%) at heavy loads. The overall effect of varying the value of ϕ and thus changing the penalty curve is thus dependent on the number of priority classes in the job-stream; with little change for $L = 1$ and $L = 2$ and a small improvement for $L = 3$. Table 4.2 shows the numeric values of $\bar{\omega}$, ω' and δ . The latter value is equal to $f\{2\omega'(\sqrt{n} - \sqrt{2})^{-1}\}$ (cf equation 3.4.11) and its sum and difference with $\bar{\omega}$ gives the 95% range of $\bar{\omega}$.

4.6 THE EFFECTS OF PENALTY CURVE VARIATION UPON THE RELATIVE RESPONSE OF A PRIORITY CLASS OF JOBS

In the previous section, the overall mean relative responses for $L = 1, 2$ and 3 and for $\phi = 1.5, 2.0$ and 3.0 were examined and discussed. Since the effect of varying ϕ was found to give a maximum of 7% improvement in overall system performance, it was decided to examine the mean relative responses of the individual priority classes for $L = 2$ and 3 to see if the changes in $\bar{\omega}_j$ benefited a particular priority class.

Figure 4.2(a) shows the relative response for $j = 1$ and $j = 2$ as a function of ρ for $\phi = 1.5$ and $L = 2$. Figures 4.2(b) and 4.2(c) are difference graphs for $j = 1$ and $j = 2$ at $\phi = 2.0$ and $\phi = 3.0$, respectively. Table 4.3 gives the $\bar{\omega}_j$ values from which Figure 4.2 was derived as well as the ω'_j values. These values are the result of calculations performed in Section 4.5. As can be seen from Figure 4.2(a), with $\phi = 1.5$ and $L = 2$, the priority class two jobs ($j = 2$) fair better than priority class one jobs for all loads and particularly when the system load is heavy ($\rho = .8$). When $\phi = 2.0$ or 3.0 , the

TABLE 4.2

RELATIVE RESPONSE FOR SIMULATION RUNS FOR L=1,2,3 WITH $\phi = 1.5, 2.0, 3.0$

L = 1										
$\phi = 1.5$					$\phi = 2.0$			$\phi = 3.0$		
\bar{m}_1	ρ	$\bar{\omega}$	ω'	δ	$\bar{\omega}$	ω'	δ	$\bar{\omega}$	ω'	δ
4	.2	1.14	.35	.016	1.14	.34	.016	1.14	.34	.016
8	.4	1.55	.73	.034	1.57	.75	.034	1.57	.78	.036
12	.6	2.15	1.25	.057	2.17	1.25	.057	2.20	1.26	.057
16	.8	3.73	2.54	.116	3.74	2.53	.116	3.76	2.52	.116
L = 2										
2	.2	1.17	.45	.017	1.17	.40	.015	1.17	.41	.015
4	.4	1.51	.95	.036	1.49	.92	.035	1.48	.89	.034
6	.6	2.25	1.93	.074	2.29	1.94	.074	2.23	1.88	.072
8	.8	4.84	4.84	.184	4.83	4.89	.186	4.78	4.90	.186
L = 3										
2	.3	1.37	.88	.054	1.35	.78	.048	1.35	.64	.039
3	.45	1.81	1.85	.113	1.73	1.72	.105	1.68	1.43	.087
4	.60	2.71	3.04	.186	2.70	3.83	.234	2.54	3.57	.220
5	.75	5.68	6.55	.400	5.50	8.01	.490	5.30	7.98	.490

FIGURE 4.2

$\bar{\omega}_j$ as a function of ρ for $L = 2$ $\phi = 1.5, 2.0, 3.0$

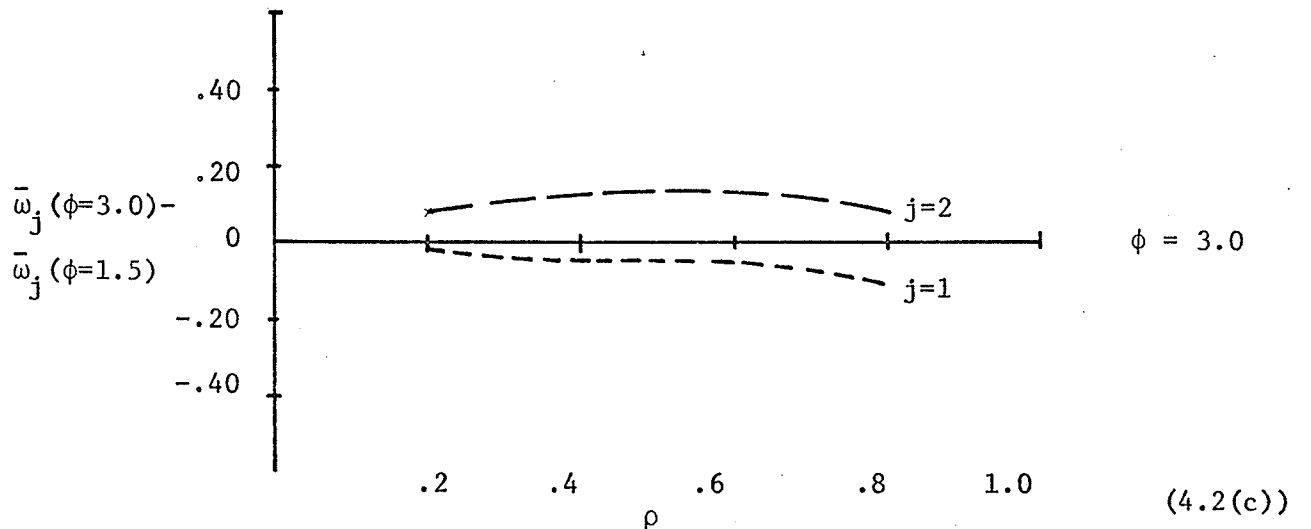
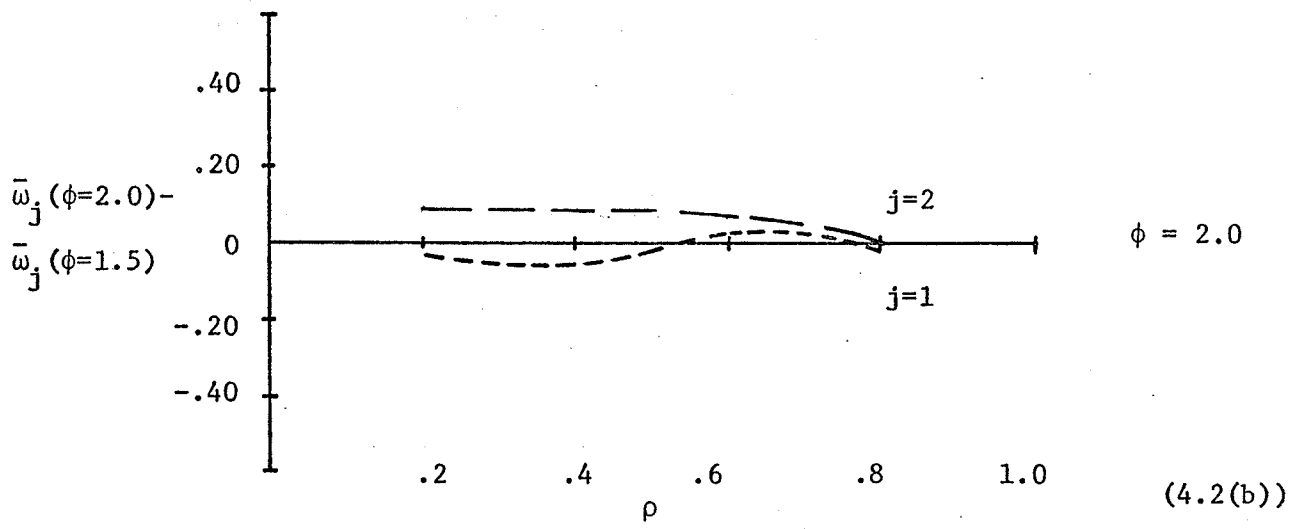
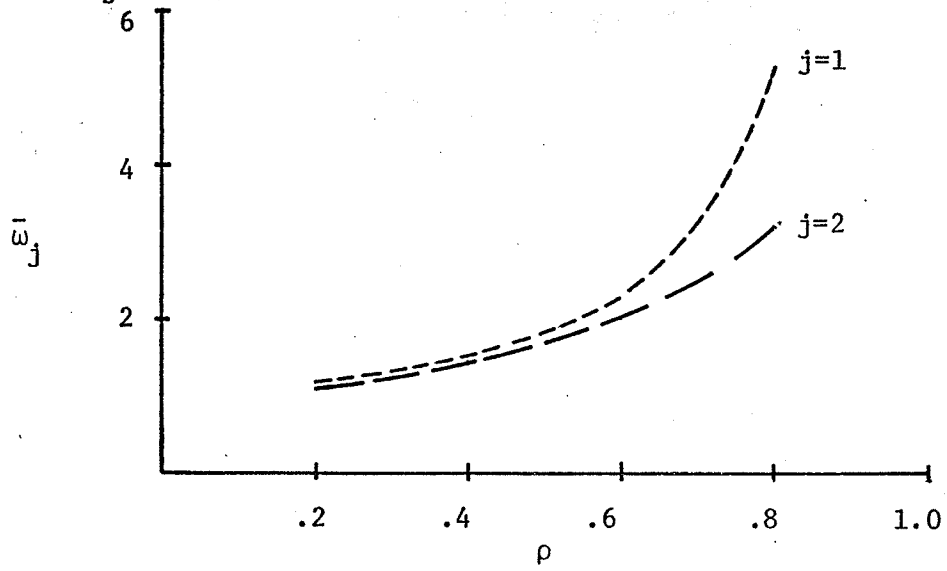


TABLE 4.3

 $\bar{\omega}_j$ AND ω_j' FOR A TWO PRIORITY CLASS JOB-STREAM

$\phi = 1.5$					
		$j = 1$		$j = 2$	
\bar{m}_1	ρ	$\bar{\omega}_1$	ω_1'	$\bar{\omega}_2$	ω_2'
2	.2	1.19	.49	1.10	.26
4	.4	1.52	1.03	1.47	.56
6	.6	2.31	2.10	2.00	.90
8	.8	5.25	5.25	3.19	1.79
$\phi = 2.0$					
2	.2	1.16	.41	1.19	.33
4	.4	1.47	.97	1.56	.63
6	.6	2.34	2.12	2.07	.90
8	.8	5.23	5.30	3.19	1.73
$\phi = 3.0$					
2	.2	1.17	.42	1.18	.33
4	.4	1.46	.93	1.59	.69
6	.6	2.26	2.04	2.13	.96
8	.8	5.14	5.32	3.27	1.70

priority class one jobs have a slightly better relative response until ρ is about .5, after which the $\bar{\omega}_1$ values become greater than the $\bar{\omega}_2$ values, indicating poorer service for priority class one jobs (Table 4.3).

As with the $\bar{\omega}$ values for the simulation run, the variation of the ϕ values has very little effect on the $\bar{\omega}_j$ values (+2% $j = 1$, -2% $j = 2$ relative to $\phi = 1.5$).

