NRL Report 6549

The Throughput and Cost Effectiveness of Monoprogrammed, Multiprogrammed, and Multiprocessing Digital Computers

BRUCE WALD

Intercept and Signal Processing Branch
Electronic Warfare Division

April 17, 1967

NAVAL RESEARCH LABORATORY
Washington, D.C.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

This report is a thesis that is being issued as an interim report on NRL Problem R06-07, Project RF-001-08-41. The manuscript was submitted for an NRL Report on February 23, 1967.

ABSTRACT

Title of Thesis: The Throughput and Cost Effectiveness of Monoprogrammed, Multiprogrammed, and Multiprocessing Digital Computers

Bruce Wald, Doctor of Philosophy, 1967

Thesis directed by: Associate Professor Alan B. Marcovitz

A generalized model of a digital computer system with a workload drawn from a proffered infinite workload was investigated to determine the factors that affect throughput and cost effectiveness to allow optimization of the system parameters. Throughput was defined as the sum of the products of the cost of each component and the proportion of the time each component was beneficially used. Cost effectiveness was defined as the quotient of throughput and system cost, with unity being the ideal quotient.

Each of the jobs constituting the workload consisted of a fixed number of segments, each alternately requiring processing and output. The memory requirements of a segment consisted of a fixed component required at all times, another fixed component required only during processing, and a third component required at all times. This third component was usually assumed to be exponentially distributed, and output time was assumed proportional to this component.

Analytic expressions for throughput and cost effectiveness in terms of the system and workload parameters were obtained for the special cases where the number of jobs simultaneously active did not exceed two and where the number of processors was no greater than the number of active jobs. Simulation was extensively employed to analyze the complex cases.

The cost effectiveness of monoprogrammed systems is limited by the nonutilization of the processor during output and by the underutilization of the memory resulting from the necessity of providing a total memory considerably larger than the average job requirements in order to prevent an inordinate amount of job termination. Multiprogramming alleviates both of these problems: more jobs are available to keep the processor busy, and the peak memory demands of the individual jobs are averaged.

Two types of multiprogrammed systems were analyzed: systems which terminate jobs in case of memory conflict, and systems which suspend jobs under these

circumstances if there is any possibility that memory can be allocated later. The suspension strategy was found to be superior.

Attention was given to the rules that an operating system might use in determining when to start a new job. In general, rules based on the number of active jobs, i.e. rules responsive to the probable distribution of memory allocation requests in the near future, were superior to rules based primarily on instantaneous memory availability. These rules were best modified, however, by consideration of the instantaneous memory situation.

Multiprocessing allows increased multiprogramming and averaging of peak memory requirements beyond the amount feasible with a single processor. As system size increases, the cost effectiveness approaches unity. However, at any given level of system cost, the decision as to whether to invest a finite additional amount of funds in additional memory or in additional processors is strongly affected by the ratio of memory cost to processor cost.

ACKNOWLEDGMENT

In addition to acknowledging the contributions of his wife, Elizabeth E. Wald, who assisted in the reduction of numerical data, and of his advisor, Dr. Alan B. Marcovitz, who made numerous helpful suggestions during the prosecution of the investigation and during the writing of this dissertation, the author would like to express his deep gratitude to the Naval Research Laboratory. This laboratory provided the research environment which suggested the investigation, supported the author's graduate studies with a Thomas A. Edison Memorial Graduate Training Fellowship, made available several hundred hours of computer time, and performed the editorial, illustration, typesetting, and reproduction services necessary for the physical production of this dissertation.

TABLE OF CONTENTS

Section	Page
ACKNOWLEDGMENT	ii
I. INTRODUCTION	1
II. THE THROUGHPUT MODEL	3
A. Purpose and Limitations	3
B. The System Model	3
C. The Workload Model	4
D. Definition of Cost Effectiveness	5
E. Methodology	6
III. THE MONOPROGRAMMED SYSTEM	8
A. Analytic Results	8
B. Numerical Examples and Discussion	9
IV. MULTIPROGRAMMING OF TWO JOBS	13
A. Constant Memory Requirements	13
B. Variable Memory Requirements – Termination Strategy	16
1. Analytic Results	17
2. Comparison of Analysis and Simulation	19
3. Numerical Examples and Discussion	20
C. Variable Memory Requirements – Suspension Strategy	24
1. Analytic Results	24
2. Comparison of Analysis and Simulation	28
3. Numerical Example and Discussion	28
V. MULTIPROGRAMMING OF MORE THAN TWO JOBS	30
A. Constant Memory Requirements	30
B. Variable Memory Requirements – Termination Strategy	31
C. Variable Memory Requirements – Suspension Strategy	40
VI. MULTIPROCESSING	48
A. Constant Memory Requirements	48
B. Variable Memory Requirements	50
1. Analytic Results	50
2. Numerical Results and Discussion	51

TABLE OF CONTENTS – Continued

Section	Page
VII. SUMMARY OF RESULTS	67
APPENDIX A – GLOSSARY OF NOTATION	74
APPENDIX B – DESCRIPTION OF SIMULATOR	77
APPENDIX C – DERIVATION OF EQUATIONS (7) THROUGH (14)	82
APPENDIX D – OPTIMUM SELECTION OF THE JOB TO TERMINATE	87
APPENDIX E – DERIVATION OF EQUATIONS (20) THROUGH (24)	90
APPENDIX F – COMPARISON OF ANALYSIS AND SIMULATION – TERMINATION STRATEGY	94
APPENDIX G – DERIVATION OF EQUATIONS (35) THROUGH (43b)	101
APPENDIX H – COMPARISON OF ANALYSIS AND SIMULATION – SUSPENSION STRATEGY	104
APPENDIX I – DERIVATION OF FIG. 24	108
APPENDIX J – ADDITIONAL TABLES – MULTIPROCESSING	109
WORKS CITED	119

LIST OF TABLES

Table		Page
1.	Monoprogrammed Case; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$	10
2.	Comparison of Monoprogrammed and Multiprogrammed (Termination Strategy) Cases; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = \mathbf{n} = 10$	21
3.	Factors Favoring and Opposing Multiprogramming	22
4.	Comparison of Monoprogrammed and Multiprogrammed (Termination Strategy) Cases; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$	22
5.	Comparison of Termination and Suspension Strategies; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$	28
6.	Effect of Decreasing μ by a Factor of 4 on the **CE** Data of Fig. 25; $\mathbf{m} = \mathbf{x} = \alpha = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$	33
7.	Effect of Varying $\mathbf{M_s}$, Termination Strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$, $\mathbf{N_j} = 5$	35
8a.	Effect of Finite Expansion Budget on **T** and **CE** for $\mu = 1$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$	60
8b.	Effect of Finite Expansion Budget on **T** and **CE** for $\mu = 0.25$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$	61
8c.	Effect of Finite Expansion Budget on **T** and **CE** for $\mu = 0.1$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$	62
B1.	Simulator Inputs	78
B2.	Simulator Outputs	79
C1.	**CE** vs $\mathbf{y_1}$, Monoprogrammed Case; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$	85
D1.	Comparison of Termination Strategies	89
F1.	Theoretical and Simulated Values of $\mathbf{P_z^+(z)}$; $\mathbf{m} = \mathbf{x} = 0$, $\alpha = \mathbf{n} = 1$	95
F2.	Theoretical and Simulated Values of **V**, γ, $\mathbf{t_{o2}}$ and **CE**; $\mathbf{m} = \mathbf{x} = 0$, $\alpha = \mu = \mathbf{n} = 1$	97
F3.	Theoretical and Simulated Values of **V**, γ, $\mathbf{t_{o2}}$ and **CE**; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = \mu = 1$, $\mathbf{n} = 10$	98
F4.	Theoretical and Simulated Values of **V**, γ, $\mathbf{t_{o2}}$ and **CE**; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{n} = 10$	99

LIST OF TABLES – Continued

Table		Page
H1a.	Theoretical and Simulated Values of V, γ_t, $1 + t'_{o2}$, and CE for $r = 2$; $m = 1$, $x = 0$, $\alpha = \mu = 1$, $n = 10$, $M_s = 1$, $y_t = M - 2$	105
H1b.	Theoretical and Simulated Values of V, γ_t, $1 + t'_{o2}$, and CE for $r = 0.5$; $m = 1$, $x = 0$, $\alpha = \mu = 1$, $n = 10$, $M_s = 1$, $y_t = M - 2$	105
H2a.	Theoretical and Simulated Values of V, γ_t, $1 + t'_{o2}$, and CE for $r = 2$; $m = x = \alpha = \mu = 1$, $n = 10$, $M_s = 2$, $y_t = M - 4$	107
H2b.	Theoretical and Simulated Values of V, γ_t, $1 + t'_{o2}$, and CE for $r = 0.5$; $m = x = \alpha = \mu = 1$, $n = 10$, $M_s = 2$, $y_t = M - 4$	107
J1.	Theoretical and Simulated Values of V, γ, t_{o2} and CE, Termination Strategy; $m = x = \alpha = \mu = 1$, $n = 10$, $N_j = N_p - 2$.	109
J2.	Theoretical and Simulated Values of $1 + t'_{o2}$, Suspension Strategy; $m = 1$, $x = 0$, $\alpha = \mu = 1$, $r = 0.5$, $n = 10$, $N_j = N_p - 2$, $M_s = 1$, $y_t = M - N_j$	110
J3a.	CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 1$)	110
J3b.	CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 2$)	111
J3c.	CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 3$)	111
J3d.	CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 4$)	112
J3e.	CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 5$)	112
J3f.	CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 6$)	113
J3g.	CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 7$)	113
J3h.	CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 8$)	114
J3i.	CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 9$)	115
J4a.	CE vs C for $\mu = 1$; $m = 1$, $m = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$	116
J4b.	CE vs C for $\mu = 0.25$; $m = 1$, $m = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$	117
J4c.	CE vs C for $\mu = 0.1$; $m = 1$, $m = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$	118

LIST OF FIGURES

Figure		Page
1.	CE vs **M**, monoprogrammed case; $\alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$	10
2.	CE vs $\mathbf{y}_1$, monoprogrammed case; $\alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$	10
3.	CE vs **M**, monoprogrammed case; $\mathbf{m} = \mathbf{x} = \alpha = 1$, $\mathbf{n} = 10$	11
4.	CE vs **M**, monoprogrammed case; $\mathbf{m} = \mathbf{x} = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$	11
5.	CE vs $\alpha\mathbf{y}_1$, monoprogrammed case; $\mathbf{m} = \mathbf{x} = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$	11
6.	CE vs **M**, monoprogrammed case; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$	12
7.	CE vs $(\mathbf{y}_1 - 1.1\,\ell n(\mathbf{n}))$, monoprogrammed case; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$	12
8.	CE vs μ, monoprogrammed and multiprogrammed cases, constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = 1$, $\mathbf{r} = 0.2$	14
9.	CE vs μ, monoprogrammed and multiprogrammed cases, constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = 1$, $\mathbf{r} = 1$	15
10.	CE vs μ, monoprogrammed and multiprogrammed cases, constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = 1$, $\mathbf{r} = 2$	15
11.	CE vs **r**, monoprogrammed and multiprogrammed cases; constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = \mu = 1$	15
12.	CE vs **r**, monoprogrammed and multiprogrammed cases, constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = 1$, $\mu = 0.25$	15
13.	Loci of $\mathbf{CE_{mono}} = \mathbf{CE_{multi}}$ in **r**, μ space, constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = 1$	16
14.	$\mathbf{P}_z^+(z) = \int_z^\infty \mathbf{p}_z(w)\,\mathbf{dw}$, multiprogrammed case, termination strategy; $\alpha = 1$	18
15.	CE vs $\mathbf{y}_1$ or $\mathbf{y}_2$ monoprogrammed case and multiprogrammed case with termination strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = \mathbf{n} = 10$	20
16.	CE vs $\mathbf{y}_1$ or $\mathbf{y}_2$, monoprogrammed case and multiprogrammed case with termination strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$	23
17.	CE vs $\mathbf{y}_1$ or $\mathbf{y}_2$, monoprogrammed case and multiprogrammed case with termination strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = \mathbf{r} = 1$, $\mathbf{n} = 10$	23
18.	CE vs $\mathbf{y}_1$ or $\mathbf{y}_2$, monoprogrammed case and multiprogrammed case with termination strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mathbf{r} = 1$, $\mathbf{n} = 10$; broken lines: $\mu = 1$ (inexpensive processor), solid lines: $\mu = 0.25$	23

Figure		Page
19.	CE vs y_1 or y_2, monoprogrammed case and multiprogrammed case with termination strategy; $m = x = \mu = r = 1$, $n = 10$; broken lines: $\alpha = 1$ (low memory variability), solid lines: $\alpha = 0.5$	23
20.	CE vs y_1 or y_2, monoprogrammed case and multiprogrammed case with termination strategy; $m = x = \alpha = \mu = r = 1$; broken lines: $n = 10$, solid lines: $n = 20$	24
21.	Worst-case suspension situation	26
22.	Better alternate suspension situations	27
23.	CE vs y_1 or y_2, monoprogrammed case and multiprogrammed cases; $m = x = \alpha = \mu = 1$, $r = 0.5$, $n = 10$	29
24.	Loci of CE $(N_j = k)$ = CE $(N_j = k + 1)$ in r, μ space, constant memory requirements; $m = x = y = 1$	31
25.	CE vs M, termination strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$, $M_s = 2$	32
26.	CE vs M, termination strategy, $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$	37
27.	CE vs M_s, termination strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$	38
28.	CE vs M, termination strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$	39
29.	CE vs M, suspension strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$, $M_s = 2$, $y_t = M - 2N_j$	41
30.	CE vs M, suspension strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$, $N_j = M_s = 2$	43
31.	CE vs M, suspension strategy; $m = 0$, $x = \alpha = \mu = r = 1$, $n = 10$, $N_j = 2$, $M_s = 1$	43
32.	CE vs M, suspension strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$, $N_j = 5$, $M_s = 1$	43
33.	CE vs M_s, suspension strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$, $M = 11$	44
34.	Synchronism of processing for rational t_o	49
35.	Worst-case suspension situation, multiprocessing	50
36.	Better suspension situation, multiprocessing	51
37.	CE vs y_1 or y_2; $m = x = \alpha = \mu = 1$, $r = n = 10$	52
38.	CE vs M/N_j; $m = x = \alpha = \mu = 1$, $r = n = 10$	52
39a.	CE vs C for $\mu = 1$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$; $N_p = 1$	54

Figure Page

39b. CE vs C for $\mu = 0.25$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$; $N_p = 1$ 54

39c. CE vs C for $\mu = 0.1$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$, $N_p = 1$ 55

40a. CE vs C for $\mu = 1$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$, $N_p = 2$ 55

40b. CE vs C for $\mu = 0.25$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$, $N_p = 2$ 55

40c. CE vs C for $\mu = 0.1$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$, $N_p = 2$ 55

41a. CE vs C for $\mu = 1$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ 57

41b. CE vs C for $\mu = 0.25$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ 57

41c. CE vs C for $\mu = 0.1$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ 57

42a. CE vs C for $\mu = 1$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ 58

42b. CE vs C for $\mu = 0.25$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ 58

42c. CE vs C for $\mu = 0.1$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ 58

43. T vs C; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ 63

44. CE vs C for nonzero job collection cost; $m = 1$, $x = 0$, $\alpha = \mu = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ 65

B1. Principal simulator routines 77

B2. JOBSEARCH routine 79

B3. WORKSEARCH routine 80

B4. TIMESEARCH routine 81

F1. Operation at high r 100

F2. Operation at low r 100

I. INTRODUCTION

The motivation underlying this study has already been described in a state-of-the-art paper (1) and will be reviewed only briefly here. Although multiprocessor digital computer systems were originally developed by the armed forces to obtain high system reliability at the expense of only moderate hardware redundancy (2), this reliability has become of increasing interest to designers of new equipment intended for installation at computing centers which are required to provide continuously available service to a multiplicity of users. Even when reliability is not of major concern, multiprocessors, and the multiprogramming operating systems that usually accompany them, also afford flexibility in the assignment of system resources to the individual problems in the instantaneous workload. This flexibility is a powerful tool in a multiple-user environment.

The object of the present study is to analyze the factors that determine the optimum hardware mix for a given workload and that influence the choice of operating system strategies for a given combination of hardware and workload. An analytic approach is carried as far as is practical, and simulation is employed in the more complex cases.

In this search for optimum hardware configurations and operating system strategies, the first task is to define the measure to be optimized. Two such measures appear reasonable: a service delay minimization and a throughput maximization.

The service delay minimization leads to a queuing model in which it is assumed that the jobs are presented to the system at various times and that the effectiveness of the system is some function of the delays between the arrival and completion of each of these jobs. Note that the density of the workload is an important factor in the model, as is the nature of the delay function to be minimized. Implicit in this model is the assumption that the system can process the workload.

The throughput maximization assumes that an infinite number of jobs are available and that the effectiveness of the system can be measured by summing the value of the jobs completed. Although the statistical distribution of job parameters is still important, no assumptions need be made about arrival rate, since the system is assumed to have a job waiting at all times. It is considerably simpler to measure the value of a job in terms of the cost of the hardware components utilized that it is to assign the cost of a delay in the queuing model.

The throughput model was selected for this study because it leads to useful results with fewer assumptions and because it has been relatively neglected by other investigators.

No criticism of the queuing model is implied; if one is faced with a given workload and a given hardware configuration to process that workload, the rational policy is to minimize in some sense the total inconvenience induced by the service delay to those users for whose benefit the hardware exists. Nevertheless, a throughput model is of considerable value in providing the data required for the choice of size, balance, and geographic distribution of computer centers, balancing economies of scale against data transmission costs, particularly in those cases where the bulk of the workload is of the traditional type in which service delay is not critical.

II. THE THROUGHPUT MODEL

A. Purpose and Limitations

The throughput model is intended to focus attention on the efficiency with which the system allocates its resources, and on the degree to which the hardware complement is appropriate to the workload. Questions relating to the service given to a particular problem generating portion of the workload are ignored; instead total productivity is to be maximized, even if this maximization leads to poor service to a particular class of problems. Thus the throughput model is not sufficient for developing operating systems, but neither is the queuing model alone. The complex problem of combining the models is left for later investigations.

In the model at hand it is assumed that the operating system draws problems from an infinite pool. All problems in this pool are assumed to have known and stationary statistics. The throughput, T*, of the system is the product of the rate of problem completion and of the mean value of the completed problems, where this value is related to the resources required for its solution. No value is given partially completed jobs, no penalties are attached to delays in completion, and no penalties are attached to refusing to attempt or to abandoning a particular problem, except for the loss in throughput caused by the wasted effort on an abandoned problem.

It is further assumed that operating system decisions and actions are accomplished at no cost, or at least that these costs may be considered to be included in the specifications of the problems themselves.

B. The System Model

The system is assumed to consist of N_p processor (computer) modules, N_c input/output (i/o) channels, and a main memory.

Every processor module is assumed to have identical characteristics and to cost one unit. The processing rate will not be explicitly defined; instead problems will be characterized by their processing time requirements.

*A glossary of notation is provided in Appendix A.

Every i/o channel is assumed to have identical characteristics, and for the purposes of this study the cost per channel, χ, will be set at zero. It is further assumed that sufficient channels are available so that no waits are caused by channel unavailability. It is assumed that problem input can take place in negligible time but that output requires significant time. All outputs are assumed to take place at a rate **r**, measured in units of memory per unit of time.

The memory is characterized by its size, **M**. The unit in which **M** is measured is deliberately undefined for greater generality, but it should be noted that in the numerical examples the unit is assumed to be much larger than one word – perhaps in the range of a page to a bank in present-day systems. The assumption that operating system functions take place at no cost in resources (or that these costs are hidden in the workload specifications) implies that the total memory **M** is available to be allocated to the workload.

It is further assumed that problems associated with dynamic allocation and reallocation of memory have been solved and that the system is always free to allocate a quantity of memory to a job if this quantity is less than the difference between **M** and the amount of memory already allocated. Thus questions of memory fracturing or checkerboarding will be ignored.

C. The Workload Model

Each job is assumed to contain the same number, **n**, of segments. Associated with each job is a memory requirement, **m**, which must be allocated continuously from job initiation to completion; **m** may be thought of as representing the permanent data base and higher level routines of the job. Initiation of any segment of a job requires the allocation of two additional increments of memory: **x**, which may be thought of as representing additional subroutines and working space required for execution, and $\underline{\mathbf{y}}$*, which may be thought of as representing the space in which the segment output will be formatted.

Execution of each segment requires τ units of processor time. This quantity will be set to one, thus fixing the unit of time for the investigation. After a segment is executed, the **x** units of memory may be released, but the $\underline{\mathbf{y}}$ units must be retained until output is complete. Output requires $\underline{\mathbf{y}}/\mathbf{r}$ units of time.

In the most general case all of these parameters could be statistically distributed. In this investigation, however, it will be assumed that only $\underline{\mathbf{y}}$ is a random variable and that the designer of the operating system is aware of the value of all constant parameters

*Random variables will be denoted by scored boldface type.

and of the parameters of the $\underline{y}$ distribution. However, the value of the $\underline{y}_i$ associated with the ith segment of a job will not be unveiled until the $(i-1)^{th}$ segment is completed.

For the most part the $\underline{y}$ distribution will be assumed to be

$$p_{\underline{y}}(y) = \alpha e^{-\alpha y}, \tag{1}$$

although other distributions will be used in special cases. It will be convenient to introduce the symbol $\underline{z}$ to represent those $\underline{y}$'s that are actually allocated. The form of the $\underline{z}$ distribution will depend on the configuration and the operating system, as not all $\underline{y}$'s have an equal probability of being allocated.

D. Definition of Cost Effectiveness

The mean value of completed jobs, **V**, will be defined as the sum of average products of resource utilization and cost. Thus when a job is in execution, all memory components contribute to the value of the job, but when a job is suspended awaiting the availability of a processor, the occupancy of **m** units of memory is simply wasted and does not contribute to the value of the job.

For any particular job, there are two components of value. During execution, which lasts one unit of time, the component for each segment is

$$1 + \mu\,(\mathbf{m} + \mathbf{x} + \underline{z}).$$

During output, which lasts $\underline{z}/\mathbf{r}$ units of time, the component for each segment is

$$\mu \frac{\underline{z}}{\mathbf{r}}(\mathbf{m} + \underline{z}).$$

Summing these components, and taking the average, yields

$$\mathbf{V} = \mathbf{n}\ \{1 + \mu\ [\mathbf{m} + \mathbf{x} + \left(1 + \frac{\mathbf{m}}{\mathbf{r}}\right)\langle\underline{z}\rangle + \frac{1}{\mathbf{r}}\langle\underline{z}^2\rangle]\}. \tag{2}$$

The mean time between job completions is denoted by $\mathbf{t_c}$. It should be noted that this does not necessarily represent the mean time to complete one job, since more than one job may be in the system at one time. It is therefore reasonable to define the throughput, **T**, of the system as

$$\mathbf{T} = \frac{\mathbf{V}}{\mathbf{t_c}}. \tag{3}$$

Under the assumption that i/o channels are available at no cost, the system cost, **C**, is given by

$$\mathbf{C} = \mathbf{N_p} + \mu\mathbf{M}. \tag{4}$$

Finally, the cost effectiveness of the system, CE, is defined as

$$CE = \frac{T}{C} \tag{5}$$

Under these definitions, no system can have a CE greater than unity, as a CE of unity implies that all processors are always performing useful work and that the memory is completely filled with jobs.

A monoprogrammed system cannot achieve a CE of unity except in trivial limiting cases. The processor will not be utilized during output, nor will the memory space required for x be utilized. The CE will approach unity (a) as μ approaches zero and r approaches infinity, thus minimizing the effect of memory underutilization and minimizing processor underutilization, or (b) as μ approaches infinity, x approaches zero, and $\underline{y}$ becomes a constant, y, thus minimizing the effect of processor underutilization and minimizing memory underutilization.

Even the multiprogrammed system has similar difficulties in approaching the CE limit. Processor underutilization can be eliminated, but this will raise CE to unity only in the limit as μ approaches zero. Memory underutilization can be eliminated only by letting $\underline{y}$ become a constant, y, and by either letting x approach zero or by not requiring the system to allocate memory for more than N_p x's. This last assumption is not inconsistent with the model but is unrealistic in actual practice where the loading of a new job cannot take place in zero time.

E. Methodology

The throughput model will be used to examine three general types of systems: monoprogrammed, multiprogrammed, and multiprocessing.

The monoprogrammed system has one processor and attempts to perform jobs in strictly serial fashion. It is representative of almost all computer systems in use until 1960 and of the majority of systems in use up to the present. It would be expected that monoprogrammed systems would yield CE's of considerably less than unity because of the unutilized processor time during output, the unutilized memory provided for x during output, and the unutilized memory provided for above average values of $\underline{y}$ when smaller values are required.

The multiprogrammed system also has one processor but attempts to keep more than one job active. It is representative of a few systems now extant and of systems that will probably dominate the computing field by 1968 or 1970. This system attempts to raise

CE by making more than one job available to the processor and by averaging the memory requirements of the active job.

The multiprocessing system not only attempts to keep more than one job active but also has more than one processor available for job execution. There are very few multiprocessors now extant except in military applications, but it is expected that multi-user systems presenting a "public utility" posture to their customers will make substantial use of such systems in the 1970's. Apart from the matter of reliability, multiprocessing would be expected to afford improved CE for large systems by allowing a considerable amount of multiprogramming with its attendant advantages of memory utilization averaging without restrictions caused by processor waits.

Two techniques have been used in this investigation: probability analysis for the simpler cases, and a simulator which is described in Appendix B. The general order of presentation is analytic results (if any), a discussion of the validity of the analysis, and a discussion of numerical examples, where these examples are derived analytically or through simulation as appropriate. Details of the derivations and the numerical examples may be found in various appendixes.

III. THE MONOPROGRAMMED SYSTEM

A. Analytic Results

The monoprogrammed system, faced with an instantaneous memory requirement greater than **M**, has no option but to terminate the job. It will be convenient to define y_1 as the amount of memory available to allocate the variable component $\underline{y}$. Then

$$y_1 = M - m - x. \tag{6}$$

The probability, P_c, that a given $\underline{y}$ will exceed y_1 is given by*

$$P_c = e^{-\alpha y_1} \tag{7}$$

and the $\underline{z}$ distribution is given by

$$p_{\underline{z}}(z) = \frac{\alpha e^{-\alpha z}}{1 - e^{-\alpha y_1}},\ 0 \le z \le y_1;\ 0 \text{ otherwise.} \tag{8}$$

Because of these conflicts, and the resulting terminations, it will require on the average the completion of more than **n** segments per job completed. The mean number of segments completed per job completed is denoted by γ and is found to be

$$\gamma = \frac{1 - Q_c^{\,n}}{P_c \cdot Q_c^{\,n-1}},\ \text{where } Q_c = 1 - P_c. \tag{9}$$

The mean output time per segment, denoted by t_o, is given by

$$t_o = \frac{\langle \underline{z} \rangle}{r} = \frac{1}{r Q_c}\left[\frac{1}{\alpha} - \left(\frac{1}{\alpha} + y_1\right) P_c\right], \tag{10}$$

and since the mean total time per segment is $1 + t_o$,

$$t_c = \gamma\,(1 + t_o). \tag{11}$$

*The deviation of equations (7) through (14) may be found in Appendix C.

The mean value of completed jobs, **V**, is found to be

$$\mathbf{V} = \mathbf{n}\left\{1 + \mu\left[\mathbf{m} + \mathbf{x} + (\mathbf{m} + \mathbf{r})\,\mathbf{t_o} + \frac{2 - (\alpha^2 \mathbf{y}_1{}^2 + 2\alpha \mathbf{y}_1 + 2)\,\mathbf{P_c}}{\alpha^2 \mathbf{r}\,(1 - \mathbf{P_c})}\right]\right\} \tag{12}$$

which leads to the desired expression for **CE**:

$$\mathbf{CE} = \frac{\mathbf{n}\left\{1+\mu\left[\mathbf{m}+\mathbf{x}+\frac{\mathbf{m}+\mathbf{r}}{\mathbf{r}(1-\mathbf{e}^{-\alpha \mathbf{y}_1})}\left(\frac{1}{\alpha}-\left(\frac{1}{\alpha}+\mathbf{y}_1\right)\mathbf{e}^{-\alpha \mathbf{y}_1}\right)+\frac{2-(\alpha^2\mathbf{y}_1{}^2+2\alpha\mathbf{y}_1+2)\mathbf{e}^{-\alpha \mathbf{y}_1}}{\alpha^2\mathbf{r}(1-\mathbf{e}^{-\alpha \mathbf{y}_1})}\right]\right\}\mathbf{e}^{-\alpha \mathbf{y}_1}(1-\mathbf{e}^{-\alpha \mathbf{y}_1})^{\mathbf{n}-1}}{(1+\mu\mathbf{M})\left\{1+\frac{1}{\mathbf{r}(1-\mathbf{e}^{-\alpha \mathbf{y}_1})}\left[\frac{1}{\alpha}-\left(\frac{1}{\alpha}+\mathbf{y}_1\right)\mathbf{e}^{-\alpha \mathbf{y}_1}\right]\right\}\left[1-(1-\mathbf{e}^{-\alpha \mathbf{y}_1})^{\mathbf{n}}\right]} \tag{13}$$

Provided that $\mathbf{P_c}$ is small, as it is in the region of maximum **CE**, equation (13) can be approximated by

$$\mathbf{CE} \approx \frac{\left[1 + \mu\left(\mathbf{m} + \mathbf{x} + \frac{\mathbf{m}}{\alpha \mathbf{r}} + \frac{1}{\alpha} + \frac{2}{\alpha^2 \mathbf{r}}\right)\right]\left[1 - (\mathbf{n} - 1)\mathbf{e}^{-\alpha \mathbf{y}_1}\right]}{\left(1 + \frac{1}{\alpha \mathbf{r}}\right)\left[1 + \mu\,(\mathbf{m} + \mathbf{x} + \mathbf{y}_1)\right]}; \tag{14}$$

but this approximation breaks down for small **M** when $\mathbf{P_c}$ is substantial.

No approximations were used in the derivation of equations (7) through (13), so it is believed that these results are exact. Simulation yielded sample means whose fiducial confidence intervals covered the analytically derived points the expected proportion of the time, thus verifying the analytic results and the proper operation of the simulator.

In principle, equation (13) could be differentiated with respect to **M** (or $\mathbf{y}_1$), the derivative equated to zero, and the resulting transcedental equation solved to yield $\mathbf{M_{opt}}$, the value of **M** that maximizes **CE**. If this solution was an explicit function of the other parameters, it could be differentiated to show the sensitivity of $\mathbf{M_{opt}}$ to these other parameters. In practice, the analytic difficulties are too formidable and Equation (13) was evaluated numerically.

B. Numerical Examples and Discussion

Table 1 displays the results of calculations based on the assumptions $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$, and $\mathbf{n} = 10$. It can be seen that the maximum **CE** of 0.381 occurs at an **M** of 6, which is sufficiently large that $\mathbf{t_o}$ has risen and γ has fallen to within about 10% of their final values.

Table 1

Monoprogrammed Case; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$.

M	y_1	t_o	γ	t_c	V	T	C	CE
3.0	1.0	2.09	167.0	516.0	67.8	.131	4.0	.033
3.5	1.5	2.85	40.0	153.8	88.8	.577	4.5	.128
4.0	2.0	3.44	21.0	93.0	108.6	1.168	5.0	.234
4.5	2.5	3.88	15.2	74.0	126.3	1.707	5.5	.310
5.0	3.0	4.21	12.7	66.3	141.3	2.130	6.0	.355
5.5	3.5	4.46	11.5	62.9	153.5	2.441	6.5	.376
6.0	4.0	4.63	10.9	61.2	163.1	2.664	7.0	.381
6.5	4.5	4.74	10.5	60.5	170.5	2.820	7.5	.376
7.0	5.0	4.83	10.3	60.1	176.1	2.930	8.0	.366
7.5	5.5	4.89	10.2	60.0	180.2	3.004	8.5	.353
8.0	6.0	4.93	10.1	59.9	183.1	3.056	9.0	.340
8.5	6.5	4.95	10.1	59.9	185.3	3.092	9.5	.325
9.0	7.0	4.97	10.0	59.9	186.7	3.116	10.0	.312
9.5	7.5	4.98	10.0	59.9	187.8	3.133	10.5	.298
10.0	8.0	4.99	10.0	60.0	188.5	3.144	11.0	.286

For the small **r** chosen, it would be expected that the value of $\mathbf{M_{opt}}$ should vary in direct correspondence with variations in **m** and **x**. This is illustrated in Fig. 1, in which the **CE** results of Table 1 are plotted along with the results of calculations made with $\mathbf{m} = 3$, $\mathbf{x} = 1$, and with $\mathbf{m} = 1$, $\mathbf{x} = 3$. Figure 2 is a replot of the data with $\mathbf{y}_1$ as the

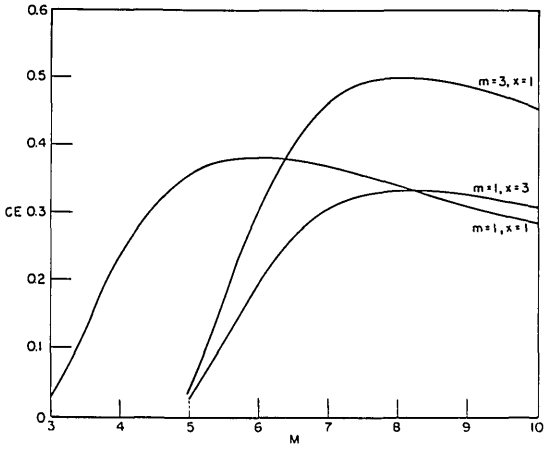


Fig. 1 – **CE** vs M, monoprogrammed case; $\alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$.

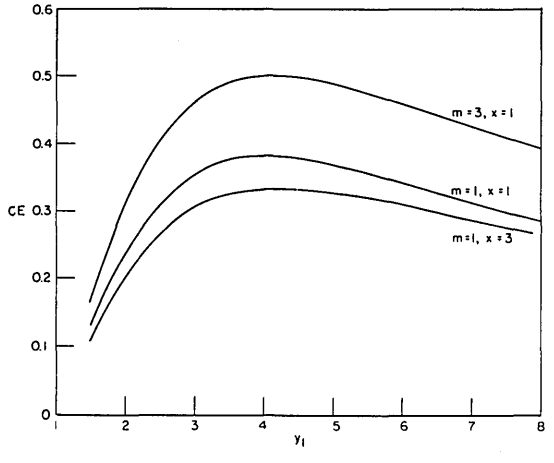


Fig. 2 – **CE** vs y_1, monoprogrammed case; $\alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$.

abscissa. The higher values of **CE** for **m** = 3 can be explained by the low **r**: memory provided for **x** is utilized only a small part of the time, while memory provided for **m** is always utilized.

Figure 3 illustrates the small magnitude of the effect of μ and **r** on **CE**. Provided that **r** is small and provided that $\mu >> \mathbf{r}$, costs and values are determined primarily by memory considerations. Increases in μ slightly increase **CE** as the under-utilization of the processor becomes relatively less significant, while decreases in **r** decrease **CE** by further lowering processor utilization.

Figure 4, on the other hand, illustrates the strong dependence of $\mathbf{M_{opt}}$ on α. This dependence is not unexpected, since $\langle \underline{y} \rangle = 1/\alpha$. In Fig. 5, the data have been replotted against a normalized abscissa of αy_1. Inspection of Fig. 5 suggests a relation of the form

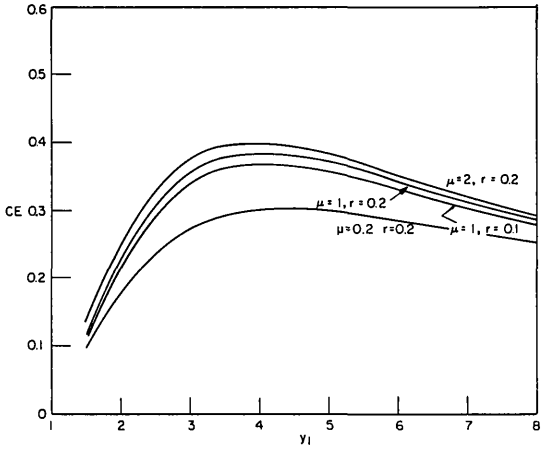


Fig. 3 – **CE** vs **M**, monoprogrammed case; **m** = **x** = α = 1, **n** = 10.

$$y_{1_{opt}} = \frac{k}{\alpha} \qquad (15)$$

where **k** has a value of about 4 for **r** = 0.2 and **n** = 10.

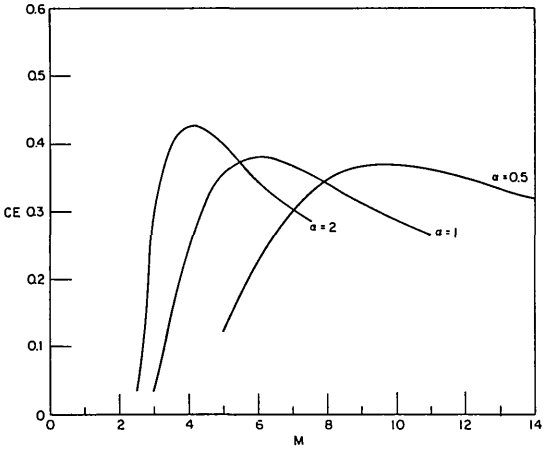


Fig. 4 – **CE** vs **M**, monoprogrammed case; **m** = **x** = μ = 1, **r** = 0.2, **n** = 10.

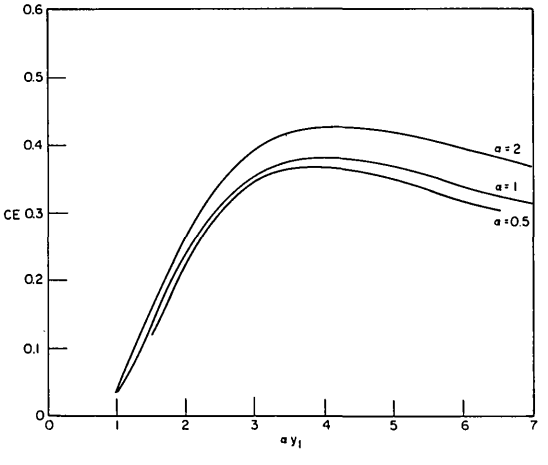


Fig. 5 – **CE** vs αy_1, monoprogrammed case; **m** = **x** = μ = 1, **r** = 0.2, **n** = 10.

It can be seen that as α decreases, the maximum **CE** decreases. Smaller values of α imply a larger variability in $\underline{y}$ and less efficiency, because more memory, which is utilized only part of the time, must be provided to accommodate this larger variability.

It would be expected that the value of **k** in equation (15) would increase as **n** increases, since larger values of **n** would require lower values of $\mathbf{P_c}$ to keep γ reasonably close to **n**. Figure 6 shows the results of calculations based on four values of **n**. As shown in Fig. 7, a reasonable fit is obtained with the normalized abscissa $\mathbf{y}_1 - 1.1\ \ell n\ (\mathbf{n})$, suggesting a relationship of the form

$$\mathbf{y}_{1_{\mathbf{opt}}} \approx \frac{(1.5 + 1.1\ \ell n\ (\mathbf{n}))}{\alpha}. \tag{16}$$

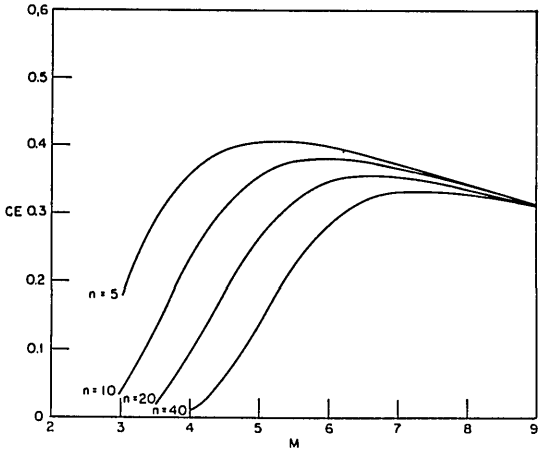


Fig. 6 – **CE** vs **M**, monoprogrammed case; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$.

