H. A. Anderson, Jr.
R. G. Sargent

Investigation into Scheduling for an Interactive Computing System

Abstract: This paper describes a statistical evaluation of the performance of the swap scheduling algorithm of an interactive computer system and an investigation into foreground-background scheduling to improve system performance. Input traffic, computer service time demands, and system performance were statistically analyzed. Based on the results of these analyses performance enhancements for the system were determined and then evaluated through use of a validated simulation model.

Introduction

The main objective of the work reported here was to integrate the results of a statistical analysis of empirical data recorded for a heavily utilized interactive computing system with theoretical results on optimal foreground-background scheduling in order to improve system performance. The commonly made assumptions used to model interactive computing systems were investigated. A simulation model was developed to investigate scheduling algorithms that might improve system performance by decreasing average response time.

Experimental system

The IBM APL/360 system studied was a dedicated interactive computing system located at the IBM T. J. Watson Research Center [1]. It serviced a population of experienced users. This system was similar to the JOSS (RAND) system [2]. The hardware configuration consisted of an IBM System/360 Model 50 computer with 512K bytes of main memory and one IBM 2314 disk unit. Users communicated with the system through typewriter terminals.

The programs and data of a user resided concurrently in an area called a workspace, the size of which was limited to a maximum of 36,000 bytes. When a user submitted an input to the system, his workspace was flagged as being active and queued for service. A swap scheduling algorithm supervised the loading and unloading of the main memory with workspaces from disk units, which were used for storing the libraries of users' workspaces. Only one disk access could be in progress at a time. Four fixed-size regions in main memory were used for storing active workspaces, each capable of storing the maximum-size workspace.

The software subsystem was comprised of two main components, the supervisor and the interpreter. Evaluation of the swap scheduling algorithm implemented in the scheduler portion of the supervisor was of main concern. (Earlier Hellerman and Ron [3] conducted a limited investigation of the performance of the same system.) The main objective of the swap scheduling algorithm was to minimize disk arm movement in swapping workspaces in and out of main memory. Selecting a workspace to be swapped into main memory was accomplished by searching a list of disk-resident workspaces that was ordered by physical address. The search commenced at the address representative of the current physical location of the disk arm. One of two criteria was applied in selecting an active task for swapping: 1) Select the first active workspace waiting for its first time slice of service; 2) select the first active workspace that is found. The swap scheduler switched between these criteria if in the first search of the list the criterion applied was unsuccessful in finding a workspace to swap.

Active workspaces residing concurrently in main memory were dispatched in round robin fashion to the interpreter. However, newly loaded active workspaces

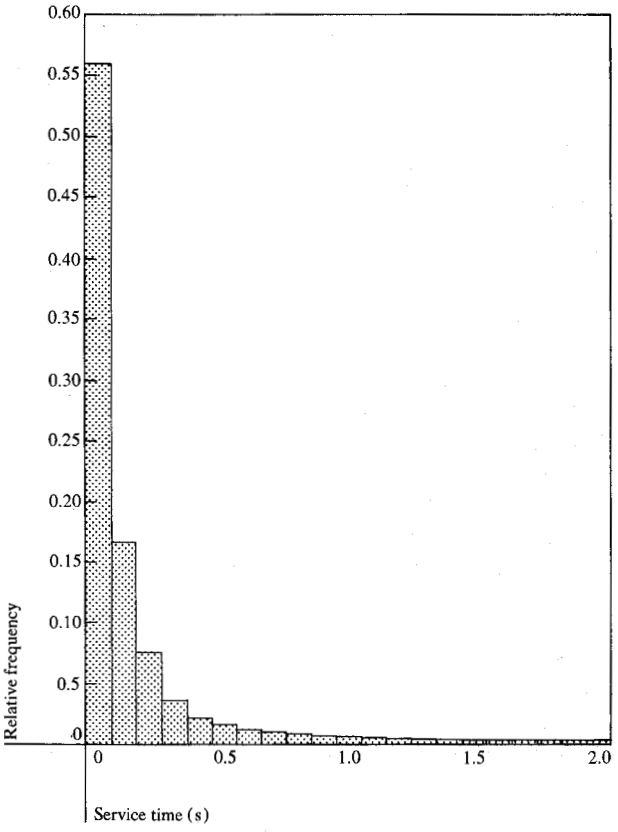


Figure 1 Computer service time histogram for all inputs.

always went to the head of the interpreter dispatching queue. A decision to swap a workspace out of main memory resulted when the four workspace regions in main memory were occupied and one of the following events occurred: 1) An input arrived in the system from a user; 2) a library directory had to be loaded; 3) ten time slices of service had been allocated without 1) or 2) occurring; or 4) a workspace in main memory required no further service and some active workspace resided on the swap disk. If an active workspace had to be swapped out, the one residing the longest time in main memory was selected.