Figure 4.3(a) shows the relative responses for $j = 1, 2$ and 3 as a function of ρ for $\phi = 1.5$. Figures 4.3(b) and 4.3(c) are the difference graphs for $\phi = 2.0$ and 3.0 , respectively, for $j = 1, 2$ and 3 . A tabular representation of the values used to plot Figure 4.3 is given in Table 4.4.

When L is equal to three (Figure 4.3), the $\bar{\omega}_j$ values vary slightly for system loads less than .5, after which the mean relative responses of the individual priority classes decrease and are inversely related to j . The step point, $\phi = 1.5$, appears to favour the low priority jobs ($j = 3$), while higher step points of $\phi = 2.0$, and 3.0 , offer improved performance for priority class 1 and 2 jobs.

The maximum improvements affected by movement of the step point upon the $\bar{\omega}_j$ values are in the order of +2% $j = 1$, -2% $j = 2$ for $L = 2$ and +7% $j = 1$, +13% $j = 2$, -5% $j = 3$ for $L = 3$ all relative to $\phi = 1.5$ (Figures 4.2, 4.3).

4.7 THE EFFECTS OF VARIATION OF THE PENALTY CURVE IN A MULTI-PRIORITY JOB-STREAM

Section 4.6 has described the effects of varying the penalty function for one, two and three priority class job-streams, in which the same penalty curve was used by each priority class in the multi-priority class simulation runs. This section describes the results from simulation runs in which the penalty curve for priority class one was varied while the penalty curves used by priority classes two and three were made equal and set to a

FIGURE 4.3

$\bar{\omega}_j$ as a function of $\rho L = 3$ $\phi = 1.5, 2.0, 3.0$

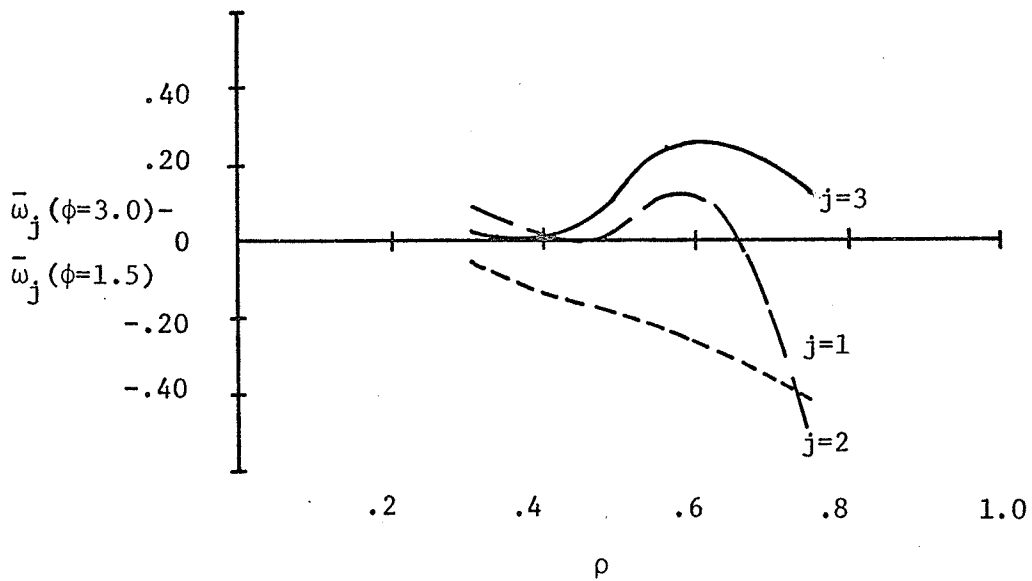
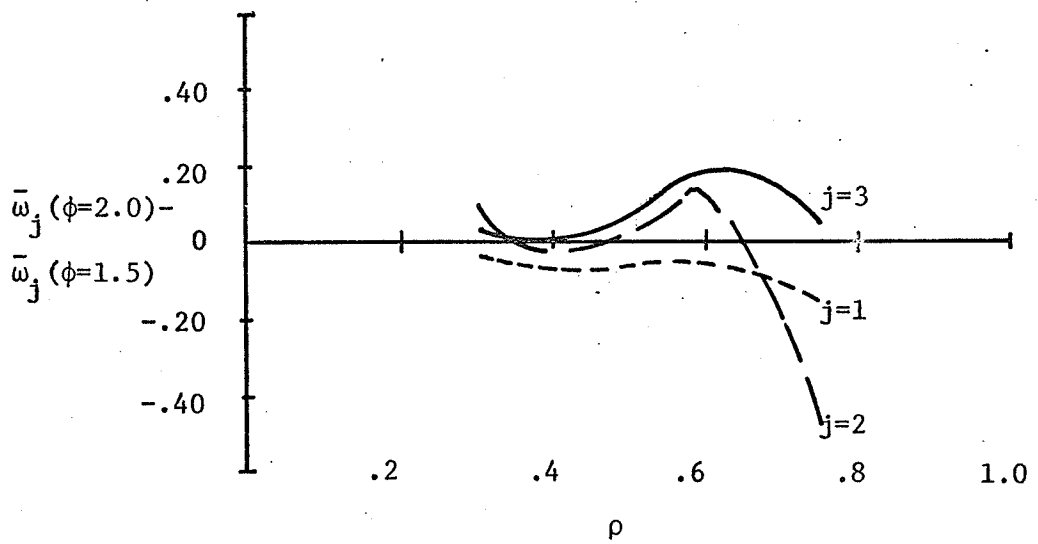
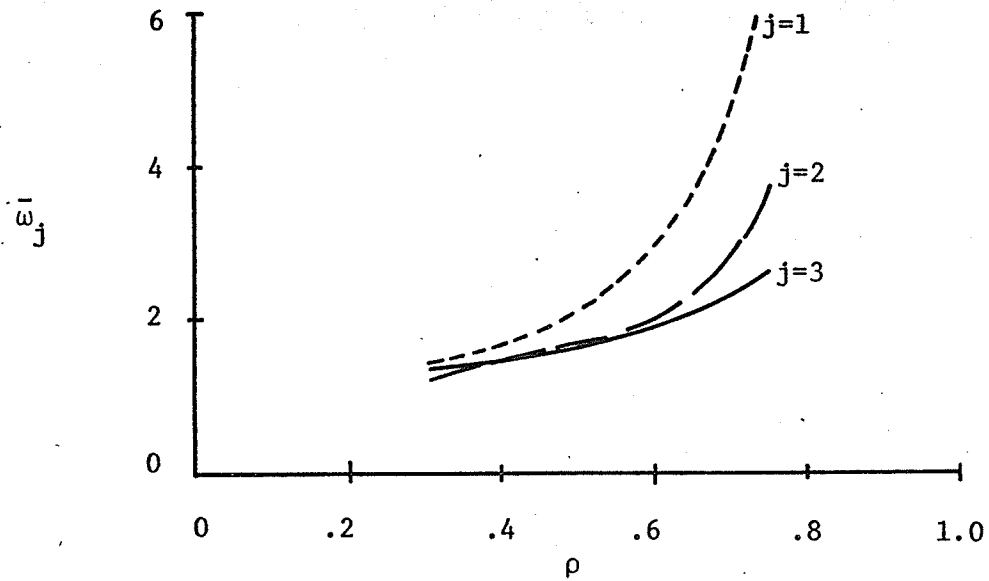


TABLE 4.4

 $\bar{\omega}_j$ AND ω_j' FOR $L = 3$, $\phi = 1.5, 2.0, 3.0$

$\phi = 1.5$							
\bar{m}_1	ρ	$j = 1$		$j = 2$		$j = 3$	
		$\bar{\omega}_1$	ω_1'	$\bar{\omega}_2$	ω_2'	$\bar{\omega}_3$	ω_3'
2	.30	1.41	.97	1.21	.52	1.36	.38
3	.45	1.86	2.06	1.62	.85	1.65	.54
4	.60	2.94	3.35	1.99	1.61	1.91	.70
5	.75	6.35	7.16	3.73	3.36	2.64	1.11
$\phi = 2.0$							
2	.30	1.36	.86	1.30	.48	1.39	.40
3	.45	1.77	1.91	1.60	.82	1.67	.54
4	.60	2.88	4.28	2.11	1.43	2.10	.73
5	.75	6.19	8.89	3.24	2.73	2.68	.97
$\phi = 3.0$							
2	.30	1.36	.71	1.30	.46	1.39	.43
3	.45	1.69	1.58	1.60	.73	1.70	.61
4	.60	2.67	3.99	2.11	1.36	2.16	.78
5	.75	5.94	8.87	3.22	2.68	2.76	.97

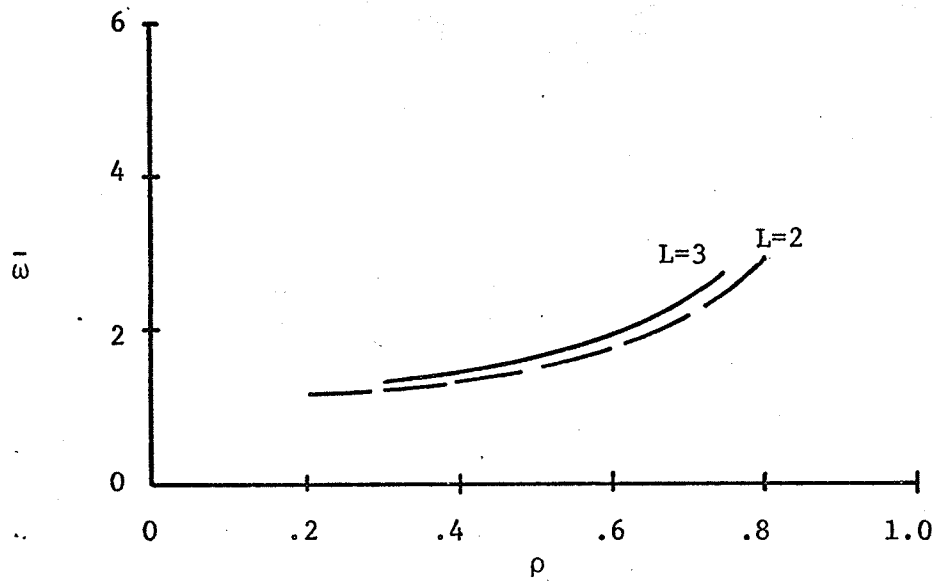
constant value of one. The job-mix and operational parameters used were the same as those shown in Table 4.1. Simulation runs were made for $L = 2$ at $\bar{m}_1 = 2, 4, 6$ and 8 for $\phi = 1.5, 2.0$ and 3.0 and for $L = 3$ at $\bar{m}_1 = 2, 3, 4$ and 5 for $\phi = 1.5, 2.0$ and 3.0 . Sixty-four hundred jobs were processed through the system on each simulation run using the same starting random number of 9875211.

Different queue selection and execution list servicing algorithms were used for these runs. The queue selection algorithm was changed from servicing in the order of decreasing penalty (Section 4.4) to first-come-first-served by priority. Thus jobs with the highest priority (lowest priority class number) were selected for transfer to the execution list first. When equal priority numbers were encountered, selection reverted to first-come-first-served. The result of this change is to give improved performance [11]. The execution list servicing algorithm was changed from servicing in the order of decreasing penalties to servicing in the order of increasing execution slot numbers. The effect of this change was virtually negligible on the system performance and since the latter algorithm was simpler, offered a small saving the model's execution time (Appendix G).