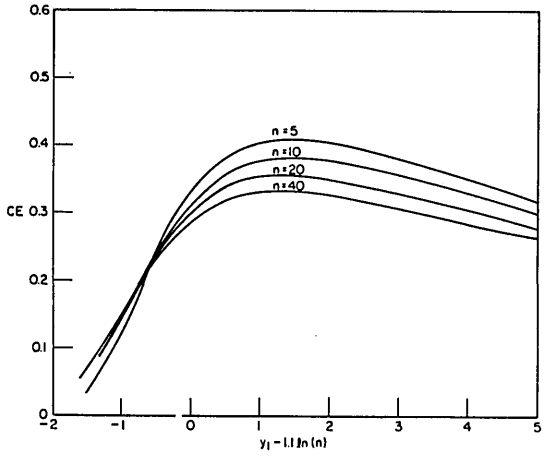


Fig. 7 – **CE** vs $(\mathbf{y}_1 - 1.1\ \ell n\ (\mathbf{n}))$, monoprogrammed case; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1, \mathbf{r} = 0.2$.

In summary, the value of $\mathbf{M_{opt}}$ is relatively insensitive to **r** and μ, provided that **r** and μ are so related that the processor is severely underutilized, and is equal to the sum of **m**, **x**, and some multiple of the mean value of $\underline{y}$. For **n** in the range 5 to 40, this multiple is approximately $1.5 + 1.1\ \ell n\ (\mathbf{n})$. The value of **CE** at $\mathbf{M_{opt}}$ is reduced by those factors which force less than full-time utilization of system resources; thus increases in **x**, **y**, and **n** and decreases in **r** tend to reduce the maximum **CE**.

IV. MULTIPROGRAMMING OF TWO JOBS

A. Constant Memory Requirements

It will be shown that multiprogramming can increase CE in two ways. The first is the reduction of processor underutilization by the provision of additional jobs to occupy the processor while one job is in the output state. The second is the averaging of the peak memory demands that occurs in a memory large enough to accommodate more than one job.

In order to focus attention upon the first and more obvious advantage of multiprogramming, a degenerate case of the workload model in which **y** becomes deterministic and equal to 1 will be considered. For this discussion it will be assumed that **m** and **x** are equal to 1. Since a suitable value of **M** will insure that no termination occur, the value of **n** becomes irrelevant.

Since the mean values of $\underline{y}$ and $\underline{y}^2$ equal 1, equation (2) reduces to

$$\frac{\mathbf{V}}{\mathbf{n}} = 1 + \mu\left(3 + \frac{2}{\mathbf{r}}\right). \tag{17}$$

For both the monoprogrammed and multiprogrammed cases, $\mathbf{t_o}$ is simply $1/\mathbf{r}$. The segment completion time for the monoprogrammed case was $1 + \mathbf{t_o}$ but may be greater than that for the multiprogrammed case because of processor unavailability. It will be convenient, therefore, to define $\mathbf{t_{o2}}$ as the mean output time plus waiting time, so that the mean segment completion time for the multiprogrammed case will be $1 + \mathbf{t_{o2}}$. Fortunately, in the specialized situation at hand the jobs will become synchronized and $\mathbf{t_{o2}}$ will be the greater of 1 (the processing time of the other job) and $\mathbf{t_o}$.

For the monoprogrammed case, $\mathbf{M_{opt}}$ is equal to 3, the sum of **m**, **x**, and **y**. For the multiprogrammed case, it might be argued that **M** is equal to 5, the sum of twice **m** and **y** for the two jobs and of a single **x** for the one job that is being processed. An argument can be made, however, for an $\mathbf{M_{opt}}$ of 6. Although there is no absolute requirement that space be reserved for the **x** of both jobs, most operating systems would insist on an M of 6. If the purpose of multiprogramming is to assure that there will be an allocated job ready for execution when the processor is finished with its previous job, allocation time for the **x** and **y** of the new job could not be tolerated. Thus the practical operating system would

preload these areas and require an **M** of 6. On the other hand, the idealized model assumes zero-time operating system loads. Accordingly the discussion of this subsection will be based on two values of **M** for the multiprogrammed case, a conservative value of 6 graphed with solid lines, and an idealized value of 5 graphed with broken lines.

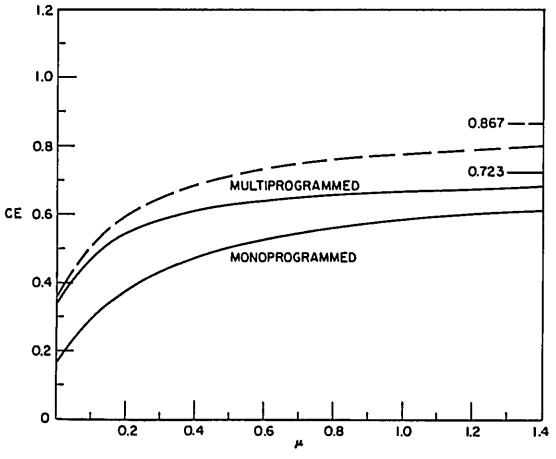


Fig. 8 – **CE** vs μ, monoprogrammed and multiprogrammed cases, constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = 1$, $\mathbf{r} = 0.2$.

In Fig. 8 the relationship between **CE** and μ has been plotted under the assumption that $\mathbf{r} = 0.2$. At this low value of **r** there is never a processor wait, and the multiprogrammed system exhibits superior **CE**. For infinitesimal μ the **CE** ratio is 2 to 1, as costs and values are determined solely by the processor and the multiprogrammed system has twice the throughput of the monoprogrammed system. For large μ the **CE**'s of the monoprogrammed and the conservative multiprogrammed systems approach the same asymptotic value, 0.723, since in the limit the cost ratio of 2 to 1 is the same as the throughput ratio. Note that in all cases **CE** increases with μ because of processor underutilization and that the **CE**'s are higher than were seen before because the variability has been removed from **y** and the hardware can better match the problem.

In Fig. 9 the situation has been plotted for $\mathbf{r} = 1$. This is the value of **r** that yields perfect synchronization with no waits for or wastage of processor time in the multiprogrammed case. Thus for the idealized **M** of 5, there is perfect resource utilization and a **CE** of unity for any value of μ. For $\mathbf{M} = 6$, the **CE** decreases with increasing μ as the effect of the wasted unit of memory becomes more significant, approaching from above the same asymptotic value, 0.833, that is approached from below by the monoprogrammed system as the effect of its half utilized processor becomes less serious with increasing μ.

In Fig. 10 the situation has been plotted for $\mathbf{r} = 2$. For this value of **r**, $\mathbf{t_o} = 0.5$ but $\mathbf{t_{o2}} = 1.0$ and there is considerable waiting for the processor in the multiprogrammed case. Since memory is being wasted during these waits, the **CE** of the multiprogrammed system is inferior, except for very small values of μ which minimize the effect of the wasted memory.

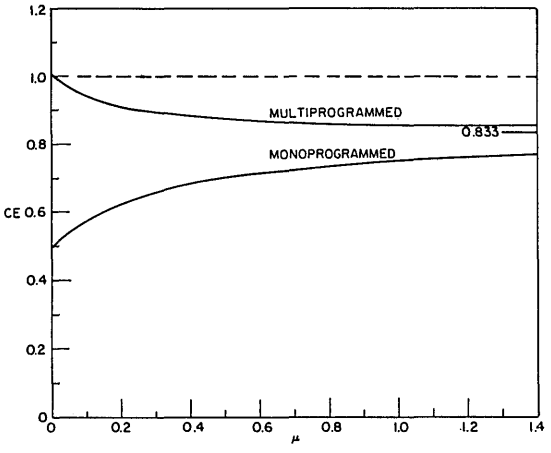


Fig. 9 – CE vs μ, monoprogrammed and multiprogrammed cases, constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = 1$, $\mathbf{r} = 1$.

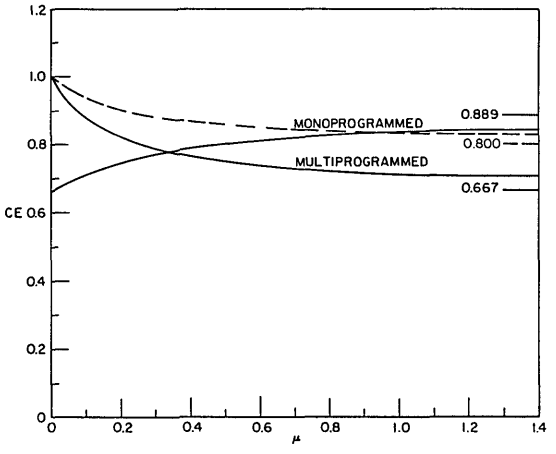


Fig. 10 – CE vs μ, monoprogrammed and multiprogrammed cases, constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = 1$, $\mathbf{r} = 2$.

In Figs. 11 and 12, **CE** has been plotted as a function of **r** for two values of μ. Note that the monoprogrammed **CE** rises with **r**, as output time and the inevitable processor wastage that accompanies it is decreased with increasing **r**. The multiprogrammed **CE**, however, increases with **r** only up to $\mathbf{r} = 1$ and then begins to fall. The rising portion of the curves are accounted for by reduced wastage in processor utilization, the falling portion by increased wastage in allocated memory which cannot be used because of processor unavailability. In Fig. 12, where μ is small and efficiency in processor utilization is paramount, the multiprogrammed **CE** dominates the monoprogrammed **CE** over most of the

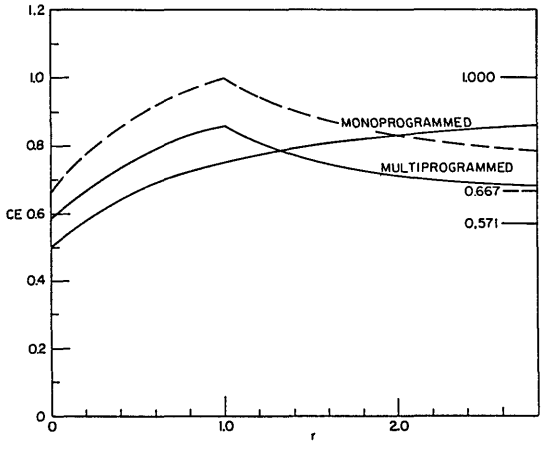


Fig. 11 – CE vs **r**, monoprogrammed and multiprogrammed cases, constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = \mu = 1$.

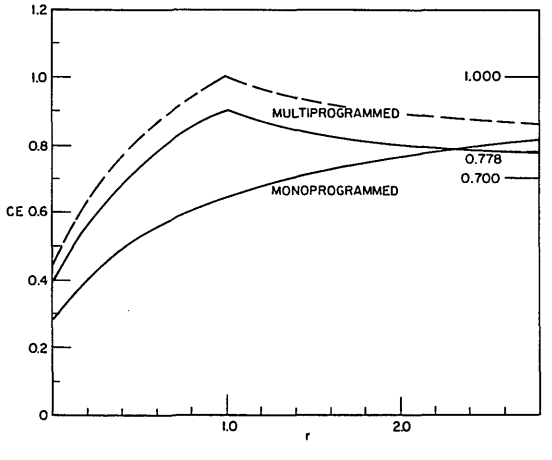


Fig. 12 – CE vs **r**, monoprogrammed and multiprogrammed cases, constant memory requirements; $\mathbf{m} = \mathbf{x} = \mathbf{y} = 1$, $\mu = 0.25$.

range of **r**; but in Fig. 11 where μ is large enough to severely penalize memory wastage, the curves cross at moderate **r**.

In summary, at low values of **r** ($r < 1$ for $M = 6$ and $r < 1.5$ for $M = 5$) multiprogramming will always afford superior **CE** irrespective of value of μ. At higher values of **r** the gain in processor utilization in multiprogramming must be balanced against the loss in memory utilization caused by waits for the processor. The crossover point will depend on the value of μ. This effect is illustrated in Fig. 13, which shows the loci of points in the **r**, μ plane at which the monoprogrammed and the multiprogrammed systems yield equal **CE**. The lower curve is for the multiprogrammed system in which $M = 6$, the upper curve is for $M = 5$. Thus in the area below both curves multiprogramming is superior, and in the area above both curves monoprogramming is superior. In the area between the curves multiprogramming is superior only if it can be accomplished with $M = 5$. It should be noted that the curves do not connect points of equal **CE** along the curves; a few **CE** values are indicated at selected points.

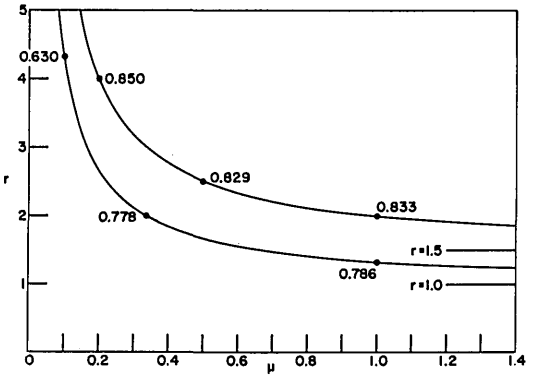


Fig. 13 – Loci of $CE_{mono} = CE_{multi}$ in **r**, μ space, constant memory requirements; $m = x = y = 1$.

It should also be noted that the perfect synchronization and unity **CE** at $r = 1$ is possible only because of the fixed memory requirements. If instead of **y** being fixed at 1, $\underline{y}$ was a random variable with a mean value of 1, there would be a significant amount of processor waiting. Further the **M**'s considered in this subsection would yield an intolerably high proportion of memory conflicts.

B. Variable Memory Requirements – Termination Strategy

The discussion of systems which multiprogram two jobs will now be continued with the probability distribution of equation (1) assumed for $\underline{y}$. Under these circumstances memory conflicts may occur. In this subsection it will be assumed that in case of a memory conflict the requesting job is terminated.

It may well seem intuitively obvious, and it will be demonstrated later, that this strategy of terminating the requesting job and wasting the work invested is inferior to that of suspending the requesting job. Nevertheless the termination strategy is of interest for at least three reasons: it is analytically tractable, it allows direct comparison

with monoprogrammed systems which must terminate in case of memory conflicts, and it was initially used (although later modified) in one of the first multiprocessor operating systems (3).

Another objection to the termination strategy is that the requesting job is always the one to be terminated, whereas about half of the time the job already allocated rather than the requesting job has had less system resources invested in it up to the time of conflict and should be terminated. It is demonstrated in Appendix D, however, that terminating the job having less system resources invested is only slightly better than the simple strategy of terminating the requesting job.

1. **Analytic Results.** In analogy with equation (6), y_2 is defined as the amount of memory available to allocate the $\underline{y}$'s of both jobs. An immediate problem is whether to state

$$y_2 = M - 2\,(m + x) \tag{18a}$$

or

$$y_2 = M - 2m - x. \tag{18b}$$

The arguments are the same as those in the previous subsection: if one accepts without question the assumption of the model that problem loading is done at no cost by the operating system, equation (18b) is entirely reasonable, as there never need to be more x's allocated then there are processors; if, on the other hand, one distrusts the assumption, equation (18a) leads to a conservative answer.

To complicate matters further the simulator steers a middle course. When a processor completes the computation of a segment, the x of that job is released. This is the only rational action at this point and suggests that the equation (18b) is more appropriate. On the other hand, as soon as the output of a segment is complete, the simulator attempts to allocate the x and $\underline{y}$ of the next segment, whether or not a processor is immediately available. This is the policy followed by all real multiprogramming systems in their desire to always have work available for the processor and suggests that the equation (18a) is more appropriate. Thus, when x is significant and t_o is of the order of the processing time, the results of simulation would not be expected to agree with analyses based on either (18a) or (18b) but would be expected to fall somewhere between.

Fortunately, the bulk of the analysis occurs in the determination of T as a function of y_2. The transformation of y_2 into M in order to determine C and CE is a relatively simple matter; so both formulations can be evaluated with little additional labor.

The first task in the analysis is to find the density function for $\underline{z}$, the allocated $\underline{y}$'s. In order to allocate a new $\underline{z}_i$ in the range $z < \underline{z}_i < z + \Delta z$, two conditions must be met: (a) $0 < \underline{z}_{i-1} < y_2 - z$ so that allocation is possible, and (b) the first $\underline{y}$ selected that

can be allocated (i.e., that satisfies $\underline{y} < y_2 - \underline{z}_{i-1}$) must also satisfy $z < \underline{y} < z + \Delta z$. This can be expressed by integrating a dummy variable, w, over the possible range of $\underline{z}_{i-1}$, yielding

$$P\,(z < \underline{z}_i < z + \Delta z) = \int_0^{y_2 - z} p_{\underline{z}_{i-1}}(w)\, P\,(z < \underline{y} < z + \Delta z \mid \underline{y} < y_2 - w)\, dw. \tag{19}$$

This equation may be solved to yield the required density function if $p_{\underline{z}_i}$ can be equated with $p_{\underline{z}_{i-1}}$. This is a reasonable assumption for a simple replacement process but in the case at hand is accurate only for large values of r. Accepting this assumption that $p_{\underline{z}_i} = p_{\underline{z}_{i-1}}$ for the moment, the required density function is*

$$p_{\underline{z}}(z) = \frac{\alpha(e^{\alpha(y_2 - z)} - 1)}{1 - e^{-\alpha(y_2 - z)}}, \tag{20}$$

whose integral is plotted in Fig. 14.

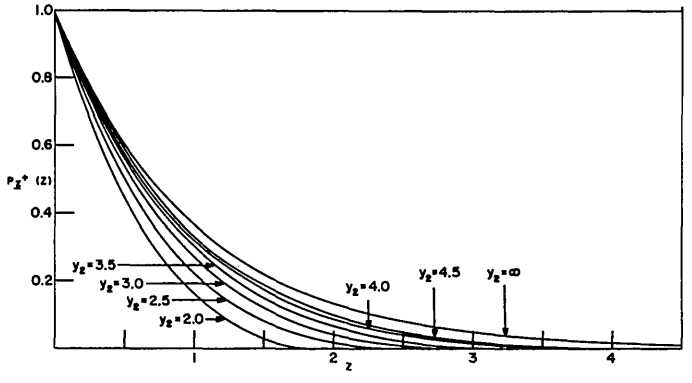


Fig. 14 – $P_{\underline{z}}^{+}(z) = \int_z^{\infty} p_{\underline{z}}(w)\, dw$, multiprogrammed case, termination strategy; $\alpha = 1$.

The probability of conflict is found to be

$$P_c = \frac{e^{-\alpha y_2} + \alpha y_2 - 1}{e^{\alpha y_2} - \alpha y_2 - 1}, \tag{21}$$

and the mean output time per segment is given by

$$t_o = \frac{e^{\alpha y_2} - \alpha y_2 - \frac{(\alpha y_2)^2}{2} - 1}{\alpha r(e^{\alpha y_2} - \alpha y_2 - 1)}. \tag{22}$$

*The deviation of equations (20) through (24) may be found in Appendix E.

When the effect of processor waits is considered, t_{o2} is found to be

$$t_{o2} = \frac{2e^{\alpha y_2}(\alpha r + e^{-\alpha r}) - (\alpha y_2)^2 - 2\alpha y_2 - (\alpha r)^2 - 2}{2\alpha r\,(e^{\alpha y_2} - \alpha y_2 - 1)} . \tag{23}$$

Finally, the mean value of a completed job is found to be

$$V = n \left\{ 1 + \mu \left[m + x + (m + r)\, t_o + \frac{2 - e^{-\alpha y_2}\left[\frac{1}{3}(\alpha y_2)^3 + (\alpha y_2)^2 + 2\alpha y_2 + 2\right]}{\alpha^2 r\,[1 - (1 + \alpha y_2)\, e^{-\alpha y_2}]} \right] \right\} . \tag{24}$$

Equation (9) holds, provided that the value of P_c given by equation (21) is used therein, but since two jobs are running at once, the mean time between job completions is given by

$$t_c = \frac{1}{2}\,\gamma\,(1 + t_{o2}) . \tag{25}$$

The CE calculations may now be completed through the use of equations (3) through (5).

2. Comparison of Analysis and Simulation. There are two approximations in the preceding analysis: the assumption of a simple replacement process in the solution of equation (19), and the use of equation (18a) or (18b) in the determination of y_2. In order to evaluate the effect of these approximations on the accuracy of analytically derived results, a comparison was made of these results with results derived from simulation. The details of this comparison are presented in Appendix F and will be reviewed only briefly here.

The assumption of a simple replacement process is valid at large values of r, because if r is large, even an above-average-sized $\underline{z}$ is unlikely to remain in memory long enough to be involved in the resolution of more than one conflict. For smaller values of r, the analysis overestimates the fraction of $\underline{z}$'s that are above a given fixed value; i.e., the actual $\underline{z}$ distribution is skewed to the left of the theoretical distribution. The amount of skewing is small and decreases as M increases.

An associated effect is that at small r the analysis underestimates P_c and therefore overestimates CE. This overestimation is severe at small M, where conflicts are frequent, and moderate (6% in a typical case) in the vicinity of M_{opt}. Fortunately, the value of M_{opt} is accurately located by the analysis.

The approximation involved in the use of equation (18a) has the opposite effect. At high values of r this equation is appropriate, but at low values of r the effective value of y_2 lies between the value given by equation (18a) and that given by equation

(18b), and the use of equation (18a) leads to an underestimation of CE. The strength of this effect is approximately that of the **z** distribution, and for values of **x** of the order of <y> the two effects nearly cancel.

In summary, for large values of **r** the assumptions made in the preceding analysis have little effect on its validity. At small values of **r**, M_{opt} is correctly located, but there may be an error in CE. For small **x** the theoretical CE is too high, but with increasing **x** the error decreases. In the vicinity of M_{opt} the magnitude of the error is not severe and passes through zero in the vicinity of $x = \langle \underline{y} \rangle$.

3. Numerical Examples and Discussion. Figure 15 compares the CE curves, and Table 2 presents additional comparisons of the monoprogrammed and the multiprogrammed systems. The workload assumed is similar to that considered in connection with Table 1, but **r** has been raised to 10. The large value of **r** assures that the multiprogrammed system will be processor bound and that CE will therefore be primarily determined by memory considerations.

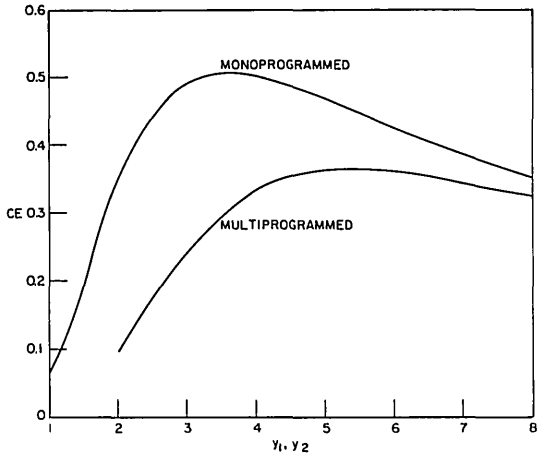


Fig. 15 – CE vs y_1 or y_2 monoprogrammed case and multiprogrammed case with termination strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = \mathbf{n} = 10$.

Since throughput is processor limited in this example, it might be expected that twice as much memory (and almost twice as expensive a system) is required for multiprogramming with no increase in throughput and that the CE of the multiprogrammed system would be little better than half that of the monoprogrammed system. It was seen in connection with Fig. 11 that under conditions of constant memory utilization the ratio was 0.571 for infinite **r**. Calculations based on equation (17) yield a ratio of 0.628 for **r** = 10. In the case at hand, however, where $\underline{y}$ is a random variable, inspection of Table 2 reveals that the CE ratio at optimum **M**'s is 0.725. Evidently under conditions of variable memory requirements there exist factors favoring multiprogramming in addition to the matter of processor utilization discussed in the previous subsection.

A study of Table 2 will reveal the key to the situation. In the monoprogrammed case the optimum value of y_1 is about 3.5 or 4 times $\langle \underline{y} \rangle$, and this is the value of y_1 that reduces γ to about 1.1**n**. In the multiprogrammed case the optimum value of y_2 (which is also that value that reduces γ to about 1.1**n**) is a little less than 5.5 $\langle \underline{y} \rangle$ and considerably

TABLE 2
Comparison of Monoprogrammed and Multiprogrammed (Termination Strategy) Cases;
$\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = \mathbf{n} = 10$.

y_1 or y_2	Monoprogrammed							Multiprogrammed						
	M	t_o	γ	t_c	V	T	CE	M	t_{o2}	γ	t_c	V	T	CE
1.0	3.0	0.04	167.0	174.0	34.9	0.20	0.050							
1.5	3.5	0.06	40.0	42.3	36.8	0.87	0.193							
2.0	4.0	0.07	21.0	22.4	38.3	1.71	0.342	6.0	1.00	54.3	54.3	36.5	0.67	0.096
2.5	4.5	0.08	15.2	16.3	39.5	2.42	0.440	6.5	1.00	29.1	29.1	37.7	1.30	0.173
3.0	5.0	0.09	12.7	13.8	40.5	2.94	0.489	7.0	1.00	19.9	19.9	38.8	1.95	0.244
3.5	5.5	0.10	11.5	12.6	41.2	3.28	0.505	7.5	1.00	15.7	15.7	39.7	2.52	0.297
4.0	6.0	0.10	10.9	11.9	41.7	3.51	0.501	8.0	1.00	13.5	13.5	40.5	3.01	0.334
4.5	6.5	0.10	10.5	11.5	42.1	3.66	0.487	8.5	1.00	12.2	12.2	41.1	3.39	0.357
5.0	7.0	0.10	10.3	11.3	42.4	3.75	0.469	9.0	1.00	11.4	11.4	41.6	3.64	0.364
5.5	7.5	0.10	10.2	11.2	42.6	3.81	0.448	9.5	1.00	10.9	10.9	41.9	3.85	0.366
6.0	8.0	0.10	10.1	11.1	42.7	3.85	0.427	10.0	1.00	10.6	10.6	42.2	3.99	0.362
6.5	8.5	0.10	10.1	11.0	42.8	3.87	0.407	10.5	1.00	10.4	10.4	42.4	4.08	0.355
7.0	9.0	0.10	10.0	11.0	42.9	3.88	0.388	11.0	1.00	10.3	10.3	42.6	4.15	0.346
7.5	9.5	0.10	10.0	11.0	42.9	3.89	0.371	11.5	1.00	10.2	10.2	42.7	4.20	0.336
8.0	10.0	0.10	10.0	11.0	42.9	3.90	0.354	12.0	1.00	10.1	10.1	42.8	4.24	0.326

less than $2y_{1_{opt}}$. Thus, even though t_{o2} of the multiprogrammed system is not much less than $1 + t_o$ of the monoprogrammed system, it is not necessary to double the memory of the multiprogrammed system to reduce γ, and hence t_c, to the value that yields maximum CE.

The situation is analogous to that facing the designer of an electric generating plant. If the plant is to serve a single customer, the ratio of plant capacity to average demand must equal the customer's ratio. If the plant is to serve many customers, and if the pattern of peak demands in uncorrelated among the customers, the plant's ratio of capacity to average demand may be substantially less than that of the customers because of the averaging of peak demands. The averaging of peak memory demands by the multiprogrammed system may likewise be as important as the more obvious advantage of minimizing unused processor time that was considered in the previous subsection.

Under conditions of variable memory demands, multiprogramming tends to raise CE through the averaging of peak memory demands. On the other hand, if t_o is small

compared with τ, the inefficiency attendant to processor waits may decrease or overcome this advantage. In this latter situation, multiprocessing becomes attractive.

Table 3 summarizes, for the single processor situation, the factors which tend to favor or oppose multiprogramming. In support of this summary, Table 4 and Fig. 16, which present a situation similar to that presented in Table 2 and Figure 15 but with **r** lowered to 0.5 to make the system less processor limited, and Figs. 17 through 20 which illustrate the other factors summarized in Table 3, are presented without further comment.

TABLE 3
Factors Favoring and Opposing Multiprogramming.

Factors Favoring Multiprogramming	Factors Opposing Multiprogramming
I/O limited problems	Processor limited problems
Expensive processor	Inexpensive processor
Variable memory requirements	Fixed memory requirements
High cost of memory conflict (e.g. large **n**)	Low cost of memory conflict

TABLE 4
Comparison of Monoprogrammed and Multiprogrammed (Termination Strategy) Cases; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$.

y_1 or y_2	Monoprogrammed							Multiprogrammed						
	M	t_o	γ	t_c	V	T	CE	M	t_{o2}	γ	t_c	V	T	CE
1.0	3.0	0.84	167.0	306.6	47.6	0.16	0.039							
1.5	3.5	1.14	40.0	85.5	56.9	0.67	0.148							
2.0	4.0	1.37	21.0	49.8	65.6	1.32	0.264	6.0	1.44	45.2	55.1	61.4	1.11	0.160
2.5	4.5	1.55	15.2	38.7	73.2	1.89	0.344	6.5	1.60	25.3	32.9	67.6	2.05	0.274
3.0	5.0	1.69	12.7	34.2	79.6	2.33	0.388	7.0	1.72	18.6	25.3	72.7	2.87	0.361
3.5	5.5	1.78	11.5	32.1	84.7	2.64	0.407	7.5	1.83	15.1	21.4	77.9	3.64	0.428
4.0	6.0	1.85	10.9	31.0	88.8	2.86	0.409	8.0	1.91	13.3	19.3	81.9	4.24	0.468
4.5	6.5	1.90	10.5	30.5	91.9	3.01	0.402	8.5	1.98	12.2	18.2	84.5	4.64	0.494
5.0	7.0	1.93	10.3	30.2	94.2	3.12	0.390	9.0	2.03	11.4	17.2	88.5	5.15	0.511
5.5	7.5	1.95	10.2	30.1	95.9	3.19	0.375	9.5	2.08	11.0	16.9	90.7	5.37	0.511
6.0	8.0	1.97	10.1	30.0	97.2	3.24	0.359	10.0	2.13	10.7	16.7	93.4	5.60	0.511
6.5	8.5	1.98	10.1	30.0	98.0	3.27	0.344	10.5	2.14	10.4	16.3	94.8	5.82	0.503
7.0	9.0	1.99	10.0	30.0	98.7	3.29	0.329	11.0	2.16	10.3	16.3	96.2	5.90	0.492
7.5	9.5	1.99	10.0	30.0	99.1	3.30	0.315	11.5	2.18	10.2	16.2	97.2	6.00	0.479
8.0	10.0	1.99	10.0	30.0	99.4	3.31	0.301	12.0	2.18	10.1	16.1	97.5	6.05	0.465

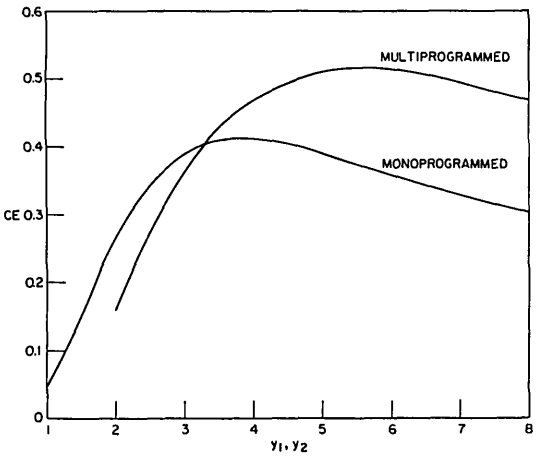


Fig. 16 – CE vs y_1 or y_2, monoprogrammed case and multiprogrammed case with termination strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$.

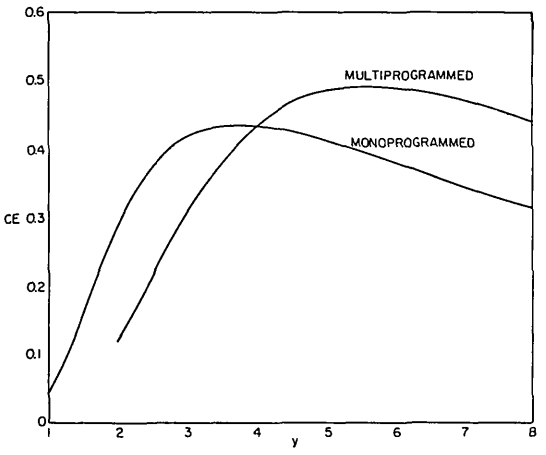


Fig. 17 – CE vs y_1 or y_2, monoprogrammed case and multiprogrammed case with termination strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = \mathbf{r} = 1$, $\mathbf{n} = 10$.

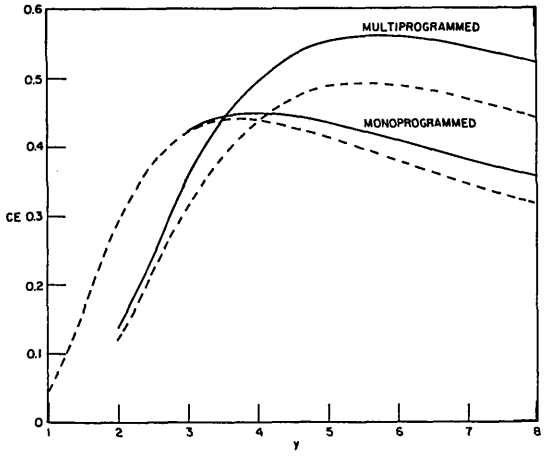


Fig. 18 – CE vs y_1 or y_2, monoprogrammed case and multiprogrammed case with termination strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mathbf{r} = 1$, $\mathbf{n} = 10$; broken lines: $\mu = 1$ (inexpensive processor), solid lines: $\mu = 0.25$.

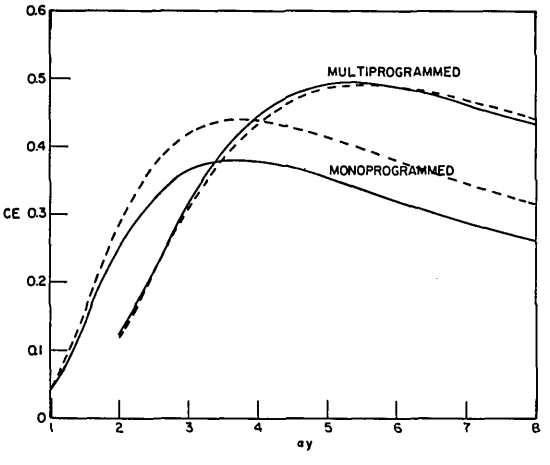


Fig. 19 – CE vs y_1 or y_2, monoprogrammed case and multiprogrammed case with termination strategy; $\mathbf{m} = \mathbf{x} = \mu = \mathbf{r} = 1$, $\mathbf{n} = 10$; broken lines: $\alpha = 1$ (low memory variability), solid lines: $\alpha = 0.5$.

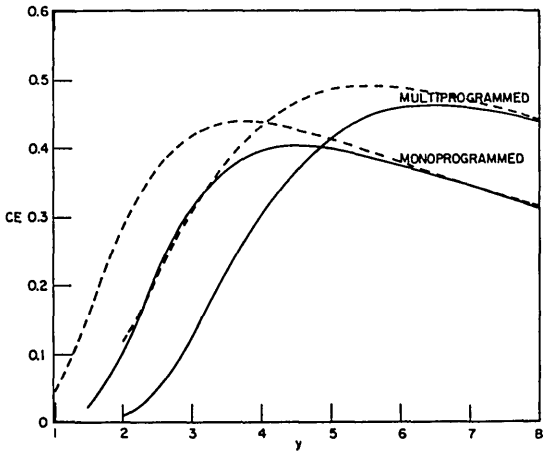


Fig. 20 – CE vs y_1 or y_2, monoprogrammed case and multiprogrammed case with termination strategy; $m = x = \alpha = \mu = r = 1$; broken lines: $n = 10$, solid lines: $n = 20$.

C. Variable Memory Requirements – Suspension Strategy

In this subsection the discussion of multiprogrammed systems will be continued with attention directed at operating systems which suspend rather than terminate a job which makes a memory request that cannot be honored immediately but that can be honored in the future. Thus if $\underline{y}$ exceeds the currently available memory, the requesting job will be suspended, unless $\underline{y}$ exceeds some limit y_t, in which case the job will be terminated. In this subsection y_t will be assumed equal to the y_2 defined by equation (18a); in the next section other values of y_t will be considered.

1. Analytic Results. Since every $\underline{y} < y_2$ is allocated, the probability of termination, P_t, in analogy with equation (7), is

$$P_t = e^{-\alpha y_2}, \tag{26}$$

and the $\underline{z}$ distribution, in analogy with equation (8), is

$$p_{\underline{z}}(z) = \frac{e^{-\alpha z}}{Q_t}, \; 0 < z < y_2; \; 0 \text{ otherwise; where } Q_t = 1 - P_t. \tag{27}$$

The symbol γ_t will be introduced to represent the ratio of segments completed to jobs completed under the suspension strategy. This quantity will usually be greater than n because of the terminations that occur when $y > y_t$. Then γ_t, in analogy with equation (9), is

$$\gamma_t = \frac{1 - Q_t^n}{P_t \cdot Q_t^{n-1}}, \tag{28}$$

and **V**, in analogy with equation (12), is

$$\mathbf{V} = \mathbf{n}\left\{1 + \mu\left[\mathbf{m} + \mathbf{x} + (\mathbf{m} + \mathbf{r})\, \mathbf{t_o} + \frac{2 - [(\alpha y_2)^2 + 2\alpha y_2 + 2]\mathbf{P_t}}{\alpha^2 \mathbf{r}\, \mathbf{Q_t}}\right]\right\}. \tag{29}$$

It follows that

$$\mathbf{t_c} = \frac{1}{2}\gamma\,(1 + \mathbf{t'_{o2}}), \tag{30}$$

where $\mathbf{t'_{o2}}$ is the mean output time plus waiting time per segment for the suspension strategy. It would not be expected that $\mathbf{t'_{o2}}$ would be equal to the $\mathbf{t_{o2}}$ for the termination strategy, as the reduction in γ accompanying the suspension strategy is obtained at the expense of increased waiting time associated with suspended jobs.

For the suspension strategy, the mean output time, $\mathbf{t_o}$, in analogy with equation (10), is

$$\mathbf{t_o} = \frac{1}{\mathbf{r}\,\mathbf{Q_t}}\left[\frac{1}{\alpha} - \left(\frac{1}{\alpha} + \mathbf{y}_2\right)\mathbf{P_t}\right], \tag{31}$$

but $\mathbf{t'_{o2}}$ will be expressed as

$$\mathbf{t'_{o2}} = \mathbf{P_{\bar{s}}}\,\mathbf{t_{o2\bar{s}}} + \mathbf{P_s}\,\mathbf{t_{o2s}}, \tag{32}$$

where

$$\mathbf{P_s} = \mathbf{P}\,(\underline{\mathbf{y}} > \mathbf{y}_2 - \underline{\mathbf{z}} \mid \underline{\mathbf{y}} < \mathbf{y}_2) \tag{33}$$

represents the probability that an allocatable request will cause suspension,

$$\mathbf{P_{\bar{s}}} = 1 - \mathbf{P_s} = \mathbf{P}\,(\underline{\mathbf{y}} < \mathbf{y}_2 - \underline{\mathbf{z}} \mid \underline{\mathbf{y}} < \mathbf{y}_2), \tag{34}$$

and $\mathbf{t_{o2\bar{s}}}$ and $\mathbf{t_{o2s}}$ are the mean output plus waiting times for segments that proceed without and with suspension, respectively.