The length of a time slice was a random variable ranging from 0 to 0.2 s. The randomness resulted from the interpreter terminating the current time slice whenever the supervisor in servicing an I/O interrupt had requested it to do so. This time slicing strategy caused the average length of a time slice to decrease as I/O activity in the system increased.

A SIPE-type [4] event-tracing monitor was implemented to record both user and internal system activity. The first analysis of the event traces revealed the fact that user inputs could be categorized into three classes: commands, used to access the workspace libraries; compute transactions, which demand only interpreter servicing; and output-bound transactions, which demand interpreter servicing but generate output at such a high rate that their service frequently has to be interrupted to prevent them from being allocated all available output buffers. The different kinds of inputs generated by the active users during a ten-minute time interval were found to form a composite input traffic that was reasonably approximated by a Poisson process [5].

Statistical properties of computer service times

The computer service times of the users' APL inputs were observed to have a long-tailed cumulative distribution function (c.d.f.) with a large estimated coefficient of variation and positive skew. The exponential distribution was not appropriate to assume for the computer service times. Table 1 summarizes the computer service time statistics for all inputs and for each of the three classes of APL inputs. These statistics indicate that even within each class computer service times varied widely. The commands had the smallest coefficient of variation because of the upper bound on the size of a workspace, which establishes a physical limitation on the amount of processing that they could demand. Output-bound transactions usually demanded much more computer processing than either of the other two classes. Inputs of this class typically requested that a computation be performed on arrays of data and that an array of results be displayed on the keyboard terminal.

Over the four-week monitoring period, the computer service time c.d.f. for all inputs did not vary significantly. In addition, independently of whether the input traffic was low or high, the percentage of the input traffic

Table 1 Statistics of service times for all inputs and for three classes of inputs.

	All inputs	*Compute transactions*	*Commands*	*Output-bound transactions*
Mean (s)	2.064	0.826	0.741	20.910
Variance	387.8	96.092	23.68	4542.9
Std. dev.	19.69	9.803	4.87	67.40
Coeff. of var.	9.54	11.86	6.57	3.23
Skew	21.07	33.69	25.61	6.22
Percentage of inputs (%)	–	83.3	11.4	5.3

in each of the three classes did not vary significantly. On the average 83.3%, 11.4%, and 5.3% of the inputs were compute transactions, commands, and output-bound transactions, respectively.

Numerous statistical characterizations can be made of the c.d.f. One of these is the histogram illustrated in Fig. 1. It indicates the nature of the long tail of the c.d.f.. A sizable majority of all inputs, 77.8%, had computer service times less than 0.2 s, whereas 6% had computer service times that exceeded 2 s. The effects of having such a positively skewed c.d.f. is quantified by the concentration curve, which reflects the percentage contribution made to the mean value by the different percentiles of the c.d.f.. The concentration curve is a plot of

$$Q(t) = \frac{\int_0^t X dF_T(x)}{E[T]} \text{ vs } F_T(t),$$

where $E[T]$ is the mean value of $F_T(t)$, the c.d.f.. The estimated concentration curve in Fig. 2 indicates that the upper 4% of the computer service times contributed 90% of the estimated value of the mean. Therefore the mean value is a sizable overestimate of the service time of the typical input.

In the theory of computer scheduling, the mean residual life time (m.r.l.) of the computer service time c.d.f. has been used to select the algorithm from a well-defined class of foreground-background scheduling algorithms that minimize average response time [6,7]. The m.r.l. is the conditional expectation of the additional service time required by an input that has received X seconds of service,

$$E[T|S = X] = \frac{\int_X^\infty [1 - F_T(t)]dt}{1 - F_T(X)},$$

where T is the additional amount of service required and S is the amount of service received. Figure 3 shows that for the APL system, the computer service time c.d.f. had an increasing m.r.l. This means that, as an input received more service, the expected amount of service remaining *increased*. This observation has a significant effect on scheduling. At any time, newly arrived inputs are expected to require the least amount of service. Therefore, if the performance optimization criterion is to minimize the mean response time, then newly arrived inputs should be serviced ahead of queued inputs that have received some service. This decision rule is included in all the foreground-background scheduling algorithms [8]. The optimality of the FB∞ scheduling algorithm over all other foreground-background scheduling algorithms in obtaining the minimum mean response time when the service times have an increasing m.r.l. was proved by van den