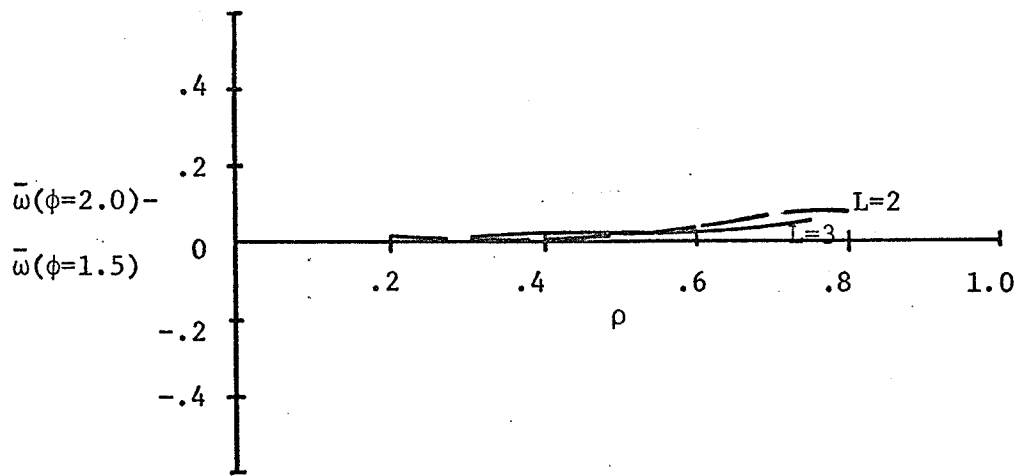
Figure 4.4 and Table 4.5 show the mean relative response, $\bar{\omega}$ as a function of the system load ρ , for $L = 2$ and 3 and $\phi = 1.5$. Figures 4.4(b) and 4.4(c) show the difference graphs of $\bar{\omega}$ for $L = 2$ and 3 when $\phi = 2.0$ and 3.0 , respectively. The effect of moving the step point away from 1.5 is to decrease the performance of the system, particularly for heavy loads. This decrease in performance is in the range of 5% for $L = 3$ at $\rho = .75$ and for $L = 2$ at $\rho = .8$. The improvements noted could arise from the changes made to the penalty curves in going from the results shown in Figure 4.1 to those shown in Figure 4.4, or more probably from the changes made to the queue selection algorithm. It has previously been determined that these latter improvements are of the correct order

FIGURE 4.4

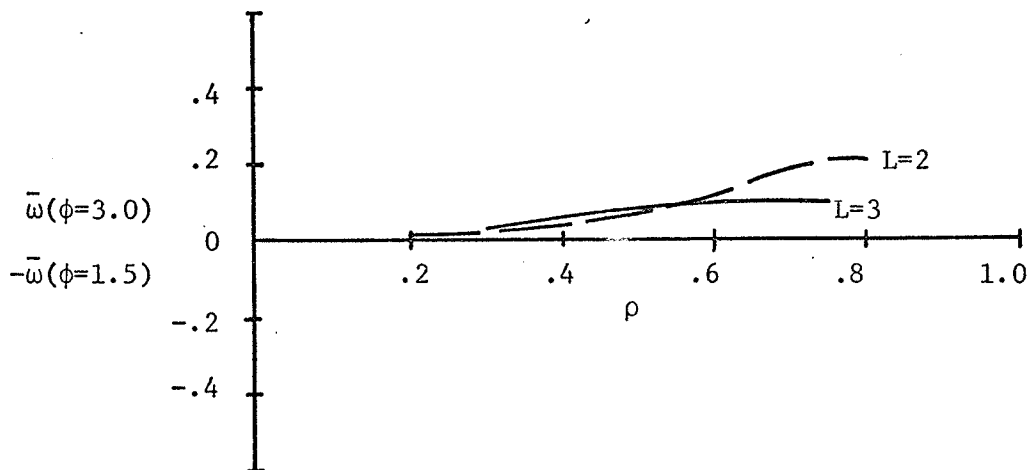
$\bar{\omega}$ as a function of ρ $L = 2, 3$ $\phi = 1.5, 2.0, 3.0$



(4.4(a))



(4.4(b))



(4.4(c))

TABLE 4.5

 $\bar{\omega}$, ω' , and δ for $L = 2, 3$ $\phi = 1.5, 2.0, 3.0$

L = 2										
	$\phi = 1.5$			$\phi = 2.0$			$\phi = 3.0$			
\bar{m}_1	ρ	$\bar{\omega}$	ω'	δ	$\bar{\omega}$	ω'	δ	$\bar{\omega}$	ω'	δ
2	.2	1.15	.36	.014	1.16	.37	.014	1.16	.39	.015
4	.4	1.38	.65	.025	1.40	.65	.025	1.42	.66	.025
6	.6	1.81	1.10	.042	1.85	1.07	.041	1.92	1.12	.042
8	.8	2.80	2.18	.083	2.88	2.14	.082	3.00	2.08	.079
L = 3										
2	.30	1.31	.62	.038	1.32	.63	.038	1.34	.66	.040
3	.45	1.53	1.03	.062	1.55	1.03	.062	1.60	1.06	.065
4	.60	1.95	1.91	.117	1.98	1.92	.117	2.05	1.89	.115
5	.75	2.72	2.81	.171	2.77	3.01	.184	2.82	2.62	.159

of magnitude to be completely explained by the change [11].

The overall mean relative response, $\bar{\omega}$, changed very little for changes in the step point. Because of this, it was decided to calculate the $\bar{\omega}_j$ values, the mean relative responses for the individual priority classes, to determine the effect of changing the ϕ values upon the individual priority classes of jobs making up the job-stream.

Figure 4.5(a) shows $\bar{\omega}_1$ and $\bar{\omega}_2$ for $L = 2$ when $\phi = 1.5$. Figures 4.5(b) and 4.5(c) are the differences observed when $\phi = 2.0$ and 3.0 , respectively. The change observed as the step point is moved is that $\phi = 3.0$ gives an improvement for $\bar{\omega}_2$ in the order of 2% and a deterioration in the order of 10% for $\bar{\omega}_1$, both at $\rho = .8$. The changes observed are negligible for loads less than .5. When Figure 4.5 is compared to Figure 4.2, it can be seen that priority class one jobs receive better service when the queue selection algorithm is first-come-first-served by priority (Figure 4.5). This improvement ranges from 8% for $\rho = .4$ to 53% for $\rho = .8$ and again this could arise from the changes made to the penalty curves or to the queue selection algorithm.

The mean relative responses by priority class, $\bar{\omega}_j$, for $L = 3$ are shown in Figure 4.6 and the tabular values are given in Table 4.7. As with the two priority class job-stream, priority one jobs receive better response than priority two jobs with the mean relative response for priority three jobs being slightly better at high loads than that of priority class two jobs. The changing of the step point from 1.5 to 2.0 and then to 3.0 brings about a deterioration in the range of 5% of the mean relative response for $j = 1$. For $j = 2$ and 3 , the changes in relative response are negligible.

FIGURE 4.5

$\bar{\omega}_j$ as a function of ρ for $L=2$ $\phi = 1.5, 2.0, 3.0$

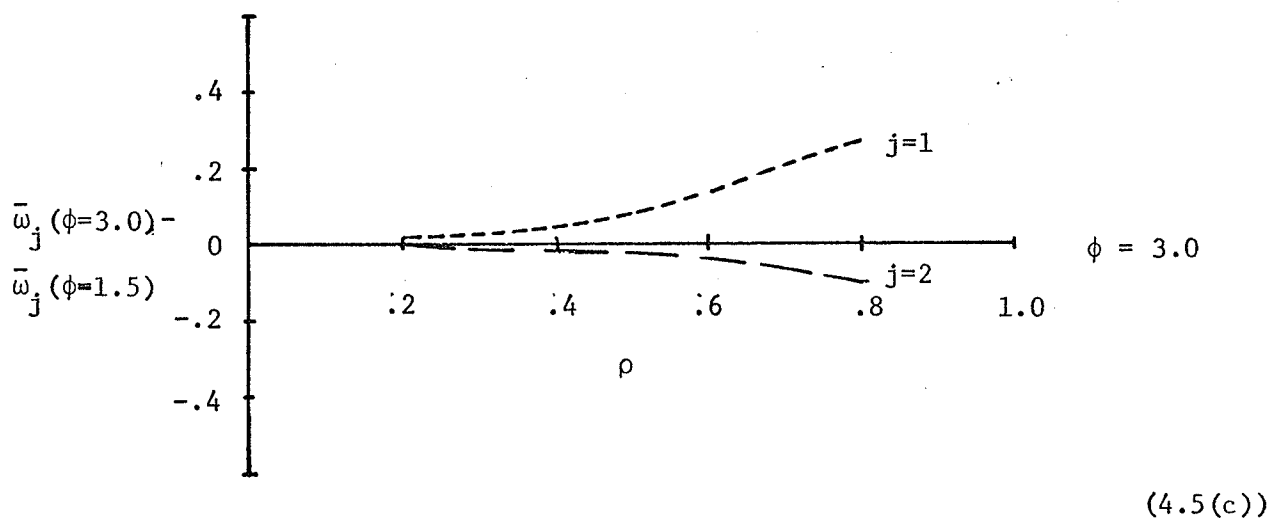
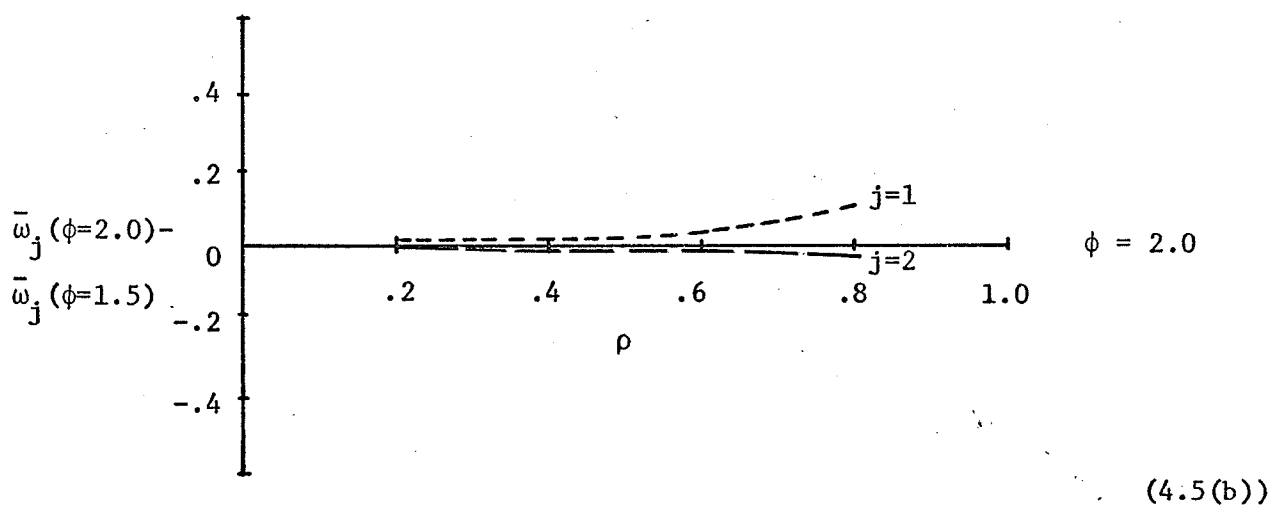
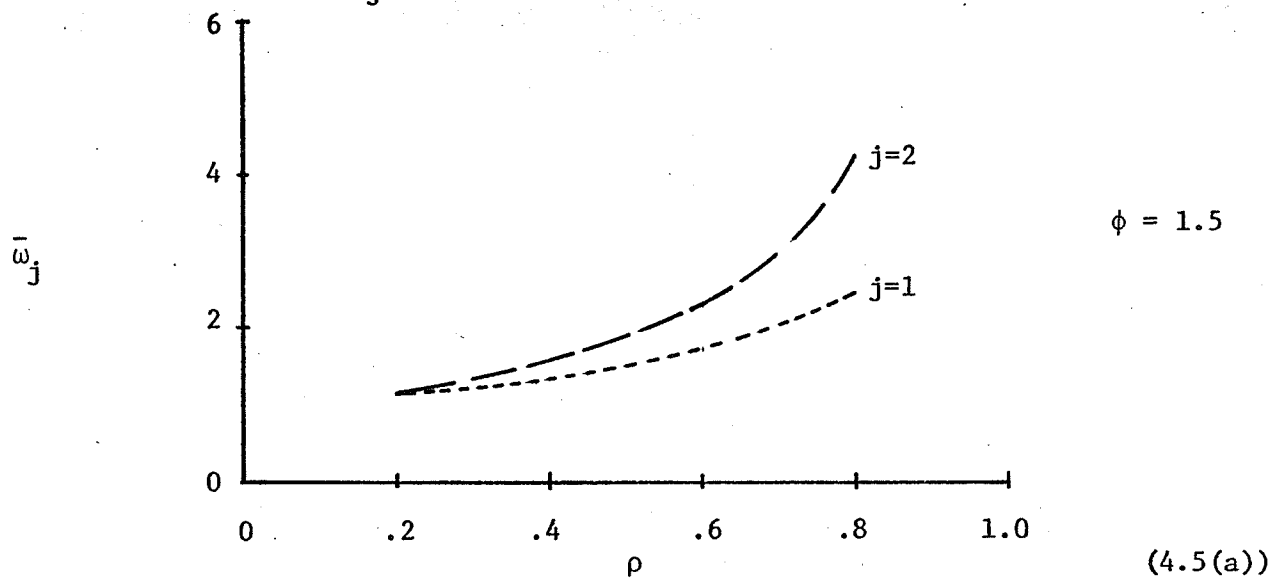