The unnormalized probability distributions for $\underline{\mathbf{z}}_{\underline{\mathbf{s}}}$ and $\underline{\mathbf{z}}_{\underline{\bar{\mathbf{s}}}}$ are found to be*

$$\mathbf{p}_{\underline{\mathbf{z}}_{\underline{\mathbf{s}}}}(\mathbf{z}) = \frac{\alpha e^{-\alpha y_2}}{\mathbf{Q_t}^2}(1 - e^{-\alpha z}),\ 0 < \mathbf{z} < \mathbf{y}_2;\ 0 \text{ otherwise}; \tag{35}$$

and

$$\mathbf{p}_{\underline{\mathbf{z}}_{\underline{\bar{\mathbf{s}}}}}(\mathbf{z}) = \frac{\alpha}{\mathbf{Q_t}^2}(e^{-\alpha z} - e^{-\alpha y_2}),\ 0 < \mathbf{z} < \mathbf{y}_2;\ 0 \text{ otherwise}; \tag{36}$$

*The derivation of equations (35) through (43) may be found in Appendix G.

and integration of these probabilities yields

$$P_s = \frac{e^{-\alpha y_2}}{Q_t^2}(\alpha y_2 + e^{-\alpha y_2} - 1) \tag{37}$$

and

$$P_{\bar{s}} = \frac{1 - (1 + \alpha y_2)\, e^{-\alpha y_2}}{Q_t^2}. \tag{38}$$

It is straightforward to find

$$t_{os} = \frac{(\alpha y_2)^2 - 2 + (2\alpha y_2 + 2)e^{-\alpha y_2}}{2\alpha r\,(\alpha y_2 + e^{-\alpha y_2} - 1)}, \tag{39}$$

$$t_{o\bar{s}} = \frac{2 - e^{-\alpha y_2}[2 + 2\alpha y_2 + (\alpha y_2)^2]}{2\alpha r\,[1 + e^{-\alpha y_2}(1 + \alpha y_2)]}, \tag{40}$$

and

$$t_{o2\bar{s}} = \frac{2\alpha r + 2e^{-\alpha r} - e^{-\alpha y_2}\,[(\alpha r)^2 + 2 + 2\alpha y_2 + (\alpha y_2)^2]}{2\alpha r\,[1 + e^{-\alpha y_2}(1 + \alpha y_2)]}, \tag{41}$$

where $t_{o\bar{s}}$ and t_{os} are the mean output times for segments that proceed without and with suspension, respectively; but there are difficulties in the calculation of t_{o2s}. The major difficulty lies in the evaluation of the suspension time.

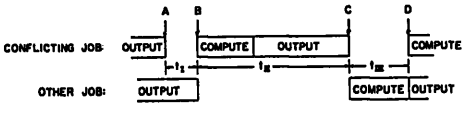


Fig. 21 – Worst-case suspension situation.

The worst case is depicted in Fig. 21. At point A a job completes its output and makes a conflicting memory request for its next segment. The job is suspended until the other job completes its output at point B. The conflicting job has been suspended for the period t_I from A to B. This job is now allocated and run, and completes the output of that segment t_{II} units of time later at point C. In this illustration it is assumed that because the conflicting job is large, the other job cannot be allocated and that it must be suspended for the period t_{II}. At point C the other job resumes, but the conflicting job cannot resume until t_{III} units of time later at point D when the processor becomes available.

Under this pessimistic view of the suspension situation, a reasonable approximation for t_{o2s} is given by

$$t_{o2s} = 2.5 + \frac{1}{2} t_o + 2t_{os} + t_{o\bar{s}} - t_{o2\bar{s}}, \tag{42}$$

and this value can be substituted into equation (32) to complete the analysis.

Now consider the better situations depicted in Fig. 22. In Fig. 22a it is assumed that the other job can resume as soon as processing is complete on the conflicting job. Although the suspension occurred because the conflicting segment could not be allocated along with the current segment of the other job, after the conflicting job starts it may be possible to restart the other job if its next segment is sufficiently smaller than its last to permit the allocation of the next segment along with the conflicting segment.

Fig. 22 – Better alternate suspension situations.

(a) Other job can resume as soon as processing is complete on the conflicting segment.

(b) Conflict can be resolved as soon as the other job releases its **x**.

In this case the conflicting job is suspended for t_I units of time, the other job is suspended for t_{III} units of time, and a reasonable approximation for t_{o2s} is given by

$$t_{o2s} = 1.5 + \frac{1}{2} t_o + t_{os} + t_{o\bar{s}} - t_{o2\bar{s}}. \tag{43a}$$

Although equation (43a) gives the lower limit on t_{o2s} under the model described by equation (18a), which implies that an **x** for both jobs is always required, the simulator does not require the maintenance of an **x** for a job which is performing output. This implies the possibility of the situation depicted in Fig. 22b, where the conflict occurs while the other job is being processed and can be resolved as soon as that job releases its **x**. It would be expected, therefore, that for workloads with a significant value of **x** the lower limit on t_{o2s} would be

$$t_{o2s} = 0.5 + t_{os}. \tag{43b}$$

This is equivalent to stating that some of the conflicts should only be charged a processor wait, i.e., that the use of equation (18a) leads to an overestimation of the probability of suspension.

2. **Comparison of Analysis and Simulation.** Comparisons were made between the results of simulation and calculations based on the analyses just presented. Details may be found in Appendix H.

Good agreement was found between the simulations and equations (28) and (29), and the values found for t'_{o2} fell between calculated values based on the upper limit of t_{o2s} given by equation (42) and on the lower limits of equations (43a) and (43b).

For $\mathbf{x} = 0$, the lower limit of equation (43a) is appropriate, and the simulated values fell approximately midway between the upper and lower limits in the region of $\mathbf{CE_{max}}$. The width of the limits was about 5% in that region. For larger values of $\mathbf{x}$, where the situation described by equation (43b) is possible, the simulated values of t'_{o2} fall below those predicted by equation (43a) for low values of y_2, but equation (43a) is a good approximation in the vicinity of $\mathbf{CE_{max}}$. The width of the limits was about 10% in that region.

3. **Numerical Example and Discussion.** Only one numerical example will be presented at this point, although additional examples may be found in the next section and in Appendix H. Table 5 presents the results for the suspension strategy applied to the same case to which the termination strategy was applied in the derivation of Table 4; data from Table 4 are reproduced in parentheses for comparison. Figure 23 illustrates the **CE** vs **M** relationship for these situations and for the monoprogrammed system.

TABLE 5
Comparison of Termination and Suspension Strategies;
$\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$.

y_2	t_{o2}'	(t_{o2})	γ_t	(γ)	t_c		V		T		CE	
2	1.57	(1.44)	21.5	(45.2)	27.7	(55.1)	63.5	(61.4)	2.29	(1.11)	0.329	(0.160)
3	2.05	(1.72)	12.7	(18.6)	19.4	(25.3)	78.4	(72.7)	4.08	(2.87)	0.510	(0.361)
4	2.21	(1.91)	10.9	(13.3)	17.5	(19.3)	88.0	(81.9)	5.10	(4.24)	0.558	(0.468)
5	2.23	(2.03)	10.4	(11.4)	16.8	(17.2)	93.6	(88.5)	5.60	(5.15)	0.561	(0.511)
6	2.25	(2.13)	10.1	(10.7)	16.4	(16.7)	97.0	(93.4)	5.90	(5.60)	0.537	(0.511)
7	2.26	(2.16)	10.0	(10.3)	16.3	(16.3)	98.5	(96.2)	6.10	(5.90)	0.507	(0.492)
8	2.22	(2.18)	10.0	(10.1)	16.1	(16.1)	99.3	(97.5)	6.15	(6.05)	0.473	(0.465)

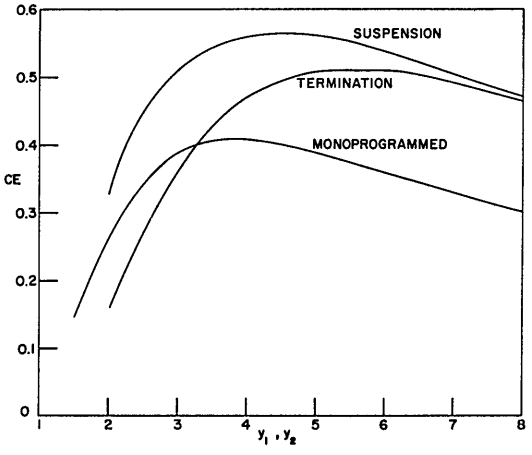


Fig. 23 – CE vs y_1 or y_2, monoprogrammed case and multiprogrammed cases; $m = x = \alpha = \mu = 1$, $r = 0.5$, $n = 10$.

Although the suspension strategy is superior to the termination strategy for all values of **M**, the difference is not substantial at values of **M** in the optimum region and above. This is not surprising, since optimum operation of the termination strategy occurs at a low P_c.

It is interesting to note that for the termination strategy optimum operation occurs with $\gamma \approx 1.1n$ and that for the suspension strategy optimum operation occurs with $t'_{o2} \approx 1.1t_o$. The ratio of γ to **n** represents the wastage of termination, while the ratio of t'_{o2} to t_o represents the wastage of suspension. Thus the t_c's are comparable for the two strategies, although the suspension strategy exhibits a higher mean value because of the completion of the costly suspended problems.

A number of cautions must be attached to the conclusion that the suspension strategy is only somewhat better than the termination strategy in the vicinity of optimum **M**. First, the model assesses no penalty for failure to complete a job; this might be an unrealistic assumption in actual practice. Second, this section is restricted to single processor systems which never attempt to run more than two jobs simultaneously. Finally, the operating systems considered always try to keep a second job active, even when it might be wiser during a peak memory utilization of one job to keep the other job out of core entirely. In the following sections these latter two restrictions will be removed.

V. MULTIPROGRAMMING OF MORE THAN TWO JOBS

A. Constant Memory Requirements

When the memory required for each job is fixed, it is possible to specify N_j, the number of jobs that can be simultaneously allocated in a given **M**. The optimum value of N_j will depend on the ratio of output time to processing time. If the number of jobs is too small, the processor will be underutilized; if the number of jobs is too large, they will have to wait for the processor and the memory devoted to these jobs will be underutilized.

The completion time of a single segment is

$$1 + t_o = 1 + y/r,$$

and the processing time of a segment is 1. If N_j is less than $1 + t_o$, there will be no processor waits and

$$t_c = n \frac{1 + t_o}{N_j}.$$

If N_j is greater than $1 + t_o$, the system will be processor limited and t_c will equal **n**.

In order to display numerical results, it will be necessary to make some assumptions as to how **M** varies as a function of N_j. The choice was made of

$$M = (m + y)N_j + 2x. \tag{44}$$

The 2**x** term can be justified by the argument that a practical operating system would reserve space not only for the **x** of the job being processed but also for the **x** of the next job to be processed.

Figure 24 displays the results of calculations made under the same assumptions of $m = x = y = 1$ that were used in the derivation of Fig. 13. Each of the digits in the body of Fig. 24 indicates the region in the **r**, μ plane where that value of N_j is optimum. Details may be found in Appendix I.

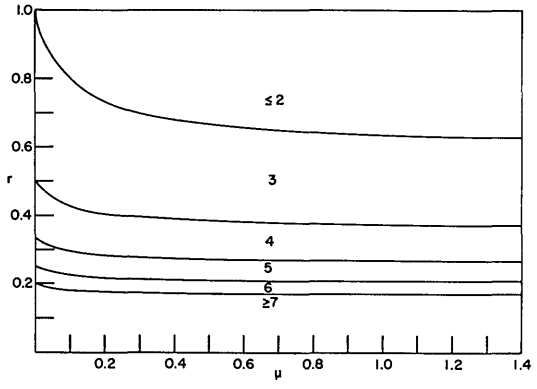


Fig. 24 – Loci of CE ($N_j = k$) = CE ($N_j = k + 1$) in r, μ space, constant memory requirements; $m = x = y = 1$.

It can be seen that if $1 + t_o = 1 + 1/r$ is an integer, that integer is the optimum value of N_j. If $1 + t_o$ is not an integer, one of the two adjacent integers will be optimum; low values of μ favoring the higher value of N_j which keeps the processor occupied at the expense of some memory underutilization, and high values of μ favoring the lower value of N_j which assures perfect memory utilization at the expense of some processor underutilization.

B. Variable Memory Requirements – Termination Strategy

When the job memory requirements are variable, it is no longer possible to specify the number of jobs that will be simultaneously active in a given M. The term N_j will be retained to signify the maximum number of jobs that the operating system ever attempts to allocate simultaneously, and the term N_a will be introduced to signify the actual number of active jobs. The algorithm simulated is that a new job is started whenever $N_a < N_j$ and the amount of unallocated memory in the system is greater than a parameter, M_s.

It would be useful at this point to describe the memory allocation algorithm simulated. The test on N_j and M_s is involved only in the decision to start a job, not in the allocation of any of its segments. When a job is started, a request is made for the allocation of an m, x, and $\underline{y}$. This request is serviced immediately, irrespective of processor availability. Under the termination strategy the job is terminated if the request cannot be granted, although there is no penalty attached to this termination, because no resources have yet been invested in the solution of the job about to be terminated. When processing of the first segment is complete, the memory used for the x of that segment is returned to the pool of available memory and a request for an i/o channel is made. In all the simulations reported herein, a number of i/o channels, N_c, greater than N_j was assumed available, and this request is always immediately granted. When output of the segment is complete, $\underline{y}$ is returned to the pool of available memory, but a request is made immediately for the allocation of the next x and $\underline{y}$, irrespective of processor availability. Termination at this point or at any later time involves the wastage of the investment already made in the job. When the output of the last segment of a job is complete, m and $\underline{y}$ are returned to available memory. N_a is increased by 1 whenever a decision to start a new job is made, and is decreased by 1 whenever a job is terminated or completed.

The behavior of the simulator in preallocating memory irrespective of processor availability is consistent with the assumptions of the model and with history but is probably far from optimum. The assumption that secondary-storage access and transfer times involved in problem loading can be neglected is justifiable if preallocation is performed, and most multiprogramming operating systems perform preallocation in their desire to ensure that there will always be work for the processor. In modern systems, where memory constitutes a major fraction of system cost, and where low-access-time high-speed secondary storages are available, these assumptions break down. If the present work were extended to allow the simulation of secondary-storage access and transfer times, it would permit realistic evaluation of alternatives to the preallocation rule.

In any case, it is necessary to consider the factors that influence the choice of N_j for a given M and a given preallocation policy. It is clearly undesirable to choose an N_j so small that the memory cannot be fully utilized. The maximum reasonable value of N_j, however, depends on M_s. The minimum reasonable value of M_s is $m + x$; attempts to operate with a lower value lead to situations where the system is attempting to start a job when no job, no matter how small its $\underline{y}$, can possibly fit. At this minimum value of M_s there is a strong incentive for choosing an N_j that is not too large for M lest an excessive number of terminations result from attempting to run too many jobs simultaneously; but as M_s is increased, the probability of starting jobs that will be terminated before completion is reduced and the selection of N_j becomes less critical.

Cases where M_s was set at its minimum value will be considered first, then cases where M_s is raised in an attempt to lower the probability of termination and improve CE will be considered. Figure 25 presents the CE vs M results of a number of simulations with various N_j with M_s set at the minimum value, and with r set to 0.2 motivate a substantial amount of multiprogramming.

It can be seen that the value of CE_{max} increases as N_j is increased from 1 to 5 but decreases slightly as N_j is increased from 5 to 6. This limit on N_j results from

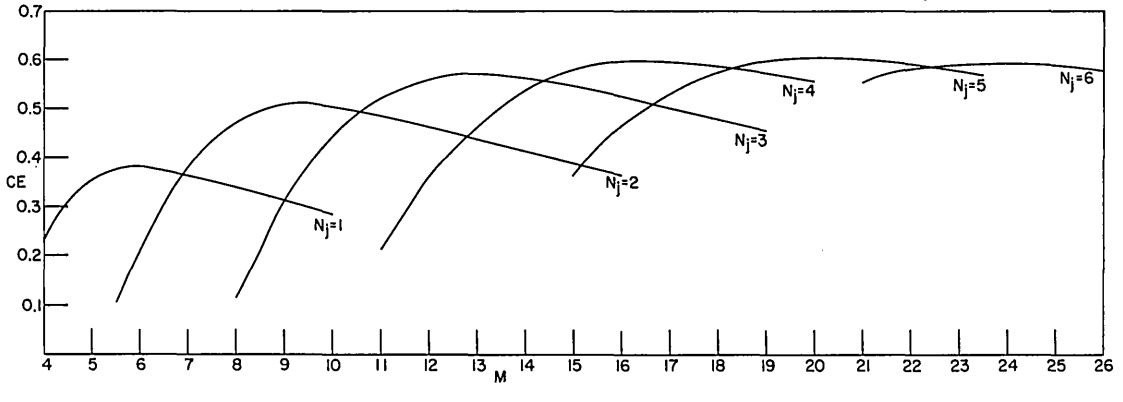


Fig. 25 – CE vs M, termination strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$, $M_s = 2$.

the system becoming processor limited, but the increase in CE_{max} as N_j increases from 1 to 5 is attributable to two causes: improved processor utilization and improved memory utilization. In order to separate these causes, additional simulations were made with the value of μ reduced by a factor of 4 to make the effect of processor utilization more noticeable. As can be seen from Table 6, the effect of this change was minimal.

TABLE 6
Effect of Decreasing μ by a Factor of 4 on the CE Data of Fig. 25; $\mathbf{m} = \mathbf{x} = \alpha = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$.

N_j	Average CE Change (%)	Change in CE_{max} (%)
2	−5.0	−8.4
3	−2.0	−3.5
4	0	−0.2
5	+2.5	+2.5
6	+3.5	+3.5

Although most of the increase in CE with increasing N_j can be attributed to improved memory utilization, it is necessary to distinguish two aspects of this question. In the monoprogrammed case, the memory occupied by **x** is always occupied, even though it is required only a fraction of the time. In a multiprogrammed system the space occupied by an **x** can be released to the system as soon as the processing of a segment is complete. This advantage of multiprogramming holds under conditions of both constant and variable memory requirements, although, as has been discussed previously, practical operating systems may use this space for the assignment of a new job before the processor is ready for it.

Under conditions of variable memory requirements, increasing the number of jobs yields a further improvement in memory utilization by permitting averaging of the peak memory demands of the individual jobs. This advantage of multiprogramming does not apply to the constant memory requirement case.

The simulator always attempts to allocate memory for a job as soon as it can, even though a processor is unavailable. It would be reasonable to conclude, therefore, that the improved memory utilization with increasing N_j observed in Fig. 25 is largely attributable to the averaging of peak memory demands. This may be confirmed by observing for each N_j the values of **M** that yield maximum CE. For $N_j = 1$, maximum CE occurs at $y_1 = 4$, which is considerably greater than $\langle \underline{y} \rangle = 1$. Each subsequent increase of one in N_j increases $y_{N_{j_{opt}}}$ only by about 1.5.

Although the improvement in memory utilization through the averaging of peak requirements continues as N_j is increased, increases in N_j beyond $1 + \langle \underline{y} \rangle / \mathbf{r}$ cause an overall decrease in memory utilization and CE as processor waits are introduced. Thus in Fig. 25 the CE_{max} for $N_j = 6$ is slightly less than the CE_{max} for $N_j = 5$.

It must be remembered that Fig. 25 is derived from an operating system strategy that attempts to start a new job (and reserve space for it) whenever N_a is less than N_j, even though the probability of successful allocation may be very low. Because of the high cost of conflict under the termination strategy, it is wiser to modify the rules for starting a new job.

Two modifications of the rule for job initiation will now be considered. The first is to retain the algorithm but to set M_s at some value higher than the minimum of $m + x$. The other is to abandon the N_j goal entirely and to rely on M_s to control the number of jobs that are allocated. The former alternative will be considered first.

Table 7 displays the results of simulations of the termination strategy made with $N_j = 5$ and with values of M_s in the range 2 to 12. The column headed "Term. Ratio" gives the ratio of jobs terminated in their initial allocation attempt to jobs completed. The column headed "Aband. Ratio" gives the ratio of jobs terminated after having previously completed at least one segment to jobs completed.

In the CE column, an asterisk indicates that some of the simulation runs yielded anomalous values less than 1/2 the usual values. In the other columns, the footnotes indicate the presence of anomalous runs yielding results a significant factor greater than the usual values. The anomalous results are generally attributable to a situation where the available memory is slightly above M_s but so slightly above that many jobs must be attempted before one is found that has a $\underline{y}$ sufficiently small that the job can be allocated. Increasing M_s from 2 to 2.5 removes most of these anomalies, but further increases in M_s are required before the maximum value of CE for a given M is obtained.

In general, the effect of increasing M_s is to reduce the probability of termination and to reduce the mean value of N_a. The reduction of the probability of termination is always beneficial. The reduction in N_a is beneficial when N_j is too large for M, and is associated with the reduction in the probability of termination, but is detrimental if carried too far. The best value of M_s is that which reduces N_a to an appropriate value for the M provided.

Thus if M is too small for N_j, M_s can be increased considerably above its minimum value and improvements in CE attained by reducing N_a. Consider the value of $M = 15$, which is very low for $N_j = 5$. This case produced a CE of 0.364 in the simulation with $M_s = 2$ plotted in Fig. 25. Increasing M_s to 8 raises the CE to 0.556. Each job, on the average, requires 3 units of memory while computing, and 2 while outputting, and is in the output state most of the time. Thus if $N_a = 3$, there would typically be about 8 units of memory available to start the fourth job. Therefore it would be expected that with $M_s = 8$ the usual values of N_a would be 3 and 4. It can be seen in Fig. 25 that for

TABLE 7

Effect of Varying M_s, Termination Strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$, $N_j = 5$

M	M_s	Term. Ratio	Aband. Ratio	γ	CE
	2	§	3.22§	21.2‡	0.364 ± 0.002*
	2.5	4.34	2.84	20.5	0.371 ± 0.004
	3	3.27	2.52	19.7	0.385 ± 0.005
	4	2.30	2.03	18.4	0.402 ± 0.004
15	5	1.43	1.36	15.9	0.451 ± 0.004
	6	0.89	0.87	14.0	0.497 ± 0.003
	8	0.25	0.25	11.3	0.556 ± 0.003
	10	0.08	0.08	10.4	0.511 ± 0.002
	12	0.02	0.02	10.1	0.407 ± 0.002
	2	6.50§	2.18§	17.9‡	0.418 ± 0.004*
	2.5	2.92†	1.99	17.6	0.430 ± 0.003
	3	2.43	1.90	17.5	0.429 ± 0.002
	4	1.76	1.57	16.7	0.442 ± 0.003
15.5	5	1.23	1.17	15.3	0.473 ± 0.004
	6	0.80	0.78	13.7	0.510 ± 0.004
	8	0.28	0.27	11.3	0.558 ± 0.002
	10	0.08	0.08	10.4	0.521 ± 0.002
	12	0.02	0.02	10.1	0.433 ± 0.002
	2	4.60†	1.54†	16.0	0.466 ± 0.003
	2.5	2.31	1.55	16.0	0.465 ± 0.004
	3	1.81	1.39	15.6	0.476 ± 0.003
	4	1.32	1.18	15.1	0.483 ± 0.002
16	5	0.97	0.91	14.2	0.502 ± 0.003
	6	0.66	0.65	13.0	0.533 ± 0.003
	8	0.28	0.28	11.3	0.560 ± 0.002
	10	0.07	0.07	10.4	0.538 ± 0.002
	12	0.02	0.02	10.1	0.446 ± 0.001
	2	3.47§	1.21‡	14.6†	0.500 ± 0.003
	2.5	1.72	1.16	14.6	0.503 ± 0.003
	3	1.42	1.12	14.5	0.505 ± 0.001
	4	1.00	0.89	13.9	0.520 ± 0.003
16.5	5	0.76	0.72	13.3	0.532 ± 0.004
	6	0.57	0.56	12.7	0.544 ± 0.004
	8	0.27	0.27	11.3	0.567 ± 0.002
	10	0.09	0.09	10.5	0.541 ± 0.002
	12	0.02	0.02	10.1	0.463 ± 0.001
	2	§	0.91§	13.7‡	0.528 ± 0.002*
	2.5	1.27	0.88	13.5	0.534 ± 0.002
	3	1.05	0.83	13.4	0.533 ± 0.002
	4	0.80	0.71	13.2	0.542 ± 0.002
17	5	0.66	0.62	12.9	0.551 ± 0.002
	6	0.49	0.47	12.2	0.559 ± 0.003
	8	0.25	0.25	11.3	0.575 ± 0.002
	10	0.10	0.10	10.5	0.547 ± 0.002
	12	0.02	0.02	10.2	0.474 ± 0.001
	2	1.65†	0.71	12.8	0.556 ± 0.002
	2.5	0.97	0.67	12.8	0.554 ± 0.003
	3	0.79	0.62	12.6	0.561 ± 0.004
	4	0.63	0.55	12.5	0.563 ± 0.003
17.5	5	0.50	0.47	12.2	0.570 ± 0.003
	6	0.39	0.38	11.8	0.578 ± 0.003
	8	0.22	0.22	11.1	0.582 ± 0.003
	10	0.10	0.10	10.5	0.557 ± 0.001
	12	0.02	0.02	10.1	0.490 ± 0.001
	2	‡	0.51†	12.2	0.572 ± 0.003
	2.5	0.76	0.55	12.2	0.576 ± 0.003
	3	0.64	0.50	12.1	0.575 ± 0.002
	4	0.49	0.43	12.0	0.578 ± 0.003
18	5	0.40	0.37	11.8	0.581 ± 0.003
	6	0.32	0.31	11.5	0.589 ± 0.002
	8	0.20	0.20	11.0	0.585 ± 0.001
	10	0.09	0.09	10.5	0.564 ± 0.001
	12	0.02	0.02	10.1	0.502 ± 0.002
	2	0.95†	0.41†	11.8	0.584 ± 0.002
	2.5	0.62	0.43	11.8	0.589 ± 0.002
	3	0.51	0.40	11.7	0.588 ± 0.003
	4	0.39	0.34	11.5	0.593 ± 0.003
18.5	5	0.31	0.29	11.4	0.595 ± 0.002
	6	0.25	0.24	11.2	0.599 ± 0.001
	8	0.17	0.16	10.9	0.592 ± 0.002
	10	0.10	0.10	10.5	0.568 ± 0.002
	12	0.03	0.03	10.2	0.511 ± 0.001
	2	0.61‡	0.34	11.4	0.597 ± 0.003
	2.5	0.44	0.31	11.3	0.596 ± 0.002
	3	0.39	0.31	11.3	0.599 ± 0.002
	4	0.30	0.27	11.2	0.001 ± 0.003
19	5	0.28	0.27	11.2	0.601 ± 0.002
	6	0.21	0.20	11.0	0.604 ± 0.002
	8	0.13	0.13	10.7	0.599 ± 0.002
	10	0.08	0.08	10.4	0.573 ± 0.002
	12	0.03	0.03	10.2	0.518 ± 0.001

*Some anomalous values less than 1/2 the usual values.

†Some anomalous values a significant factor less than 2 times greater than the usual values.

‡Some anomalous values between 2 and 10 times the usual values.

§Some anomalous values greater than 10 times the usual values.

TABLE 7–Continued

Effect of Varying M_s, Termination Strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$, $N_j = 5$

M	M_s	Term. Ratio	Aband. Ratio	γ	CE
	2	0.55†	0.27	11.1	0.600 ± 0.002
	2.5	0.35	0.24	11.0	0.602 ± 0.002
	3	0.30	0.23	11.0	0.603 ± 0.004
	4	0.24	0.21	11.0	0.607 ± 0.002
19.5	5	0.20	0.19	10.9	0.608 ± 0.002
	6	0.17	0.17	10.9	0.604 ± 0.002
	8	0.11	0.11	10.6	0.605 ± 0.002
	10	0.07	0.07	10.4	0.579 ± 0.001
	12	0.04	0.04	10.2	0.529 ± 0.001
	2	0.44†	0.21	10.9	0.603 ± 0.001
	2.5	0.26	0.18	10.8	0.605 ± 0.002
	3	0.24	0.19	10.8	0.608 ± 0.002
	4	0.18	0.16	10.8	0.607 ± 0.001
20	5	0.15	0.14	10.7	0.607 ± 0.002
	6	0.12	0.11	10.6	0.608 ± 0.002
	8	0.10	0.10	10.5	0.601 ± 0.002
	10	0.06	0.06	10.3	0.581 ± 0.001
	12	0.03	0.03	10.2	0.536 ± 0.002
	2	0.27†	0.14†	10.7	0.602 ± 0.002
	2.5	0.21	0.15	10.7	0.601 ± 0.003
	3	0.19	0.15	10.6	0.600 ± 0.002
	4	0.16	0.14	10.7	0.603 ± 0.002
20.5	5	0.12	0.11	10.5	0.605 ± 0.002
	6	0.11	0.10	10.5	0.607 ± 0.002
	8	0.07	0.07	10.4	0.602 ± 0.001
	10	0.05	0.05	10.3	0.584 ± 0.001
	12	0.03	0.03	10.2	0.541 ± 0.001
	2	‡	0.12	10.5	0.601 ± 0.002
	2.5	0.17	0.11	10.5	0.601 ± 0.003
	3	0.15	0.12	10.5	0.598 ± 0.002
	4	0.10	0.10	10.5	0.601 ± 0.002
21	5	0.10	0.09	10.4	0.600 ± 0.002
	6	0.08	0.08	10.4	0.598 ± 0.002
	8	0.06	0.06	10.3	0.598 ± 0.002
	10	0.04	0.04	10.2	0.583 ± 0.001
	12	0.03	0.03	10.2	0.544 ± 0.001
	2	0.16‡	0.08	10.4	0.595 ± 0.002
	2.5	0.12	0.09	10.4	0.596 ± 0.002
	3	0.11	0.09	10.4	0.596 ± 0.002
	4	0.08	0.07	10.3	0.597 ± 0.002
21.5	5	0.08	0.07	10.4	0.597 ± 0.002
	6	0.06	0.06	10.3	0.597 ± 0.002
	8	0.05	0.05	10.3	0.595 ± 0.002
	10	0.04	0.04	10.2	0.578 ± 0.001
	12	0.02	0.02	10.2	0.547 ± 0.001
	2	0.09†	0.06	10.3	0.591 ± 0.002
	2.5	0.09	0.07	10.3	0.591 ± 0.002
	3	0.08	0.06	10.3	0.589 ± 0.003
	4	0.07	0.07	10.3	0.587 ± 0.007
22	5	0.05	0.04	10.2	0.593 ± 0.002
	6	0.05	0.04	10.2	0.592 ± 0.002
	8	0.04	0.04	10.2	0.589 ± 0.002
	10	0.02	0.02	10.2	0.576 ± 0.002
	12	0.02	0.02	10.2	0.550 ± 0.002

†Some anomalous values a significant factor less than 2 times greater than the usual values.

‡Some anomalous values between 2 and 10 times the usual values.

M = 15 a CE of 0.556 lies between the curves for $N_j = 3$ and $N_j = 4$. The overall effect of raising M_s to 8 has been to reduce N_a to an appropriate value for that M and r.

However, if M_s is raised to 12, it would be expected that the usual value of N_a would be 2, with a third job started only occasionally. The CE value of 0.407 found in

Table 7 lies just above the $N_j = 2$ curve in Fig. 25. Since this is too small a N_a for an efficient use of a memory of size 15, the CE is less than could be obtained with a smaller M_s.

Note that increases in M_s have a much stronger effect on reducing the termination ratio than on the abandonment ratio. For small M_s the ratio of terminations (including jobs which have completed no segments) to abandonments is as high as 3 rather than the 1.1 that would be expected if a conflict was equally likely at any segment. This large ratio of terminations to abandonments results from situations where the system is attempting to start a job when the other jobs are occupying an above-average amount of memory, and many allocations must be attempted before one can be found with a sufficiently small $\underline{y}$ to fit. Later allocations of that job are less likely to conflict, because the other jobs are more likely than not to contract in the direction of their average size. On the other hand, for large M_s the ratio is about 1, implying that initial allocation attempts are nearly always successful because a job is started only when the available memory is much larger than the average required by a job.

Because of the freedom not to attempt a particular job and the assumption of zero operating system costs in the model, it is the abandonment ratio rather than the termination ratio that controls CE; i.e., there is no penalty associated with the termination of a job before the processing of its first segment. In practice, where operating system functions do utilize system resources, values of M_s above the minimum become still more attractive.

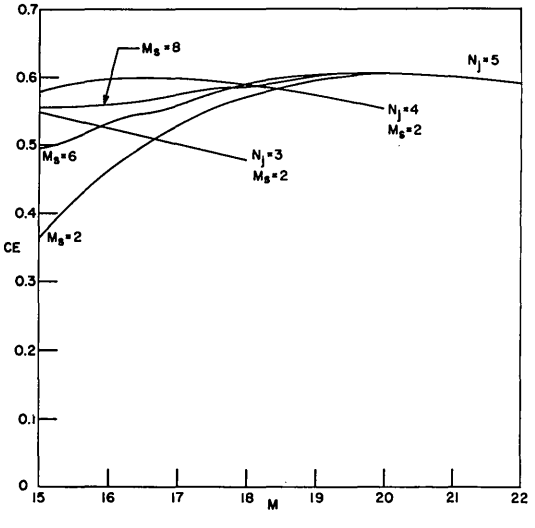


Fig. 26 – CE vs M, termination strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$.

Figure 26 combines a portion of Fig. 25 with some of the CE data from Table 7. Note that for values of M less than 18, i.e., values of M too small for reasonable operation with $N_j = 5$ and $M_s = 2$, the CE's obtainable with lower values of N_j can be approached, but not obtained, by maintaining N_j at 5 and increasing M_s.

The preceding discussion concerned the effect of increasing M_s above its minimum value when M was too small for N_j. Inspection of Table 7 and Fig. 26 reveals that for large values of M increases in M_s have little effect until M_s reaches such a large value that the system does not start the fifth job

and CE declines. This suggests that the system is not trying to start too many jobs and leads to the question of whether the system is being too conservative and should not attempt to start a sixth job when the other jobs are occupying a below-average amount of memory.

The general answer to this question is in the negative, because at the reasonably large values of **n** considered it is improbable that the fortuitious situation which permitted the initiation of the sixth job will persist long enough for the jobs in the system to go to completion without conflicts and resulting termination. While it is possible that a sophisticated operating system should consider how near the jobs in the system are to completion in deciding whether or not to allocate a new job when the amount of available memory is marginal, for the unsophisticated system simulated it is wiser to rely on average memory requirements than on the instantaneously available memory. Average memory requirements are relied upon in the selection of a suitable N_j, although it is still desirable to use values of M_s above the minimum.

This point can be demonstrated by considering the alternative strategy in which the N_j maximum is abandoned and the decision to start a new job is based entirely on whether the available memory exceeds M_s.

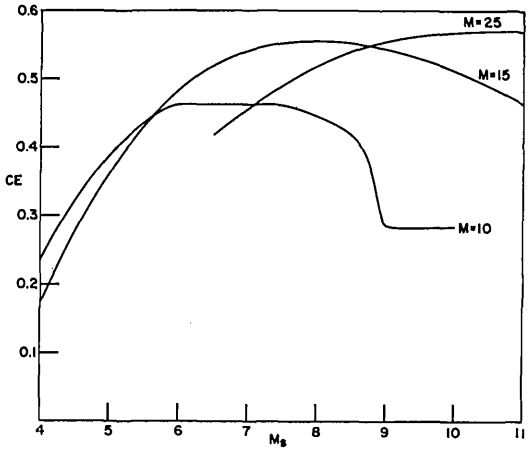


Fig. 27 – CE vs M_s, termination strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$.

Figure 27 illustrates the effect of M_s on CE for three different values of **M**. The shape of the curve for **M** = 10 deserves some comment. For M_s in the range 9 to 10, the CE curve is flat. In this region, one job will always be started, but a second cannot possibly be started until the first is completed or terminated. The value of CE is in agreement with that shown in Fig. 25 for $N_j = 1$. There is another flat portion in the M_s range 6 to 7.5. The value of CE, however, is less than that shown in Fig. 25 for $N_j = 2$. Suppose that two jobs are active, and that one is performing output and the second is about to perform output. The first job occupies a minimum of 1 unit of memory and an average of 2. The second job releases its **x** and its memory requirements shrink to those of the first job. The available memory now averages 6 units and may be as much as 8. A third job is started therefore. As soon as the first or second job completes output, a new **x** and $\underline{y}$ is requested, even if the third job is still processing. The total memory

requirement is now three **m**'s, two **x**'s, and three **y**'s which can easily exceed **M** and cause termination. Thus the decision to start the third job was unwise. In the case shown in Fig. 25 for $N_j = 2$ the third job is never attempted.

The discussion of the termination strategy will now be concluded with the presentation of Fig. 28 which shows the **CE** vs **M** relationships for a system which always attempts to start a new job whenever the available memory exceeds M_s. Superimposed on this figure is the upper envelope of Fig. 25, i.e., the **CE** vs **M** relationship for optimum N_j and $M_s = 2$. It can be seen that this limit is sometimes equaled (for small **M**) but never exceeded, and it should be recalled that somewhat better values of **CE** can be obtained with optimum N_j and moderately increased M_s.

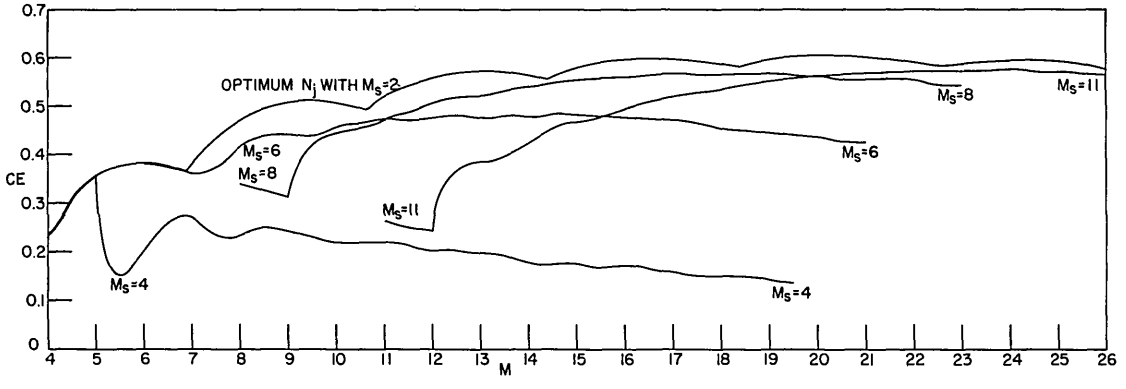


Fig. 28 – **CE** vs **M**, termination strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$.

Note that for each of the four values of M_s considered, the **CE** vs **M** curve in the range $M_s < M < M_s + 1$ follows the $N_j = 1$ curve of Fig. 25. As **M** exceeds $M_s + 1$, a second job is sometimes started. For $M_s = 4$, $M = 5$, the probability of conflict for the second job is so high that this increase in **M** causes a significant loss in **CE**. For $M_s = 11$, $M = 12$, the probability of conflict is so low that the **CE** nearly doubles as **M** exceeds 12. Intermediate values of M_s exhibit intermediate behavior.

Generalizing, the termination strategy with substantial **n** exacts a heavy penalty for a memory conflict and forces considerable conservatism in job initiation. Since each job requires a wide variety of $\underline{y}$'s before it is completed, average memory requirements are more appropriate in making the initiation decision than are the instantaneous requirements of the jobs in the systems. In the present model, where each job has the same mean memory requirements, the best strategy of those simulated is to pick a value of N_j appropriate to **M** and to set M_s somewhat above the minimum value.

C. Variable Memory Requirements—Suspension Strategy

The heavy penalty associated with a memory conflict under the termination strategy led to considerable conservatism in job initiation and therefore to underutilization of the memory. Another difficulty with the termination strategy is that in real life jobs must be done, and a strategy which has a significant probability of abandoning a given job completely is unsatisfactory. Both of these arguments favor the suspension strategy, but the caution must be offered at the outset that the model is biased in favor of the suspension strategy.

The model assumes that operating system functions are obtained at no cost, or that the costs are included within the workload specifications. Under the termination strategy there are exactly **n** memory allocations and operating system decisions per job, and it is not too unreasonable to suppose that these costs can be included in the workload. Under the suspension strategy the number of times the operating system will have to make decisions about a particular job is unpredictable.