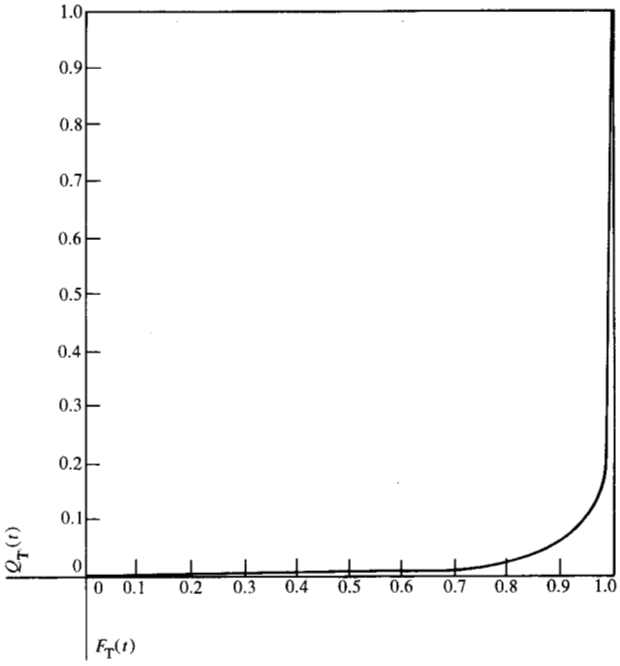
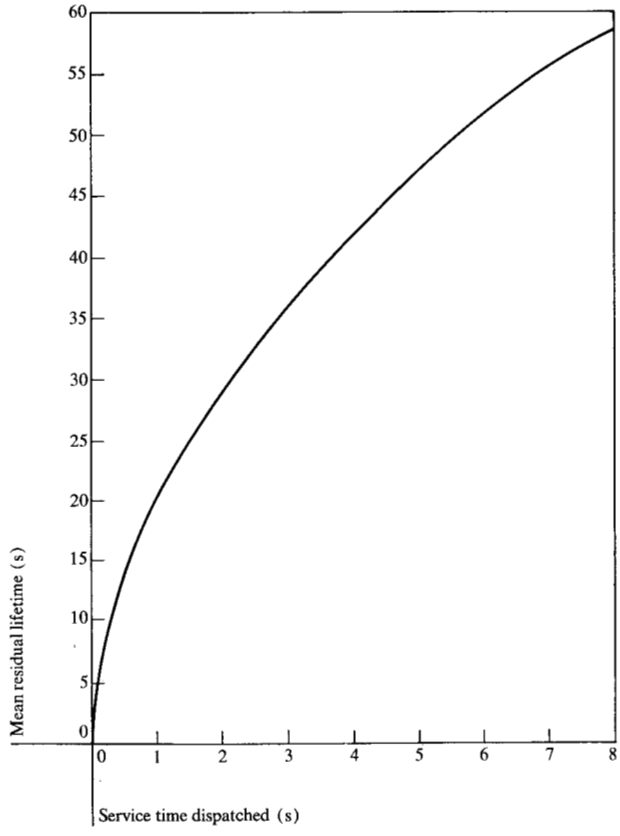


Figure 2 Computer service time concentration curve for all inputs.

Figure 3 Mean lifetimes for all inputs.