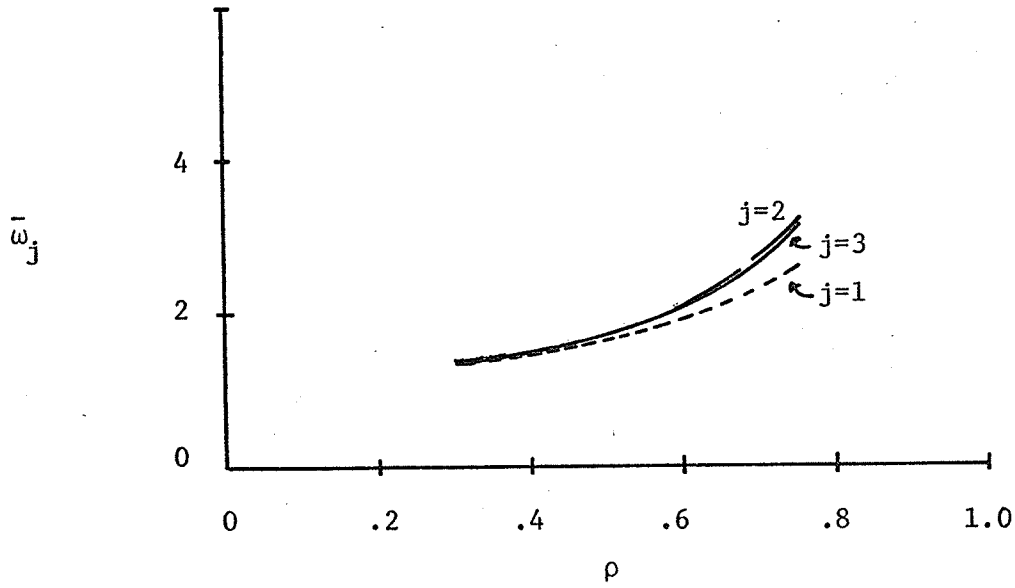
TABLE 4.6

 $\bar{\omega}_j, \omega_j'$ FOR $L = 2$ $\phi = 1.5, 2.0, 3.0$

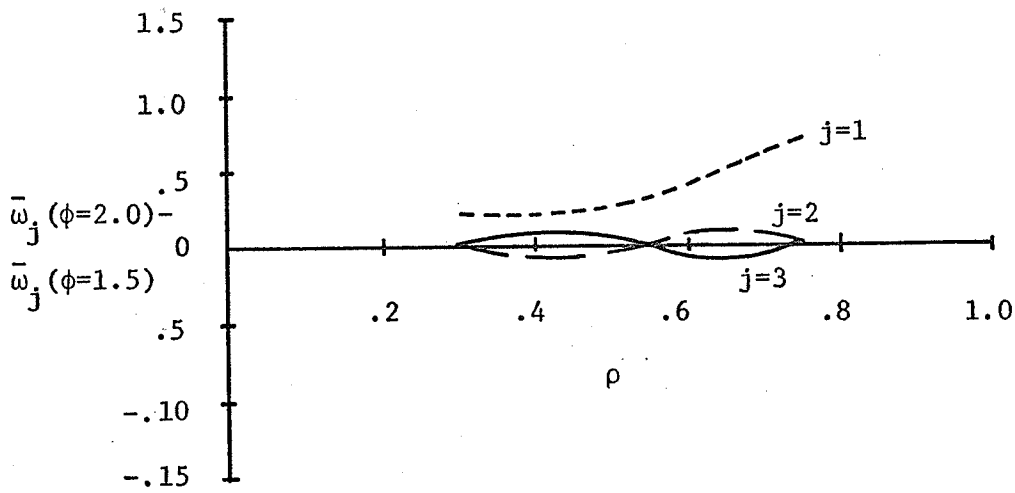
$\phi = 1.5$					
$j = 1$			$j = 2$		
\bar{m}_1	ρ	$\bar{\omega}_1$	ω_1'	$\bar{\omega}_2$	ω_2'
2	.2	1.15	.38	1.16	.31
4	.4	1.34	.62	1.58	.53
6	.6	1.70	1.03	2.28	1.24
8	.8	2.45	1.78	4.25	2.94
$\phi = 2.0$					
2	.2	1.16	.39	1.16	.31
4	.4	1.35	.62	1.57	.72
6	.6	1.74	1.00	2.27	1.25
8	.8	2.56	1.74	4.22	2.96
$\phi = 3.0$					
2	.2	1.17	.41	1.16	.31
4	.4	1.38	.64	1.56	.72
6	.6	1.84	1.07	2.24	1.24
8	.8	2.73	1.67	4.15	2.99

FIGURE 4.6

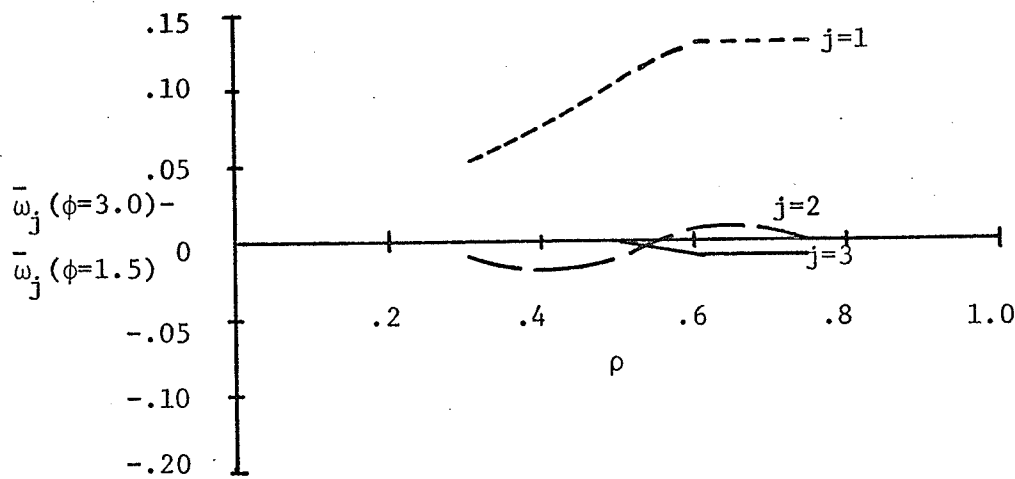
$\bar{\omega}_j$ as a function of ρ $L = 3$ $\phi = 1.5, 2.0, 3.0$



(4.6(a))



(4.6(b))



(4.6(c))

TABLE 4.7

 $\bar{\omega}_j, \omega'_j$ FOR $L = 3, \phi = 1.5, 2.0, 3.0$

$\phi = 1.5$							
		$j = 1$		$j = 2$		$j = 3$	
\bar{m}_1	ρ	$\bar{\omega}_1$	ω'_1	$\bar{\omega}_2$	ω'_2	$\bar{\omega}_3$	ω'_3
2	.30	1.30	.64	1.33	.57	1.36	.39
3	.45	1.50	1.08	1.66	.89	1.66	.60
4	.60	1.88	2.06	2.18	1.39	2.16	.91
5	.75	2.59	2.97	3.14	2.23	3.09	2.41
$\phi = 2.0$							
2	.30	1.32	.65	1.33	.56	1.36	.39
3	.45	1.52	1.09	1.65	.85	1.67	.62
4	.60	1.92	2.06	2.19	1.44	2.15	.92
5	.75	2.66	3.22	3.14	2.25	3.09	1.56
$\phi = 3.0$							
2	.30	1.35	.70	1.32	.56	1.36	.39
3	.45	1.59	1.12	1.64	.85	1.66	.62
4	.60	2.01	2.00	2.19	1.54	2.15	.92
5	.75	2.72	2.74	3.14	2.28	3.08	1.56

4.8 EFFECTS OF CHANGING THE QUEUE SELECTION ALGORITHM

Two changes were made in going from the results of Section 4.6 to the results of Section 4.7 and consequently it is not completely clear which of the changes gave rise to the observed improvements, although it is known from previous results [11] that changing the queue selection algorithm can produce improvements of the size observed.

In order to determine the effects of the two changes separately, a further calculation was carried out using the penalty curves defined in Section 4.2:

$$\alpha_i(t) = 1 \quad u_i(t) \leq \phi, \quad \dots(4.8.1)$$

or

$$\alpha_i(t) = 1000 \quad u_i(t) > \phi, \quad \dots(4.8.2)$$

where $u_i(t)$ is the current lack of attention of job i at time t and ϕ is the chosen step point. The penalty curves were the same for each priority class of jobs. Two step points, $\phi = 1.5$ and $\phi = 3.0$ and the first-come-first-served by priority queue selection algorithm were used. The results are given in Table 4.8.