In discussing the suspension strategy it is necessary to clarify how N_a is defined. In the case of the termination strategy there was no problem; a job could be in a processing state, in which case its memory occupancy was $m + x + \underline{y}$, it could be in the process of output, in which case its occupancy was $m + \underline{y}$, or it could be allocated and awaiting a processor, in which case its occupancy was $m + x + \underline{y}$. There was always a $\underline{y}$ allocated, since as soon as the output of one segment was finished, the system attempted to allocate the next segment. All allocation requests were immediately honored, or else the job was terminated. In the suspension strategy, however, not all allocation requests will be immediately honored. Such suspended jobs will still be counted in determining N_a. Most of the time these jobs will be occupying **m** units of memory, but in those cases when the suspension occurred on the initial allocation request after the job was initiated the suspended job occupies no memory.

Thus, under the suspension strategy the importance of choosing appropriate limits on the job initiation parameters is increased because of the risk that the memory may become filled with the **m**'s of suspended jobs. This was not a problem under the termination strategy, where M_s could be used to control the number of active jobs if N_j was set too large for **M**.

Another complication attendant to the suspension strategy is the choice of y_t, which is the maximum $\underline{y}$ that can be requested before the requesting job is terminated. The

maximum possible value of y_t is $M - (m + x)$. A very conservative value of y_t, used in the analysis of the previous section is

$$y_t = M - N_j(m + x). \tag{45}$$

The use of values of y_t close to the maximum carries the risk that all other jobs in the system will have to be completed and release all of their memory (including m) before the requesting job can resume. On the other hand it will be seen later that the value of y_t specified by equation (45) is too conservative.

Even when this conservative value of y_t is employed, the suspension strategy has a better CE than the termination strategy. Figure 29 displays the CE vs M results of a number of simulations which employed the same workload model as was used in the derivation of Fig. 25 but which used the suspension strategy with $M_s = 2$ and the y_t of equation (45). The broken lines in Fig. 29 reproduce the envelope of Fig. 25.

As before, CE_{max} increases as N_j increases from 1 to 5, but the optimum range for a given N_j occurs at lower values of M for the suspension strategy. Since the suspension strategy permits higher values of N_j (up to 6, at which point the system becomes processor limited), and since higher values of N_j yield better memory utilization (and better processor utilization, although the effect of processor utilization is second order at the level of μ employed), higher CE's are available at a given M when the suspension strategy is employed.

For $N_j = 1$, where the two strategies are indistinguishable, the value of M_{opt} is about 6. Under the termination strategy, each increase of one in N_j increases M_{opt} by

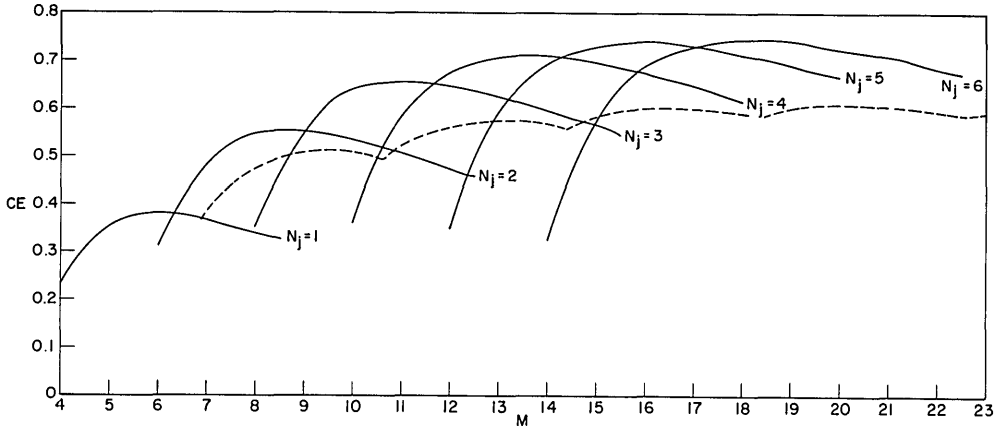


Fig. 29 – CE vs M, suspension strategy; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$, $M_s = 2$, $y_t = M - 2\,N_j$.

about 3.5 (even though the average memory utilization of a job is nearly 3) in order to keep the conflict probability acceptably low. Under the suspension strategy, each increase of one in N_j increases M_{opt} by only 2.5.

One reservation must be offered at this point. The simulator attempts to allocate the memory of jobs for which no processor is available. The suspension strategy overcomes this defect by placing jobs in a suspended state and deferring this allocation. Had the simulator attempted to allocate memory for the **x** and $\underline{y}$ of only those jobs for which a processor was available or about to be available, the difference between the strategies might have been less pronounced.

Attention must be given to the optimum value of y_t. A number of simulations were performed in order to investigate the appropriate value of this parameter. Figure 30 displays the results of simulations with $N_j = 2$ employing the same workload model as in Figs. 25 and 29. The $y_t = M - 4$ curve corresponds to the $N_j = 2$ curve of Fig. 29.

It can be seen that increasing y_t up to one unit of memory above the value specified by equation (45) causes a moderate improvement in **CE**, especially at low values of **M**. Further increases in y_t, however, cause the simulations to become unstable. To account for this instability suppose that $M = 6$ and $y_t = 3 + \epsilon$. If two jobs are running, and if the request is for an unusually large **y**, the usual procedure is to suspend that job until the other job completes the output of a segment, at which time its **x** and **y** will become available. If the request is for a $\underline{y}$ of $3 + \delta$, however, and if $\delta < \epsilon$, the job will not be terminated because $\underline{y} < y_t$. On the other hand, the memory requirements of the job are greater than 5, so it cannot be allocated as long as the other job is occupying its **m**. This suggests an upper limit on the value of y_t of the form

$$y_t = M - N_j\, m - x \tag{46}$$

as the maximum value of y_t.

In an attempt to confirm equation (45), additional simulations were made with other values of **m** and **x**. For $N_j = 2$, $x = 0$, the maximum value of y_t was found to be $M - 2m$. For $N_j = 2$, $m = 0$, $x = 1$, the case shown in Fig. 31, the maximum value of y_t was $M - 1$. In all cases, attempts to run with a y_t above that specified by equation (46) resulted in unstable simulations and usually in marked decreases in **CE**. For all cases with $N_j = 2$, the optimum **CE** was obtained with y_t set at the limit, although the differences between the use of equations (45) and (46) were substantial only at small values of **M**.

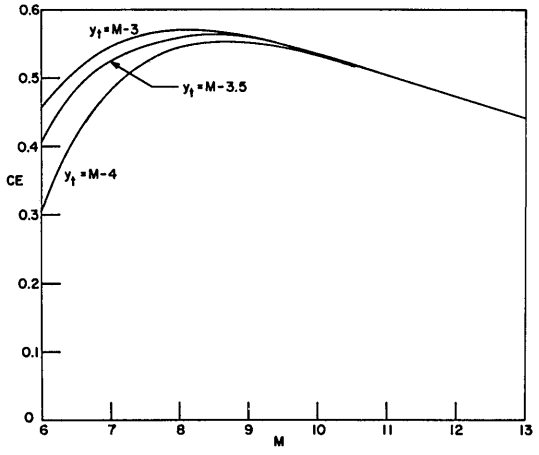


Fig. 30—CE vs **M**, suspension strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1, \mathbf{r} = 0.2, \mathbf{n} = 10, \mathbf{N_j} = \mathbf{M_s} = 2$.

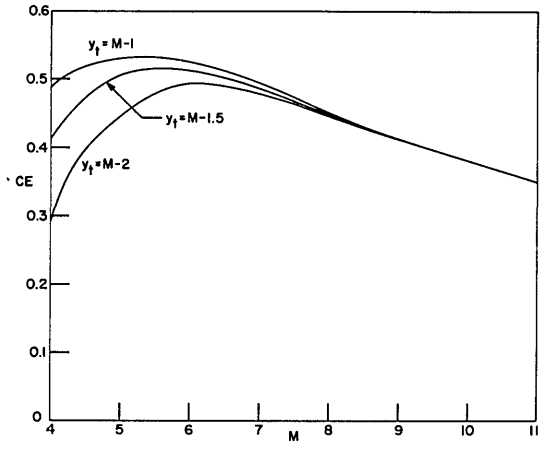


Fig. 31—CE vs **M**, suspension strategy; $\mathbf{m} = 0$, $\mathbf{x} = \alpha = \mu = \mathbf{r} = 1$, $\mathbf{n} = 10$, $\mathbf{N_j} = 2$, $\mathbf{M_s} = 1$.

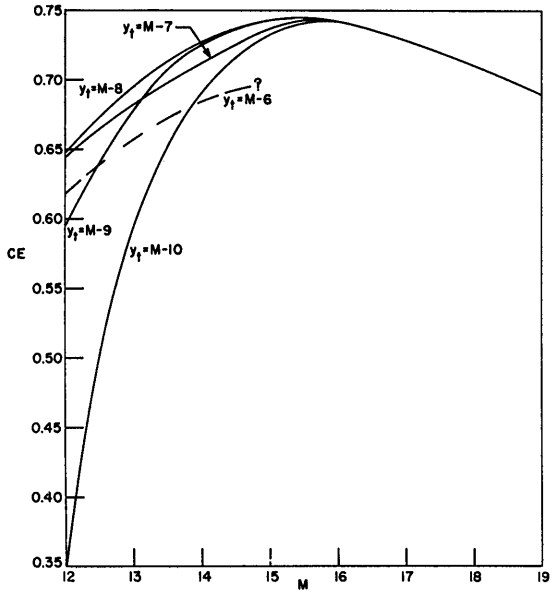


Fig. 32—CE vs **M**, suspension strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$, $\mathbf{N_j} = 5$, $\mathbf{M_s} = 1$.

Figure 32 presents the case where $\mathbf{m} = \mathbf{x} = 1$, and $\mathbf{N_j} = 5$. Note that the ordinate scale of this figure is considerably magnified with respect to the other **CE** scales in order to display relatively weak phenomena.

As before, the value of $\mathbf{y_t}$ derived from equation (45) is too conservative, but optimum performance results from increasing that $\mathbf{y_t}$ by only 2, rather than the 4 suggested by equation (45). Further increases in $\mathbf{y_t}$ lead to unstable simulations, but with less catastrophic **CE** reductions than were found for $\mathbf{N_j} = 2$. The broken curve in Fig. 32 for $\mathbf{y_t} = \mathbf{M} - 6$ (the value suggested by equation (46)) locates the means of many simulations whose behavior varied substantially, but none of which was better than those with $\mathbf{y_t} = \mathbf{M} - 8$.

Although the best performance resulted from $\mathbf{y_t} = \mathbf{M} - 8$, very nearly equal performance was obtained from $\mathbf{y_t} = \mathbf{M} - 9$. Thus

$$\mathbf{y_{t_{opt}}} \approx \mathbf{M} - \mathbf{N_j m} - (\mathbf{N_j} - 1)\,\mathbf{x} \tag{47}$$

would be a reasonably approximation for the limited range of workloads and $\mathbf{N_j}$ is considered.

In the previous subsection, which was devoted to the termination strategy, two approaches were considered for dynamically varying the number of active jobs in response to memory availability: retaining an N_j limit and a conservative M_s, and abandoning the N_j limit and depending entirely on M_s. For the suspension strategy the choice of approach is clouded by the fact that the suspension strategy has, in effect, the option of reducing the number of active jobs by suspending a job and reducing its memory requirements to **m**, although such jobs are counted in determining N_a. Thus, while in the case of the termination strategy there was considerable motivation for increasing M_s above the minimum value, particularly when N_j was too large for **M**, there is much less motivation for increasing M_s above the minimum in the suspension strategy, provided that N_j and y_t are reasonable.

In a number of simulations with reasonable N_j limits relative to **M**, increasing M_s had no perceptible effect on **CE** until it was increased to a value roughly an average job requirement above the minimum value, at which point **CE** began to decrease.

On the other hand, if N_j is unreasonably high, or if this limit is removed entirely, an above-minimum value of M_s provides some protection against inefficient operation.

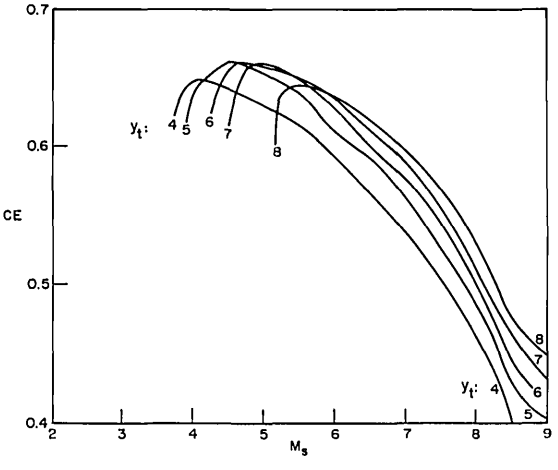


Fig. 33 – **CE** vs M_s, suspension strategy; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = 0.2$, $\mathbf{n} = 10$, $\mathbf{M} = 11$.

Figure 33 displays the results of a number of simulations in which the N_j limit was removed. The workload model is the same as was used in the simulations displayed in Fig. 29.

The simulator has an internal limitation of $N_a \leq 10$. An attempt to allocate an eleventh job will cause the simulation to terminate. The simulations in question were run therefore at $\mathbf{M} = 11$, assuring that an attempt to run an eleventh job would always be unwise.

Inspection of Fig. 33 reveals that for optimum combinations of M_s and y_t the **CE** attains the value shown in Fig. 29 for $\mathbf{M} = 11$ and $N_j = 3$ but that the optimum range is rather narrow; i.e. **CE** is very sensitive to y_t and M_s. For values of M_s above 5, **CE** declines, because the memory is being underutilized. At these large values of M_s, conservative values of y_t lead to unnecessary termination and further reduction in **CE**.

For values of M_s below 5, **CE** declines rapidly and the simulations become unstable. The extent to which M_s can be decreased below 5 is affected by the value of y_t; small

values of y_t assure that unusually large $\underline{y}$'s will not cause suspension and allow operation at somewhat smaller values of M_s, although reductions in y_t lower the maximum available CE.

The unstable behavior shown in Fig. 33 for small M_s and in Fig. 32 for large y_t was investigated by inspecting snapshots of the internal state of the simulated system at periodic intervals. It was found that this unstable behavior was associated with situations in which N_a has grown to an unreasonably large value for the M in question.

Suppose that some job requests an unusually large $\underline{y}$ that is just less than some large y_t. That y cannot be allocated, and the requesting job is suspended. Every time additional memory is available, a determination is made as to whether that job can be allocated and resumed. It is entirely possible, however, that there are other suspended jobs requiring smaller amounts of memory, and that the newly available memory will be assigned to one of these other jobs. Thus the job that has requested an unusually large $\underline{y}$ may be suspended for an extremely long time if y_t is large enough that the job will not be terminated and if N_j is large enough to make it likely that there will be more than one suspended job or if N_j is unlimited, in which case the newly available space may be devoted to starting another job. During this extremely long suspension time, the space occupied by the m of the suspended job is wasted, and the effective numbers of jobs will have been reduced.

The explanation of the unstable simulations is that in a simulation of reasonable length, i.e., of a length that yields consistent results with conservative values of M_s and y_t, these effects occur a very small number of times, and the simulated results will be markedly affected by exactly how many long suspensions occur and, in the case of suspensions which persist until the end of the simulation, by the time of their occurrence. For example, in Fig. 32, for large values of y_t some simulations yielded results consistent with those obtained with optimum y_t whereas others yielded considerably poorer results. The former were those simulations in which the long suspensions did not occur; the latter were those in which an m was wasted and N_j was reduced below its optimum value.

In the cases depicted in Fig. 33, if the requested $\underline{y}$ lay very close to y_t, but if M_s was below $x + y_t$, every time the available memory was between M_s and $x + \underline{y}$ a new job would be started, and the suspended job could be resumed only in the rare instances when the available memory was increased from some value less than M_s to some value greater than $x + \underline{y}$.

Some of the simulations on which Fig. 33 is based even terminated with $N_a > 10$. In one sense this is merely an artifact of the simulator: if the very large $\underline{y}$ was requested on the first segment of a job, the **m** was never allocated; thus the suspended job was occupying no memory even though the simulator counted it in evaluating N_a. In a practical sense, however, this constitutes a real problem. Practical operating systems must have some limit on the number of active jobs, as they must keep the specifications of these jobs in main memory.

The conclusion that can be drawn is that if the suspension strategy is used, either N_j must be carefully selected or else more sophisticated control over the job initiation and suspension process must be provided than was available in the simulator. If N_j is too small, the memory will be underutilized. If N_j is too large, the mechanisms provided in the simulator will not give satisfactory performance. Under these circumstances, low values of y_t cause a substantial amount of termination, while high values of y_t will allow long suspensions unless M_s is raised to $x + y_t$, which would cause serious memory underutilization. Although somewhat smaller values of M_s gave satisfactory results in the simulations, this may well be due to the low probability of requesting a $\underline{y}$ between $M_s - x$ and y_t during the finite duration of any one simulation.

It seems reasonable to expect that improvements could be achieved by the use of more sophisticated operating systems than the one simulated. The problems just described suggest a number of candidates for subsequent investigations.

The operating system should be aware it has suspended a job and should vary its rules for initiating new jobs, and possibly for continuing the processing of other jobs, while a job is suspended. Intuitively it seems sensible that while a job is suspended, but occupying memory, it is usually unwise to start a new job. If new jobs are not started, the entire memory would eventually become available for the allocation of the suspended job.

Another way of handling the situation would be to prevent any subsequent allocation requests from the other jobs from being honored until the suspended job was allocated. Although this would permit allocation of the suspended job more quickly in most cases, the other jobs would still be occupying their **m**'s, and the maximum request that could be

honored would be smaller. Either y_t would have to be set to $M - N_j m - x$, or else requests for larger $\underline{y}$'s would have to be deferred until the other jobs began to release their **m**'s, and the other jobs would have to be permitted to continue to have their allocation requests honored so that they could be completed. This suggests that the action to be taken when an allocation request is suspended is probably determined by the size of that request which can be used to derive the expected time required before it can be honored under alternative strategies.

A simple technique worth investigating would be to keep track of how long a job has been suspended. If this time is unreasonably long, then the measures already taken are probably not sufficient and some more drastic measures should be taken.

If while a job is suspended the other jobs are permitted to make allocation requests in the hope that they will eventually be completed and release their **m**, and if one of these jobs makes an allocation request that cannot be granted, a serious problem ensues. If termination is not permitted, that job too must be suspended, and processor utilization may decline markedly while the conflicts are being resolved. It seems clear that the rules for handling conflicting memory requests should vary according to how many jobs are already suspended and how the system is trying to collect memory for them. If secondary-storage access and transfer times are reasonable, the complete roll-out of one or more of the suspended jobs would seem attractive. If these times were very short, then roll-out might be useful in resolving even the simplest conflicts.

The analysis of these techniques would require detailed consideration of operating system processing costs and secondary-storage access and transfer times. These factors are ignored in the throughput model, and the analysis of the more sophisticated operating systems lies beyond the scope of the present investigation.

VI. MULTIPROCESSING

Two motivations for multiprogramming were examined in the previous section: the provision of sufficient work to assure high processor utilization and the averaging of peak memory demands. The amount of multiprogramming that could be accomplished without loss in CE was limited, however, by the availability of the processor. If an attempt was made to keep too many programs simultaneously active, each program would suffer substantial waits for the processor, and the memory allocated to these programs would not be beneficially utilized during these processor waits.

This situation provides the motivation for multiprocessing even if questions of reliability are ignored. Multiprocessing permits increased multiprogramming with consequently increased averaging of peak memory demands and increased efficiency in memory utilization.

If the assumption of infinite workload contained within the throughput model is valid, very high values of CE can be obtained by specifying large values of N_p and **M**, and selecting N_j appropriately. CE can be made to approach unity if the system performs no preallocation of memory. In this section, however, attention will be restricted to systems with a relatively small number of processors, and the interactions between N_p, N_j, C, and CE for such systems will be analyzed.

A. Constant Memory Requirements

Whereas the motivation for multiprocessing arises primarily from the increased opportunities for averaging of peak memory demands afforded thereby, it follows that under conditions of constant memory requirements the motivation for multiprocessing would be rather weak. Nevertheless small improvements in CE are sometimes possible through multiprocessing.

Assume that the workload model is the same as that considered in the previous section, i.e., $\mathbf{m} = \mathbf{x} = \mathbf{y} = 1$. Assume further that $\mathbf{M} = N_j(\mathbf{m} + \mathbf{y}) + (N_p + 1)\mathbf{x}$. This latter assumption can be justified by arguing that an **m** is required for every active job, that a **y** is required for every active job except during waits for the processor (which will be minimized), that an **x** is required for every job being processed, and that an additional **x** should be preallocated to minimize waits by the processors.

In the previous section it was seen that for $N_p = 1$ the optimum value of N_j was $1 + t_o = 1 + y/r$ if that quantity was an integer. For example, for $r = 0.5$, $y = 1$, $t_o = 2$, the optimum N_j is 3. Under these circumstances the processor is fully utilized and the only inefficiency in memory utilization is that involved in the preallocation of an **x**. **CE** is 1 at $\mu = 0$, and decreases with increasing μ as the effect of the underutilized memory becomes more significant, reaching 0.917 at $\mu = 0.25$ and 0.875 at $\mu = \infty$.

For larger N_p the optimum N_j is $N_p(1 + t_o)$, provided that the result is an integer. For the case at hand, if N_p is increased from 1 to 2, the optimum N_j increases from 3 to 6. Again the processors are fully utilized, and the only inefficiency in memory utilization is that involved in the preallocation of a single **x**, which is now a smaller fraction of **M**. Thus **T** is doubled, but **M** and **C** are increased by ratios of somewhat less than 2. **CE** is 1 at $\mu = 0$, 0.947 at $\mu = 0.25$, and 0.933 at $\mu = \infty$.

Somewhat more motivation for multiprocessing can be found at values of **r** that yield noninteger values of t_o, especially when a small multiple of t_o is an integer. Then if N_p is set to the multiplier and N_j is set to $N_p(1 + t_o)$, perfect processor utilization and high memory utilization can be achieved.

Suppose that $r = 2/3$, then $t_o = 1.5$, and from the arguments of the previous section the optimum value of N_j for $N_p = 1$ must be either 2 or 3 depending on the value of μ.

For $N_j = 2$ the system will be idle part of the time and **CE** = 0.8 for all values of μ. For $N_j = 3$ there will be processor waits and unutilized allocated memory during these waits. **CE** is 1 at $\mu = 0$, 0.833 at $\mu = 0.25$, and 0.750 at $\mu = \infty$.

If N_p is increased to 2, the optimum N_j is 5. As shown in Fig. 34, perfect synchronism is achieved and $t_c = 1/2$. In this illustration A and B identify which processor is processing each segment, and the broken lines indicate the periods of output.

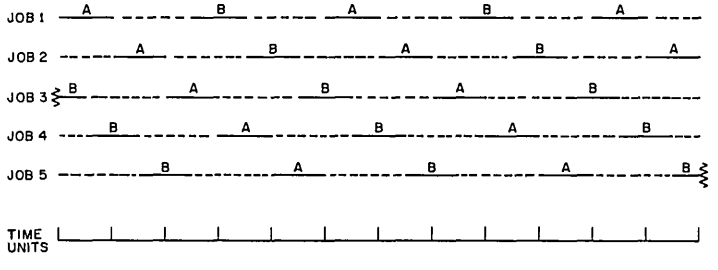


Fig. 34 – Synchronism of processing for rational t_o.

Under these circumstances there is perfect processor utilization and the only memory underutilization is that resulting from the preallocation of an **x**. **CE** is 1 at $\mu = 0$, 0.948 at $\mu = 0.25$, and 0.923 at $\mu = \infty$.

When $N_p(1 + t_o)$ is not an integer, the optimum value of N_j is one of the two adjacent integers, the lower value being preferable at high μ and the higher value being preferable at low μ.

B. Variable Memory Requirements

1. **Analytic Results.** Analytic results for multiprocessing systems were obtained only for the special case of $N_j = N_p = 2$. For workloads in which $t_o << \tau$, multiprogramming requires multiprocessing in order to attain improved memory utilization through peak averaging without suffering a loss in memory utilization through processor waits. For $N_j = N_p = 2$ only minor modifications need to be made to the development of Section IV, and the assumption made therein that $y_2 = M - 2(m + x)$ becomes even more reasonable.

For the termination strategy the development of Appendix E that led to equations (20) through (24) is applicable, except that there can no longer be any processor waits. Accordingly t_o rather than t_{o2} should be used in equation (25) for t_c. The only other modification required is the substitution of $N_p = 2$ into equation (4) for the system cost.

For the suspension strategy, the development of Appendix G must be modified. In addition to the elimination of processor waits, which requires the substitution of $t_{o\bar{s}}$ for $t_{o2\bar{s}}$ in equation (32), the duration of the suspensions must be reevaluated.

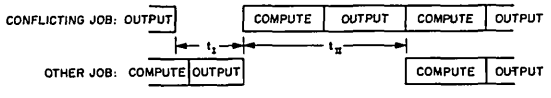


Fig. 35 – Worst-case suspension situation, multiprocessing.

Fig. 35 depicts the worst-case suspension situation. It is assumed that the memory request made by the conflicting job is very much larger than average and that the other job cannot have its next segment allocated until output of the conflicting segment is complete. Under these circumstances,

$$\begin{aligned} t_{o2s} &= t_{os} + t_I + t_{II} \\ &= t_{os} + 1/2\,(1 + t_o) + 1 + t_{os} \\ &= 1.5 + 1/2\, t_o + 2\, t_{os}. \end{aligned} \tag{48}$$

Figure 36 depicts better suspension situations that might occur when the conflicting memory request is not extremely large or when the next request of the other job is smaller than average. In Fig. 36a it is assumed that under these circumstances the other job can resume as soon as processing of the conflicting segment is complete, and in Fig. 36b

(a) Other job can resume on completion of processing of the conflicting segment.

(b) Other job can be allocated in parallel with the conflicting segment.

Fig. 36 – Better suspension situation, multiprocessing

it is assumed that the other job can be allocated in parallel with the conflicting segment. The former situation is more likely for large **x**, the latter for small **x**.

Then, for the situation of Fig. 36a

$$\begin{aligned} t_{o2s} &= t_{os} + t_I + t_{III} \\ &= t_{os} + 1/2\,(1 + t_o) + 1 \\ &= 1.5 + 1/2\, t_o + t_{os}, \end{aligned} \tag{49a}$$

and for the situation of Fig. 36b

$$\begin{aligned} t_{o2s} &= t_{os} + t_I \\ &= 0.5 + 1/2\, t_o + t_{os}. \end{aligned} \tag{49b}$$

Thus, in modifying the development of Appendix G to cover the case of $N_p = 2$, equation (48) instead of equation (42) should be used in the determination of the upper limit on the quantity to be multiplied by P_s in equation (32), and equation (49a) for large **x** and equation (49b) for small **x** should be used instead of equations (43a) or (43b) for the lower limit.

Appendix J contains comparisons between calculated and simulated values of the quantities that determine CE. For the termination strategy, agreement was excellent at the higher values of **r** that suggest the use of an N_p equal to N_j. For the suspension strategy, the relative width of the limits on $1 + t'_{o2}$, and hence on t_c and CE, was about 6% in the region of M_{opt}. In this region the simulated values fell about midway between the calculated limits.

2. **Numerical Results and Discussion.** In Section IV the multiprogramming of two jobs with $\alpha = 1$ and $r = 10$ was considered in connection with Table 2 and Fig. 15. Since t_o was approximately $0.1\,\tau$, there was considerable waiting for the processor; the inefficiency in memory utilization involved in allocated jobs waiting for the processor overwhelmed the gains in peak demand averaging, and the performance was inferior to that of the monoprogrammed system.

In Fig. 37 the effect of the addition of a second processor is shown. Processor waits are eliminated, and since the processors are each utilized about 90% of the time, the advantage of averaging peak memory demands is fully available.

In Fig. 38 the process has been carried further by setting $N_p = 9$ and $N_j = 10$. Further gains in **CE** have been attained by permitting still more averaging of peak memory demands.

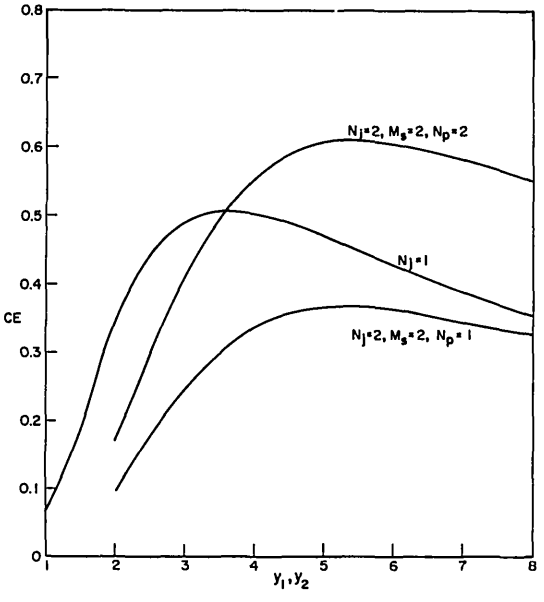


Fig. 37 – CE vs y_1 or y_2; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = \mathbf{n} = 10$.

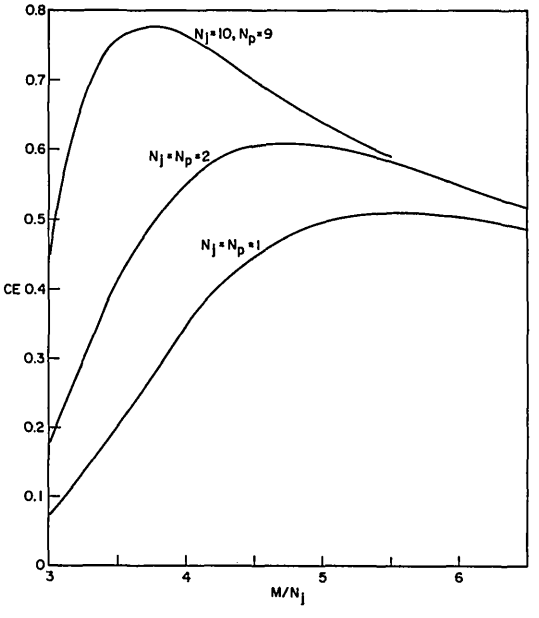


Fig. 38 – CE vs M/N_j; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{r} = \mathbf{n} = 10$.

Figure 38 illustrates the effect noted earlier that if one's budget were truly infinite and if an infinite workload were truly available, very high values of **CE** could be obtained with an infinite system. With such a system a ratio of N_j to N_p of $1 + t_o$ would yield perfect processor utilization, and a ratio of **M** to $(\mathbf{m} + \mathbf{x} + \langle \underline{y} \rangle)$ of N_j would yield perfect memory utilization. Of course the cost of such a system would be infinite.

It is possible to avoid this absurdity by focusing attention on how **CE** varies with **C**. This will make it possible to illustrate how some realistic problems may be attacked through the use of the throughput model. The problems considered are:

1. Given a specific budget, **C**, determine the optimum hardware mix (i.e. N_p and **M**) and N_j that optimizes **CE** and therefore **T**;

2. Given a specific system, and a specific expansion budget, Δ**C**, determine the system changes that optimize **CE** and therefore **T**;

3. Given a specific required **T**, determine the system of maximum **CE**, and therefore minimum **C**, that can supply the required **T**;

4. Given additional costs involved in increasing the workload, find the system that maximizes **CE**.

The first two problems may be attacked by considering the **CE** vs **C** relations for various N_p and N_j. The only assumption that need be made is that there is a potential workload larger than the workload that can be processed with any system that can be procured with the available funding; then the assumption of an infinite workload made by the throughput model is justifiable.

The third problem can be approached by considering the **T** vs **C** relations for various N_p and N_j. No exact solution is possible under the throughput model, which assumes an infinite workload. Although the model can predict that a system will produce a throughput **T** when offered an infinite workload, it does not automatically follow that the same system will process all elements of a finite workload corresponding to the same **T** without introducing very long solution delays. Thus the exact solution of the third problem requires consideration of queuing models, which, it was stated at the outset, lie beyond the scope of the present investigation. Nevertheless the approximate solution to this problem afforded by the throughput model may provide a useful lower limit on the scale of the system required to produce a given **T**.

The fourth problem involves the consideration of such factors as data transmission costs, the analysis of which also lies beyond the scope of the present investigation. Nevertheless one brief example of this type of problem will be given.

The workload model that was investigated in some detail was one in which $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, and $\mathbf{n} = 10$. The systems that were considered had $\mathbf{r} = 0.5$ and three values of μ: 1, 0.25, and 0.1.

A value of $\mu = 1$ leads to systems in which the bulk of the cost results from providing the memory. This value of μ may be unrealistically large at present, but current trends in batch fabrication of digital logic suggest that the cost of processing elements is declining more rapidly than the cost of memory. If the unit of memory is thought to be a bank or a small module (perhaps 8192 or 16384 words), the μ range of 0.1 to 0.25 is representative of systems delivered in the early 1960's and the μ range of 0.25 to 1 is representative of systems announced in the middle 1960's. In both cases the lower end of the range is more representative of sales prices, and the upper end of the range is more representative of production costs.

In all of the simulations the suspension strategy was used, y_t was set to $(M - N_j)$, and M_s was set to 1. For large values of M and N_j this led to the instabilities discussed in the previous section, although they were usually not severe. For values of N_j up to 5, the estimated standard deviation of the means of simulations made with the same parameters was usually less than 0.005 absolute. For values of N_j above 5, equally consistent results were obtained for M of the order of M_{opt} or above, but standard deviations of 0.01, and occasionally more, were obtained for values of M well below M_{opt}.

It was seen earlier in this section that under conditions of constant memory requirements the optimum ratio of N_j to N_p is $1 + t_o$. In the case at hand, t_o is very nearly 2, which suggests an optimum ratio of 3. However, Figs. 39 and 40, for $N_p = 1$ and $N_p = 2$, respectively, demonstrate that under conditions of variable memory requirements the optimum ratio is sensitive to μ.

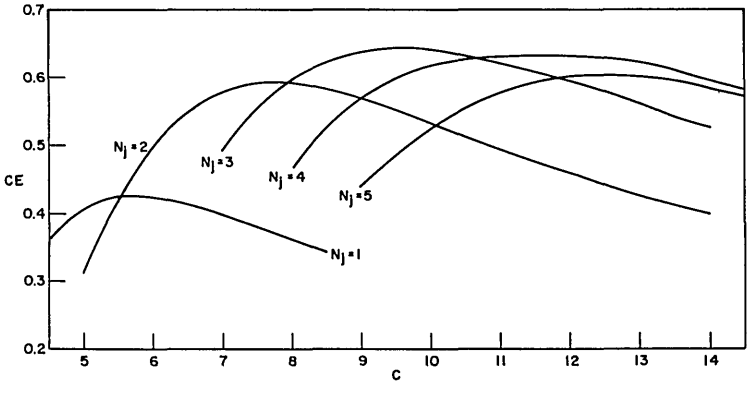


Fig. 39a – CE vs C for $\mu = 1$; $m = 1$, $x = 0$, $a = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$, $N_p = 1$.

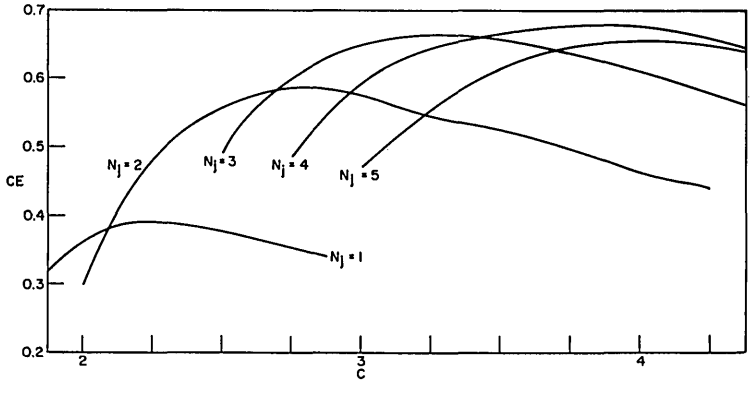


Fig. 39b – CE vs C for $\mu = 0.25$; $m = 1$, $x = 0$, $a = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$, $N_p = 1$.

For $N_p = 1$ and $\mu = 1$ (Fig. 39a), CE is largely determined by efficiency in memory utilization. Although maximum CE is obtained with $N_j = 3$, nearly equal values of CE are obtainable with $N_j = 4$ at higher C levels. As N_j is raised from 3 to 4, the inefficiency involved in the allocation of memory to problems for which the processor is unavailable is nearly counterbalanced by the increased averaging of peak memory demands and higher processor utilization.

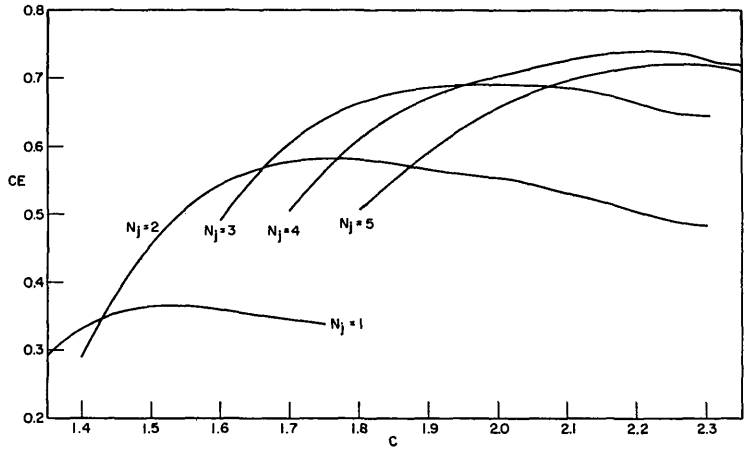


Fig. 39c – CE vs C for μ = 0.1; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $a = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 0$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$, $\mathbf{N_p} = 1$.

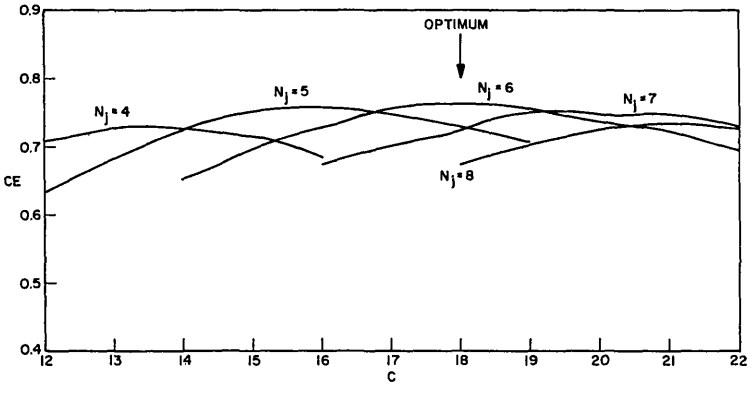


Fig. 40a – CE vs C for $\mu = 1$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $a = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$, $\mathbf{N_p} = 2$.

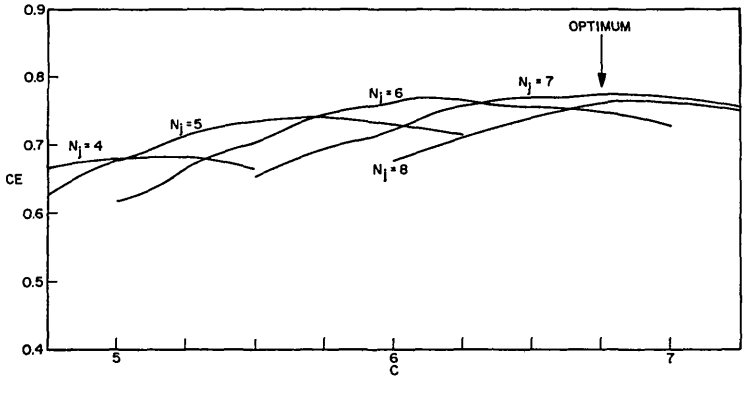


Fig. 40b – CE vs C for μ = 0.25; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $a = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$, $\mathbf{N_p} = 2$.