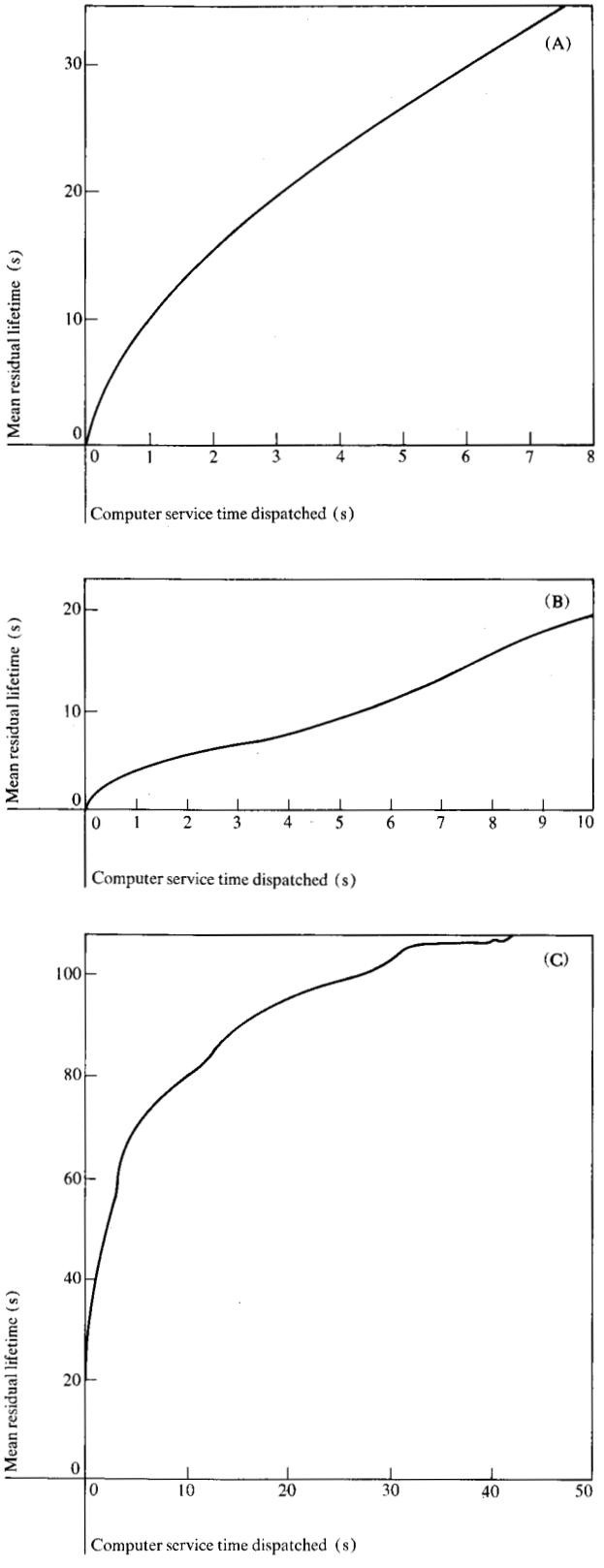


Figure 4 Mean residual lifetimes for each of the three input classes (A) computer transactions, (B) commands, (C) output-bound transactions.

Heever [6]. Therefore, we investigated a FB∞ swap scheduling algorithm for improving system responsiveness. The FB∞ algorithm can be implemented by selecting the first active workspace from the set of active workspaces that have been ordered by the amount of service received, and for those that have received the same amount of service, by the time of their activation. A swap scheduling algorithm should make use of the statistical properties of the service times. This point is further emphasized by the fact that the m.r.l. for each class of input was increasing (Fig. 4).

Performance measurement

At the time we were measuring performance, the system was undergoing long periods of heavy usage with noticeable degradation in responsiveness. The causes of the performance degradation were not well understood. It was hoped that our performance evaluation would uncover how system responsiveness could be improved.

Performance measures for interactive computing systems should reflect how the system appears to the users. In this evaluation, three performance measures were selected: reaction time for all inputs, response time for compute transactions, and response time for commands. Only the characterization of system performance degradation in terms of the average value of reaction time will be discussed in this section. The same methodology was applied to characterizing both response time measures and is discussed in [5]. *Reaction time* is the elapsed time between receipt of an input and dispatching of the first time slice of service to the corresponding workspace. It is a measure of how effective a scheduler is in dispatching service to a newly arrived input.

The average value of reaction time $\overline{R}$ was affected by the workspace swapping rate $\overline{D}$. A high swapping rate was accompanied by high computer processor utilization. These observations are summarized in Table 2. Performance degradation was apparent when $\overline{R}$ and the maximum observed value of reaction time R_{max} exceeded 1 and 4 s, respectively (Fig. 5). The large values for R_{max} revealed that the states of some active disk-resident workspaces were not being examined by the scheduler for relatively long periods of time. This was caused by shortcomings in the swap scheduling algorithm, i.e., 1) the search technique used in selecting a workspace to swap into main memory, and 2) the switching between search criteria. It was concluded that workspaces awaiting their first time slice of service were not consistently obtaining high priority in being swapped into the main memory. Therefore, the majority (70%) of the inputs that required ≤ 0.1s of service were being affected by congestion in the system, resulting in decreased throughput. By improving the swap scheduling algorithm, sys-

tem throughput should be increased. Before a simulation model was developed to investigate improvements in the swap scheduling algorithm, this conclusion was quantified by applying standard data analysis techniques to the performance data.