An improvement of 7 to 10% was observed in the relative response of the system as the step point was moved from 1.5 to 3.0. The relative response of priority class one jobs was observed to improve 10 to 20% depending on the system load. The mean relative responses for priority class two jobs were found to change very little while priority class three jobs experienced a decrease in service of approximately 5% as the step point was moved from 1.5 to 3.0.

Figure 4.7 shows the $\bar{\omega}$ values plotted as functions of ρ with $\phi = 1.5$ and 3.0, $L = 3$ for the three combinations of queue selection algorithms and penalty curves. Curve (a) is for results calculated using the first-come-first-served by penalty queue selection algorithm with equal penalty curves

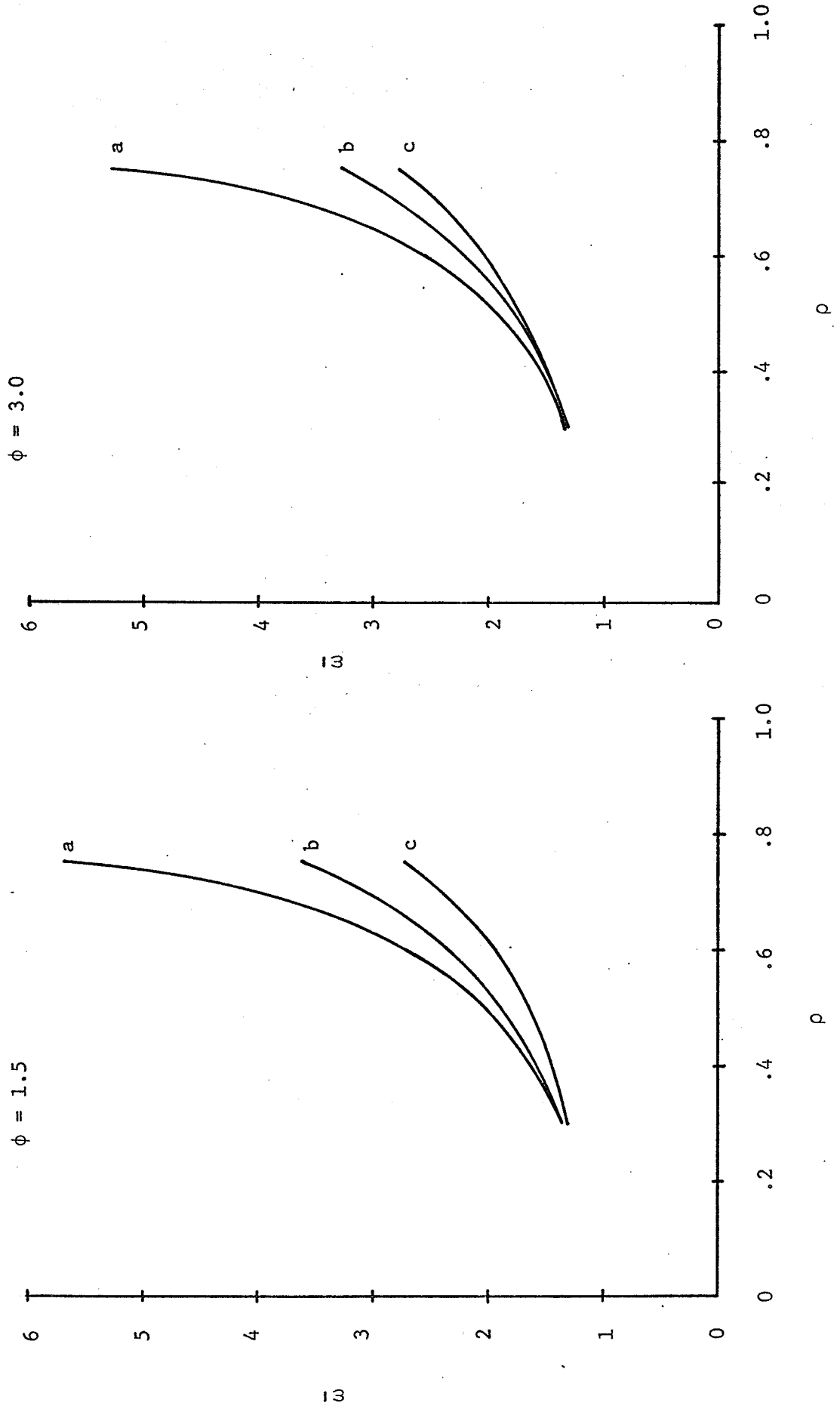
TABLE 4.8

 $\bar{\omega}$, ω' , $\bar{\omega}_j$ AND ω'_j FOR $L = 3$, $\phi = 1.5$ AND 3.0

$\phi = 1.5$									
\bar{m}_1	ρ	$\bar{\omega}$	ω'	$\bar{\omega}_1$	ω'_1	$\bar{\omega}_2$	ω'_2	$\bar{\omega}_3$	ω'_3
2	.3	1.37	.80	1.38	.87	1.31	.51	1.38	.37
3	.45	1.72	1.37	1.75	1.49	1.62	.88	1.66	.60
4	.60	2.37	2.30	2.43	2.48	2.17	1.65	2.09	.89
5	.75	3.64	3.38	3.79	3.59	3.19	2.64	2.84	1.43
$\phi = 3.0$									
2	.3	1.35	.66	1.35	.70	1.30	.48	1.39	.43
3	.45	1.62	1.03	1.62	1.11	1.60	.71	1.71	.63
4	.60	2.19	2.16	2.21	2.34	2.12	1.46	2.19	.87
5	.75	3.29	3.07	3.35	3.25	3.14	2.55	2.97	1.42

FIGURE 4.7

$\bar{\omega}$ as a function of ρ $L = 3$ for three combinations of queue selection algorithms and penalty curves



for each priority class (Table 4.2). Curve (b) is for the first-come-first-served by priority algorithm with identical penalty curves for each priority class. Curve (c) is obtained from results calculated when the first-come-first-served by priority queue selection algorithm was used and the penalty curves were those of Section 4.7.

Comparison of the results obtained using the first-come-first-served by penalty algorithm (Curve (a)) and the first-come-first-served by priority algorithm (Curve (b)) show that for a load of .75, the improvement in performance due to the use of the latter queue selection algorithm was 35% and 38% for $\phi = 1.5$ and 3.0, respectively. When the penalty curves defined in Section 4.7 were used, the improvements were 52% for $\phi = 1.5$ and 47% for $\phi = 3.0$ for $\rho = .75$. Thus improvements of 17% and 9% were made by using the penalty curves of Section 4.7. These results would suggest that under these particular circumstances, where priority one jobs are the most numerous and sensitive to delay, that system performance can be improved by concentrating on providing the best service to the highest priority jobs.

Examination of the relative responses by priority class for $L = 3$ when the queue selection algorithm is based on penalty (Figure 4.3) shows that the order of response by priority class is 3, 2, 1 with priority three jobs obtaining better service. In contrast, when the queue selection algorithm is first-come-first-served by priority and the penalty curves for priorities 2 and 3 are constant and equal to one, the order of increasing relative responses becomes 1, 2, 3 at low loads and 1, 3, 2 at high loads, with priority one jobs receiving better service throughout. When the penalty curves of each priority class are the same and the queue selection algorithm is first-come-first-served by priority, the order of increasing relative responses becomes 2, 3, 1 for $\phi = 1.5$ at low loads, 3, 2, 1 for $\phi = 1.5$ at high loads

and 2, 1, 3 for $\phi = 3.0$ for low loads and 3, 2, 1 at high loads. In all cases the priority one jobs receive poorer relative response than the lower priority jobs.

Thus it appears that the order of relative responses of the individual priority classes is a function of the system load, the penalty curves used and the queue selection algorithm.

The largest changes in the relative responses of the individual priority classes of jobs were obtained when both the queue selection algorithm and the penalty curves were changed. Table 4.9 gives the percentage changes made for loads of .3 and .75 when the queue selection algorithm and the penalty curves were both changed.

4.9 THE VARIANCE OF THE RELATIVE RESPONSE FOR CHANGING PENALTY CURVES

The measure of performance used, the mean relative response has been seen to change but little ($\pm 10\%$) for changes in the step point for both of the queue selection algorithms used. However, it may be considered that system performance is more satisfactory if the standard deviations of the mean relative responses have been decreased even though the $\bar{\omega}$ and $\bar{\omega}_j$ values are not changed very much as the step point changes.

A multi-priority job-stream with $L = 3$ is probably the most interesting case for examination of the variance of the mean relative responses since it most closely resembles a real system. Also, under the present circumstances the priority one jobs are the most numerous and sensitive to delay. Thus for these reasons the standard deviations of the ω_i values for $j = 1$, $L = 3$ and $\phi = 1.5, 2.0$ and 3.0 for the two queue selection algorithms were taken from the results obtained for Sections 4.6 and 4.7 and plotted in Figure 4.8 to determine if an improvement was made in terms of decreased standard deviations for the priority one jobs.

TABLE 4.9
 PERCENTAGE IMPROVEMENTS IN $\bar{\omega}_j$ VALUES FOR QUEUE SELECTION BY PRIORITY
 OVER QUEUE SELECTION BY PENALTY