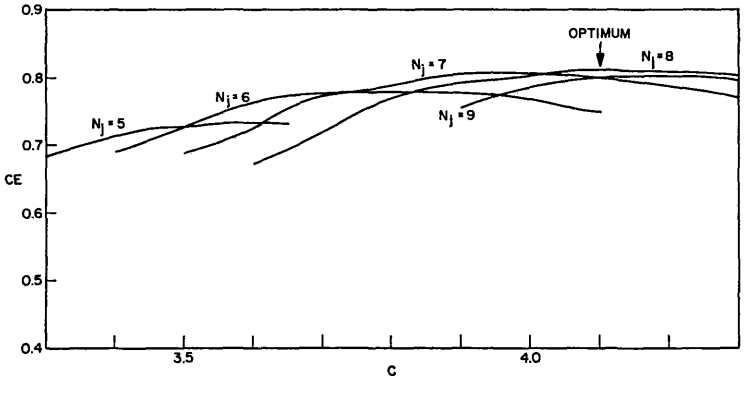


Fig. 40c – CE vs C for μ = 0.1; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $a = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$, $\mathbf{N_p} = 2$.

For $\mu = 0.25$ (Fig. 39b), somewhat higher values of CE are obtainable with $N_j = 4$, than with $N_j = 3$. At this lower value of μ efficiency in processor utilization becomes more significant. At $N_j = 3$ the processor is not in use all the time, because the variability in $\underline{y}$ leads to variability in output time.

For $\mu = 0.1$ (Fig. 39c), significantly higher values of CE are obtainable with $N_j = 4$ than with $N_j = 3$. At this very low value of μ, questions of memory utilization become relatively unimportant, and the motivation to minimize idle processor time becomes paramount. Processor utilization at $N_j = 4$ is near perfect, however, so the additional very small gain in processor utilization available by increasing N_j to 5 is insufficient to compensate for the accompanying decrease in memory utilization.

For $N_p = 2$, inspection of Fig. 40 reveals that the optimum values of N_j are 6 for $\mu = 1$, 7 for $\mu = 0.25$, and 8 for $\mu = 0.1$.

Next to be considered is the question of the optimum C level at which to increase N_p. Figure 41 displays the CE vs C curves in the crossover regions between optimum operation with $N_p = 1$ and $N_p = 2$, and Fig. 42 displays the CE vs C curves for the optimum N_p, N_j combinations over a wider range of C. Again the form of the curves is rather sensitive to μ.

At $\mu = 1$ and $N_p = 1$ (Fig. 42a), CE increases with C up to C = 9.5, $N_j = 3$. As C increases from 9.5 to 9.7, there is a very slight decrease in CE, and as C exceeds 9.7 the introduction of the second processor is appropriate. In the C range 9.7 to 18.7 the general trend of the CE vs C curve is upward, although there are some small local declines in the N_j crossover regions. Maximum CE for $N_p = 2$ is attained at about C = 18 or 18.5, and the third processor should be introduced at C = 18.7. Above this value of C the general trend of the curve is upward, with very small local declines, and the fourth processor should be introduced at about C = 26.4.

At $\mu = 0.25$ and $N_p = 1$, CE increases with C up to C = 3.87, $N_j = 4$. Above this point CE declines as additional memory is being underutilized and an additional processor cannot be procured for a small increment in C without decreasing M and underutilizing the processors. The second processor should be added at about C = 4.48, but the CE available at C = 3.87 is not restored until C reaches nearly 5, i.e. until the original memory is restored. The general trend is upward until maximum CE for $N_p = 2$ is reached at C = 6.75, $N_j = 7$. The trend is now downward until the third processor is introduced at about C = 7.7.

At $\mu = 0.1$ and $N_p = 1$, CE increases with C up to C = 2.25, $N_j = 4$. Above this point CE declines significantly, as additional memory is being underutilized and an additional processor cannot be procured for a small increment in C without substantially decreasing

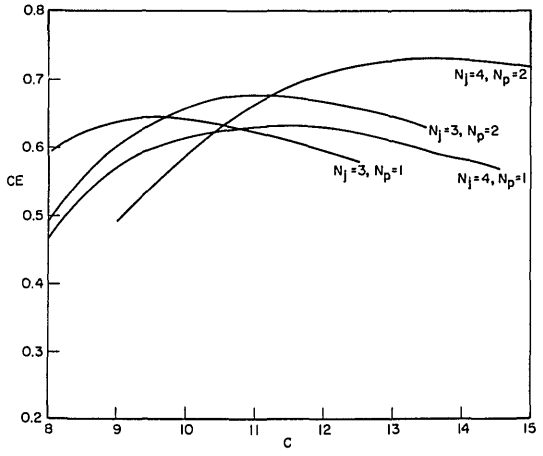


Fig. 41a – CE vs C for $\mu = 1$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$.

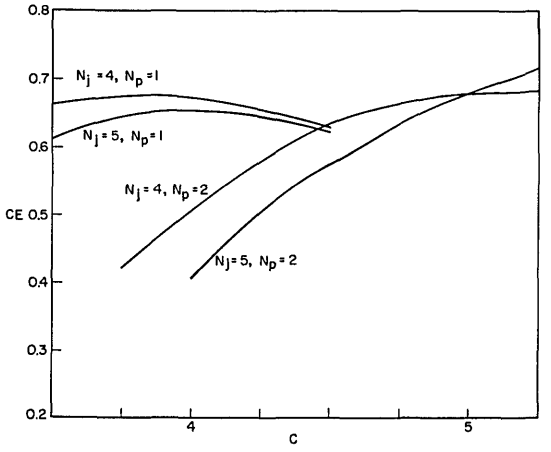


Fig. 41b – CE vs C for $\mu = 0.25$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$.

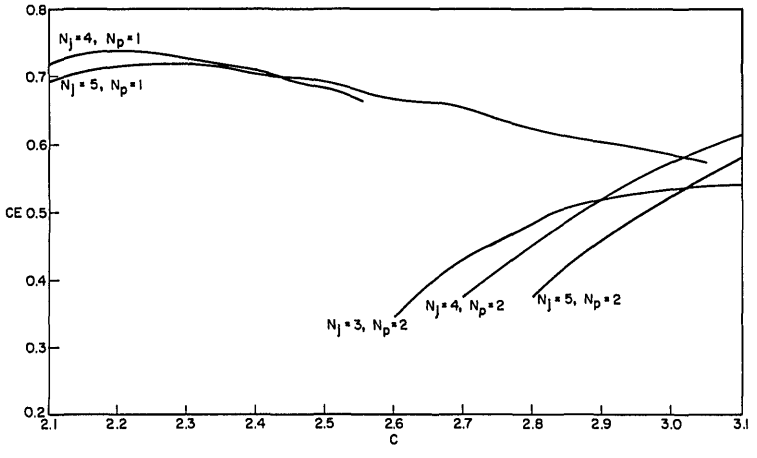


Fig. 41c – CE vs C for $\mu = 0.1$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$.

M and severely underutilizing the processors. The second processor should be added at **C** = 3.02, but the **CE** available at **C** = 2.25 is not restored until **C** reaches about 3.53, i.e., until considerable additional memory to keep the additional processor busy can be procured. Maximum **CE** for $\mathbf{N_p} = 2$ is reached at about **C** = 4.1, and above this value **CE** declines until the third processor is added at **C** = 4.86; the value of **CE** available at **C** = 4.1 being attained again at **C** = 5.35.

The four practical problems enumerated earlier may now be attacked through the use of these figures and the tables in Appendix J on which the figures are based.

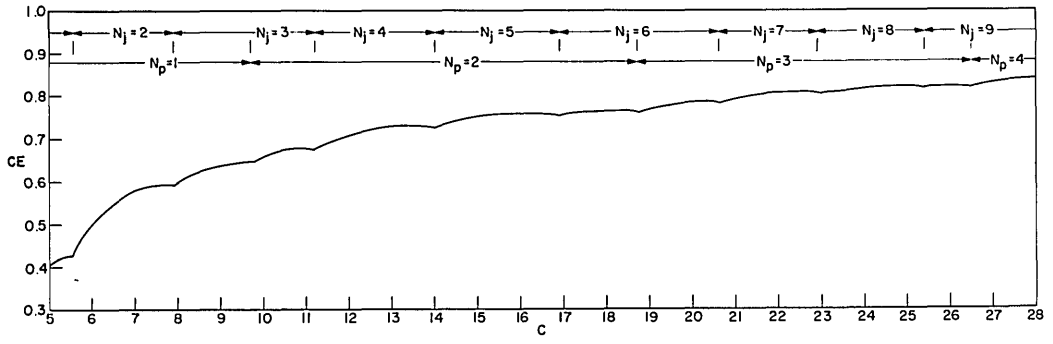


Fig. 42a—CE vs C for $\mu = 1$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$.

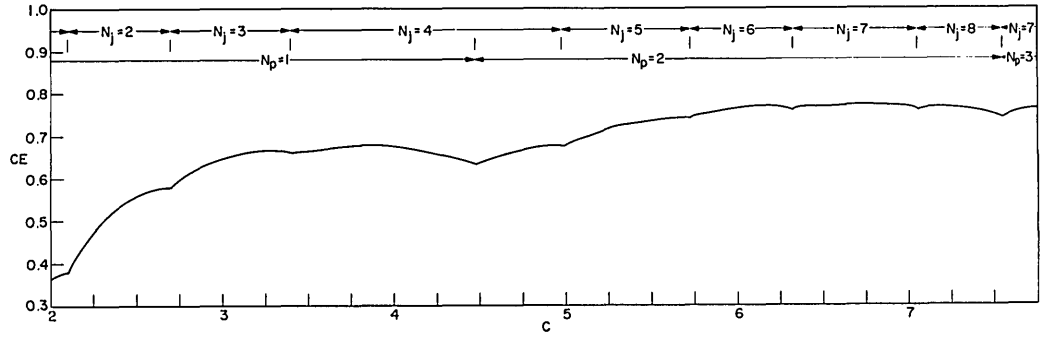
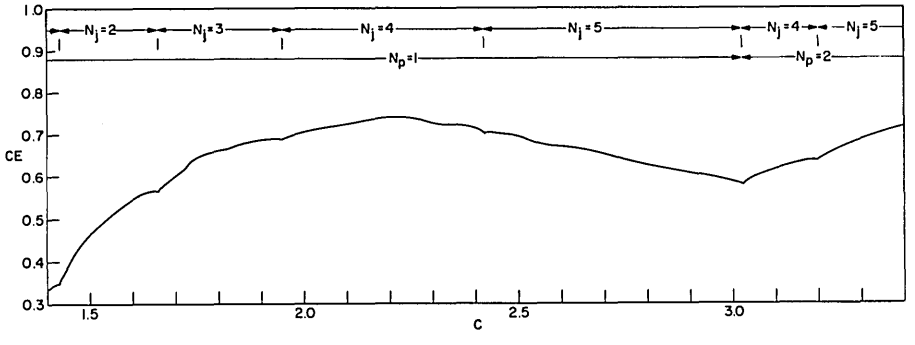


Fig. 42b—CE vs C for $\mu = 0.25$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$.

Part 1



Part 2

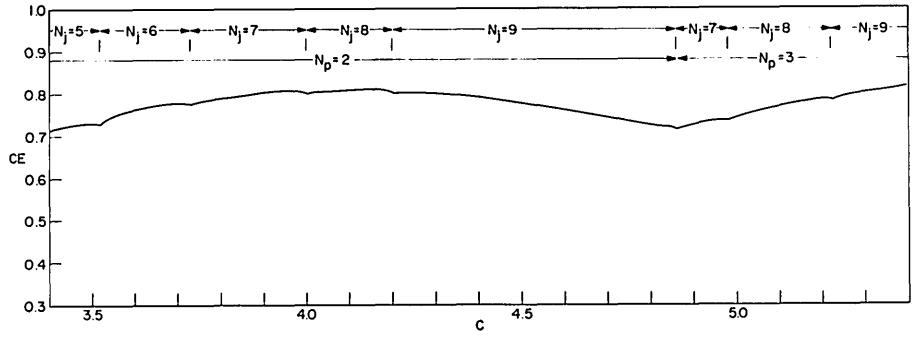


Fig. 42c—CE vs C for $\mu = 0.1$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$.

The first problem was the selection of the optimum hardware mix and N_j for a given **C**. The optimum mix can be found by direct inspection of Fig. 42 or of the bold-faced values in Table J4.

Two reservations must be offered at this point: the discussion has assumed that **M** is a continuous variable and that the hardware budget is fixed. In practice one must procure memory in integer number of modules, and the decision as to what to procure may influenced by the possibility of future availability of additional funding, especially when the system is being purchased rather than leased and components cannot be returned.

Although the data plotted in Figs. 39 through 42 and displayed in Table J3 was simulated at memory intervals of 0.5 or finer, the abcissa ticks on the figures and the entries of Table J4 correspond to integer multiples of the unit of memory. The question of preparation for future expansion will be deferred until after discussion of the second problem, that of the optimum investment of an expansion budget, Δ**C**, which is to be used in upgrading a system which is already optimal at some cost level, **C**.

This problem can be attacked by inspection of Fig. 42, or by reference to Table 8, which displays the optimal system changes and their effect on **T** and **CE** for three values of Δ**C** and various initial **C**. The assumption was made in the preparation of Table 8 that it was possible to reduce **M** to help pay for an additional processor.

The most fortuitous expansion situation is where the existing system is at the minimum **M** and **C** level for a given N_p. In this situation the expansion budget should be invested in additional memory and N_j increased to utilize this memory. Usually **T** will be greatly increased, and **CE** will be improved.

The most unfortunate expansion situation, especially at low μ, is where the system is already at its optimum **M** for its N_p and Δ**C** is small. In this case, although any increase in **C** will increase **T**, the relative increase in **T** may be less than that in **C**, and **CE** will therefore decline. The minimum Δ**C** that is required before an additional processor should be procured is about 0.2 for $\mu = 1$, is 0.6 or 0.9 (for the second and third processors, respectively) for $\mu = 0.25$, and is about 0.8 for $\mu = 0.1$. The minimum Δ**C** required to maintain **CE**, however, is about 0.2 for $\mu = 1$, slightly over 1 for $\mu = 0.25$, and about 1.25 for $\mu = 0.1$.

It should now become apparent that particularly at low μ it may be wise to purchase a suboptimal initial system if a later expansion is anticipated and the option of returning memory is not available, in order that the final system can be optimal. This will generally involve procuring more processors and less memory for the initial system than would be done if the future expansion were unlikely.

Table 8a – Effect of Finite Expansion Budget on T and CE for $\mu = 1$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$.

C	$\Delta C = 0.5$						$\Delta C = 1$						$\Delta C = 1.5$					
	ΔN_p	ΔN_j	ΔM	$\Delta C(\%)$	$\Delta T(\%)$	$\Delta CE(\%)$	ΔN_p	ΔN_j	ΔM	$\Delta C(\%)$	$\Delta T(\%)$	$\Delta CE(\%)$	ΔN_p	ΔN_j	ΔM	$\Delta C(\%)$	$\Delta T(\%)$	$\Delta CE(\%)$
5	0	0	0.5	10.0	16.1	5.5	0	1	1	20.0	48.6	23.8	0	1	1.5	30.0	74.2	34.0
6	0	0	0.5	8.3	17.2	8.2	0	0	1	16.7	34.6	15.3	0	0	1.5	25.0	46.1	16.9
7	0	0	0.5	7.1	8.5	1.3	0	1	1	14.3	17.5	2.8	0	1	1.5	21.4	30.0	7.0
8	0	0	0.5	6.3	10.8	4.2	0	0	1	12.5	20.0	6.7	0	0	1.5	18.8	27.9	7.7
9	0	0	0.5	5.6	6.6	0.9	1	0	0	11.1	11.4	3.1	1	0	0.5	16.7	23.2	5.6
10	0	0	0.5	5.0	7.4	2.3	0	0	1	10.0	13.0	2.7	0	1	1.5	15.0	18.7	3.2
11	0	1	0.5	4.5	6.4	1.8	0	1	1	9.1	14.4	4.9	0	1	1.5	13.6	20.9	6.4
12	0	0	0.5	4.2	5.6	1.3	0	0	1	8.3	11.3	2.8	0	0	1.5	12.5	16.0	3.1
13	0	0	0.5	3.8	4.1	0.3	0	0	1	7.7	7.3	−0.4	0	1	1.5	11.5	13.1	1.4
14	0	1	0.5	3.6	5.5	1.8	0	1	1	7.1	11.0	3.6	0	0	1.5	10.7	15.2	4.1
15	0	0	0.5	3.3	3.9	0.6	0	0	1	6.7	7.6	0.8	0	0	1.5	10.0	10.4	0.4
16	0	1	0.5	3.1	2.8	−0.3	0	1	1	6.3	6.2	−0.1	0	1	1.5	9.4	9.8	0.4
17	0	0	0.5	2.9	3.5	0.6	0	0	1	5.9	6.7	0.8	0	0	1.5	8.8	9.9	1.0
18	0	0	0.5	2.8	4.2	1.4	1	0	0	5.5	6.2	0.7	1	0	0.5	8.3	10.1	1.7
19	0	0	0.5	2.6	3.6	1.0	0	0	1	5.3	7.4	2.0	0	0	1.5	7.9	10.2	2.1
20	0	0	0.5	2.5	2.6	0.1	0	1	0	5.0	5.9	0.9	0	1	1.5	7.5	10.4	2.7
21	0	0	0.5	2.4	5.3	1.8	0	0	1	4.8	6.9	2.0	0	0	1.5	7.1	9.6	2.3
22	0	0	0.5	2.3	2.5	0.2	0	1	1	4.5	4.6	0.1	0	1	1.5	6.8	6.9	0.1
23	0	0	0.5	2.2	2.2	0	0	0	1	4.3	5.6	1.2	0	0	1.5	6.5	7.6	1.0
24	0	0	0.5	2.1	2.1	0	0	0	1	4.2	4.2	0	0	1	1.5	6.3	6.1	−0.2
25	0	1	0.5	2.0	1.8	−0.2	0	1	1	4.0	3.9	−0.1	1	1	0.5	6.0	6.1	0.1
26	1	0	−0.5	1.9	2.1	0.2	1	0	0	3.8	5.3	1.4	1	0	0.5	5.8	7.0	1.1
27	0	0	0.5	1.8	2.5	0.7	0	0	1	3.7	4.7	1.0						
28	0	0	0.5	1.8	2.9	0.1	0	0	1	3.6	3.5	−0.1						

Table 8b – Effect of Finite Expansion Budget on T and CE for $\mu = 0.25$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$.

C	$\Delta C = 0.5$						$\Delta C = 1$						$\Delta C = 1.5$					
	ΔN_p	ΔN_j	ΔM	$\Delta C(\%)$	$\Delta T(\%)$	$\Delta CE(\%)$	ΔN_p	ΔN_j	ΔM	$\Delta C(\%)$	$\Delta T(\%)$	$\Delta CE(\%)$	ΔN_p	ΔN_j	ΔM	$\Delta C(\%)$	$\Delta T(\%)$	$\Delta CE(\%)$
2	0	1	2	25.0	93.1	54.5	0	2	4	50.0	16.6	77.3	0	3	6	75.0	218	81.8
2.25	0	1	2	22.2	51.5	24.0	0	1	4	44.4	98.7	37.6	0	2	6	66.7	133	39.8
2.5	0	1	2	20.0	38.0	15.0	0	2	4	40.0	65.2	18.0	0	2	6	60.0	92.0	20.0
2.75	0	0	2	18.2	32.1	11.8	0	1	4	36.4	53.7	12.7	0	1	6	54.6	69.1	9.4
3	0	1	2	16.7	19.9	2.7	0	1	4	33.3	38.9	4.2	1	1	2	50.0	47.8	−1.5
3.25	0	1	2	15.4	17.2	1.6	0	1	4	30.7	28.9	−1.4	1	1	2	46.2	46.9	0.5
3.5	0	0	2	14.3	16.0	1.5	1	0	0	28.6	23.3	−4.1	1	1	2	42.9	46.8	2.7
3.75	0	0	2	13.3	9.9	−3.0	1	0	0	26.7	27.8	0.9	1	1	2	40.0	49.0	6.4
4	1	0	−2	12.5	6.3	−5.5	1	1	0	25.0	26.5	1.2	1	1	2	37.5	46.9	9.2
4.25	1	0	−2	11.8	13.8	1.8	1	1	0	23.5	35.0	9.3	1	2	2	35.3	53.3	13.3
4.5	0	1	2	11.1	19.0	7.1	0	1	4	22.1	41.1	15.6	0	2	6	33.3	60.2	20.2
4.75	0	1	2	10.5	18.7	7.4	0	2	4	21.1	38.5	14.4	0	2	6	31.6	50.9	14.7
5	0	0	2	10.8	18.7	7.9	0	1	4	20.0	34.5	12.1	0	2	6	30.0	46.9	13.0
5.25	0	1	2	9.5	14.1	4.2	0	1	4	19.0	27.2	6.9	0	2	6	28.6	39.0	8.1
5.5	0	1	2	9.1	13.2	3.8	0	2	4	18.2	23.6	4.6	0	2	6	27.3	32.9	4.4
5.75	0	0	2	8.7	13.3	4.2	0	1	4	17.4	21.9	3.8	0	2	6	26.1	29.3	2.6
6	0	1	2	8.3	9.2	0.8	0	1	4	16.7	17.3	0.5	0	2	6	25.0	22.9	−1.7
6.25	0	1	2	8.0	9.3	1.2	0	2	4	16.0	16.0	0	1	1	2	24.0	23.9	−0.1
6.5	0	0	2	7.7	7.4	−0.3	0	1	4	15.4	12.5	−2.5	1	1	2	23.1	24.2	0.9
6.75	0	1	2	7.4	6.1	−1.2	1	0	0	14.8	13.3	−1.3	1	1	2	22.2	25.3	2.5
7	0	1	2	7.1	5.0	−2.0	1	1	0	14.3	15.4	1.0	1	1	2	21.4	27.0	4.6
7.25	1	−1	−2	6.9	6.8	−0.1	1	0	0	13.8	18.0	3.7	1	1	2	20.7	29.0	6.9
7.5	1	0	−2	6.7	10.2	3.3	1	0	0	13.8	21.2	7.0	1	1	2	20.0	30.7	8.9
7.75	0	0	2	6.5	10.5	3.8	0	2	4	12.9	19.9	6.2	0	2	6	19.4	27.0	6.4
8	0	1	2	6.3	10.0	3.5	0	1	4	12.5	18.5	5.3						
8.25	0	1	2	6.1	9.2	2.9	0	1	4	12.1	15.4	2.9						
8.5	0	1	2	5.9	7.7	1.7												

Table 8c – Effect of Finite Expansion Budget on **T** and **CE** for $\mu = 0.1$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $M_s = 1$, $y_t = M - N_j$.

C	ΔC = 0.5						ΔC = 1						ΔC = 1.5					
	ΔN_p	ΔN_j	ΔM	ΔC(%)	ΔT(%)	ΔCE(%)	ΔN_p	ΔN_j	ΔM	ΔC(%)	ΔT(%)	ΔCE(%)	ΔN_p	ΔN_j	ΔM	ΔC(%)	ΔT(%)	ΔCE(%)
1.4	0	2	5	35.7	178	105	0	3	10	71.4	282	123	0	4	15	107	276	81.8
1.5	0	2	5	33.3	102	51.7	0	3	10	66.7	149	49.5	0	3	15	100	153	26.3
1.6	0	2	5	31.2	61.1	32.4	0	3	10	62.5	99.4	22.7	1	2	5	93.8	118	12.6
1.7	0	2	5	29.3	58.8	22.8	0	2	10	58.8	72.0	8.3	1	2	5	88.3	101	6.5
1.8	0	2	5	27.7	40.5	10.0	0	2	10	55.6	48.0	−4.9	1	2	5	83.3	90.8	4.1
1.9	0	2	5	26.3	31.6	4.2	0	2	10	52.7	35.3	−11.4	1	2	5	79.0	87.1	4.5
2.0	0	1	5	25.0	23.4	−1.3	0	1	10	50.0	25.0	−16.7	1	1	5	75.0	82.0	4.0
2.1	0	1	5	23.8	14.8	−7.3	1	0	0	47.7	25.5	−15.0	1	2	5	71.4	80.7	5.4
2.2	0	1	5	22.7	8.2	−11.8	1	1	0	45.5	26.1	−13.3	1	2	5	68.3	77.1	5.2
2.3	0	1	5	21.7	5.1	−13.6	1	1	0	43.5	35.8	−5.4	1	3	5	65.3	80.2	9.0
2.4	0	1	5	20.8	2.7	−15.0	1	1	0	41.7	41.3	0.3	1	3	5	62.5	83.3	12.8
2.5	0	0	5	20.0	1.3	−15.6	1	0	0	40.0	47.4	5.3	1	2	5	60.0	85.4	15.9
2.6	1	−1	−5	19.2	8.6	−8.9	1	1	0	38.5	58.7	14.6	1	3	5	57.7	90.3	20.7
2.7	1	0	−5	18.5	16.5	−1.7	1	1	0	37.1	63.6	19.3	1	3	5	55.6	92.2	23.5
2.8	1	0	−5	17.8	28.9	9.4	1	2	0	35.7	71.1	26.1	1	4	5	53.6	96.1	27.7
2.9	1	0	−5	17.2	38.3	18.0	1	2	0	34.5	67.6	24.6	1	4	5	51.8	98.3	30.6
3.0	1	0	−5	16.7	45.6	24.8	1	2	0	33.3	83.0	37.3	1	4	5	50.0	98.8	32.5
3.1	0	2	5	16.1	43.8	23.9	0	4	10	32.3	74.1	31.6	0	5	15	48.4	82.8	23.2
3.2	0	1	5	15.6	40.2	21.3	0	3	10	31.3	65.0	25.7	0	4	15	46.9	70.7	16.2
3.3	0	2	5	15.1	32.7	15.3	0	4	10	30.3	52.1	16.7	0	4	15	45.5	54.4	6.1
3.4	0	2	5	14.7	29.0	12.5	0	4	10	29.4	43.2	10.7	1	2	5	44.2	46.1	1.3
3.5	0	2	5	14.3	25.7	10.0	0	4	10	28.6	36.6	6.2	1	3	5	42.9	45.6	1.9
3.6	0	2	5	13.9	21.0	6.2	0	3	10	27.8	27.2	−0.5	1	2	5	41.7	43.0	0.9
3.7	0	2	5	13.5	17.6	3.6	0	3	10	27.1	22.1	−3.9	1	2	5	40.6	42.7	1.5
3.8	0	2	5	13.2	14.7	1.3	0	2	10	26.4	16.4	−7.9	1	2	5	39.5	42.0	1.8
3.9	0	2	5	12.8	11.0	−1.6	1	0	0	25.7	13.1	−10.0	1	2	5	38.5	40.6	1.5
4.0	0	2	5	12.5	10.9	−3.2	1	1	0	25.0	15.9	−7.3	1	2	5	37.5	40.9	2.5
4.1	0	1	5	12.3	5.3	−6.2	1	0	0	24.4	18.3	−4.9						
4.2	0	1	5	11.9	3.4	−7.5	1	0	0	23.8	21.3	−2.0						
4.3	0	0	5	11.6	1.6	−9.0	1	0	0	23.3	22.7	0.5						
4.4	1	−2	−5	11.3	1.1	−9.2												
4.5	1	−1	−5	11.1	6.4	−4.2												
4.6	1	−1	−5	10.9	12.6	1.5												

For example, suppose that $\mu = 0.1$ and that an initial budget of 3 is available. The optimal system at this **C** is one with $\mathbf{N_p} = 1$, **M** = 20, and **CE** = 0.585. Suppose further, however, that it is likely that an additional budget of 0.5 will become available but very unlikely that an additional budget of 1 can be obtained. If 0.5 unit of funding are to be used in expanding the system, and if memory cannot be returned for credit, the only way these funds can be applied is in the increase of **M** from 20 to 25. This increase in **M** would hardly increase **T**, however, because the system is processor limited, and **CE** would decline to 0.470. On the other hand, if the system initially procured was one with $\mathbf{N_p} = 2$ and **M** = 10, its **CE** would be 0.575, which is only slightly less than 0.585. The expansion budget of 0.5 could then be used in increasing **M** from 10 to 15, and **CE** would then be increased to 0.730.

The third type of problem concerned the selection of a system to produce a given **T**. Figure 43 graphs **T** vs **C** for the workload and models considered in Figs. 39 through 42.

Each of the three solid lines on Fig. 43 trace the **T** vs **C** relationship for the optimum $\mathbf{N_p}$ and $\mathbf{N_j}$ combination for that **C** and μ. On this log-log plot, points with the same

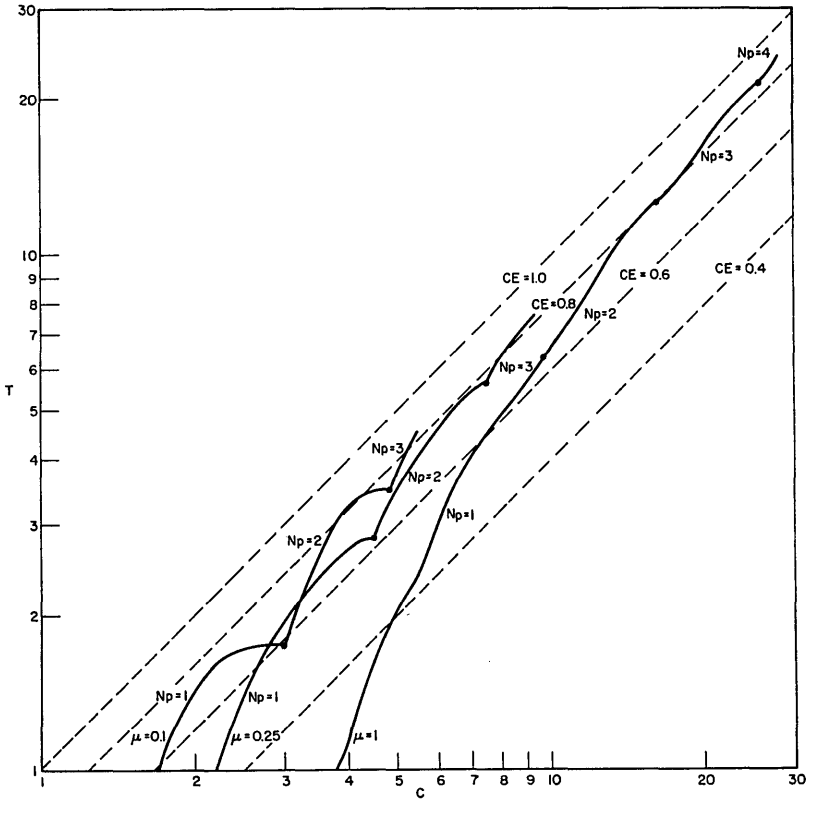


Fig. 43 – **T** vs **C**; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $a = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$.

CE = T/C are connected by straight lines of unit slope. The four broken lines on Fig. 43 are the loci of points in the T, C plane corresponding to CE values of 1, 0.8, 0.6 and 0.4.

The effect of the necessary adjustments in N_j as C is varied is hardly visible in Fig. 43, although careful inspection of the trace for $\mu = 1$ will reveal reversals in the sign of the second derivative at the values of C which are optimum for a given N_j and at the values of C where the optimum N_j changes.

The behavior at the N_p crossovers is clearly visible for the lower values of μ, however. The regions of reduced slope correspond to systems whose throughput is processor limited.

Considerable caution must be applied in attacking problems of the third type through the use of the type of relations depicted in Fig. 43. The throughput model assumes that an infinite workload is available and that the system need not be able to complete every type of problem represented in the workload. On the other hand, if a finite workload is specified, it is likely that there is an implied requirement that every job in the workload be processed. It may be necessary, for example, to procure a system with a very large memory which will be poorly utilized on the average in order to be able to process problems with very large memory requirements which constitute a small proportion of the workload.

The specification of a finite workload is often accompanied by the specification of a maximum mean or worst-case processing delay. The throughput model is unable to shed any light on these matters. It is possible for example, that if low delays are specified, or if the jobs tend to arrive in bunches, it may be necessary to provide more processors in the system than would be predicted by throughput considerations.

Nevertheless relations of the type summarized in Fig. 43 do provide a lower bound on the quantity of equipment required to provide a given T. This bound is about all that throughput analyses can contribute to the solution of the third type of problem; complete solution would require the specification of finite limits on statistically distributed parameters, maximum mean and worst-case delays, and job arrival rate statistics. The analysis of the factors lies beyond the scope of the present investigation.

Solution of the fourth type of problem, accounting for extra costs involved in increasing the workload, also lies beyond the scope of the present investigation, but a problem of this type will be sketched.

Suppose that a very large budget is available to build a computer center and that a determination of the optimum size center is desired. Assume that a given workload is available at the center, and that equal workloads are available at a large number of other sites which are evenly distributed over geographic area. There will be additional data

transmission costs involved in processing the workloads of the remote sites, and it will be assumed that these costs are proportional to the distance between the sites and the center.

This is an extension of the third type of problem, in that what is required is the solution of a number of subproblems, each subproblem involving a specified workload and data transmission cost. The use of throughput analysis requires a number of additional assumptions.

First, it is necessary to assume that service delays and the possibility of abandoning specific jobs may be ignored. This is the same assumption that was required in order to attack the third type of problem. Next, it is necessary to ignore job arrival rate statistics and deal only with average throughput. If this is acceptable, then it is reasonable to characterize the throughput requirement of each site as corresponding to the continuous multiprogramming of some number, λ, of jobs. Finally, it is necessary to make some assumption as to how the data transmission costs accrue.

A reasonable assumption might be as follows: there are no data transmission costs involved in processing the local workload of λ jobs, but there is a cost involved in adding each remote site. The cost for each site is assumed to depend only on its distance from the center and to be independent of how many jobs (up to the limit of λ jobs per $\mathbf{n}(1 + \mathbf{t_o})$ units of time) originating at this site are actually processed. It is then reasonable to approximate the cost of connecting the ith site as $\mathbf{k}\sqrt{\mathbf{i} - 1}$, where the square root corresponds to the assumption of a site distribution uniform over area and data transmission costs linearly proportional to distance. Thus if $\mathbf{I}$ sites are connected, the multiprogramming of up to $\mathbf{I}\lambda$ jobs is possible and the data transmission costs are

$$\mathbf{k} \sum_{\mathbf{i}=1}^{\mathbf{I}} \sqrt{\mathbf{i} - 1}.$$

Figure 44 graphs the **CE** vs **C** curves for the model of Fig. 42a with the costs adjusted as indicated and for $\lambda = 1$. Note that very large systems are no longer optimal

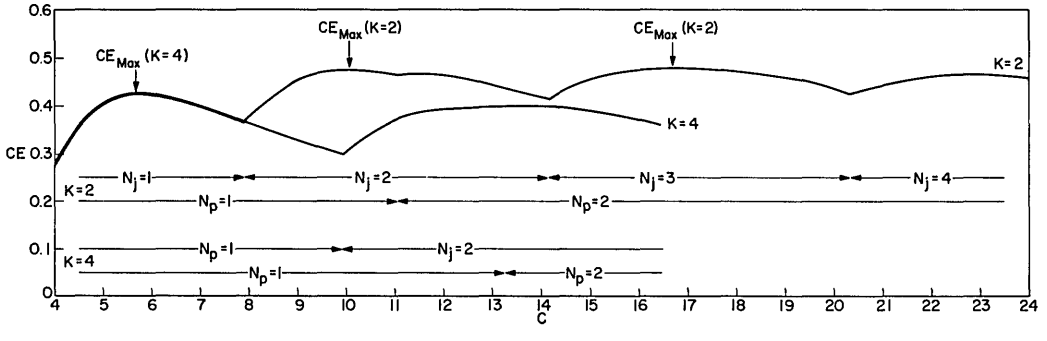


Fig. 44—**CE** vs **C** for nonzero job collection cost; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = \mu = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$.

and that for both values of **k** considered it is worthwhile to add the second processor before adding the third job. For **k** = 2 the optimal systems are $N_p = 1$ and $N_j = 2$, or $N_p = 2$ and $N_j = 3$. For **k** = 4 the cost of bringing the additional workload to the center is so high that the optimal strategy is to confine the center to the local workload.

For larger values of λ it will usually be desirable when **k** is very large to multiprogram the λ jobs of the local workload and to provide a suitable number of processors for this workload. For smaller values of **k** the optimal system may be quite large, but it will always be finite provided that **k** is not zero.

VII. SUMMARY

A throughput model was used to analyze the cost effectiveness, **CE**, of digital computer systems. Attention was directed at monoprogrammed systems, in which only one job is active at a time and only one processor is available, and at multiprogrammed systems, in which more than one job may be active at a time and more than one processor may be available.

The throughput model assumes that a finite system is offered an infinite workload of stationary statistics and that system productivity is to be measured in terms of the resources used to solve a problem and the rate of problem completion relative to the hardware provided. Questions relating to delays in problem solution and to which types of problems are eventually solved are ignored. System performance can then be characterized by **CE**, which is the weighted average of the fraction of the time each system component is beneficially used. Beneficial use corresponds to those periods during which the component is required to work on a job which is eventually completed. The weighting is by component cost.

The jobs constituting the workload were assumed to consist of a number, **n**, of segments. An increment of memory, **m**, was assumed to be required throughout the active life of the job, implying that complete roll-out of a suspended job was not considered. Beneficial use of **m** accrues while a job is being processed by a processor (computer) module or is outputting but not while the job is waiting for a processor or suspended waiting for additional memory. Another increment of memory, **x**, was assumed to be required while each segment was being processed; and a third increment of memory, **y**, was assumed to be required during the processing and output of a segment. Output time was assumed proportional to **y**, and input requirements were not separately stated. Operating system overhead functions were assumed to be included in the workload description of the individual jobs.

Although the simulator permitted the random variation of all these workload parameters according to a linear or an exponential distribution, in all simulations reported herein only $\underline{y}$ was allowed to vary within a single simulation run and only the exponential distribution was used. The cost and availability of i/o channels were ignored in these simulations. The analyses were performed under the same restrictions, with the further limitation that N_j, the maximum number of jobs active at one time, could not exceed 2.

It was shown that the monoprogrammed system usually cannot achieve high values of CE, because parts of the system must be unutilized some fraction of the time. The processor is not being beneficially used during output, and the fraction of wasted processor availability is equal to the output time divided by the sum of processing time and output time. The memory of a monoprogrammed system is also underutilized, because the component of memory devoted to **x** is utilized only when the processor is active.

Additional memory underutilization results from workloads with variable memory requirements. Since the effort devoted to processing segments of jobs that are terminated before completion due to an unallocatable memory request in connection with a later segment does not constitute beneficial use, it is necessary to provide memory for the component $\underline{y}$ of several times that component's average value. The optimum ratio of the memory provided for $\underline{y}$ to $\langle\underline{y}\rangle$ varies about logarithmically with **n**, the number of memory allocation requests in a job.

Multiprogramming can improve processor utilization. Under conditions of constant memory requirements and output time, if the ratio of output time to processing time is some integer **k**, the multiprogramming of **k** + 1 jobs will assure full processor utilization. If the ratio is not an integer, maximum CE results from the multiprogramming of approximately **k** + 1 jobs. The multiprogramming of a number of jobs equal to the next integer below **k** + 1 leads to some processor underutilization, while the multiprogramming of an additional job leads to some memory underutilization because an allocated job may be occupying memory but cannot proceed while it is waiting for the unavailable processor. The exact number of jobs to be multiprogrammed depends on the value of μ (the ratio of the cost of a unit of memory to the cost of a processor) and of the output time ratio in question.

Under conditions of constant memory requirements, multiprogramming offers the potential of improving memory utilization if, after the output of a segment is complete, the **x** and **y** of the next segment are not allocated until a processor is available for the execution of that segment. Both the simulator and most practical operating systems, however, preallocate the **x** and **y** of the next segment as soon as output of the previous segment is complete.