Data analysis for performance evaluation

In the data analysis, a 3^2 factorial design (f.d.) of experiments [9] was used to investigate the effects that the number of active users and the input rate per user had on the average value of the logarithm of reaction time. The levels of these factors were 26-27, 30-31, and 34-35 users and 1.3, 1.6, and 1.9 inputs/minute/user, respectively. Substantial performance degradation occurred over this region. The recorded data provided eight independent replications for each of the nine experiments. An experiment consisted of selecting a prerecorded ten-minute period and then summarizing system activity and computing the average value of the following logarithmic transformation of reaction time:

$$\overline{R^*} = \frac{1}{N}\sum_{i=1}^{N} \log_{10}(R_i + 0.5),$$

where the R_i's were untransformed observations. The histogram of the transformed reaction times was reasonably approximated by a normal distribution. Therefore, the statistical F-test was used in the analysis of variance (ANOVA) with a reasonable degree of confidence.

From the ANOVA for the 3^2 f.d. it was concluded that the effect of the number of active users was statistically significant at the 5-percent significance level, whereas the other effects and interactions were not statistically significant (Table 3). However, the sum of squares due to error was relatively large. Therefore, although the number of active users had a statistically significant effect, it was not a wholly reliable factor upon

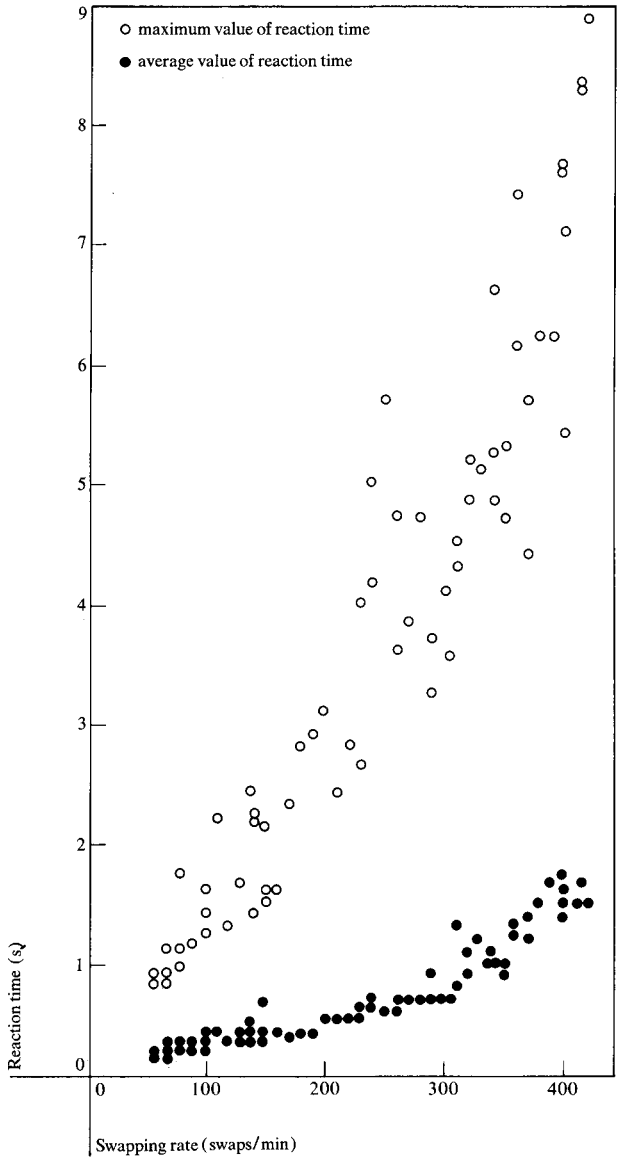


Figure 5 Average and maximum observed reaction time versus swapping rate.

Table 2 Empirical performance data ordered by average reaction time.