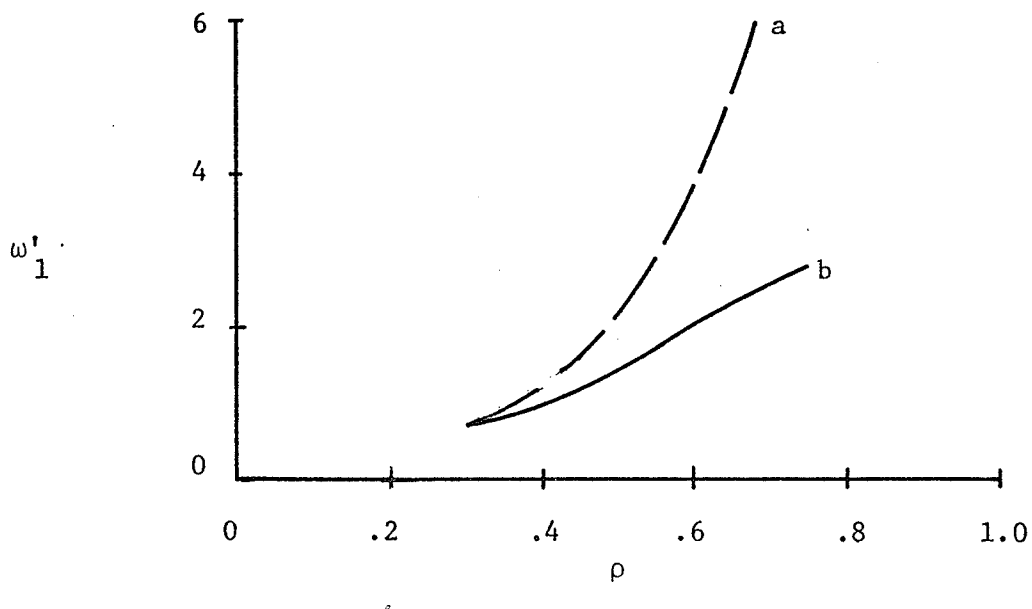
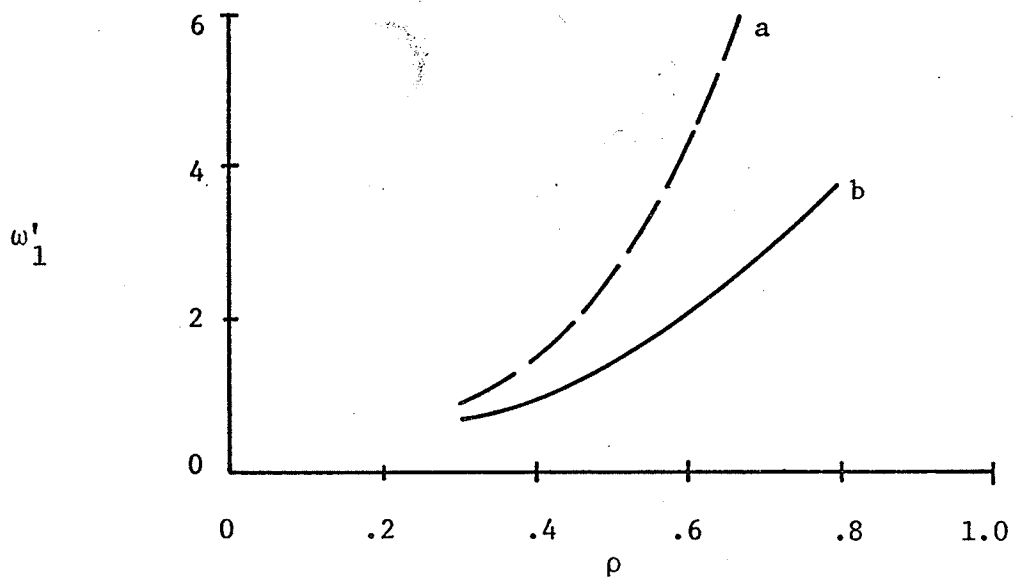
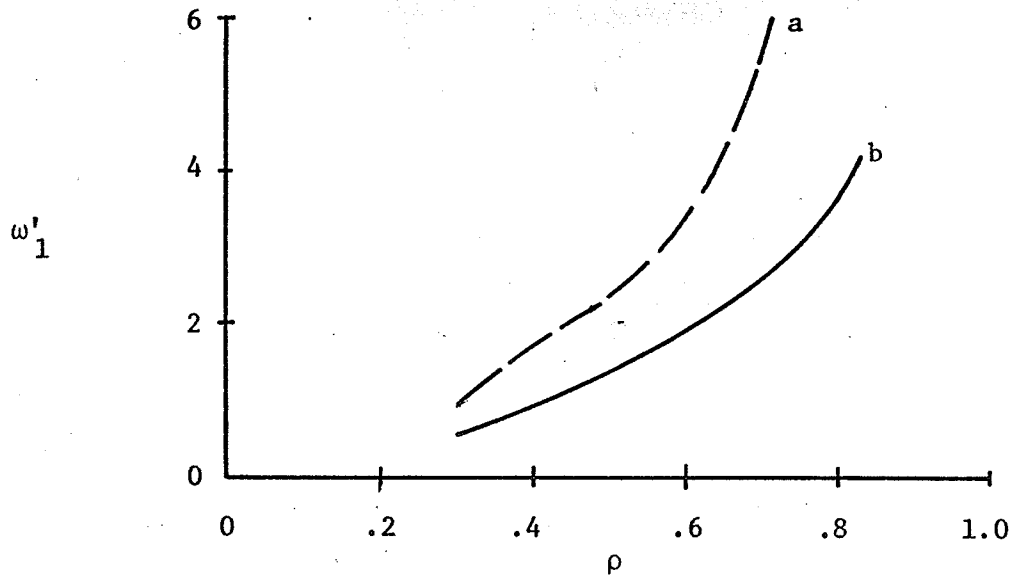
$\phi = 1.5$									
ρ	$j = 1$			$j = 2$			$j = 3$		
	$\bar{\omega}_1^a$	$\bar{\omega}_1^b$	%	$\bar{\omega}_2^a$	$\bar{\omega}_2^b$	%	$\bar{\omega}_3^a$	$\bar{\omega}_3^b$	%
.3	1.30	1.41	+8	1.33	1.21	-9	1.36	1.36	0
.75	2.59	6.35	+59	3.14	3.73	+16	3.09	2.64	-14
$\phi = 2.0$									
.3	1.32	1.36	+2	1.33	1.30	-2	1.36	1.39	+2
.75	2.66	6.19	+57	3.14	3.24	+3	3.09	2.68	-13
$\phi = 3.0$									
.3	1.35	1.36	+1	1.32	1.30	-1	1.36	1.39	+2
.75	2.72	5.94	+54	3.14	3.22	+3	3.08	2.76	-11

a Queue selection algorithm is first-come-first-served by priority class.
 Penalty curves are the same for each priority class.

b Queue selection algorithm is on the basis of first-come-first-served by penalty. $\alpha_i(t) = 1$ $u_i(t) \leq \phi$, $\alpha_i(t) = 1000$ $u_i(t) > \phi$ for $\gamma_i = 1$ and $\alpha_i(t) = 1$ for $\gamma_i = 2$ and 3.

FIGURE 4.8

ω'_1 as a function of ρ for the two queue selection algorithms $L = 3$



Examination of the graphs shows that changing the step point has little effect on the standard deviations. The decrease in the standard deviation of the priority one jobs due to the changing of the penalty curves and the queue selection algorithm is of the same order of magnitude as the improvements affected to the $\bar{\omega}$ and $\bar{\omega}_j$ values. The first-come-first-served by priority queue selection algorithm tends to produce a more uniform set of relative responses than the first-come-first-served by penalty algorithm. The only exception to this is when $\phi = 3.0$ and the system load, ρ , is equal to .3. Under these conditions, the two algorithms produce approximately the same results. A sample distribution of the relative responses is given in Appendix F.

4.10 CONCLUSIONS

The results obtained from the simulation runs in which the same penalty curve was used by all priority classes (Section 4.5) indicate that the effect of changing the step point is slight for all the various system loads used with the changes being in the order of 7% for $L = 3$, 2% to 5% for $L = 2$ and negligible for $L = 1$.

In Section 4.6 the effects of changing the penalty curve upon the individual priority classes of jobs in a multi-priority job-stream were examined, using the first-come-first-served by penalty queue selection algorithm. For $L = 2$ the changing of the step point from $\phi = 1.5$ to $\phi = 3.0$ improved the service to priority one jobs by about 2% and decreased the relative response of priority one jobs by 7%. It is perhaps noteworthy that for all step points, the lower priority jobs ($j = 2$) received better relative responses than the higher priority jobs for all values of ρ greater than about .5.

In the case of the job-streams where L was equal to 3, the effects of changing the penalty functions was most noticeable for heavy system loads

with improvements in the relative response for priority one and two jobs in the order of 6% and 13%, respectively. Also the order of service was found to be 3, 2, 1 at heavy loads ($\rho = .75$) with the low priority jobs ($j = 3$) receiving the best service. When the system load was less than about .45, the relative responses were found to be approximately the same for all priority classes.

Section 4.7 examined the effects of changing penalty curves upon multi-priority job-streams in which the penalties of priority classes two and three were constant and equal to one. The effect of changing the step point from $\phi = 1.5$ to $\phi = 3.0$ was found to be to improve the performance of the system by a small amount (4% $L = 2$, 7% $L = 3$). Examination of the \bar{w}_j values calculated in Section 4.7 showed that with $L = 2$ the changing of the penalty curves improved the relative responses for priority two jobs by 2% and deteriorated the service to priority one jobs by about 11%. For $L = 3$ the effect of changing the penalty curve on the priority two and three jobs was negligible. Priority one jobs experienced a decrease in service in the order of 5%.

The simulation model used to obtain the results of Section 4.7 had a different queue selection algorithm and the penalty curves for priority two and three jobs were defined differently than those used for Sections 4.5 and 4.6. The changing of the selection algorithm and the penalty curves for $j = 2$ and 3 affected a major improvement of 55 to 60 percent of all step points for priority one jobs. Also, the order of relative responses for the individual priority classes changed giving the best service to priority one jobs. This was on contrast to the previous simulation runs (Sections 4.5 and 4.6), in which priority three jobs received the best service.

In Section 4.8 the penalty curves of Section 4.5 were used with:

$$\left. \begin{array}{l} \alpha_i(t) = 1 \quad u_i(t) \leq \phi \\ \alpha_i(t) = 1000 \quad u_i(t) > \phi \end{array} \right\} \gamma_i = 1 ,$$

and $\alpha_i(t) = 1, \gamma_i = 2,3 .$

The queue selection algorithm was that of Section 4.7, first-come-first-served by priority. The results calculated indicated that 9% to 17% (dependent on ϕ) of the improvement made in system performance in Section 4.7 was due to the change in the penalty curves for priorities two and three. Examination of the $\bar{\omega}_j$ values showed that the order of the responses for the different priority classes was influenced by the queue selection algorithm, the penalty curves and the system load.

Examination of the standard deviations of the highest priority jobs (Section 4.9) indicated that the improvements in the relative responses in terms of decreased variance, due to changing the queue selection algorithm and the penalty curves for priority two and three jobs, was of the same order of magnitude as the improvements made to the $\bar{\omega}$ values by these changes.

In summary, in the given circumstances, some change is to be had by movement of the step point but major improvements in system performance can be gained by basing the queue selection algorithm on priority and by concentrating on servicing the high priority jobs by altering the penalty curves of the low priority jobs so that the high priority jobs obtain the largest quanta.

REFERENCES

1. Nielsen, N.R., The Simulation of Time Sharing Systems, Comm. ACM Vol. 10, No. 7 (July 1967), pp 397-412.
2. Scherr, A.L., An Analysis of Time Shared Computer Systems, MAC-TR-18, M.I.T. Project MAC, Cambridge, Mass., (1965).
3. Chai, L.A., Study of the Performance of a Scheduling Algorithm for a Time-Slicing Supervisor, M.Sc. Thesis, University of Manitoba (1968).
4. Greenberger, M., The Priority Problem and Computer Time Sharing, Management Science 12, (July 1966), pp 888-906.
5. Feller, W., An Introduction to Probability and Its Applications, 2nd ed., John Wiley, New York (1957).
6. Fife, D., An Optimization Model for Time Sharing, A.F.I.P. Conference Proceedings, Vol. 28 (1966) pp 97-104.
7. Coffman, E.G. and Kleinrock, L., Computer Scheduling Methods and Their Countermeasures, A.F.I.P.S. Conference Proceedings, 32 (SJCC 1968).
8. Abate, J., Dubner, H., Weinburg, S.B., Queueing Analysis of the IBM 2314 Disk Storage Facility, J.ACM (October 1968), pp 577-589.
9. Rourke, T.A., Clark, S.R., Wren, J.M., Investigation of the Reproducibility of Simulation Studies of Computer Systems, (to be published).
10. Deming, W.E., Some Theory of Sampling, John Wiley, New York (1950), 526.
11. McDonald, B.H., Study of Resource Allocation in Computer Systems by Simulation, M.Sc. Thesis, University of Manitoba (1969).
12. Naylor, T.H., Balintfy, J.L., Burdick, D.S., and Chu, K., Computer Simulation Techniques, John Wiley, New York (1966), pp 49-54.
13. Blatny, J., (private communication).