Under conditions of variable memory requirements, multiprogramming further improves memory utilization by allowing the averaging of peak memory demands. Thus the ratio of the memory provided for the variable components of the jobs to the sum of their average values need not be as high as it was in the monoprogrammed case. If the variable component of the memory required by a job is a significant fraction of the total requirement,

the gain in memory utilization resulting from the averaging of peak memory demands is more significant than the gain in memory utilization that would result if no preallocation was performed.

The practice of preallocation derives from the desire in an era of expensive processors to minimize the processor underutilization involved in performing memory allocations while the processor is waiting. Whereas the cost of processors relative to memory appears to be declining, it would be desirable, if the present work were being extended to include an accounting for the effect of secondary storage transfer times, to allow variations in the preallocation rule. It is conjectured that the number of preallocations permitted simultaneously should be a function of the number of processors in the system. The optimum number of preallocations might be of the order of the number of processors multiplied by the ratio of loading time to segment processing plus output time.

The multiprogrammed system must have some means of disposing of requests for memory allocations which cannot be granted immediately. The two simple policies studied were termed the termination strategy and the suspension strategy.

The termination strategy involves the termination of any job that makes a memory request that cannot be immediately granted. Since the time spent on a problem that is terminated before completion does not constitute beneficial use, frequent terminations lead to low **CE**. The frequency of terminations can be reduced by providing a memory substantially larger than the average memory requirements of the workload processor, but this memory is underutilized. Better values of **CE** under comparable conditions can be obtained through the use of the suspension strategy, which causes the termination only of jobs which make very large memory requests and which more usually suspends jobs until the requested memory can be allocated. These suspensions do decrease **CE**, however, since they increase the problem completion time and therefore reduce the throughput available from a given configuration. From the standpoint of utilization, suspension leads to the underutilization of the memory assigned to the **m** of a suspended job and to the underutilization of the processor that could be working on that job if the job could be allocated immediately.

When two jobs of $\mathbf{n} = 10$ are multiprogrammed, maximum **CE** under the termination strategy occurs when the total memory is of such a size that the wastage of termination is about 10%, i.e. when about 11 segments must be processed per job completion or about one job in 6 is not completed. Maximum **CE** under the suspension strategy occurs when the wastage of suspension is about 10%, i.e. when the mean job completion time is about 10% greater than what it would be if there was a very large memory and no suspensions. For a system of given size, the **CE** under the suspension strategy is greater. Conversely,

the use of the suspension strategy will permit the attainment of a given CE with a smaller system. For one of the models studied, the minimum **M** required for CE = 0.5 was about 7.2 for the suspension strategy and 8.8 for the termination strategy. The comparable figures for CE = 0.6 were 9.4 and 16.5.

If one employs the generally inferior termination strategy which places a heavy penalty on memory conflicts, some means must be provided to reduce the conflict probability to an acceptably low level. The means studied were to provide a limit, N_j, on the number of jobs that would be run simultaneously, and to specify M_s, the minimum amount of memory required to be available before a new job would be started. The minimum reasonable value of M_s is $m + x$, the minimum memory requirement of a job.

When M_s is set at this minimum level, N_j must be selected appropriately for the total amount of memory available. If N_j is too high, not enough jobs will be running simultaneously and the memory will be underutilized. If N_j is set too low, there will be a significant number of terminations. Nevertheless, even with M_s at the minimum level, significant improvement in memory utilization is possible. For example, for one particular workload model the optimum CE was obtained when the ratio of memory provided for the variable components to the sum of their average values was about 4 for $N_j = 1$, 2.8 for $N_j = 2$, 2 for $N_j = 3$, and seemed to be approaching a limit of about 1.5 for very large N_j.

Some improvement can be gained, particularly at larger values of N_j, by setting M_s above its minimum value. An optimum value of M_s improves CE only slightly at the best **M** for the N_j selected, but significantly improves CE at values of **M** less than optimum for the N_j by effectively reducing N_j. Except at very low values of N_j, the optimum increase in M_s was several times the average value of the variable component.

Attempts to abandon the N_j limit entirely and to depend only on M_s to control job initiation yielded somewhat poorer results. Under these circumstances, if M_s is set very high there will be significant memory underutilization. If M_s is set to a lower value, additional jobs will be started when the sum of the variable components of the other jobs is less than average. This sum is likely to grow to the average value or above before the jobs are completed, and terminations result. Thus when average memory requirements are known and do not vary too much from job to job, it is wiser to depend on N_j which takes into account the average memory requirements and to use M_s only to reduce the number of running jobs in those unusual circumstances where the total memory requirement is very considerably above average.

In the suspension strategy, when a job makes a memory request that cannot be allocated, the job is suspended until the request can be honored, but that job is still counted

in totaling the number of active jobs, N_a, to be compared with the N_j limit. If, however, the variable component of the request exceeds the limit y_t, the job is terminated.

Since the penalty associated with a memory conflict is very much less severe under the suspension strategy, less memory needs to be provided for a given N_j and better memory utilization results. For example, when the same workload discussed earlier was run under the suspension strategy, even with a conservatively low value of y_t that caused some unnecessary termination, the optimum ratio of memory provided for the variable components to the sum of their average values was about 2.3 for $N_j = 2$, 1.8 for $N_j = 3$, and seemed to be approaching a limit of 1.0 for very large N_j. Although the suspension strategy slows the rate of segment completion when there are memory conflicts, the reduction in termination restores the rate of job completion and significantly better values of **CE** are available for a given **M** than with the termination strategy.

If y_t is set to a conservative value, or if N_j is not too large, there is little incentive for raising M_s above its minimum value. At large values of **M** and N_j, however, a problem develops. The natural inclination is to set y_t to a rather large value to avoid terminations. If this is done, however, a very large memory request may not be allocated for a long time, if ever.

While this effect can be controlled by manipulating M_s and y_t, better solutions likely involve variation of the termination strategy. This seems to be a fruitful area for subsequent investigations. Among the techniques that should be considered are limitations on the number of jobs that can be suspended, dynamic variation of M_s according to the number of jobs suspended or how long a suspended job has been waiting, complete roll-out of suspended jobs, and collection of available memory until large requests can be serviced. Investigation of these techniques would probably require simulation of secondary storage access and transfer times and might require simulation of operating system overhead requirements. The assumption that these requirements can be included in the workload specification is reasonable only for those activities whose frequency can be predicted without a knowledge of the operating system algorithm and the environment. For example, the completion of a job requires the allocation of **n** segments of that job, but also requires a number of suspensions and reallocations that cannot be predicted without a calculation of the probability of suspension.

While increases in memory coupled with appropriate increases in N_j will always permit more averaging of peak memory demands, a point is reached where the resulting gain in memory utilization is overwhelmed by the loss in memory utilization resulting from allocated jobs waiting for the processor. Another way of stating this is to note that as the memory is increased, the throughput increases more and more slowly and

finally levels off as the system becomes completely processor limited. Thus, as the memory is increased and N_j optimally adjusted for the memory, the system CE will increase and then decrease. This effect provides the incentive for multiprocessing. Accordingly, from the CE standpoint there is little incentive for multiprocessing under conditions of constant memory demands.

Under conditions of variable memory requirements, multiprocessing allows increases in memory size and consequently improved memory utilization through peak averaging. In the limit, an infinite system can accomplish perfect averaging of the workload variability and a CE of unity could be approached.

For example, a workload with a high ratio of processing to output time was considered under the generally inferior termination strategy. With one processor a memory had to be supplied for $\underline{y}$ about 3.5 times its average value, and system CE was slightly above 0.5. With two processors and two jobs the ratio of the optimum memory provided for both $\underline{y}$'s to the sum of their average values was about 2.75, and the CE was slightly above 0.6. With nine processors and ten jobs the memory ratio was about 1.8, and the CE was about 0.78.

However, the higher values of CE can be obtained only with large and costly systems. For the example just discussed, the costs of the three systems were 6.5, 11.5, and 48. In a real application neither infinite funds nor an infinite workload will be available.

Nevertheless, there are a number of problems that can be attacked through the application of the results of throughput analysis. Among these problems are the determination of the optimum expenditure of a fixed budget for the initial construction or for the expansion of a computer center, the determination of the most economical hardware mix that will produce a specified throughput, and the determination of the optimum size system to process a workload which includes additional job collection costs.

The first type of problem can be solved if it is reasonable to assume that there is a potential workload considerably larger than the capacity of any system that can be procured with the available funding. In these circumstances the assumption of an infinite workload in the throughput model is acceptable, and the problem reduces to the determination of the configuration costing the specified amount that yields the highest CE or throughput when offered an infinite workload.

Although throughput increases monotonically with cost, CE does not, even when the optimum combination of processors and memory in every cost interval is considered. Particularly at low values of μ there are regions where the throughput is rising very slowly as the system cost is increased and the CE is therefore declining. This situation

occurs when throughput is processor limited and increases in memory increase throughput only slightly, but when an additional processor cannot be efficiently utilized if it is necessary to reduce the memory to hold the system cost constant.

The optimum utilization of an expansion budget less than the cost of a processor may involve the procurement of an additional processor and a reduction in memory to help finance it. If the system is purchased rather than rented a return of memory for credit may not be possible. Under these circumstances it is necessary to consider the probability of expansion in the determination of the initial system to procure, and it may be desirable to procure a system whose throughput is less than the maximum obtainable at that cost level in order that the system can later be expanded into an optimum system at a higher cost level.

The solution of these types of problems requires the determination of the optimum ratio of the number of jobs multiprogrammed to the number of processors, even though this ratio is not a purchase parameter. For high values of μ this ratio is one greater than the ratio of mean output time to processing time per segment. This is the same result that was found under conditions of constant memory requirements. At low values of μ, however, considerable memory underutilization is acceptable if the valuable processors are kept busy thereby, and the optimum ratio is somewhat higher.

The problem of determining the minimum cost system that will process a specified workload can only be approximately solved through the use of the throughput model. Although it is easy to determine the minimum cost system that will yield the specified throughput when offered an infinite workload, there is no assurance that this system will process every component of a specific workload of equivalent throughput without introducing very long solution delays. The complete solution of this problem would require the consideration of queuing models that lie beyond the scope of the present investigation, but the approximate solution afforded by the throughput model places a lower limit on the size of the system that can process a specified workload.

The problem of balancing the high efficiency of large systems against job collection costs is of current interest because of the rapid growth of remote computing. There are situations in which the optimum size of a computing center is primarily determined by the data transmission costs involved in the processing of remote workloads. Again the throughput model cannot accurately solve this problem, but useful approximations can be obtained.

APPENDIX A

GLOSSARY OF NOTATION

Symbol	Page First Appearing	Definition
C	5	System cost
CE	6	Cost effectiveness
M	4	System memory
M_s	31	Minimum available memory required to start another job
N_a	31	Number of active jobs
N_c	3	Number of i/o channels
N_j	30	Maximum number of active jobs permitted
N_p	3	Number of processor (computer) modules
P	78	Simulator input parameter (See Appendix B, Table B1)
P_c	8	Probability of a memory allocation conflict
P_s	25	Probability that an allocation request that does not cause termination causes suspension
$P_{\bar{s}}$	25	$1 - P_s$
P_t	24	Probability that an allocation request causes termination under the suspension strategy
Q	79	Simulator output parameter (See Appendix B, Table B2)
Q_c	8	$1 - P_c$
Q_t	24	$1 - P_t$
T	3	Throughput
V	5	Mean value of completed jobs
Z_i	82	Probability that a job is in its ith segment
Z_i'	88	$1 - Z_i$
c	94	Conflict ratio
m	5	Base memory requirement of a job
n	5	Number of segments in a job
r	5	I/o channel rate
t_c	5	Mean time between job completions

Symbol	Page First Appearing	Definition
t_o	8	Mean output time per segment
t_{o2}	13	Mean output time plus waiting time for the processor (termination strategy)
t'_{o2}	25	Mean output time plus waiting time for the processor plus suspension time (suspension strategy)
t_{os}	26	Mean output time for segments that caused suspension
$t_{o\bar{s}}$	26	Mean output time for segments that did not cause suspension
t_{o2s}	25	Mean output time plus waiting times for segments that caused suspension
$t_{o2\bar{s}}$	50	Mean output time plus waiting time for the processor for segments that did not cause suspension
t_I	26	Mean time between an allocation request and the allocation of a suspended job
t_{II}	26	Mean time required for processing and output of a suspended segment
t_{III}	26	Processing time of the next segment after a conflict is resolved
$\mathbf{x}$	4	Memory required (in addition to $\mathbf{m}$) during processing of a segment
$\underline{\mathbf{y}}$ or $\mathbf{y}$	4	Memory required (in addition to $\mathbf{m}$ and $\mathbf{x}$) to hold output data during processing and output of a segment
y_t	24	Maximum $\underline{\mathbf{y}}$ request allowed without termination
y_1	8	Memory available for the allocation of one $\underline{\mathbf{y}}$
y_2	17	Memory available for the allocation of two $\underline{\mathbf{y}}$'s
$\underline{\mathbf{z}}$	5	An allocated $\underline{\mathbf{y}}$
$\underline{\mathbf{z}}_s$	25	A $\underline{\mathbf{z}}$ that causes suspension
a	5	Distribution parameter (1/mean) of an exponential distribution
γ	8	Mean number of segments completed per job completed
γ_t	24	γ for the suspension strategy
τ	4	Processing time of a segment
μ	5	Cost of a unit of system memory relative to the cost of one processor
χ	4	Cost of an i/o channel relative to the cost of one processor

Bold faced, scored type indicates random variables. In general, $\mathbf{P}$ denotes a discrete probability, $\mathbf{p}$ a probability distribution, $\mathbf{P}^+$ a cumulative distribution, and $\mathbf{Q}$ an improbability. Thus

$$\mathbf{p}_{\underline{\mathbf{z}}}(\mathbf{w})\, d\mathbf{w} = \mathbf{P}(\mathbf{w} < \underline{\mathbf{z}} < \mathbf{w} + d\mathbf{w}),$$

$$\mathbf{P}^+_{\underline{\mathbf{z}}}(\mathbf{w}) = \mathbf{P}(\underline{\mathbf{z}} > \mathbf{w}) = \int_{\mathbf{w}}^{\infty} \mathbf{p}_{\underline{\mathbf{z}}}(\mathbf{v})\, d\mathbf{v},$$

and

$$\mathbf{Q} = 1 - \mathbf{P}.$$

APPENDIX B

DESCRIPTION OF SIMULATOR

A system simulator was constructed in the Neliac language (4, 5) and run on the AN/GYK-3(V) computer (2, 6) at the Naval Research Laboratory.

Table B1 lists the inputs to a simulation. In this table, an entry in the Range column implies that the input parameter is an integer in the specified range. The lack of an entry implies that any positive real quantity can be simulated. An entry in the Usual column implies that most or all simulations were conducted with that value of the parameter, with parentheses indicating most and the lack of parentheses indicating all.

Table B-2 lists the primary outputs that were obtained from all simulations and relates them to the quantities discussed in the text. Optional outputs included results of calculations based on the analyses of Sections III and IV and periodic snapshots of the state of the simulated system.

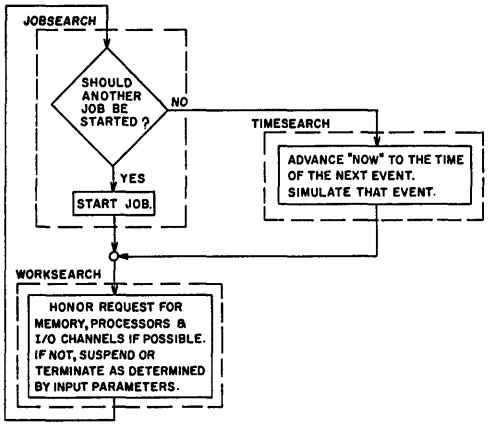


Fig. B1 – Principal simulator routines.

Figure B-1 shows the relationship between three of the important routines in the simulator. The routine WORKSEARCH attempts to keep active jobs progressing by assigning system resources, although under some circumstances it must terminate jobs. When it has serviced all jobs it passes control to JOBSEARCH.

The JOBSEARCH routine determines if it is appropriate to start a new job. If a new job is started, control is passed back to WORKSEARCH so that the new job may be serviced. If it is not an appropriate time to start a new job, nothing more can be done at the simulated instant and control is passed to the TIMESEARCH routine.

Table B1
Simulator Inputs

Parameter (P) Number	Symbol	Range	Usual
1	N_p	1-10	
2	N_c	1-10	10
3	r		
4	M		
5	(Note A)	0-3	
6	N_j	1-10	
7	M_s		
8	$(x + y)_t$		
9	(Note B)	0-4	
10	(Note B)	0-4	
11	$\langle \underline{r} \rangle$		1
12	(Note C)	0-2	0
13	$\langle \underline{m} \rangle$		
14	(Note C)	0-2	0
15	$\langle \underline{x} \rangle$		
16	(Note C)	0-2	0
17	$\langle \underline{y} \rangle$		
18	(Note C)	0-2	(2)
19	$\langle \underline{n} \rangle$		(10)
20	(Note C)	0-2	0
21	μ		
22	χ		0
23	Simulation time		(20,000)
24	Snapshot interval		

NOTE A: P_5 controls job initiation. The condition for starting a new job is

$$[(P_5 = 0) \wedge (N_a < N_j)] \vee [(P_5 = 1) \wedge (M_a > M_s)]$$
$$\vee [(P_5 = 2) \wedge (N_a < N_j) \wedge (M_a > M_s)]$$
$$\vee [(P_5 = 3) \wedge [(N_a < N_j) \vee (M_a > M_s)]]$$

where M_a is the available (unallocated) memory.

NOTE B: P_9 and P_{10} control the actions to be taken in case of i/o channel and memory conflicts respectively. 0: suspend; 1: terminate; 2,3: unimplemented.

NOTE C: These parameters control the statistical distributions of the quantities whose means are given above. 0: point; 1: linear; 2: exponential; e.g.,

if $P_{18} = 0$, $p_{\underline{y}}(y) = \delta(y - \langle \underline{y} \rangle)$.

if $P_{18} = 1$, $p_{\underline{y}}(y) = 1/2 \langle \underline{y} \rangle$, $0 < y < 2 \langle \underline{y} \rangle$;
$= 0$, elsewhere.

if $P_{18} = 2$, $p_{\underline{y}}(y) = 1/\langle \underline{y} \rangle \, e^{-y/\langle \underline{y} \rangle}$, $y > 0$;
$= 0$, $y < 0$.

Table B2
Simulator Outputs.

Parameter (Q) Number	Description	Relation to quantities in text
1	Number of jobs attempted	$c = Q_1/Q_3$
2	Number of jobs terminated	TERM. RATIO $= Q_2/Q_3$
3	Number of jobs completed	
4	Total value	$V = Q_4/Q_3$
5	Mean value	$V = Q_5$
6	Simulated time	$t_c = Q_6/Q_3$
7	Cost	$C = Q_7$
8	Number of segments completed	$\gamma = Q_8/Q_3$
9	Number of jobs terminated after completing at least one segment	ABAND. RATIO $= Q_9/Q_3$
10	Throughput	$T = Q_{10}$
11	Cost effectiveness	$CE = Q_{11}$
12-18	Number of allocated $\underline{z}$'s greater than 0.5 (0.5) 3.5	$P^{+}_{\underline{z}}(k/2) = (Q_{11+k})/Q_8$, $k = 1, 2, \ldots, 7$

The TIMESEARCH routine locates the time of the next completion of processing or outputting, simulates the actions directly implied by that completion, and passes control to WORKSEARCH so that the other jobs may be serviced.

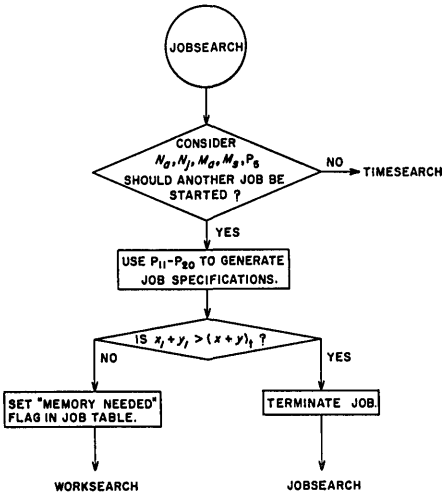


Fig. B2 – JOBSEARCH routine.

The principal functions of these three routines are illustrated in Figs. B-2 through B-4.

The amount of actual computer time required to perform a simulation depends on the number of segments completed and the amount of termination involved. Typical simulation runs involved the completion of about 20,000 segments and required from three to five minutes of processing by the computer, which has internal speeds roughly comparable to those of an IBM 7090. Most runs were repeated five times to obtain more reliable averages and estimates of accuracy. Over 300 hours of computing time were involved in the simulations reported herein.

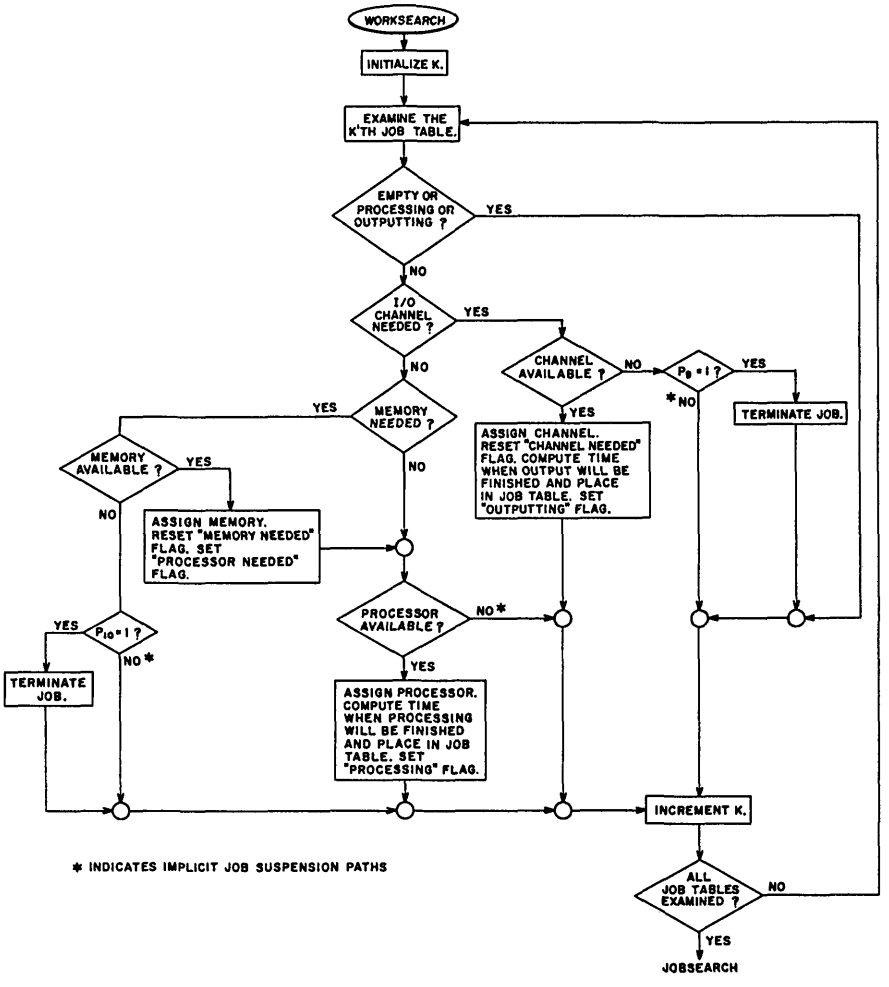


Fig. B3 – WORKSEARCH routine.

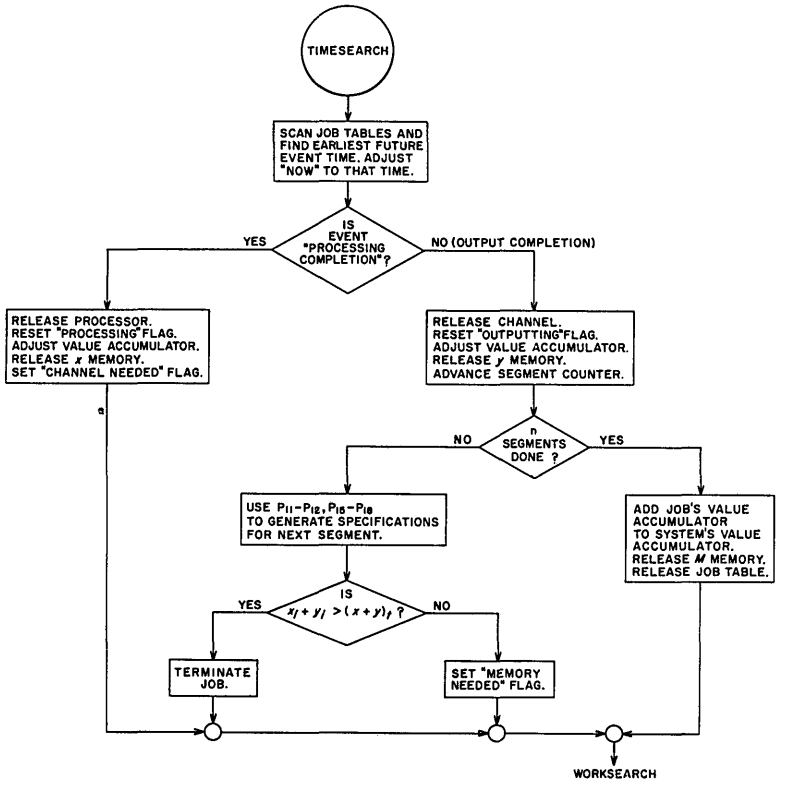


Fig. B4 – TIMESEARCH routine.

APPENDIX C

DERIVATION OF EQUATIONS (7) THROUGH (14)

Since the $\underline{y}$ distribution is given as

$$p_{\underline{y}}(y) = \alpha e^{-\alpha y}, \tag{1}$$

the conflict probability is

$$P_c = P(\underline{y} > y_1) = \int_{y_1}^{\infty} p_{\underline{y}}(y)\, dy = e^{-\alpha y_1}, \tag{7}$$

and the $\underline{z}$ distribution is the $\underline{y}$ distribution truncated at y_1 and normalized, viz.,

$$p_{\underline{z}}(z) = \frac{\alpha e^{-\alpha z}}{1 - e^{-\alpha y_1}},\ 0 \le z \le y_1;\ 0 \text{ otherwise.} \tag{8}$$

Now let Z_i denote the probability that the job is in its ith segment. Obviously,

$$\sum_{i=1}^{n} Z_i = 1. \tag{C1}$$

At the conclusion of each segment there is a probability P_c that the next segment cannot be allocated. Then, defining $Q_c = 1 - P_c$,

$$Z_{i+1} = Q_c Z_i \qquad (n > i \ge 1) \tag{C2}$$

or

$$Z_k = Q_c^{k-1} Z_1. \qquad (n \ge k \ge 1) \tag{C3}$$

This can be combined with equation (C1) to yield

$$1 = Z_1 \sum_{i=0}^{n-1} Q_c^i \tag{C4}$$

If this is combined with the familiar relation for the sum of a geometric series

$$\sum_{i=0}^{\ell} r^i = \frac{1 - r^{\ell+1}}{1 - r}.$$

there results

$$1 = Z_1 \frac{1 - Q_c^{\,n}}{1 - Q_c}, \tag{C5}$$

which can be solved for

$$Z_1 = \frac{1 - Q_c}{1 - Q_c^{\,n}} = \frac{P_c}{1 - Q_c^{\,n}}$$

$$Z_n = Q_c^{\,n-1}\, Z_1 = \frac{Q_c^{\,n-1}\, P_c}{1 - Q_c^{\,n}}. \tag{C6}$$

Since every job in the nth state goes to completion,

$$\gamma = \frac{1}{Z_n} = \frac{1 - Q_c^{\,n}}{P_c\, Q_c^{\,n-1}}. \tag{9}$$

Since the output time of a segment is $\underline{z}/r$, the mean output time is

$$t_o = \frac{1}{r}\int w\, p_{\underline{z}}(w)\, dw = \frac{1}{r\, Q_c}\int_0^{y_1} \alpha w e^{-\alpha w}\, dw = \frac{1}{r\, Q_c}\left[-\left(\frac{1}{\alpha} + w\right) e^{-\alpha w}\right]_0^{y_1}$$

$$= \frac{1}{r\, Q_c}\left[\frac{1}{\alpha} - \left(\frac{1}{\alpha} + y_1\right) P_c\right], \tag{10}$$

and since the mean total time per segment is $1 + t_o$, by the definition of γ,

$$t_c = \gamma\,(1 + t_o). \tag{11}$$

To compute the mean value, first find

$$\langle \underline{z}^2 \rangle = \int w^2\, p_{\underline{z}}(w)\, dw = \frac{1}{Q_c}\int_0^{y_1} \alpha w^2 e^{-\alpha w}\, dw = \frac{1}{Q_c}\left[-\frac{e^{-\alpha w}}{\alpha^2}((\alpha y)^2 + 2\alpha y + 2)\right]_0^{y_1}$$

$$= \frac{1}{\alpha^2(1 - P_c)}\,[2 - e^{-\alpha y_1}(\alpha^2 y_1^{\,2} + 2\alpha y_1 + 2)] = \frac{2 - (\alpha^2 y_1^{\,2} + 2\alpha y_1 + 2)P_c}{\alpha^2\,(1 - P_c)}; \tag{C7}$$

then recall that $t_o = \langle \underline{z}/r \rangle$, and substitute equation (C7) in

$$V = n\left\{1 + \mu\left[m + x + \left(1 + \frac{m}{r}\right)\langle \underline{z} \rangle + \frac{1}{r}\langle \underline{z}^2 \rangle\right]\right\} \tag{2}$$

to yield

$$V = n\left\{1 + \mu\left[m + x + (m + r)\, t_o + \frac{2 - (\alpha^2 y_1{}^2 + 2\,\alpha y_1 + 2)\, P_c}{\alpha^2 r\,(1 - P_c)}\right]\right\}. \tag{12}$$

Substitution of equations (3), (4) and (11) into equation (5) yields

$$CE = \frac{V}{(1 + \mu M)\,(1 + t_o)\,\gamma}, \tag{C8}$$

and substitution of equations (7), (9) and (10) into equation (C7) yields

$$CE = \frac{n\left\{1+\mu\left[m+x+\frac{m+r}{r(1-e^{-\alpha y_1})}\left(\frac{1}{\alpha}-\left(\frac{1}{\alpha}+y_1\right)e^{-\alpha y_1}\right)+\frac{2-(\alpha^2 y_1{}^2+2\alpha y_1+2)e^{-\alpha y_1}}{\alpha^2 r\,(1-e^{-\alpha y_1})}\right]\right\}e^{-\alpha y_1}(1-e^{-\alpha y_1})}{(1+\mu M)\left\{1+\frac{1}{r(1-e^{-\alpha y_1})}\left[\frac{1}{\alpha}-\left(\frac{1}{\alpha}+y_1\right)e^{-\alpha y_1}\right]\right\}[1-(1-e^{-\alpha y_1})^n]} \tag{13}$$

To derive the approximation for CE, assume that P_c is very small. Then it is reasonable to approximate γ by

$$\gamma = \frac{1 - Q_c{}^n}{P_c\, Q_c{}^{n-1}} = \frac{1 - (1 - P_c)^n}{P_c\,(1 - P_c)^{n-1}} \approx \frac{1 - (1 - nP_c)}{P_c\,(1 - (n - 1)\,P_c)} = \frac{n}{1 - (n - 1)\,P_c}, \tag{C9}$$

and to approximate $p_{\underline{z}}(z)$ by $p_{\underline{y}}(y)$. Substituting $1/\alpha$ and $2/\alpha^2$, the mean and mean square values of the exponential distribution, into equation (2) yields

$$V \approx n\left[1 + \mu\left(m + x + \frac{m}{\alpha r} + \frac{1}{\alpha} + \frac{2}{\alpha^2 r}\right)\right] \tag{C10}$$

and into (10) yields

$$t_o \approx \frac{1}{\alpha r}. \tag{C11}$$

Substitution of equations (C9), (C10), and (C11) into equation (C8) and reexpression of P_c and M in terms of y_1 then yields

$$CE \approx \frac{\left[1 + \mu\left(m + x + \frac{m}{\alpha r} + \frac{1}{\alpha} + \frac{2}{\alpha^2 r}\right)\right][1 - (n - 1)e^{-\alpha y_1}]}{\left(1 + \frac{1}{\alpha r}\right)[1 + \mu\,(m + x + y_1)]} \tag{14}$$

Table C1 compares the values of CE resulting from the use of equations (13) and (14). It can be seen that (14) is a reasonable approximation of (13) in the vicinity of CE_{max},

Table C1
CE vs y_1, Monoprogrammed Case; $m = x = \alpha = \mu = 1$, $r = 0.2$, $n = 10$.

y_1	CE	
	Equation (13)	Equation (14)
2	0.234	negative
2.5	0.310	0.150
3	0.355	0.291
3.5	0.376	0.355
4	0.381	0.378
4.5	0.376	0.381
5	0.366	0.371
5.5	0.353	0.357
6	0.340	0.344

where the assumption of low P_c is valid. In this particular case, the use of equation (14) overestimates the value of y_1 that yields CE_{max} by about 0.3 memory units. Since CE is changing rather slowly with variations in y_1, the value of CE_{max} from the approximation is quite accurate.

The approximation for CE can be differentiated with respect to y_1 to yield

$$\frac{\partial (CE)}{\partial y_1} = \frac{[1 + \mu(m + x + y_1)]\, \alpha(n - 1)e^{-\alpha y_1} - [1 - (n - 1)e^{-\alpha y_1}]\, \mu}{[1 + \mu(m + x + y_1)]^2} \frac{V}{t_o}. \tag{C12}$$

Equation of the numerator of equation (C12) to zero yields

$$[1 + \mu(m + x + y_1)]\, \alpha(n - 1)e^{-\alpha y_1} = [1 - (n - 1)e^{-\alpha y_1}]\, \mu \tag{C13a}$$

or

$$\left(\frac{1}{\mu} + m + x + y_1\right) \alpha\,(n - 1) = e^{\alpha y_1} - (n - 1) \tag{C13b}$$

and manipulation into a form that rapidly converges produces

$$e^{\alpha y_1} = \left(\frac{1}{\alpha} + \frac{1}{\mu} + m + x + y_1\right) \alpha\,(n - 1) \tag{C14a}$$

or

$$\alpha y_1 = \ln\left(\frac{1}{\alpha} + \frac{1}{\mu} + m + x + y_1\right) + \ln \alpha + \ln (n - 1) \tag{C14b}$$

for the value of y_1 that maximizes CE. For a reasonable initial estimate, this expression usually converges within 1% of the final value in three or less iterations.

It was seen in connection with Table C1 that the use of the approximation led to an overestimation of the optimum y_1. If n is reduced from 10 to 5, however, the approximation leads to an underestimation of the optimum y_1 of about 1 unit of memory. The explanation is that at the lower value of n the effects of a termination on throughput are not so deleterious, optimum operation occurs with a higher P_c, and the approximation of

equation (C9) is no longer valid. Fortunately the same effect reduces the harm attendant to operating at low values of y_1, and operation at the value obtained through the use of equations (C14b) leads to an actual CE only a few percent less than the maximum possible value.

APPENDIX D

OPTIMUM SELECTION OF THE JOB TO TERMINATE

A simple modification of the termination policy considered in the text would be, at the time of a memory conflict, to terminate that job which had made the least progress toward completion and thereby minimize the wasted effort of the system.

Since each $\underline{y}$ is an independent sample of the distribution, the conflict probabilities will remain unchanged. The state transition probabilities must be modified, however. For the standard strategy,

$$Z_{i+1} = Q_c Z_i \; (n > i \geq 1). \tag{C1}$$

For the alternative strategy, if there is a conflict, and if the running program is at an earlier state, the running program will be terminated. Thus there must be added to the transition probability of the potentially conflicting program a factor of

$$Z_i P_c \sum_{j=1}^{i-1} Z_j.$$

On the other hand, if the other program is at a later state and has a conflict, the program at step i will be terminated and a compensating factor must be subtracted. Thus the transition probability for the alternate strategy is given by

$$Z_{i+1} = Z_i \left[Q_c + P_c \left(\sum_{j=1}^{i-1} Z_j - \sum_{k=i+1}^{n-1} Z_k \right) \right] (n > i \geq 1), \tag{D1a}$$

where

$$\sum_{i=1}^{n} Z_i = 1. \tag{D1b}$$

This is a set of nonlinear equations that does not have a convenient closed form solution. However, summing the first $n - 1$ equations of (D1a) yields

$$\sum_{i=1}^{n-1} Z_{i+1} = Q_c \sum_{i=1}^{n-1} Z_i + P_c \sum_{i=1}^{n-1} Z_i \left(\sum_{j=1}^{i-1} Z_j - \sum_{k=i+1}^{n-1} Z_k \right). \tag{D2}$$

Substitution of equation (D1b) and the definition $Z' = 1 - Z$ into equation (D2) yields

$$Z_1' = Q_c Z_n' + P_c \left[\sum_{i=1}^{n-1} Z_i \left(\sum_{j=1}^{i-1} Z_j - \sum_{k=i+1}^{n-1} Z_k \right) \right]. \tag{D3}$$

But the expression within the brackets sums to zero, since each possible combination of two different subscripts appears twice with opposite signs; i.e.,

$$\sum_{i=1}^{n-1} Z_i \left(\sum_{j=1}^{n-1} Z_j - \sum_{k=i+1}^{n-1} Z_k \right) = \sum_{i=1}^{n-1} \sum_{\substack{m=1 \\ \neq i}}^{n-1} Z_i Z_m \text{ sign } (i - m) = 0.$$

Thus equation (D3) simplifies to

$$Z_1' = Q_c Z_n'. \tag{D4}$$

Now it is necessary to manipulate equation (D1a) so that later state probabilities are not needed in the calculation of earlier state probabilities. Substitution of the definition of Q_c into equation (D1a) permits this transformation, viz.,

$$\begin{aligned}
Z_{i+1} &= Z_i \left[1 + P_c \left(\sum_{j=1}^{i-1} Z_j - \sum_{k=i+1}^{n-1} Z_k - 1 \right) \right] \\
&= Z_i \left[1 + P_c \left(\sum_{j=1}^{i-1} Z_j - \sum_{k=i+1}^{n-1} Z_k - \sum_{m=1}^{n} Z_m \right) \right] \\
&= Z_i \left[1 - P_c \left(Z_i + Z_n + 2 \sum_{j=i+1}^{n-1} Z_j \right) \right] \\
&= Z_i \left[1 - P_c \left(Z_i + Z_n + 2 \left(1 - \sum_{k=1}^{i} Z_k - Z_n \right) \right) \right] \\
&= Z_i \left[1 - P_c \left(2 - Z_n + Z_i - 2 \sum_{k=1}^{i} Z_k \right) \right] \\
&= Z_i \left[1 - P_c \left(2 - \left(Z_n - Z_i + 2 \sum_{k=1}^{i} Z_k \right) \right) \right] \quad (n > i \geq 1).
\end{aligned} \tag{D5}$$

Equations (D4) and (D5) form a suitable set for iterative solution. If a tentative Z_1 is selected, a corresponding Z_n is immediately available, and the recurrence relationships can be used to calculate the intermediate probabilities. The probabilities are summed, the sum compared to unity, and Z_1 adjusted accordingly. Convergence is reasonably rapid.

Exhaustive study of this strategy would require the computer solution of many cases through the use of these equations. From a limited amount of hand computations, however, some conclusions can be drawn. Typical calculated results are shown in Table D1.

Table D1
Comparison of Termination Strategies.

Strategy	$n = 3, P_c = 0.5$			$n = 3, P_c = 0.1$			$n = 4, P_c = 0.1$			
	Z_1	Z_2	Z_3	Z_1	Z_2	Z_3	Z_1	Z_2	Z_3	Z_4
Simple	0.571	0.286	0.143	0.369	0.332	0.299	0.291	0.262	0.236	0.212
Alternative	0.590	0.228	0.182	0.373	0.324	0.303	0.297	0.254	0.230	0.219

It can be seen that in one case ($n = 3$, $P_c = 0.5$) the alternative strategy yielded a significantly larger Z_n and therefore smaller γ. This case, however, assumed a very large value of P_c. It was seen earlier that maximum CE occurred when γ was not much more than $1.1n$. In the second example $\gamma \approx 1.1n$, and in the third example $\gamma \approx 1.15n$. It seems safe to conclude that for those combinations of n and P_c that yield a reasonable value of γ the difference in strategies is not significant. For this reason the bulk of the study was performed under the assumption that the job that makes the conflicting request is the one that will be terminated. Nevertheless, if an operating system is forced to terminate a problem, it would appear worthwhile for it to terminate the problem on which the least has been accomplished, provided that the termination costs are not significantly increased thereby and provided that both problems are drawn from the same population, i.e., are expected to require the same total number of segments and to be equally likely to request the same segment sizes.

APPENDIX E

DERIVATION OF EQUATIONS (20) THROUGH (24)

Equation (19) may be rewritten in terms of density functions. The left side is $p_{\underline{z_i}}(z)\,\Delta z$. The conditional probability in the integrand is $p_{\underline{y}}(z)\,\Delta z$ normalized by $P(\underline{y} < y_2 - w)$. The result is

$$p_{\underline{z_i}}(z) = \int_0^{y_2 - z} p_{\underline{z_{i+1}}}(w) \frac{p_{\underline{y}}(z)}{\int_0^{y_2 - w} p_{\underline{y}}(y)} dw. \tag{E1}$$

In order to solve this equation it will be necessary to equate $p_{\underline{z_i}}$ with $p_{\underline{z_{i+1}}}$. This assumption would be unquestioned if the process under study were a simple replacement process, i.e., if $\underline{z_i}$ was the first $\underline{y}$ less than $y_2 - \underline{z_{i-1}}$ and if $\underline{z_{i+1}}$ was the first y less than $y_2 - \underline{z_i}$, etc.

Actually, $\underline{z_i}$ may be replacing either $\underline{z_{i-2}}$ or $\underline{z_{i-1}}$ depending on which job has completed its output first. If $\underline{z_{i-2}}$ is larger than usual, $\underline{z_{i-1}}$ is likely to be smaller than usual and it is not unlikely that output from $\underline{z_{i-1}}$ will be completed before that from $\underline{z_{i-2}}$. Thus the relatively large $\underline{z_{i-2}}$ will have interfered with two allocations, that of $\underline{z_{i-1}}$ and of $\underline{z_i}$. The net effect will be to raise P_c and to skew $p_{\underline{z}}(z)$ to the left.

If r is large, even an unusually large $\underline{z}$ would not have a t_o greater than τ, and the simple replacement operation would be assured. The smaller the value of r, the greater the underestimation of P_c. Thus in the development to follow, the results will constitute a lower bound on P_c and an upper bound on CE, with the actual situation approaching the bound as r increases.

Equating of the densities and manipulation produces an integral equation:

$$p_{\underline{z}}(z) = \int_0^{y_2 - z} p_{\underline{z}}(w) \frac{p_{\underline{y}}(z)}{\int_0^{y_2 - w} p_{\underline{y}}(y)\,dy} dw$$

$$= \int_0^{y_2 - z} \frac{p_{\underline{z}}(w)\,\alpha e^{-\alpha z}}{\int_0^{y_2 - w} \alpha e^{-\alpha y}\,dy} dw$$

$$= \int_0^{y_2 - z} \frac{p_{\underline{z}}(w)\,dw}{1 - e^{-\alpha(y_2 - w)}} \cdot \alpha e^{-\alpha z}.$$

Then,

$$\frac{1}{\alpha} e^{\alpha z} p_{\underline{z}}(z) = \int_0^{y_2 - z} \frac{p_{\underline{z}}(w)\, dw}{1 - e^{-\alpha(y_2 - w)}} . \tag{E2}$$

This integral equation can be differentiated by Leibnitz's rule to yield

$$e^{\alpha z} p_{\underline{z}}(z) + \frac{1}{\alpha} e^{\alpha z} p'_{\underline{z}}(z) = \frac{p_{\underline{z}}(y_2 - z)}{1 - e^{-\alpha z}} \frac{\partial}{\partial_z}(y_2 - z),$$

which reduces to

$$p'_{\underline{z}}(z) + \alpha p_{\underline{z}}(z) + \frac{\alpha p_{\underline{z}}(y_2 - z)}{e^{\alpha z} - 1} = 0. \tag{E3}$$

The differential equation (E3) is solved by functions of the form

$$p_{\underline{z}}(z) = \lambda\, [e^{\alpha(y_2 - z)} - 1]. \tag{E4}$$

To verify that equation (E4) solves (E2) substitution is used:

$$\frac{1}{\alpha} e^{\alpha z} \lambda\, (e^{\alpha(y_2 - z)} - 1) = \int_0^{y_2 - z} \frac{\lambda\, (e^{\alpha(y_2 - w)} - 1)}{1 - e^{-\alpha(y_2 - w)}}\, dw,$$

$$\frac{\lambda}{\alpha}(e^{\alpha y_2} - e^{\alpha z}) = \lambda \int_0^{y_2 - z} e^{\alpha(y_2 - w)}\, dw$$

$$= \frac{\lambda}{\alpha}[-e^{\alpha(y_2 - w)}]_0^{y_2 - z} = \frac{\lambda}{\alpha}(e^{\alpha y_2} - e^{\alpha z}),$$

and λ is evaluated by noting that $p_{\underline{z}}(z)$ must have a unit integral over the range $0 < z < y_2$ and be zero outside that range. Thus

$$1 = \int_0^{y_2} p_{\underline{z}}(z)\, dz = \lambda \int_0^{y_2} (e^{\alpha(y_2 - z)} - 1)\, dz$$

$$= \lambda \left[\frac{e^{\alpha(y_2 - z)}}{-\alpha} - z \right]_0^{y_2}$$

$$= \frac{\lambda}{\alpha}(e^{\alpha y_2} - 1 - \alpha y_2)$$

or

$$\lambda = \frac{\alpha}{e^{\alpha y_2} - 1 - \alpha y_2} . \tag{E5}$$

Substitution of equation (E5) into equation (E4) yields

$$p_{\underline{z}}(z) = \frac{\alpha(e^{\alpha(y_2-z)} - 1)}{e^{\alpha y_2} - 1 - \alpha y_2}. \tag{20}$$

The probability of conflict is found by integrating the probability distributions:

$$\begin{aligned}
P_c &= P(\underline{y} > y_2 - \underline{z}) \\
&= \int_0^{y_2} p_{\underline{z}}(z)\, P(\underline{y} > y_2 - z)\, dz \\
&= \int_0^{y_2} p_{\underline{z}}(z) \left[\int_{y_2-z}^{\infty} p_{\underline{y}}(y)\, dy \right] dz \\
&= \int_0^{y_2} \lambda\, (e^{\alpha(y_2-z)} - 1)\, e^{-\alpha(y_2-z)}\, dz \\
&= \lambda \int_0^{y_2} (1 - e^{\alpha(z-y_2)})\, dz \\
&= \lambda \left(y_2 - \frac{1}{\alpha} + \frac{1}{\alpha} e^{-\alpha y_2} \right) \\
&= \frac{e^{-\alpha y_2} + \alpha y_2 - 1}{e^{\alpha y_2} - \alpha y_2 - 1}.
\end{aligned} \tag{21}$$

The mean value of $\underline{z}$ is found as

$$\begin{aligned}
\langle \underline{z} \rangle &= \int_0^{y_2} z p_{\underline{z}}(z)\, dz = \lambda \int_0^{y_2} (z e^{\alpha(y_2-z)} - z)\, dz \\
&= \lambda \left[-e^{\alpha y_2} \left(\frac{1}{\alpha^2} + \frac{z}{\alpha} \right) e^{-\alpha z} - \frac{z^2}{2} \right]_0^{y_2} \\
&= \frac{\lambda}{\alpha^2} \left[-1 - \alpha y_2 + e^{\alpha y_2} - \frac{(\alpha y_2)^2}{2} \right] \\
&= \frac{e^{\alpha y_2} - \alpha y_2 - \frac{(\alpha y_2)^2}{2} - 1}{r(e^{\alpha y_2} - \alpha y_2 - 1)}
\end{aligned} \tag{E6}$$

and the mean output time per segment is

$$t_o = \frac{\langle \underline{z} \rangle}{r} = \frac{e^{\alpha y_2} - \alpha y_2 - \frac{(\alpha y_2)^2}{2} - 1}{\alpha r (e^{\alpha y_2} - \alpha y_2 - 1)}. \tag{22}$$

If $y_2 < r$, then all $\underline{z} < r$ and $t_{o2} = 1$. In the more usual case of $y_2 > r$, there will be a contribution of 1 to t_{o2} for those $\underline{z}$ less than r and a contribution of $\underline{z}/r$ for those $\underline{z}$ greater than r. Then,

$$t_{o2} = \int_0^r p_{\underline{z}}(z)\,dz + \int_r^{y_2} \frac{z}{r}\, p_{\underline{z}}(z)\,dz$$

$$= \lambda \int_0^r (e^{\alpha(y_2 - z)} - 1)\,dz + \frac{\lambda}{r}\left[-e^{\alpha y_2}\left(\frac{1}{\alpha^2} + \frac{z}{\alpha}\right)e^{-\alpha z} - \frac{z^2}{2}\right]_r^{y_2}$$

$$= \lambda\left[-\frac{1}{\alpha}e^{\alpha(y_2 - z)} - z\right]_0^r + \frac{\lambda}{r}\left[-\frac{1}{\alpha^2} - \frac{y_2}{\alpha} - \frac{y_2^2}{2} + \left(\frac{1}{\alpha^2} + \frac{r}{\alpha}\right)e^{\alpha(y_2 - r)} + \frac{r^2}{2}\right]$$

$$= \frac{\lambda}{2\alpha^2 r}[-2\,\alpha r e^{\alpha(y_2 - r)} - 2(\alpha r)^2 + 2\alpha r e^{\alpha y_2} - 2 - 2\alpha y_2 - (\alpha y_2)^2 + 2(1 + \alpha r)e^{\alpha(y_2 - r)} + (\alpha r)^2]$$

$$= \frac{\lambda}{2\alpha^2 r}[2e^{\alpha y_2}(\alpha r + e^{-\alpha r}) - (\alpha y_2)^2 - 2\alpha y_2 - (\alpha r)^2 - 2]$$

$$= \frac{2e^{\alpha y_2}(\alpha r + e^{-\alpha r}) - (\alpha y_2)^2 - 2\alpha y_2 - (\alpha r)^2 - 2}{2\alpha r(e^{\alpha y_2} - \alpha y_2 - 1)}. \tag{23}$$

Next the mean square value of $\underline{z}$ is calculated:

$$\langle \underline{z}^2 \rangle = \int_0^{y_2} z^2\, p_{\underline{z}}(z)\,dz = \int_0^{y_2} z^2 \frac{\alpha(e^{-\alpha z} - e^{-\alpha y_2})}{1 - (1 + \alpha y_2)\,e^{-\alpha y_2}}\,dz$$

$$= \frac{\alpha}{1 - (1 + \alpha y_2)\,e^{-\alpha y_2}} \int_0^{y_2} (e^{-\alpha z} - e^{-\alpha y_2})\,z^2\,dz$$

$$= \frac{\alpha}{1 - (1 + \alpha y_2)\,e^{-\alpha y_2}} \left[-\frac{e^{-\alpha z}}{\alpha^3}((\alpha z)^2 + 2\alpha z + 2) - e^{-\alpha y_2}\frac{z^3}{3}\right]_0^{y_2}$$

$$= \frac{1}{\alpha^2[1 - (1 + \alpha y_2)\,e^{-\alpha y_2}]}\left[2 - e^{-\alpha y_2}\left(\frac{(\alpha y_2)^3}{3} + (\alpha y_2)^2 + 2\alpha y_2 + 2\right)\right]$$

and this value and the value of $\langle \underline{z} \rangle$ from equation (E6) can be substituted into equation (2) to yield.

$$V = n\left\{1 + \mu\left[m + x + (m + r)t_o + \frac{2 - e^{-\alpha y_2}\left(\frac{(\alpha y_2)^3}{3} + (\alpha y_2)^2 + 2\alpha y_2 + 2\right)}{\alpha^3 r\,[1 - (1 + \alpha y_2)\,e^{-\alpha y_2}]}\right]\right\}. \tag{24}$$

APPENDIX F

COMPARISON OF ANALYSIS AND SIMULATIONS – TERMINATION STRATEGY

As discussed in the text and in Appendix E, two assumptions are involved in the derivation of equation (20) which are only approximately correct. The first is the definition of y_2 in accordance with equation (18) when the actual behavior of the simulator, and of practical operating systems, is described by neither equation (18a) nor by equation (18b). The second is the equation of the two probability functions in equation (19).

The latter problem will be considered first. It will be convenient to deal with the integral of the $\underline{z}$ distribution,

$$P^{+}_{\underline{z}}(z) \equiv P(\underline{z} > z) = \int_{z}^{y_2} \frac{\alpha(e^{\alpha(y_2-w)} - 1)}{e^{\alpha y_2} - \alpha y_2 - 1}\, dw$$

$$= \frac{e^{\alpha(y_2-z)} - \alpha y_2 - 1 + \alpha z}{e^{\alpha y_2} - \alpha y_2 - 1}.$$

Table F1 compares the theoretical values of $P^{+}_{\underline{z}}(z)$ with the results of simulation for various values of **M** and **r**. In these simulations, **m** and **x** were set to 0 to remove any uncertainty in the definition of y_2. In this, and the following tables, results derived from simulation are enclosed in parentheses and include the one-sigma confidence limits. Where the confidence limits are omitted, they are less than 1/2 unit in the least significant place displayed in the mean.

Table F1 confirms the prediction that the analytic result is accurate only for large values of **r**, and that for smaller values of **r** the distribution is skewed to the left; i.e., large values of $\underline{z}$ occur less often than predicted by equation (20).

It will be convenient to define a conflict ratio, **c**, as the average number of attempted allocations per successful allocation. This is done because the simulator was not programmed to record P_c directly, and the value of γ determined by simulation cannot be used to determine P_c when $n = 1$, because in that case $\gamma = 1$ for any value of P_c. The conflict ratio, **c**, on the other hand, can be easily determined from the simulator outputs.

Table F1

Theoretical and Simulated Values of $\mathbf{P}^{+}_{\underline{z}}(\mathbf{z})$; $\mathbf{m} = \mathbf{x} = 0$, $\alpha = \mathbf{n} = 1$.

y_2		$\overline{P^{+}_{\underline{z}}(z)}$					
		z = 0.5	z = 1.0	z = 1.5	z = 2.0	z = 2.5	z = 3.0
2.0	Theory	0.452	0.164	0.034			
	r = 10	(0.451 ± 0.001)	(0.163 ± 0.001)	(0.035 ± 0.001)			
	r = 2	(0.451 ± 0.002)	(0.163 ± 0.001)	(0.034 ± 0.001)			
	r = 1	(0.425 ± 0.002)	(0.150 ± 0.001)	(0.030 ±)			
	r = 0.5	(0.407 ± 0.002)	(0.131 ± 0.001)	(0.023 ±)			
	r = 0.2	(0.385 ± 0.002)	(0.109 ± 0.002)	(0.016 ±)			
2.5	Theory	0.501	0.228	0.083	0.017		
	r = 10	(0.501 ± 0.001)	(0.230 ± 0.001)	(0.082 ± 0.001)	(0.018 ±)		
	r = 2	(0.500 ± 0.001)	(0.226 ± 0.001)	(0.082 ± 0.001)	(0.016 ±)		
	r = 1	(0.479 ± 0.002)	(0.205 ± 0.001)	(0.072 ± 0.001)	(0.014 ±)		
	r = 0.5	(0.466 ± 0.001)	(0.190 ± 0.001)	(0.060 ± 0.001)	(0.011 ±)		
	r = 0.2	(0.488 ± 0.003)	(0.171 ± 0.002)	(0.050 ± 0.001)	(0.008 ±)		
3.0	Theory	0.539	0.273	0.123	0.045	0.009	
	r = 10	(0.541 ± 0.001)	(0.274 ± 0.001)	(0.124 ± 0.001)	(0.045 ± 0.001)	(0.008 ±)	
	r = 2	(0.526 ± 0.001)	(0.265 ± 0.001)	(0.120 ± 0.001)	(0.044 ± 0.001)	(0.009 ±)	
	r = 1	(0.517 ± 0.002)	(0.248 ± 0.002)	(0.105 ± 0.001)	(0.036 ± 0.001)	(0.007 ±)	
	r = 0.5	(0.506 ± 0.001)	(0.236 ± 0.001)	(0.096 ± 0.001)	(0.030 ± 0.001)	(0.005 ±)	
	r = 0.2	(0.496 ± 0.003)	(0.223 ± 0.003)	(0.086 ± 0.001)	(0.024 ± 0.001)	(0.003 ±)	
3.5	Theory	0.562	0.303	0.153	0.069	0.025	0.005
	r = 10	(0.562 ± 0.002)	(0.301 ± 0.001)	(0.152 ± 0.001)	(0.070 ± 0.001)	(0.025 ± 0.001)	(0.005 ±)
	r = 2	(0.552 ± 0.002)	(0.295 ± 0.001)	(0.148 ± 0.001)	(0.067 ± 0.001)	(0.024 ± 0.001)	(0.005 ±)
	r = 1	(0.545 ± 0.002)	(0.281 ± 0.001)	(0.136 ± 0.001)	(0.059 ± 0.001)	(0.020 ± 0.001)	(0.004 ±)
	r = 0.5	(0.534 ± 0.002)	(0.271 ± 0.002)	(0.127 ± 0.001)	(0.041 ± 0.001)	(0.016 ±)	(0.003 ±)
	r = 0.2	(0.528 ± 0.002)	(0.258 ± 0.003)	(0.116 ± 0.002)	(0.045 ± 0.001)	(0.012 ±)	(0.002 ±)

In the notation of Table B2, where Q_1 is the number of jobs attempted and Q_3 is the number of jobs completed, $c = Q_1/Q_3$ for the case $n = 1$. It remains, however, to calculate the value of c predicted by equation (20).

Let $P_c(w)$ denote the probability that if $\underline{z} = w$ the next $\underline{y}$ will be greater than $y_2 - w$ and will cause a conflict, and let $Q_c(w) = 1 - P_c(w)$. Now for any particular value of w, the expected number of allocation attempts required for a successful allocation, $c(w)$, can be found by summing the products of the number of attempts and the probability that that attempt will be the first successful one. Thus

$$c(w) = Q_c(w) + 2P_c(w)\,Q_c(w) + 3P_c^2(w)\,Q_c(w) + \ldots$$

$$= \frac{Q_c(w)}{P_c(w)} \sum_{i=1}^{\infty} i\,P_c^i(w) = \frac{Q_c(w)}{P_c(w)} \frac{P_c(w)}{[1 - P_c(w)]^2} = \frac{1}{Q_c(w)}.$$

$Q_c(w)$ is simply

$$Q_c(w) = P(\underline{y} < y_2 - w) = \int_0^{y_2 - w} \alpha e^{-\alpha y}\,dy = 1 - e^{-\alpha(y_2 - w)},$$

and integration of $c(w)$ over the $\underline{z}$ distribution yields

$$c = \int_0^{y_2} \frac{P_{\underline{z}}(w)}{Q_c(w)}\,dw = \frac{1}{e^{\alpha y_2} - \alpha y_2 - 1} \int_0^{y_2} \frac{\alpha[e^{\alpha(y_2 - w)} - 1]}{1 - e^{-\alpha(y_2 - w)}}\,dw$$

$$= \frac{1}{e^{\alpha y_2} - \alpha y_2 - 1} \int_0^{y_2} \alpha[e^{\alpha(y_2 - w)}]\,dw = \frac{e^{\alpha y_2} - 1}{e^{\alpha y_2} - \alpha y_2 - 1}.$$

In Table F2, the theoretical and actual values of c, V, t_c and CE are compared for the same simulations that were used in the construction of Table F1. Again the analysis is found to be correct for large r, but inaccurate for small r.

At small r the actual values of c are considerably larger than the theoretical values because the large $\underline{z}$'s stay in memory longer than average and tend to conflict with an above-average number of other jobs. The large $\underline{z}$'s are smaller in number than predicted, so the actual values of V are substantially less than the theoretical. The actual values of t_c, which are the same as t_{o2} because $n = 1$, are lower than predicted because of the deficiency of large $\underline{z}$'s, but this is not enough to overcome the low values of V, which depend to a substantial extent on $\langle \underline{z}^2 \rangle$, so the actual values of CE are lower than predicted.

Table F2

Theoretical and Simulated Values of V, γ, t_{o2} and CE; $m = x = 0$, $\alpha = \mu = n = 1$.

y_2	r	c	V	$t_{o2} = t_c$	CE
2.0	10	1.455 (1.449 ± 0.005)	1.592 (1.591 ± 0.001)	1.000 (1.000 ±)	0.531 (0.530 ± 0.001)
	2	1.455 (1.456 ± 0.004)	1.785 (1.785 ± 0.002)	1.000 (1.000 ±)	0.595 (0.595 ± 0.001)
	1	1.455 (1.56 ± 0.01)	2.03 (1.97 ± 0.01)	1.025 (1.023 ±)	0.659 (0.642 ± 0.001)
	0.5	1.455 (1.67 ± 0.01)	2.51 (2.32 ± 0.01)	1.195 (1.162 ± 0.001)	0.699 (0.664 ± 0.001)
	0.2	1.455 (1.72 ± 0.02)	3.95 (3.77 ± 0.01)	1.928 (1.756 ± 0.004)	0.683 (0.628 ± 0.001)
2.5	10	1.288 (1.287 ± 0.003)	1.708 (1.709 ± 0.002)	1.000 (1.000 ±)	0.488 (0.488 ± 0.001)
	2	1.288 (1.319 ± 0.007)	1.980 (1.972 ± 0.002)	1.001 (1.001 ±)	0.565 (0.563 ± 0.001)
	1	1.288 (1.38 ± 0.01)	2.32 (2.23 ± 0.01)	1.049 (1.044 ±)	0.632 (0.610 ± 0.001)
	0.5	1.288 (1.44 ± 0.01)	3.00 (2.73 ± 0.01)	1.275 (1.230 ± 0.001)	0.672 (0.634 ± 0.001)
	0.2	1.288 (1.50 ± 0.01)	5.04 (4.19 ± 0.01)	2.160 (1.963 ± 0.006)	0.683 (0.628 ± 0.001)
3.0	10	1.176 (1.184 ± 0.002)	1.808 (1.808 ± 0.001)	1.000 (1.000 ±)	0.452 (0.452 ± 0.001)
	2	1.176 (1.23 ± 0.01)	2.161 (2.136 ± 0.002)	1.003 (1.003 ±)	0.538 (0.532 ± 0.001)
	1	1.176 (1.26 ± 0.01)	2.60 (2.48 ± 0.01)	1.074 (1.064 ±)	0.605 (0.582 ± 0.001)
	0.5	1.176 (1.30 ± 0.01)	3.48 (3.16 ± 0.01)	1.345 (1.296 ± 0.001)	0.647 (0.609 ± 0.001)
	0.2	1.176 (1.33 ± 0.01)	6.12 (5.13 ± 0.02)	2.356 (2.160 ± 0.004)	0.650 (0.594 ± 0.001)
3.5	10	1.118 (1.121 ± 0.002)	1.893 (1.889 ± 0.002)	1.000 (1.000 ±)	0.421 (0.420 ± 0.001)
	2	1.118 (1.16 ± 0.01)	2.322 (2.293 ± 0.003)	1.007 (1.007 ±)	0.512 (0.506 ± 0.001)
	1	1.118 (1.19 ± 0.01)	2.86 (2.73 ± 0.01)	1.097 (1.086 ±)	0.579 (0.558 ± 0.001)
	0.5	1.118 (1.22 ± 0.01)	3.93 (3.57 ± 0.01)	1.405 (1.353 ± 0.002)	0.622 (0.587 ± 0.001)
	0.2	1.118 (1.25 ± 0.03)	7.15 (6.02 ± 0.02)	2.517 (2.317 ± 0.006)	0.593 (0.577 ± 0.001)

For large values of **r**, close agreement was found between the theoretical and simulated results for all values of **n**, **m**, and **x** tried. Equations (20) through (24) were found to agree with the results of simulation for large **r**. The remainder of this discussion will be concerned with simulations conducted with the relatively small **r** = 0.5.

Table F3 displays the results of such a simulation in which **n** was raised to 10 and **m** was raised to 1. The $\underline{z}$ distribution is not shown, since it agreed with that shown in Table F1.

Table F3
Theoretical and Simulated Values of **V**, γ, t_{o2} and **CE**; **m** = 1, **x** = 0, $\alpha = \mu = 1$, **n** = 10.

M	V	γ	t_{o2}	CE
4.0	46.0 (46.6 ± 0.3)	54.3 (72.3 ± 2.6)	1.39 (1.33 ± 0.01)	0.142 (0.112 ± 0.005)
4.5	52.8 (53.1 ± 0.1)	29.0 (37.4 ± 1.0)	1.55 (1.45 ± 0.01)	0.259 (0.211 ± 0.006)
5.0	59.2 (57.9 ± 0.2)	19.9 (24.2 ± 0.2)	1.69 (1.59 ± 0.01)	0.368 (0.308 ± 0.002)
5.5	65.0 (62.9 ± 0.2)	15.7 (18.7 ± 0.1)	1.81 (1.70 ± 0.01)	0.453 (0.383 ± 0.004)
6.0	70.1 (67.3 ± 0.3)	13.5 (15.4 ± 0.1)	1.91 (1.80 ± 0.01)	0.511 (0.466 ± 0.003)
6.5	74.4 (71.4 ± 0.3)	12.2 (13.7 ± 0.1)	1.98 (1.89 ± 0.01)	0.545 (0.481 ± 0.002)
7.0	78.0 (75.0 ± 0.4)	11.4 (12.5 ± 0.1)	2.04 (1.96 ± 0.01)	0.561 (0.508 ± 0.002)
7.5	80.9 (78.0 ± 0.4)	10.9 (11.7 ± 0.1)	2.09 (2.02 ± 0.01)	0.564 (0.520 ± 0.002)
8.0	83.2 (80.9 ± 0.5)	10.6 (11.1 ± 0.1)	2.13 (2.07 ± 0.01)	0.558 (0.526 ± 0.002)
8.5	85.0 (82.7 ± 0.2)	10.4 (10.8 ± 0.1)	2.15 (2.11 ± 0.01)	0.547 (0.519 ± 0.002)
9.0	86.3 (84.0 ± 0.2)	10.3 (10.5 ± 0.1)	2.17 (2.13 ± 0.01)	0.531 (0.510 ± 0.002)
9.5	87.3 (85.4 ± 0.4)	10.2 (10.4 ± 0.1)	2.18 (2.15 ± 0.01)	0.514 (0.497 ± 0.002)
10.0	88.1 (87.1 ± 0.4)	10.1 (10.3 ± 0.1)	2.19 (2.18 ± 0.01)	0.496 (0.485 ± 0.002)

It can be seen that the theory overestimates **V** to a relatively minor extent. This effect, which results from the $\underline{z}$ distribution, is partially compensated by the overestimation of t_{o2}. In the region of maximum **CE**, **V** is overestimated by 3.7%, but $1 + t_{o2}$ (the denominator in the **T** calculations) is overestimated by 2.3%. The net effect on **CE** is toward an overestimation of less than 2%.

The value of γ, on the other hand, is seriously underestimated by the theory, particularly at small values of **M**. This results from the underestimation of P_c, the effect of which is magnified (especially at small **M**, where P_c is sizable) by the substantial value of **n**.

The net effect is that the theory substantially overestimates **CE**, about 6% at M_{opt} and considerably more at lower values of **M**.

If **x** is now raised to 1 but the other parameters held constant, there is a startling change in the situation, as revealed in Table F4. In this case **V** and t_{o2} agree reasonably

Table F4
Theoretical and Simulated Values of V, γ, t_{o2} and CE; $\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{n} = 10$.

M	V	γ	t_{o2}	CE
6.0	56.0 (61.4 ± 0.2)	54.3 (45.2 ± 0.4)	1.39 (1.44 ±)	0.123 (0.160 ± 0.002)
6.5	62.8 (67.6 ± 0.2)	29.0 (25.3 ± 0.3)	1.55 (1.60 ±)	0.226 (0.274 ± 0.003)
7.0	69.2 (72.7 ± 0.2)	19.9 (18.6 ± 0.1)	1.69 (1.72 ±)	0.323 (0.361 ± 0.002)
7.5	75.0 (77.9 ± 0.2)	15.7 (15.1 ± 0.1)	1.81 (1.83 ±)	0.400 (0.428 ± 0.004)
8.0	80.1 (81.9 ± 0.2)	13.5 (13.3 ± 0.2)	1.91 (1.91 ±)	0.454 (0.468 ± 0.004)
8.5	84.4 (84.5 ± 0.6)	12.2 (12.2 ± 0.1)	1.98 (1.98 ±)	0.488 (0.494 ± 0.003)
9.0	88.0 (88.5 ± 0.2)	11.4 (11.4 ± 0.1)	2.05 (2.03 ±)	0.507 (0.511 ± 0.003)
9.5	90.9 (90.7 ± 0.3)	10.9 (11.0 ± 0.1)	2.09 (2.08 ± 0.01)	0.513 (0.511 ± 0.002)
10.0	93.2 (93.4 ± 0.2)	10.6 (10.7 ± 0.1)	2.13 (2.13 ± 0.01)	0.512 (0.511 ± 0.002)
10.5	94.7 (94.8 ± 0.4)	10.4 (10.4 ±)	2.15 (2.14 ± 0.01)	0.505 (0.503 ± 0.001)
11.0	96.3 (96.2 ± 0.2)	10.3 (10.3 ±)	2.17 (2.16 ± 0.01)	0.494 (0.492 ± 0.001)
11.5	97.3 (97.2 ± 0.3)	10.2 (10.2 ±)	2.18 (2.18 ± 0.01)	0.481 (0.479 ± 0.001)
12.0	98.1 (97.5 ± 0.5)	10.1 (10.1 ±)	2.19 (2.18 ± 0.01)	0.468 (0.465 ± 0.001)

well with the analysis, but the nature of the disagreement in γ has been reversed. Now the analysis has seriously overestimated γ, and thereby underestimated **CE**, at low **M**; while at moderate to high **M** the values of γ predicted by the analysis agree with the results of the simulation.

The explanation lies in the other approximation of the analysis, the use of equation (18a), which implies that an **x** must always be saved for both jobs. Recall that the simulator attempts to allocate the memory for the processing of a segment of a job as soon as output of the previous segment is complete, whether or not the processor is immediately available. At high values of **r**, where $\mathbf{t_o}$ is likely to be less than τ, the situation depicted in Fig. F1 will usually pertain. At point A, processing is complete on a segment of the first job, and its **x** is released. The second job now begins processing. At point B, output is complete for the first job, and the simulator attempts to allocate an **x** and a $\underline{\mathbf{y}}$ for its next segment. Since the second job is still being processed, it is occupying an **x** and a $\underline{\mathbf{y}}$; thus both **x**'s are involved in the allocation attempt, and the use of equation (18a) is reasonable.

At low values of **r**, where $\mathbf{t_o}$ is greater than τ, the situation depicted in Fig. F2 is more likely to pertain. As before, the first job begins output at point A, and completes output at point B, at which time an attempt is made to allocate an **x** and $\underline{\mathbf{y}}$ for its next segment. In this case, however, the second job, which began processing at point A, went into the output state and released its **x** at point C, which occurred before point B. Thus only one **x** was involved in the allocation and the use of equation (18b) would be reasonable. At low values of **r**, then, the use of equation (18a) underestimates $\mathbf{y}_2$ by some

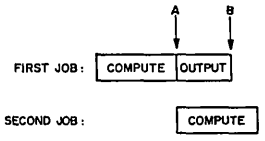


Fig. F1 – Operation at high **r**.

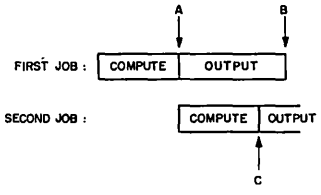


Fig. F2 – Operation at low **r**.

fraction of **m**. A comparison of the simulated γ columns of Tables F3 and F4 reveals that for these cases the fraction is very close to 0.5.

In summary, at large values of **r** the development of equations (20) through (24) and the use of equation (18a) is fully justified and leads to accurate **CE** results. As **r** decreases, two phenomena begin to be effective. The first, which depends only on **r**, leads to an overestimation of **CE** by the theory. The second, which also depends on **x**, leads to an underestimation of **CE**. These effects are most marked at small **M**, are mild in the vicinity of $\mathbf{M}_{opt}$, and do not cause serious mislocation of the value of $\mathbf{M}_{opt}$, at least for **r** as small as 0.5. At smaller values of **r** the multiprogramming of more than two jobs would be attractive. Furthermore, at values of **x** comparable to $\langle \underline{y} \rangle$ the two effects approximately cancel. It is concluded, therefore, that in the region where they would be applied, the equations developed in the text possess sufficient accuracy for engineering purposes.

APPENDIX G

DERIVATION OF EQUATIONS (35) THROUGH (43b)

The probability that a particular $y < y_2$ will cause suspension is

$$P(y > y_2 - \underline{z} \mid y < y_2) = P(\underline{z} > y_2 - y \mid y < y_2)$$

$$= \int_{y_2 - y}^{y_2} \frac{\alpha e^{-\alpha z}}{Q_t} dz = \frac{e^{-\alpha y_2}}{Q_t} (e^{\alpha y} - 1).$$

Since every $y < y_2$ is allocated and becomes a $\underline{z}$, this probability can be multiplied by equation (27) to yield,

$$p_{\underline{z}_s}(z) = \frac{\alpha e^{-\alpha y_2}}{Q_t^2} (1 - e^{-\alpha z}),\ 0 < z < y_2;\ 0 \text{ otherwise.} \tag{35}$$

Similarly

$$P(y < y_2 - \underline{z} \mid y < y_2) = 1 - \frac{e^{-\alpha y_2}}{Q_t} (e^{\alpha y} - 1) = \frac{1 - e^{-\alpha(y_2 - y)}}{Q_t},$$

which when multiplied by (27) yields

$$p_{\underline{z}_s}(z) = \frac{\alpha}{Q_t^2} (e^{-\alpha z} - e^{-\alpha y_2}),\ 0 < z < y_2;\ 0 \text{ otherwise.} \tag{36}$$

It should be noted that equations (35) and (36) are unnormalized probability density functions but that their sum is the normalized probability density function, equation (27).

Integration of equation (35) yields

$$P_s = \int_0^{y_2} p_{\underline{z}_s}(z)\, dz = \frac{e^{-\alpha y_2}}{Q_t^2} \int_0^{y_2} \alpha(1 - e^{-\alpha z})\, dz$$

$$= \frac{e^{-\alpha y_2}}{Q_t^2} (\alpha y_2 + e^{-\alpha y_2} - 1) \tag{37}$$

and

$$P_{\bar{s}} = 1 - P_s = \frac{1 - e^{-\alpha y_2}(1 + \alpha y_2)}{Q_t^2}. \tag{38}$$

These probabilities are the normalizing factors for equations (35) and (36).

It is then straightforward to find

$$t_{os} = \int_0^{y_2} \frac{z}{r} \frac{p_{Z_s}(z)}{P_s} dz$$

$$= \frac{1}{r(\alpha y_2 + e^{-\alpha y_2} - 1)} \int_0^{y_2} \alpha z(1 - e^{-\alpha z})\, dz$$

$$= \frac{1}{r(\alpha y_2 + e^{-\alpha y_2} - 1)} \left[\frac{\alpha z^2}{2} + \left(\frac{1}{\alpha} + z\right) e^{-\alpha z}\right]_0^{y_2}$$

$$= \frac{(\alpha y_2)^2 - 2 + (2\alpha y_2 + z)\, e^{-\alpha y_2}}{2\alpha r(\alpha y_2 + e^{-\alpha y_2} - 1)}, \tag{39}$$

$$t_{o\bar{s}} = \int_0^{y_2} \frac{z}{r} \frac{p_{Z_s}(z)}{P_{\bar{s}}} dz$$

$$= \frac{1}{r[1 - e^{-\alpha y_2}(1 + \alpha y_2)]} \int_0^{y_2} \alpha z(e^{-\alpha z} - e^{-\alpha y_2})\, dz$$

$$= \frac{1}{r[1 - e^{-\alpha y_2}(1 + \alpha y_2)]} \left[\left(\frac{1}{\alpha} + z\right) e^{-\alpha z} + \frac{\alpha z^2}{2} e^{-\alpha y_2}\right]_{y_2}^{0}$$

$$= \frac{2 - e^{-\alpha y_2}[2 + 2\alpha y_2 + (\alpha y_2)^2]}{2\alpha r[1 + e^{-\alpha y_2}(1 + \alpha y_2)]}, \tag{40}$$

and, in analogy with equation (23),

$$t_{o2\bar{s}}(r < y_2) = \int_0^{r} \frac{p_{Z_{\bar{s}}}(z)}{P_{\bar{s}}} dz + \int_r^{y_2} \frac{z}{r} \frac{p_{Z_{\bar{s}}}(z)}{P_{\bar{s}}} dz$$

$$= \frac{1}{1 - e^{-\alpha y_2}(1 + \alpha y_2)} \left\{ \left[e^{-\alpha z} + \alpha z e^{-\alpha y_2} \right]_r^0 + \frac{1}{r} \left[\left(\frac{1}{\alpha} + z \right) e^{-\alpha z} + \frac{\alpha z^2}{2} e^{-\alpha y_2} \right]_{y_2}^{r} \right\}$$

$$= \frac{2\alpha r + 2e^{-\alpha r} - e^{-\alpha y_2} \left[(\alpha r)^2 + 2 + 2\alpha y_2 + (\alpha y_2)^2 \right]}{2\alpha r \left[1 - e^{-\alpha y_2} (1 + \alpha y_2) \right]}. \tag{41}$$

In order to find t_{o2s}, however, it will be necessary to choose one of the models illustrated in Figs. 21 and 22 and to make some approximations.

For the model illustrated in Fig. 21, t_{o2s} will be the sum of t_{os} and the waiting times. The waits suffered by the conflicting job are t_I and t_{III}. If it can be assumed that the conflict is equally likely to occur anywhere in the course of the other job, then $t_I = (1 + t_o)/2$. This is a reasonable approximation if the conflict probability is low enough that synchronization effects can be neglected. t_{III} is simply $r = 1$. In addition to the waits suffered by the conflicting job, the other job waits for t_{II} units of time. It is reasonable to charge this wait to the conflicting job's t_{o2s}. It would not be proper, however, to simply add $t_{II} = 1 + t_{os}$, as the other job has already been charged an average amount of processor wait in the calculation of its t'_{o2}. Thus if the t_{II} is to be added to the delays, a compensating factor equal to the difference between t'_{o2} and t_o must be subtracted. Under the assumption that the probability of suspension is small, it would be reasonable to approximate this compensating factor by the difference between $t_{o2\bar{s}}$ and $t_{o\bar{s}}$, i.e., to assume that the other job did not cause suspension when its current segment was first allocated. Then it is possible to write

$$\begin{aligned} t_{o2s} &= t_{os} + t_I + t_{III} + t_{II} - (t_{o2\bar{s}} - t_{o\bar{s}}) \\ &= 2.5 + 1/2\, t_o + 2t_{os} + t_{o\bar{s}} - t_{o2\bar{s}}. \end{aligned} \tag{42}$$

For the model illustrated in Fig. 22a, the waiting times to be added to t_{os} are t_I for the conflicting job and t_{III} less the compensating factor for the other job. Under the same assumptions that were used in the derivation of equation (42), it is possible to write for this case

$$\begin{aligned} t_{o2s} &= t_{os} + t_I + t_{III} - (t_{o2\bar{s}} - t_{o\bar{s}}) \\ &= 1.5 + 1/2\, t_o + t_{os} + t_{o\bar{s}} - t_{o2\bar{s}}. \end{aligned} \tag{43a}$$

For the model illustrated in Fig. 22b, the waiting time is t_I, which now has a mean value of 0.5. Thus it is possible to write

$$t_{o2s} = 0.5 + t_{os} \tag{43b}$$

subject to the restriction that t_{o2s} cannot be less than r.