APPENDIX A

MONTE CARLO TECHNIQUES USED BY THE JOB GENERATOR

A.1 INTRODUCTION

In the research described, the central processor request times were taken to be normally distributed about given mean values, with given standard deviations. The arrival times were taken to have a Poisson distribution, requiring the interarrival times to follow an exponential distribution [5]. The power residue or multiplicative congruential method was used to generate random numbers with a rectangular distribution and these numbers were transformed as required, to either a normal distribution or an exponential distribution.

A.2 THE POWER RESIDUE METHOD

The method uses a constant k , a starting value, n_0 , and a modulus m to generate a sequence, (n_i) , of non-negative, randomly distributed integers, each of which is less than the modulus, m . The sequence is generated by means of the recursive formula:

$$n_{i+1} = kn_i \pmod{m}, \quad \dots(\text{A.2.1})$$

where $k = 65539$ and $m = 2^{31}$.

Any positive, odd integer less than $2^{31}-1$ may be used as the initial value, n_0 .

To obtain a real number, y_i , randomly distributed in the interval $(0,1)$ a further calculation is made using the formula:

$$y_i = n_i / (2^{31}-1), \quad \dots(\text{A.2.2})$$

where n_i is obtained from equation (A.2.1).

A further discussion of the power residue method is given in reference [12].

A.3 GENERATION OF NORMAL RANDOM VARIATES

The Central Limit Theorem is used in the generation of a normal random variate, x_i , with a given mean, m_x and a given standard deviation s_x , in the formula:

$$x_i = s_x (12/j)^{1/2} \left(\sum_{i=1}^j y_i - j/2 \right) + m_x, \quad \dots (A.3.1)$$

where y_i is a uniformly distributed, random, real number between 0 and 1 determined by the power residue method (Section A.2), and j is the number of new values of y_i to be used in the generation of x_i . To reduce execution time j was chosen as 12 and thus equation (A.3.1) reduces to:

$$x_i = s_x \sum_{i=1}^{12} (y_i - 6.0) + m_x \quad \dots (A.3.2)$$

According to the Central Limit Theorem, the set of values, x_i , approach a true normal distribution asymptotically as j approaches infinity [12].

A.4 THE GENERATION OF EXPONENTIALLY DISTRIBUTED VARIATES

Exponentially distributed variates are generated by taking a uniformly distributed, random, real number, y_i between 0 and 1, and transforming it to an exponentially distributed variate z_i by means of the equation [13]:

$$z_i = m_z \log_e (1/y_i), \quad \dots (A.4.1)$$

where m_z is the mean value of the exponentially distributed variates z .

The probability that y_i will fall in a particular range of size Δy is equal to the area under the constant probability curve and may therefore be expressed as $c\Delta y$. All y_i values in this range are transformed to values of z_i and they will occur in a particular range of size Δz . The probability density function at this range may be represented as $P(z)$ and it follows that:

$$P(z) \Delta z = c\Delta y \quad \dots (A.4.2)$$

and thus,

$$\lim_{\Delta y \rightarrow 0} \frac{\Delta y}{\Delta z} = \frac{P(z)}{c} = \frac{dy}{dz} \quad \dots (A.4.3)$$

From equation (A.4.1)

$$\frac{dy}{dz} = \frac{1}{m_z} e^{-z/m_z}, \quad \dots (A.4.4)$$

and since the integral of $P(z)$ over the range $(0, \infty)$ is equal to unity, c must have the value -1 . Therefore:

$$P(z) = \frac{1}{m_z} e^{-z/m_z} \quad \dots (A.4.5)$$

From equation (A.4.5) the probability of a value z less than T can be found

from:

$$\begin{aligned} P(z < T) &= \int_0^T P(z) dz = \left[-e^{-z/m_z} \right]_0^T \\ &= 1 - e^{-T/m_z}. \end{aligned} \quad \dots (A.4.6)$$

A.5 SUMMARY

The central processor request times which are normally distributed random variates with a given mean and standard deviation are generated by applying the Central Limit Theorem (Section A.3) to twelve uniform random numbers in the range $(0,1)$, obtained by the power residue method (Section A.2). The interarrival times, which are exponentially distributed variates with a given mean are generated by applying the transformation described in Section A.4 to uniform random numbers in the range $(0,1)$ (Section A.2).

APPENDIX B
SAMPLE JOB STREAM DATA

The first few jobs generated for a three priority class job stream are shown. The job-mix and operational parameters used to generate this job stream are given in Table 3.1.

PRIORITY CLASS ONE JOBS

ARRIVAL TIME (seconds)	CPU TIME REQUEST (seconds)
2.855	.342
25.277	.809
27.999	.611
39.102	.888
69.057	.762
82.401	1.417
99.913	1.124
106.717	1.302
110.960	1.005
119.487	1.006
123.985	.999
124.345	.975
129.525	1.360
151.892	.736
153.940	.793
156.534	.958
157.204	.921
159.958	.567

PRIORITY CLASS TWO JOBS

ARRIVAL TIME (seconds)	CPU TIME REQUEST (seconds)
18.600	4.555
65.661	3.623
100.166	5.357
122.877	3.017
140.203	4.767
168.842	4.648
203.736	3.678
242.837	3.184
302.607	2.883
296.020	2.883
308.115	4.570
327.728	3.319

PRIORITY CLASS THREE JOBS

ARRIVAL TIME (seconds)	CPU TIME REQUEST (seconds)
13.384	17.301
95.339	8.306
102.075	12.183
276.362	17.128
452.974	22.431
531.685	15.253
587.726	23.007
645.158	16.839
707.240	18.563
717.864	19.123
717.128	23.521
959.588	15.971

APPENDIX C

DATA MAINTAINED ON JOBS PROCESSED BY THE SIMULATION MODEL

The three lists maintained in the simulation model; the generation list, the execution list and the waiting queue are each made up of the following sub-lists:

- (1) the job number,
- (2) the job priority class,
- (3) the arrival time of the job,
- (4) the CPU time requested.

As well as the above the execution list maintains a record of the quanta allotted to each job, the penalties attributed to each job and the CPU time received by each job during each round-robin cycle.

APPENDIX D

SAMPLE SERIES OF AUTOCORRELATED FINAL PENALTIES

The sample data presented on the following page illustrates the autocorrelation that can occur between final penalties when a temporary heavy load is imposed upon the simulated system by arrival of large number of jobs in a short interval. The mean final penalty for the complete job stream from which this table was extracted was 142.40. The number of jobs in the job stream was 1600 and the operational and job-mix parameters used were:

$$t_{rr} = 6.00 \text{ seconds,}$$

$$t_m = 0.06 \text{ seconds,}$$

$$t_s = 0.01 \text{ seconds,}$$

$$L = 3,$$

$$n_q = 20,$$

$$n_e = 3,$$

$$\bar{m}_1 = 1.0 \text{ seconds,}$$

$$\bar{r}_1 = 6.0 \text{ seconds,}$$

$$k = 4,$$

$$t_d = 1.0 \text{ seconds.}$$

SAMPLE DATA

PRIORITY CLASS	ARRIVAL TIME	CPU TIME REQUEST	FINAL PENALTY
1	450.075	.746	1.670
2	452.856	2.478	1.058
2	453.344	4.120	2.656
1	456.219	1.223	337.690
2	456.902	3.398	16.624
1	459.577	.905	697.658
1	460.500	1.308	97.904
2	460.603	4.518	79.320
1	460.800	.856	906.115
1	467.111	.903	59.773
2	467.115	3.894	125.143
1	467.385	.957	880.006
1	468.868	1.675	879.793
1	471.891	1.225	905.529
1	474.955	1.280	826.782
1	478.133	1.427	622.389
1	480.037	.938	696.449
1	481.258	1.650	461.516
1	484.806	1.337	147.529
1	489.295	1.477	2.102
1	489.646	.989	24.254
1	489.966	1.151	35.341
1	492.811	1.179	5.299
1	497.574	.889	1.619

APPENDIX E

SAMPLE MEAN FINAL PENALTIES AND STANDARD DEVIATIONS

n	L = 1		L = 2		L = 3	
	P _n	S _n	P _n	S _n	P _n	S _n
50	5.78	7.11	9.70	25.65	204.98	301.72
	3.59	5.73	16.23	62.46	144.14	273.41
	4.66	7.27	21.33	80.87	112.52	251.46
	2.38	2.71	31.47	131.00	49.10	93.66
	2.54	3.59	159.53	318.85	166.03	281.21
	3.41	5.79	66.99	163.78	78.31	205.33
	9.29	34.77	22.04	78.32	152.58	264.66
	2.41	2.47	43.48	149.81	171.08	325.33
	2.32	2.69	49.87	130.35	119.45	253.15
	3.03	4.02	30.64	110.73	173.72	308.21
800	4.06	10.37	57.89	173.59	134.23	258.39
	3.83	6.35	51.75	156.55	99.82	233.55
	4.54	7.70	42.06	136.93	141.66	276.52
	4.33	12.99	51.77	163.01	107.84	250.30
	4.64	12.15	40.01	132.57	127.90	262.22
	6.33	23.18	66.55	182.98	97.68	226.53
	4.82	13.46	49.47	157.18	138.21	281.85
	5.29	30.17	45.90	147.35	108.03	233.51
	3.97	5.98	46.00	143.20	165.49	295.48
	4.64	10.54	53.02	156.37	114.78	254.27
1600	4.10	9.07	56.02	169.00	98.80	232.39
	4.51	10.16	39.79	135.82	130.74	264.25
	5.28	25.85	38.97	132.39	157.46	295.32
	4.56	12.78	52.74	157.80	106.10	240.92
	4.65	16.76	46.24	146.84	144.49	283.47
	4.28	14.51	50.22	155.79	147.49	275.89
	4.67	18.35	44.48	145.04	118.16	250.76

continued...

(continued...)

n	L = 1		L = 2		L = 3	
	P_n	S_n	P_n	S_n	P_n	S_n
4.61	15.20	50.69	155.86	135.25	270.82	
4.15	7.96	43.03	143.46	149.36	280.61	
4.67	15.69	40.28	136.88	140.09	280.76	

APPENDIX F

SAMPLE DISTRIBUTION OF RELATIVE RESPONSES

The distribution presented is for a three priority class job stream with a system load of .6 and using the penalty curve with ϕ equal to 1.5.

UPPER CLASS LIMIT	PRIORITY 1	PRIORITY 2	PRIORITY 3	TOTAL
2.729	3277	976	255	4508
4.458	769	171	32	972
6.187	308	50	2	360
7.916	165	20	0	185
9.645	103	12	0	115
11.374	60	5	0	65
13.103	61	0	0	61
14.832	40	1	0	41
16.561	32	0	0	32
18.290	24	0	0	24
20.019	13	0	0	13
21.748	11	0	0	11
23.477	5	0	0	5
25.206	3	0	0	3
26.935	2	0	0	2
28.664	1	0	0	1
30.393	0	0	0	0
32.122	1	0	0	1
33.851	0	0	0	0
∞	1	0	0	1
TOTALS:	4876	1235	289	6400

APPENDIX G

COMPARISON OF THE PERFORMANCE OF TWO EXECUTION LIST SERVICING ALGORITHMS

In Chapter IV two algorithms were used to determine the order in which jobs in the execution list with processing time available were serviced. A description of the selection algorithms is given below.