APPENDIX H

COMPARISON OF ANALYSIS AND SIMULATION – SUSPENSION STRATEGY

The development of the text did not yield a specific value for t_{o2s} which is used in the calculation of t'_{o2} and t_c but instead found limits on t_{o2s}. It is of interest to determine the width of these limits and to verify that the actual values of t_{o2s} fall within them.

Table H1 compares the results of analysis and simulation for a workload with $x = 0$. In the $1 + t'_{o2}$ and CE columns, two calculated values are given: the left values based on equation (42) and the right values based on equation (43a). The use of equation (43b) would be inappropriate, as this equation implies that the release of an x allows a job to resume, which is impossible for $x = 0$.

For the case $r = 2$, the simulated values generally fall within or very near the limits. The value of $y_{2_{opt}}$ is correctly located by both sets of calculated values. In this vicinity, the limits are separated by about 5%. Note that at small values of y_2 the simulated values fall about midway between the limits but that as y_2 increases the simulated values approach the limit based on equation (42). Although the values based on equation (42) are not reached until y_2 is considerably above $y_{2_{opt}}$, the use of equation (42) would result in an error of less than 2% in the calculation of CE_{max}.

The situation is similar for $r = 0.5$, except that the limits are somewhat further apart because t'_{o2} is larger and therefore constitutes a larger fraction of $1 + t'_{o2}$. In this case, however, the simulated values cross the calculated values based on the equation (42) at moderate values of y_2, and the use of this equation overestimates the value of CE_{max} by about 1%.

Table H2 compares the results of analysis and simulation for a workload with $x = 1$. Since the release of an x may permit the conflicting job to be allocated, the situation described by equation (43b) is possible and three limits are given.

Inspection of Table H2 reveals that equation (43a) yields a good approximation to the simulated results in the vicinity of $y_{2_{opt}}$. Smaller values of y_2 yield results between those predicted by equations (43a) and (43b). In the vicinity of $y_{2_{opt}}$, the separation of the limits is nearly 10%, but the use of equation (43a) leads to errors of not much more than 1%.

Table H1a

Theoretical and Simulated Values of V, γ_t, $1 + t'_{o2}$, and CE for $r = 2$;
$m = 1$, $x = 0$, $\alpha = \mu = 1$, $n = 10$, $M_s = 1$, $y_t = M - 2$.

y_2	V	γ_t	$1 + t'_{o2}$			CE		
			Equation (42)		Equation (43a)	Equation (42)		Equation (43a)
2	34.0 (33.4 ± 0.1)	21.0 (20.8 ± 0.3)	2.47	(2.23 ±)	2.11	0.262	(0.287 ± 0.004)	0.307
3	38.7 (38.1 ± 0.1)	12.7 (12.8 ±)	2.33	(2.21 ±)	2.11	0.436	(0.449 ± 0.001)	0.481
4	41.6 (41.3 ± 0.1)	10.9 (10.8 ± 0.1)	2.21	(2.18 ±)	2.09	0.493	(0.501 ± 0.002)	0.522
5	43.3 (43.2 ± 0.1)	10.3 (10.3 ±)	2.15	(2.13 ±)	2.08	0.489	(0.492 ± 0.001)	0.506
6	44.2 (44.2 ± 0.1)	10.1 (10.1 ±)	2.11	(2.10 ±)	2.08	0.461	(0.461 ± 0.001)	0.467
7	44.6 (44.5 ± 0.1)	10.0 (10.0 ±)	2.09	(2.09 ±)	2.06	0.425	(0.424 ± 0.001)	0.431
8	44.8 (44.7 ± 0.2)	10.0 (10.0 ±)	2.07	(2.08 ±)	2.06	0.393	(0.391 ± 0.001)	0.395

Table H1b

Theoretical and Simulated Values of V, γ_t, $1 + t'_{o2}$, and CE for $r = 0.5$;
$m = 1$, $x = 0$, $\alpha = \mu = 1$, $n = 10$, $M_s = 1$, $y_t = M - 2$.

y_2	V	γ_t	$1 + t'_{o2}$			CE		
			Equation (42)		Equation (43a)	Equation (42)		Equation (43a)
2	53.5 (54.0 ± 0.2)	21.0 (21.2 ± 0.1)	3.72	(3.30 ± 0.01)	3.00	0.275	(0.310 ± 0.002)	0.339
3	68.4 (68.0 ± 0.3)	12.7 (12.7 ± 0.1)	3.67	(3.54 ± 0.02)	3.15	0.489	(0.504 ± 0.002)	0.570
4	78.0 (77.8 ± 0.2)	10.9 (10.9 ±)	3.50	(3.52 ± 0.01)	3.18	0.584	(0.583 ± 0.001)	0.643
5	83.6 (84.0 ± 0.4)	10.3 (10.3 ±)	3.39	(3.44 ± 0.02)	3.20	0.598	(0.591 ± 0.002)	0.634
6	87.0 (86.7 ± 0.5)	10.1 (10.1 ±)	3.30	(3.34 ± 0.01)	3.20	0.580	(0.570 ± 0.002)	0.598
7	88.5 (89.0 ± 0.7)	10.0 (10.0 ±)	3.24	(3.30 ± 0.01)	3.20	0.546	(0.538 ± 0.002)	0.553
8	89.3 (88.8 ± 0.6)	10.0 (10.0 ±)	3.22	(3.24 ± 0.01)	3.20	0.504	(0.497 ± 0.002)	0.507

It can be seen that the actual values of t'_{o2} are somewhat lower in comparison to the equation (43a) values for the higher value of **r**. This reflects the increasing probability of the situation described by equation (43b) as **r** increases, relatively more time is spent by each job in the computing state, and the probability that one job is in the computing state when the other job makes the conflicting allocation request increases.

In summary, although the development of the text did not yield explicit equations for CE, the limits derived are close enough to each other to produce results that are useful in an engineering sense. Furthermore, consideration of the value of **x** should suggest which limit is nearer the actual values for near-optimum memory sizes.

Table H2a
Theoretical and Simulated Values of V, γ_t, $1 + t'_{o2}$, and CE for $r = 2$;
$m = x = \alpha = \mu = 1$, $n = 10$, $M_s = 2$, $y_t = M - 4$.

y_2	V	γ_t	$1 + t'_{o2}$				CE			
			Equation (42)		Equation (43a)	Equation (43b)	Equation (42)		Equation (43a)	Equation (43b)
2	44.0 (43.4 ± 0.1)	21.0 (20.9 ± 0.4)	2.47	(2.03 ±)	2.11	2.00	0.242	(0.294 ± 0.006)	0.284	0.299
3	48.7 (48.2 ± 0.1)	12.7 (12.6 ± 0.1)	2.33	(2.07 ±)	2.11	2.00	0.411	(0.462 ± 0.005)	0.454	0.479
4	51.6 (51.3 ± 0.2)	10.9 (10.9 ± 0.1)	2.21	(2.08 ±)	2.09	2.02	0.476	(0.502 ± 0.004)	0.503	0.521
5	53.3 (53.1 ± 0.1)	10.3 (10.3 ±)	2.15	(2.08 ±)	2.08	2.05	0.481	(0.496 ± 0.002)	0.498	0.505
6	54.2 (54.2 ± 0.2)	10.1 (10.1 ±)	2.11	(2.08 ±)	2.08	2.06	0.462	(0.469 ± 0.001)	0.469	0.474
7	54.6 (54.7 ± 0.1)	10.0 (10.0 ±)	2.09	(2.07 ±)	2.06	2.06	0.434	(0.437 ± 0.001)	0.440	0.440
8	54.8 (54.7 ± 0.2)	10.0 (10.0 ±)	2.07	(2.07 ±)	2.06	2.06	0.407	(0.406 ± 0.001)	0.409	0.409

Table H2b
Theoretical and Simulated Values of V, γ_t, $1 + t'_{o2}$, and CE for $r = 0.5$;
$m = x = \alpha = \mu = 1$, $n = 10$, $M_s = 2$, $y_t = M - 4$.

y_2	V	γ_t	$1 + t'_{o2}$				CE			
			Equation (42)		Equation (43a)	Equation (43b)	Equation (42)		Equation (43a)	Equation (43b)
2	63.5 (63.9 ± 0.3)	21.0 (20.7 ± 0.4)	3.72	(2.68 ± 0.02)	3.00	2.65	0.232	(0.329 ± 0.004)	0.288	0.328
3	78.4 (77.7 ± 0.4)	12.7 (12.7 ± 0.1)	3.67	(3.05 ± 0.02)	3.15	2.84	0.421	(0.510 ± 0.002)	0.490	0.544
4	88.0 (88.1 ± 0.2)	10.9 (10.9 ±)	3.50	(3.21 ± 0.01)	3.18	3.07	0.512	(0.558 ± 0.002)	0.564	0.584
5	93.6 (93.7 ± 0.7)	10.3 (10.3 ±)	3.39	(3.25 ± 0.02)	3.20	3.15	0.536	(0.561 ± 0.002)	0.569	0.577
6	97.0 (97.8 ± 0.6)	10.1 (10.1 ±)	3.30	(3.25 ± 0.01)	3.20	3.17	0.529	(0.537 ± 0.002)	0.546	0.551
7	98.5 (98.6 ± 0.5)	10.0 (10.0 ±)	3.24	(3.24 ± 0.01)	3.20	3.19	0.507	(0.507 ± 0.002)	0.513	0.515
8	99.3 (99.2 ± 0.5)	10.0 (10.0 ±)	3.22	(3.22 ± 0.01)	3.20	3.20	0.474	(0.473 ± 0.002)	0.477	0.477

APPENDIX I

DERIVATION OF FIG. 24

Let the ratio of t_o to r lie between the integers $k - 1$ and k. Then if $N_j \leq k$, the system will be output limited and $t_c/n = (r + t_o)/N_j$. The CE is equal to V/n (which is independent of N_j) divided by the product of t_c/n and C. Therefore CE will vary as the quotient of N_j divided by C. Since C must vary no more strongly than linearly with N_j, (and can vary linearly only if μ is infinite), there is no value of $N_j < k$ that yields a larger CE than that for $N_j = k$.

If $N_j > k$, the system will be processor limited and $t_c/n = r$. Increasing N_j does not decrease t_c and cannot decrease C (it must increase C unless $\mu = 0$); thus there is no value of $N_j > k + 1$ that yields a larger CE than that for $N_j = k + 1$.

Specializing to the case at hand, where $r = y = 1$ and $t_o/r = 1/\mathbf{r}$, if $1/\mathbf{r}$ lies between the integers $k - 1$ and k, maximum CE must result from N_j equal to k or $k + 1$.

The loci of Fig. 24 are obtained by equating the denominators of the CE expressions under the assumption

$$M = 2N_j + 2,$$

which implies that an x will be reserved for the running job and the next job to be processed, which yields

$$\frac{1 + 1/\mathbf{r}}{k}(1 + \mu(2k + 2)) = 1 + \mu(2k + 4)$$

or

$$1 + 1/\mathbf{r} = \frac{k + \mu(2k^2 + 4k)}{1 + \mu(2k + 2)}$$

or

$$\mathbf{r} = \frac{1 + \mu(2k + 2)}{k - 1 + \mu(2k^2 + 2k - 2)}.$$

APPENDIX J

ADDITIONAL TABLES – MULTIPROCESSING

Table J1 compares the results of simulation and analysis for the termination strategy with $N_j = N_p = 2$. For the range of **M** and **r** shown agreement is quite good. At smaller values of **r** the assumption of a simple replacement process in the calculation of the $\underline{z}$ distribution breaks down, and at small values of **M** (and large γ) this can cause significant errors in **CE**; but the range shown is the range of practical interest.

Table J1
Theoretical and Simulated Values of **V**, γ, t_{o2} and **CE**, Termination Strategy;
$\mathbf{m} = \mathbf{x} = \alpha = \mu = 1$, $\mathbf{n} = 10$, $N_j = N_p = 2$.

M	r	V	γ	$t_{o2} = t_o$	CE
7	10	38.8 (39.7 ±)	19.9 (19.6 ± 0.2)	0.072 (0.073 ±)	0.404 (0.414 ± 0.004)
	2	45.8 (45.9 ± 0.1)	19.9 (19.7 ± 0.3)	0.360 (0.366 ± 0.001)	0.371 (0.381 ± 0.006)
8	10	40.5 (40.7 ±)	13.5 (13.5 ±)	0.084 (0.085 ± 0.001)	0.554 (0.555 ± 0.002)
	2	48.9 (49.1 ± 0.1)	13.5 (13.4 ±)	0.419 (0.420 ± 0.001)	0.511 (0.516 ± 0.002)
9	10	41.6 (41.6 ±)	11.4 (11.5 ±)	0.091 (0.091 ± 0.001)	0.607 (0.605 ± 0.002)
	2	51.3 (51.6 ± 0.1)	11.4 (11.4 ±)	0.456 (0.456 ± 0.002)	0.562 (0.563 ± 0.003)
10	10	42.2 (42.1 ± 0.1)	10.6 (10.6 ±)	0.095 (0.095 ± 0.001)	0.606 (0.606 ± 0.001)
	2	53.0 (53.1 ± 0.2)	10.6 (10.7 ±)	0.477 (0.477 ± 0.003)	0.563 (0.562 ± 0.001)
11	10	42.6 (42.6 ±)	10.3 (10.3 ±)	0.098 (0.098 ±)	0.582 (0.583 ± 0.001)
	2	53.9 (53.8 ± 0.1)	10.3 (10.3 ±)	0.489 (0.487 ± 0.002)	0.544 (0.541 ± 0.001)
12	10	42.8 (42.7 ±)	10.1 (10.1 ±)	0.099 (0.099 ±)	0.550 (0.550 ±)
	2	54.4 (54.0 ± 0.2)	10.1 (10.1 ±)	0.494 (0.489 ± 0.002)	0.515 (0.512 ± 0.001)

Optimum operation occurs at a value of **M** sufficient to reduce γ to about 11 and to raise **V** to within a few percent of its limiting value. The effect of the approximation in the $\underline{z}$ distribution is small in this region for $\mathbf{r} > 2$. Smaller values of **r** are of only limited interest, since they would allow efficient operation with only one processor.

Nevertheless, the case $\mathbf{r} = 0.5$ was studied in connection with the suspension strategy. Agreement between the results of simulation and analysis was excellent for **V** and γ, but the analysis yields only limits on t'_{o2}.

Table J2
Theoretical and Simulated Values of $1 + t'_{o2}$, Suspension Strategy;
$m = 1$, $x = 0$, $\alpha = \mu = 1$, $r = 0.5$, $n = 10$, $N_j = N_p - 2$, $M_s = 1$, $y_t = M - N_j$.

M	$1 + t'_{o2}$			
	Equation (48)	Simulated	Equation (49a)	Equation (49b)
4	3.57	(2.94 ± 0.02)	3.06	2.86
5	3.51	(3.26 ± 0.03)	3.13	3.03
6	3.41	(3.23 ± 0.02)	3.18	3.12
7	3.26	(3.18 ± 0.02)	3.10	3.07
8	3.15	(3.11 ± 0.01)	3.06	3.05
9	3.06	(3.08 ± 0.01)	3.02	3.02
10	3.01	(3.03 ± 0.01)	3.01	3.01

Table J2 displays the limits and the simulated values for $x = 0$. In this case the lower limit of equation (49b) is more appropriate, although there is little difference between that limit and the limit of equation (49a) except at the lowest values of M. In the region of optimum operation ($M \approx 7$) the two lower limits are separated by about 1% and the separation between the upper limit and the lower limits is about 6%. Furthermore, the simulated data fall approximately midway between the upper and lower limits.

Tables J3 and J4 display the results of the simulations on which Figs. 39 through 42 are based.

Table J3a
CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$,
$M_s = 1$, $y_t = M - N_j$ ($N_j = 1$).

M	CE		
	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$
3	0.279	0.244	0.225
3.5	0.363	0.321	0.293
4	0.407	0.364	0.333
4.5	0.424	0.385	0.356
5	0.423	0.390	0.364
5.5	0.413	0.386	0.364
6	0.398	0.377	0.359
6.5	0.381	0.367	0.356
7	0.363	0.355	0.347
7.5	0.345	0.342	0.339
8	0.328	0.330	0.331
8.5	0.313	0.318	0.322
9	0.298	0.306	0.315
9.5	0.284	0.295	0.307
10	0.272	0.286	0.300

Table J3b

CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 2$).

M	CE					
	$N_p = 1$			$N_p = 2$		
	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$
4	0.312	0.300	0.293	0.288	0.221	0.189
4.5	0.420	0.401	0.385	0.402	0.308	0.259
5	0.504	0.482	0.463	0.474	0.366	0.305
5.5	0.545	0.523	0.505	0.520	0.406	0.339
6	0.581	0.561	0.546	0.548	0.432	0.361
6.5	0.589	0.576	0.562	0.560	0.450	0.377
7	0.593	0.585	0.577	0.563	0.458	0.388
7.5	0.582	0.581	0.578	0.564	0.466	0.399
8	0.572	0.575	0.579	0.553	0.465	0.401
8.5	0.554	0.554	0.575	0.541	0.463	0.403
9	0.532	0.542	0.565	0.521	0.451	0.399
9.5	0.516	0.535	0.559	0.506	0.444	0.396
10	0.499	0.525	0.554	0.491	0.437	0.396
10.5	0.478	0.507	0.539	0.472	0.424	0.388
11	0.466	0.498	0.534	0.457	0.418	0.383
11.5	0.445	0.479	0.518	0.443	0.410	0.380
12	0.425	0.461	0.503	0.425	0.397	0.373
12.5	0.413	0.450	0.496	0.410	0.387	0.367
13	0.398	0.438	0.485	0.396	0.377	0.360

Table J3c

CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 3$).

M	CE					
	$N_p = 1$			$N_p = 2$		
	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$
6	0.491	0.492	0.493	0.492	0.402	0.347
6.5	0.559	0.560	0.560	0.556	0.458	0.394
7	0.597	0.598	0.600	0.603	0.500	0.432
7.5	0.622	0.630	0.643	0.634	0.532	0.460
8	0.637	0.645	0.657	0.657	0.555	0.485
8.5	0.643	0.660	0.678	0.673	0.578	0.508
9	0.641	0.663	0.683	0.675	0.587	0.519
9.5	0.633	0.662	0.689	0.674	0.594	0.530
10	0.621	0.656	0.689	0.664	0.594	0.535
10.5	0.612	0.648	0.689	0.658	0.595	0.542
11	0.594	0.634	0.684	0.645	0.589	0.545
11.5	0.598	0.622	0.674	0.629	0.582	0.543
12	0.562	0.610	0.662	0.616	0.575	0.546
12.5	0.542	0.592	0.650	0.598	0.564	0.535
13	0.527	0.578	0.644	0.583	0.556	0.530

Table J3d

CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 4$).

M	N_p					
	1			2		
	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$
7	0.468	0.487	0.506	0.493	0.423	0.377
7.5	0.527	0.546	0.565	0.544	0.467	0.416
8	0.572	0.594	0.612	0.587	0.508	0.450
8.5	0.598	0.623	0.646	0.629	0.547	0.486
9	0.611	0.638	0.665	0.663	0.578	0.522
9.5	0.625	0.658	0.689	0.688	0.611	0.549
10	0.628	0.662	0.702	0.708	0.635	0.575
10.5	0.629	0.670	0.714	0.718	0.653	0.595
11	0.629	0.674	0.723	0.728	0.666	0.615
11.5	0.629	0.678	0.734	0.730	0.675	0.629
12	0.619	0.672	0.737	0.725	0.678	0.638
12.5	0.611	0.666	0.737	0.722	0.680	0.645
13	0.595	0.654	0.723	0.718	0.684	0.654
13.5	0.583	0.646	0.720	0.702	0.675	0.650
14	0.569	0.632	0.712	0.687	0.666	0.648

Table J3e

CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 5$).

M	N_p								
	1			2			3		
	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$
9	0.525	0.558	0.592	0.568	0.508	0.463	0.527	0.413	0.348
9.5	0.555	0.592	0.628	0.597	0.538	0.489	0.566	0.452	0.377
10	0.577	0.613	0.655	0.635	0.574	0.525	0.600	0.480	0.403
10.5	0.591	0.634	0.680	0.661	0.605	0.554	0.633	0.514	0.433
11	0.598	0.644	0.694	0.682	0.628	0.580	0.653	0.533	0.453
11.5	0.603	0.653	0.709	0.705	0.661	0.610	0.684	0.568	0.483
12	0.601	0.653	0.715	0.725	0.680	0.639	0.695	0.582	0.502
12.5	0.595	0.650	0.719	0.738	0.696	0.660	0.718	0.608	0.525
13	0.590	0.650	0.720	0.751	0.715	0.684	0.726	0.620	0.540
13.5	0.575	0.636	0.709	0.755	0.725	0.700	0.736	0.635	0.560
14	0.562	0.623	0.704	0.757	0.734	0.714	0.751	0.654	0.580
14.5	0.552	0.618	0.699	0.754	0.738	0.722	0.748	0.658	0.590
15	0.541	0.609	0.693	0.751	0.740	0.730	0.749	0.665	0.599
15.5	0.522	0.589	0.675	0.740	0.736	0.730	0.742	0.664	0.604
16	0.512	0.580	0.670	0.731	0.731	0.730	0.734	0.664	0.606
16.5	0.501	0.570	0.662	0.719	0.722	0.729	0.726	0.663	0.610
17	0.488	0.556	0.650	0.707	0.715	0.725	0.714	0.657	0.608

Table J3f

CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 6$).

M	N_p					
	2			3		
	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$
12	0.655	0.617	0.583	0.628	0.529	0.457
12.5	0.670	0.635	0.603	0.656	0.558	0.483
13	0.695	0.665	0.638	0.685	0.588	0.514
13.5	0.715	0.690	0.668	0.699	0.602	0.537
14	0.724	0.705	0.689	0.729	0.639	0.569
14.5	0.742	0.725	0.710	0.735	0.648	0.581
15	0.756	0.745	0.724	0.746	0.663	0.598
15.5	0.760	0.754	0.750	0.758	0.680	0.615
16	0.762	0.762	0.762	0.767	0.695	0.633
16.5	0.763	0.769	0.773	0.775	0.707	0.651
17	0.754	0.764	0.775	0.782	0.720	0.665
17.5	0.745	0.760	0.775	0.783	0.727	0.677
18	0.738	0.757	0.776	0.775	0.724	0.680
18.5	0.730	0.752	0.778	0.769	0.722	0.683
19	0.723	0.750	0.775	0.764	0.722	0.685
19.5	0.710	0.740	0.773	0.753	0.718	0.687
20	0.695	0.728	0.766	0.744	0.713	0.684
20.5	0.680	0.715	0.755	0.729	0.703	0.679
21	0.665	0.703	0.748	0.720	0.698	0.677

Table J3g

CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$ ($N_j = 7$).

M	N_p					
	2			3		
	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$
14	0.675	0.658	0.642	0.630	0.552	0.493
14.5	0.688	0.675	0.664	0.659	0.582	0.524
15	0.702	0.695	0.685	0.688	0.614	0.554
15.5	0.710	0.708	0.703	0.723	0.650	0.590
16	0.722	0.722	0.723	0.742	0.672	0.614
16.5	0.742	0.747	0.754	0.757	0.690	0.635
17	0.746	0.756	0.767	0.772	0.710	0.658
17.5	0.751	0.768	0.780	0.778	0.722	0.673
18	0.748	0.768	0.788	0.789	0.735	0.690
18.5	0.747	0.770	0.795	0.803	0.757	0.713
19	0.746	0.773	0.803	0.805	0.763	0.723
19.5	0.739	0.770	0.806	0.807	0.766	0.734
20	0.730	0.766	0.803	0.802	0.769	0.738
20.5	0.722	0.760	0.803	0.797	0.769	0.742
21	0.712	0.754	0.799	0.791	0.768	0.745
21.5	0.702	0.748	0.795	0.784	0.764	0.746
22	0.688	0.734	0.788	0.774	0.760	0.746
22.5	0.674	0.722	0.778	0.765	0.754	0.745
23	0.660	0.710	0.769	0.755	0.747	0.743

Table J3h
CE vs M; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$,
$y_t = M - N_j$ ($N_j = 8$).

M	N_p					
	2			3		
	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$	$\mu = 1.0$	$\mu = 0.25$	$\mu = 0.1$
16	0.673	0.673	0.673	0.679	0.615	0.562
16.5	0.687	0.693	0.696	0.704	0.643	0.590
17	0.700	0.710	0.719	0.717	0.660	0.610
17.5	0.713	0.727	0.742	0.734	0.682	0.634
18	0.733	0.753	0.772	0.748	0.698	0.656
18.5	0.733	0.757	0.781	0.767	0.722	0.680
19	0.734	0.762	0.790	0.780	0.739	0.702
19.5	0.730	0.763	0.796	0.798	0.760	0.727
20	0.728	0.763	0.803	0.807	0.774	0.744
20.5	0.723	0.762	0.804	0.806	0.778	0.753
21	0.722	0.764	0.809	0.817	0.792	0.769
21.5	0.714	0.759	0.808	0.815	0.794	0.778
22	0.702	0.749	0.803	0.817	0.801	0.787
22.5	0.695	0.744	0.802	0.813	0.800	0.789
23	0.689	0.740	0.801	0.805	0.797	0.793
23.5	0.673	0.728	0.790	0.800	0.797	0.793
24	0.663	0.718	0.784	0.794	0.794	0.794
24.5	0.650	0.705	0.774	0.785	0.789	0.793
25	0.640	0.697	0.767	0.775	0.783	0.789

Table J3i

CE vs M; m = 1, x = 0, α = 1, r = 0.5, n = 10, M_s = 1, $y_t = M - N_j$ (N_j = 9).

M	N_p								
	2			3			4		
	μ = 1.0	μ = 0.25	μ = 0.1	μ = 1.0	μ = 0.25	μ = 0.1	μ = 1.0	μ = 0.25	μ = 0.1
19	0.702	0.729	0.757	0.720	0.682	0.648			
19.5	0.708	0.740	0.770	0.740	0.705	0.673			
20	0.707	0.740	0.778	0.758	0.727	0.698	0.758	0.674	0.607
20.5	0.710	0.747	0.790	0.777	0.749	0.724	0.775	0.695	0.628
21	0.712	0.753	0.800	0.788	0.765	0.742	0.781	0.704	0.640
21.5	0.703	0.748	0.797	0.802	0.783	0.765	0.792	0.719	0.657
22	0.702	0.748	0.803	0.810	0.794	0.780	0.801	0.730	0.673
22.5	0.695	0.744	0.800	0.815	0.798	0.792	0.818	0.752	0.694
23	0.687	0.740	0.798	0.816	0.808	0.802	0.827	0.764	0.709
23.5	0.679	0.733	0.795	0.815	0.813	0.808	0.833	0.773	0.723
24	0.668	0.724	0.790	0.815	0.815	0.815	0.835	0.778	0.730
24.5	0.656	0.714	0.783	0.811	0.817	0.819	0.836	0.781	0.740
25	0.645	0.703	0.775	0.807	0.815	0.823	0.834	0.785	0.744
25.5	0.633	0.694	0.768	0.803	0.815	0.826	0.827	0.784	0.745
26	0.622	0.684	0.758	0.795	0.810	0.823	0.823	0.783	0.749
26.5	0.610	0.672	0.750	0.787	0.804	0.822	0.820	0.783	0.752
27	0.600	0.663	0.742	0.782	0.803	0.825	0.814	0.781	0.753
27.5	0.590	0.654	0.734	0.773	0.796	0.820	0.806	0.778	0.752
28	0.580	0.645	0.726	0.762	0.788	0.814	0.798	0.770	0.753

Table J4a

CE vs C for $\mu = 1$; $\mathbf{m} = 1$, $\mathbf{x} = 0$, $\alpha = 1$, $\mathbf{r} = 0.5$, $\mathbf{n} = 10$, $\mathbf{M_s} = 1$, $\mathbf{y_t} = \mathbf{M} - \mathbf{N_j}$.

C	CE																		
	$N_j = 1$,	$N_j = 2$		$N_j = 3$		$N_j = 4$		$N_j = 5$			$N_j = 6$		$N_j = 7$		$N_j = 8$		$N_j = 9$		
	$N_p = 1$	$N_p = 1$	$N_p = 2$	$N_p = 1$	$N_p = 2$	$N_p = 1$	$N_p = 2$	$N_p = 1$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 4$
5	**0.407**	0.312																	
6	0.423	**0.504**	0.288																
7	0.398	**0.581**	0.474	0.491															
8	0.363	0.593	0.548	**0.597**	0.492	0.468													
9	0.328	0.572	0.563	**0.637**	0.603	0.572	0.493												
10	0.298	0.532	0.553	0.641	**0.657**	0.611	0.587	0.525											
11	0.272	0.499	0.521	0.621	**0.675**	0.628	0.663	0.577	0.568										
12		0.466	0.491	0.594	0.664	0.629	**0.708**	0.598	0.635	0.527									
13		0.425	0.457	0.562	0.645	0.619	**0.728**	0.601	0.682	0.600									
14		0.398	0.425	0.527	0.616	0.595	0.725	0.590	**0.725**	0.653	0.655								
15			0.396		0.583	0.569	0.718	0.562	**0.751**	0.695	0.695	0.628							
16							0.687	0.541	**0.757**	0.726	0.724	0.685	0.675						
17								0.512	0.751	0.751	**0.756**	0.729	0.702	0.630					
18								0.488	0.731	0.749	**0.762**	0.746	0.722	0.688	0.673				
19									0.707	0.734	0.754	**0.767**	0.746	0.742	0.700	0.679			
20										0.714	0.738	**0.782**	0.748	0.772	0.733	0.717			
21											0.723	0.775	0.746	**0.789**	0.734	0.748	0.702		
22											0.695	0.764	0.730	**0.805**	0.728	0.780	0.707	0.720	
23											0.665	0.744	0.712	0.802	0.722	**0.807**	0.712	0.758	
24												0.720	0.688	0.791	0.702	**0.817**	0.702	0.788	0.758
25													0.660	0.744	0.689	**0.817**	0.687	0.810	0.781
26														0.755	0.663	0.805	0.668	**0.816**	0.801
27															0.640	0.794	0.645	0.815	**0.827**
28																0.775	0.622	0.807	**0.835**
29																	0.600	0.795	**0.834**
30																	0.580	0.782	0.823
31																		0.762	0.814

Table J4b

CE vs C for $\mu = 0.25$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$.

C	CE																		
	$N_j = 1$,	$N_j = 2$		$N_j = 3$		$N_j = 4$		$N_j = 5$			$N_j = 6$		$N_j = 7$		$N_j = 8$		$N_j = 9$		
	$N_p = 1$	$N_p = 1$	$N_p = 2$	$N_p = 1$	$N_p = 2$	$N_p = 1$	$N_p = 2$	$N_p = 1$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 4$
2	**0.364**	0.300																	
2.25	0.390	**0.482**																	
2.5	0.377	**0.561**		0.492															
2.75	0.355	0.585		**0.598**		0.487													
3	0.330	0.575	0.221	**0.645**		0.594													
3.25	0.306	0.542	0.366	**0.663**		0.638		0.558											
3.5	0.286	0.525	0.432	0.656	0.402	**0.662**		0.613											
3.75		0.498	0.458	0.634	0.500	**0.674**	0.423	0.644											
4		0.461	0.465	0.610	0.555	**0.672**	0.508	0.653											
4.25		0.438	0.451	0.578	0.587	**0.654**	0.578	0.650	0.508										
4.5			0.437		0.594	0.632	**0.635**	0.623	0.574										
4.75			0.418		0.589		**0.666**	0.609	0.628										
5			0.397		0.575		0.678	0.580	**0.680**		0.617								
5.25			0.377		0.556		0.684	0.556	**0.715**	0.413	0.665								
5.5							0.666		**0.734**	0.480	0.705		0.658						
5.75									0.740	0.533	**0.745**		0.695						
6									0.731	0.582	**0.762**	0.529	0.722		0.673				
6.25									0.715	0.620	**0.764**	0.588	0.756		0.710				
6.5										0.654	0.757	0.639	**0.768**	0.552	0.753				
6.75										0.665	0.750	0.663	**0.773**	0.614	0.762		0.729		
7										0.664	0.728	0.695	**0.766**	0.672	0.763	0.615	0.740		
7.25										0.657	0.703	0.720	0.754	0.710	**0.764**	0.660	0.753		
7.5												0.724	0.734	0.735	**0.749**	0.698	0.748		
7.75												0.722	0.710	**0.763**	0.740	0.739	0.740	0.682	
8												0.713		0.769	0.718	**0.774**	0.724	0.727	
8.25												0.698		0.768	0.697	**0.792**	0.703	0.765	
8.5														0.760		**0.801**	0.684	0.794	
8.75														0.747		0.797	0.663	**0.808**	
9																0.794	0.645	**0.815**	0.674
9.25																0.783		**0.815**	0.704
9.5																		0.810	0.730
9.75																		0.803	0.764
10																		0.788	0.778

Table J4c

CE vs C for $\mu = 0.1$; $m = 1$, $x = 0$, $\alpha = 1$, $r = 0.5$, $n = 10$, $M_s = 1$, $y_t = M - N_j$.

C	CE																		
	$N_j = 1$	$N_j = 2$		$N_j = 3$		$N_j = 4$		$N_j = 5$			$N_j = 6$		$N_j = 7$		$N_j = 8$		$N_j = 9$		
	$N_p = 1$	$N_p = 1$	$N_p = 2$	$N_p = 1$	$N_p = 2$	$N_p = 1$	$N_p = 2$	$N_p = 1$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 2$	$N_p = 3$	$N_p = 4$
1.4	0.333	0.293																	
1.5	0.364	0.463																	
1.6	0.359	0.546		0.493															
1.7	0.347	0.577		0.600		0.506													
1.8	0.331	0.579		0.657		0.612													
1.9	0.315	0.565		0.683		0.665		0.592											
2.0	0.300	0.554		0.689		0.702		0.655											
2.1		0.534		0.684		0.723		0.694											
2.2		0.503		0.662		0.737		0.715											
2.3		0.485		0.644		0.723		0.720											
2.4			0.189			0.712		0.704											
2.5			0.305					0.693											
2.6			0.361		0.347			0.670											
2.7			0.388		0.432		0.377	0.650											
2.8			0.401		0.485		0.450	0.625											
2.9			0.399		0.519		0.522	0.605	0.463										
3.0			0.396		0.535		0.575	0.585	0.525										
3.1			0.383		0.545		0.615		0.580										
3.2			0.373		0.546		0.638		0.639		0.583								
3.3			0.360		0.530		0.654		0.684		0.638								
3.4							0.648		0.714		0.689		0.642						
3.5									0.730		0.724		0.685						
3.6									0.730		0.762		0.723		0.673				
3.7									0.725		0.775		0.767		0.719				
3.8											0.776		0.788		0.772				
3.9										0.348	0.775		0.803		0.790		0.757		
4.0										0.403	0.766		0.803		0.803		0.778		
4.1										0.453	0.748		0.799		0.809		0.800		
4.2										0.502		0.457	0.788		0.803		0.803		
4.3										0.540		0.514	0.769		0.801		0.798		
4.4										0.580		0.569		0.493	0.784		0.790		
4.5										0.599		0.598		0.554	0.767		0.775		
4.6										0.606		0.633		0.614		0.562	0.758		
4.7										0.608		0.665		0.658		0.610	0.742		
4.8												0.680		0.690		0.656	0.726		
4.9												0.685		0.723		0.702	0.710	0.648	
5.0												0.684		0.738		0.744		0.698	
5.1												0.677		0.745		0.769		0.742	
5.2														0.746		0.787		0.780	
5.3														0.743		0.793		0.802	
5.4																0.794		0.815	
5.5																0.789		0.823	

WORKS CITED*

1. B. Wald, "Multiprocessor Operating Systems," NRL Report 6531, April 6, 1967 (State-of-the-art paper submitted to the University of Maryland, October 1965).

2. B. Wald, "Utilization of a Multiprocessor in Command and Control," **Proceedings of the IEEE**, vol. 54, pp. 1885-1888, December 1966.

3. R. Thompson and J. Wilkinson, "The D825 Automatic Operating and Scheduling Program," 1963 **Proceedings AFIPS SJCC**, pp. 139-146, May 1963.

4. Halstead, M. H., "Machine Independent Computer Programming," Washington:Spartan, 1962.

5. D. Davis, "User's Guide to AOSP Compatible NELIAC Compiler for the AN/GYK-3(V) Computer (Burroughs D825)," NEL, undated.

6. J. Anderson et al., "D825 – A Multiple-Computer System for Command and Control," 1962 **Proceedings AFIPS SJCC**, pp. 86-96, December 1962.

*For a more complete bibliography see Ref. 1.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Naval Research Laboratory Washington, D. C. 20390	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

THE THROUGHPUT AND COST EFFECTIVENESS OF MONOPROGRAMMED, MULTIPROGRAMMED, AND MULTIPROCESSING DIGITAL COMPUTERS

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

An interim report on the problem

5. AUTHOR(S) *(First name, middle initial, last name)*

Bruce Wald

6. REPORT DATE April 17, 1967	7a. TOTAL NO. OF PAGES 132	7b. NO. OF REFS 6
8a. CONTRACT OR GRANT NO. NRL Problem R06-07 b. PROJECT NO. RF-001-08-41 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) NRL Report 6549 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Department of the Navy (Office of Naval Research), Washington, D.C. 20360

13. ABSTRACT

A generalized model of a digital computer system with a workload drawn from a proffered infinite workload was investigated to determine the factors that affect throughput and cost effectiveness to allow optimization of the system parameters. Throughput was defined as the sum of the products of the cost of each component and the proportion of the time each component was beneficially used. Cost effectiveness was defined as the quotient of throughput and system cost, with unity being the ideal quotient.

Each of the jobs constituting the workload consisted of a fixed number of segments, each alternately requiring processing and output. The memory requirements of a segment consisted of a fixed component required at all times, another fixed component required only during processing, and a third component required at all times. This third component was usually assumed to be exponentially distributed, and output time was assumed proportional to this component.

Analytic expressions for throughput and cost effectiveness in terms of the system and workload parameters were obtained for the special cases where the number of jobs simultaneously active did not exceed two and where the number of processors was no greater than the number of active jobs. Simulation was extensively employed to analyze the complex cases.

The cost effectiveness of monoprogrammed systems is limited by the nonutilization of the processor during output by the underutilization of the memory resulting from the necessity of providing a total memory considerably larger than the average job requirements in order to prevent an inordinate amount of job termination. Multiprogramming alleviates both of these problems: more jobs are available to keep the processor busy, and the peak memory demands of the individual jobs are averaged.

(continued)

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0101-807-6801

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
Computers						
Digital computers						
Costs						
Theory						
Design						
Simulation						
Monoprogramming						
Multiprogramming						
Multiprocessing						
Mathematical models						
Optimization						

Abstract (continued)

Two types of multiprogrammed systems were analyzed: systems which terminate jobs in case of memory conflict, and systems which suspend jobs under these circumstances if there is any possibility that memory can be allocated later. The suspension strategy was found to be superior.

Attention was given to the rules that an operating system might use in determining when to start a new job. In general, rules based on the number of active jobs, i.e. rules responsive to the probable distribution of memory allocation requests in the near future, were superior to rules based primarily on instantaneous memory availability. These rules were best modified, however, by consideration of the instantaneous memory situation.

Multiprocessing allows increased multiprogramming and averaging of peak memory requirements beyond the amount feasible with a single processor. As system size increases, the cost effectiveness approaches unity. However, at any given level of system cost, the decision as to whether to invest a finite additional amount of funds in additional memory or in additional processors is strongly affected by the ratio of memory cost to processor cost.