- (1) The first algorithm used selected jobs for servicing in the order of decreasing penalty. That is, the job with the highest penalty was serviced first, then the job with the second highest penalty was serviced next and so on. In the event of two jobs having equal highest penalties, the job that had the earliest arrival time was selected for servicing. If two jobs had equal earliest arrival times, the job with the lowest execution slot number was selected for servicing. The probability of of two equal earliest arrival times was however, quite remote since seven decimal digits of precision was maintained for arrival times.
- (2) The second algorithm used selected jobs for servicing in the order of increasing execution slot numbers. Thus the first job encountered with processing time available, in a scan of the execution list in the direction of increasing slot numbers, was selected for servicing.

In order to compare the two algorithms, two simulation runs were made under identical conditions except that the execution list selection algorithm was changed from one run to the next. Table G.1 gives the operational and job-mix parameters used in the two simulation runs. The penalty curves were defined so that for any job i at time t :

$$\alpha_i(t) = 1 \quad u_i(t) \leq \phi,$$

$$\alpha_i(t) = 1000 \quad u_i(t) > \phi,$$

for $j = 1$ where j is the priority class of job i , and $\alpha_i(t) = 1$, $j = 2$.

TABLE G.1
JOB-MIX AND OPERATIONAL PARAMETERS

JOB-MIX PARAMETERS:

Proportionality Constant	k	=	4.0
Mean Interarrival Time	\bar{r}_1	=	20.0 seconds
Number of Priority Classes	L	=	2
Starting Random Number		=	9875211

OPERATIONAL PARAMETERS:

Length of Execution List	n_e	=	3 jobs
Length of Waiting Queue	n_q	=	20 jobs
Round-Robin Cycle Time	t_{rr}	=	6.0 seconds
Minimum Time Slice	t_m	=	0.06 seconds
Supervisor Cycle Time	t_s	=	0.01 seconds
Constant Device Time	t_d	=	1.0 seconds
Number of Jobs	n	=	6400 jobs
Step Point	ϕ	=	1.5

Table G.2 gives the mean relative response, $\bar{\omega}$, and the standard deviation ω' , and the mean relative response by priority, $\bar{\omega}_j$, and its standard deviation, ω'_j . Table G.2(a) shows the values calculated using the selection algorithm based on penalties and Table G.2(b) contains the values calculated when the selection algorithm is based on slot numbers. Comparison of the two sets of figures indicates that the execution list servicing algorithm has essentially no effect on the performance of the system under these circumstances. Since the algorithm based upon execution list slot numbers is the simpler of the two and offers a saving in the run time of the simulation model, it appears to be the most preferable of the two.

TABLE G.2

(a)¹

\bar{m}_1	ρ	$\bar{\omega}$	ω'	$\bar{\omega}_1$	ω'_1	$\bar{\omega}_2$	ω'_2
2	2	1.15	.36	1.15	.38	1.16	.31
4	4	1.38	.65	1.34	.62	1.58	.73
6	6	1.81	1.10	1.70	1.03	2.28	1.24
8	8	2.80	2.18	2.45	1.78	4.25	2.94

(b)²

2	2	1.15	.36	1.15	.37	1.16	.31
4	4	1.38	.64	1.33	.61	1.58	.73
6	6	1.81	1.10	1.70	1.03	2.28	1.24
8	8	2.80	2.17	2.45	1.78	4.25	2.94

¹ Servicing of jobs was in the order of decreasing penalty, that is, the job with the highest penalty was serviced first.

² Servicing of jobs was on the basis of first found, first served, in a scan of the execution list in the direction of increasing slot numbers.

APPENDIX H
THE MEANS OF THE P_n VALUES

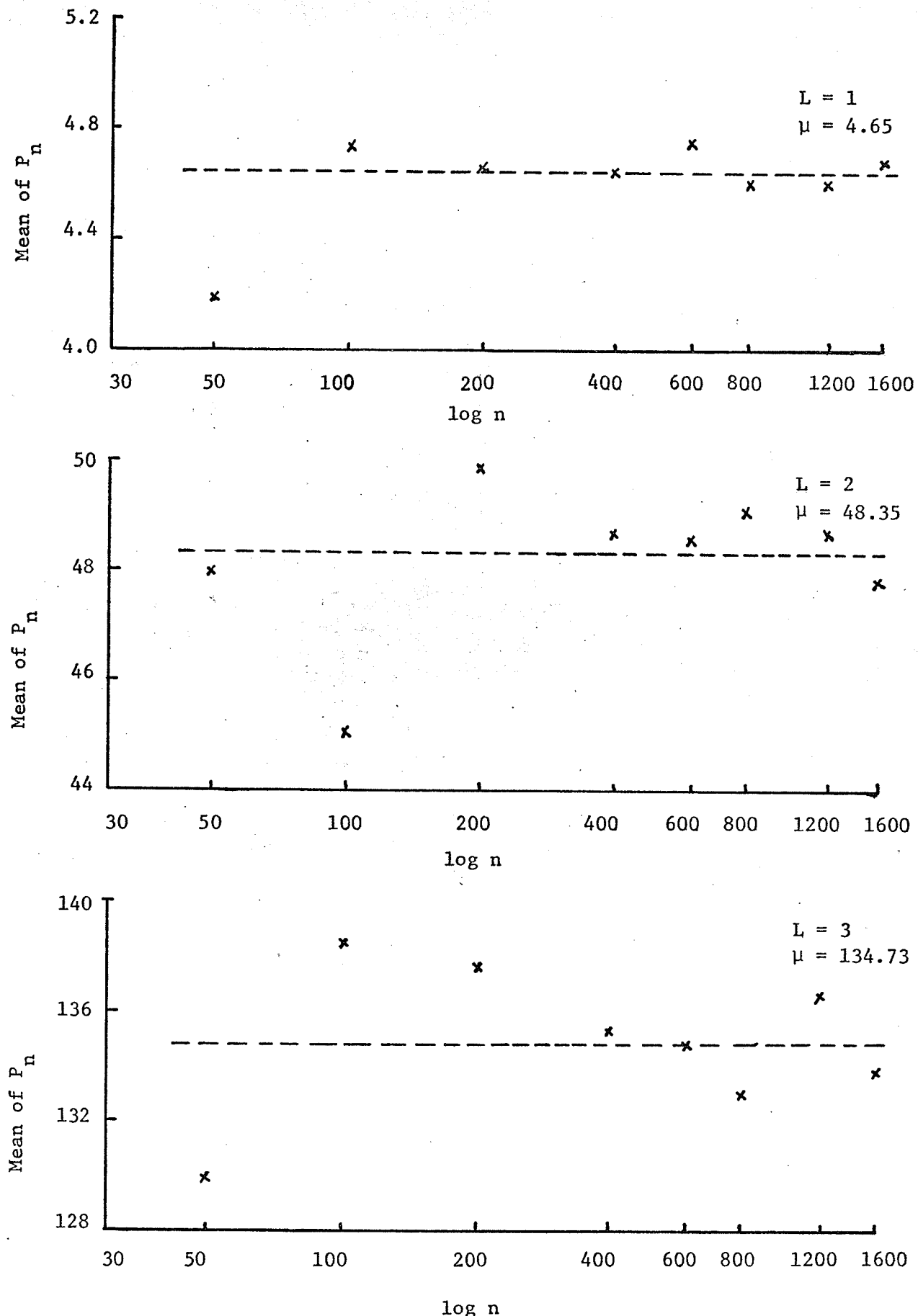
In Chapter III, the reproducibility of the results of simulation studies was examined. The measure of performance used was P_n , the mean of the final penalties attributed to the n jobs in the simulation run. The runs were replicated so that one hundred and twenty observations were obtained for each value of n . Thus 120 P_n values were determined for $n = 50, 100, 200, 400, 600, 800, 1200$ and 1600 for one, two and three priority class job-streams ($L = 1, 2, 3$).

From the 120 P_n values for each value of n the means of the P_n values were calculated and are listed in Table H.1. Figure H.1 is a scatter diagram of the values listed in Table H.1 with the population mean, μ , being shown as a broken line.

TABLE H.1
MEANS OF P_n VALUES

n	L = 1	L = 2	L = 3
50	4.19	47.99	129.94
100	4.74	45.01	138.45
200	4.66	49.77	137.63
400	4.64	48.64	135.12
600	4.75	48.50	134.70
800	4.60	49.01	132.90
1200	4.60	48.61	136.49
1600	4.68	47.76	133.78

FIGURE H.1 - SCATTER DIAGRAM OF MEAN P_n VALUES $L = 1, 2, 3$



APPENDIX I

THE DISTRIBUTIONS OF THE CENTRAL PROCESSOR TIME REQUESTS

The job-stream used as input by the simulation model was produced by means of Monte Carlo techniques (Appendix A) with the CPU time requests being normally distributed about given mean, \bar{m}_j , for each priority class j . The standard deviation of each distribution was taken to be equal to $\bar{m}_j/4$. A property of the model was that the mean CPU time request for each priority class was linked to that of the previous class by means of a proportionality constant k by the equation

$$\bar{m}_j = k \bar{m}_{j-1} . \quad \dots(I.1)$$

The mean number of jobs of a particular priority class was determined from n , the total number of jobs, L , the number of priority classes and k , the proportionality constant. For a job-stream with three priority classes ($L = 3$);

$$n_1 = k n_2 , \quad \dots(I.2)$$

and

$$n_2 = k n_3 . \quad \dots(I.3)$$

Since the total number of jobs is n ;

$$n = n_1 + n_2 + n_3 , \quad \dots(I.4)$$

or
$$n = (k^2 + k + 1) n_3 . \quad \dots(I.5)$$

Thus the proportion of jobs belonging to each priority class j is given by;

$$\frac{k^2}{k^2+k+1} , \frac{k}{k^2+k+1} \text{ and } \frac{1}{k^2+k+1} \text{ for } j = 1, 2, \text{ and } 3, \text{ respectively.}$$

The probability density function, for the normally distributed random variate x with mean μ and standard deviation σ is given by [5]

$$f(x) = \frac{1}{\sqrt{2\pi} \sigma} \exp\left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\} \quad \dots(I.6)$$

By substituting for μ , σ and various values of x , the normal frequency curves may be found.

When the input job-stream for a simulation run consists of three priority classes of jobs, there are three normal frequency curves, one for each priority class. Figure I.1 shows the normal frequency curves for $j = 1, 2$ and 3, adjusted by a factor n_j/n so as to normalize the sum of the areas under the curves to one. The value of k was 2 and the normalizing factors used were:

4/7 for $j = 1$,

2/7 for $j = 2$,

and 1/7 for $j = 3$.

The mean CPU time request for priority class one was taken to be one.

FIGURE I.1 - NORMAL FREQUENCY CURVES $L=3, k=2, \bar{m}_1=1.$ 