Number of active users	*Disk acess rate* (acess/10 min)	*Average reaction* time (s)	*Maximum reaction* time (s)
25	1132	0.30	2.8
30	1388	0.32	1.6
35	2110	0.45	2.3
30	2517	0.63	2.7
25	2717	0.71	3.7
35	2972	0.84	5.4
25	3243	1.20	4.2
35	4149	1.57	8.3
30	3883	1.73	6.2

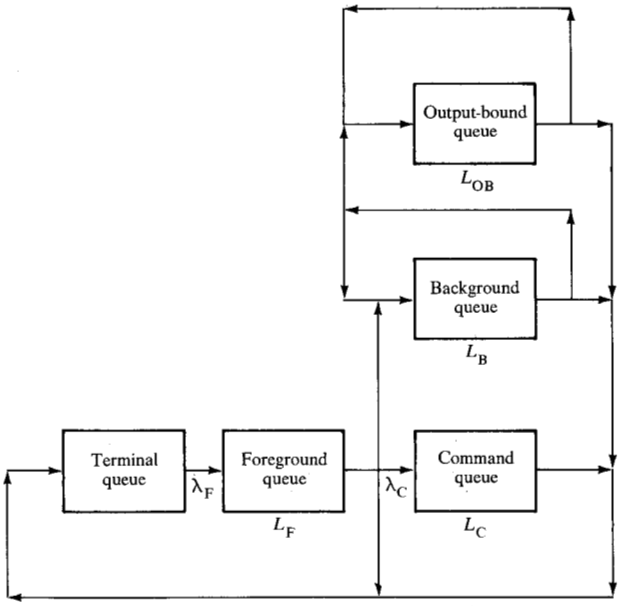


Figure 6 Queueing network idealization of system.

Table 3 Analysis of effects and interactions of the factors U and I on average of logarithm of reaction time.

Average logarithm of reaction time for the 3^2 factorial design of experiments

Number of active users (U)	*Input rate per user (I in inputs/min)* 1.36	1.62	1.88
26 or 27	0.016	−0.078	0.002
30 or 31	0.040	0.073	0.033
35 or 36	0.042	0.072	0.077

Analysis of variance for 3^2 factorial design of experiments

Sum of squares due to	*Mean sum of squares*	*F-ratio*
Linear effect of U	0.085	7.90
Quadratic effect of U	0.001	0.11
Linear effect of I	0.000	0.03
Lin. U × lin. I	0.005	0.42
Quad. U × lin. I	0.001	0.08
Quadratic effect of I	0.003	0.25
Lin. U × quad. I	0.026	2.43
Quad. U × quad. I	0.020	1.82
Error	0.011	

which to predict performance. It did not appropriately account for the workload serviced by the system during an experiment.

Stepwise regression analysis [10] was next used to develop a more general linear model of $\overline{R}^*$, one that took into account the workload. Our objective was to identify the performance factors that had the most significant effect on $\overline{R}^*$, i.e., to diagnose how the swapping algorithm manifested its effect on performance. Many performance factors were investigated, such as swapping rate, processor utilization, traffic intensity, and number of active users. In addition other performance factors were investigated that were derived from conceptualizing the system as a closed network of queues (Fig. 6). The workspaces are the customers circulating around the network. When a workspace is awaiting a user input, the workspace is in the terminal queue. Immediately after the user enters an input, the corresponding workspace leaves the terminal queue and joins the foreground queue to receive its first time slice. If more service is required and the input is not a command, the workspace joins the background queue and is given time slices in a round-robin fashion.

If a workspace in the background queue becomes output-bound, it joins the output-bound queue. In the output-bound queue the workspace receives time slices of service whenever it releases all of the output buffers allocated to it. Those workspaces that contain commands join the command queue after leaving the foreground queue. Workspaces in the command queue are serviced one at a time. When service is completed for a workspace, it always rejoins the terminal queue. For a given number of users (workspaces) the status of the system can be defined by the number of workspaces in the foreground queue, L_F, the background queue, L_B, the output-bound queue, L_{OB}, and the command queue, L_C. The average value of the queue lengths over a sample interval, $\bar{L}_F$, $\bar{L}_B$, $\bar{L}_{OB}$, and $\bar{L}_C$, were estimated for each ten-minute interval analyzed in the 3^2 f.d.

The most general results in queuing theory is Little's formula,

$$E[W] = E[L]/\lambda,$$

where $E[W]$ is the expected time in the queue, λ is the input rate to the queue, and $E[L]$ is the expected length of the queue. If newly arrived inputs are not delayed in obtaining service by inputs that have received some service, then the only significant factor affecting $\overline{R}^*$ is the estimated time spent in the foreground queue,

$$\overline{W}_F = \overline{L}_F/\bar{\lambda}_F,$$

where $\bar{\lambda}_F$ is the estimated input rate to the foreground queue. However, the linear model determined by stepwise regression established that $\overline{W}_F$ and D were both significant factors. The model was

$$\overline{R}^* = -6.47 + 2.42 \times 10^{-3}\, D + 0.41\, \overline{W}_F.$$

The swapping rate D was related to overall congestion within the system. It provided information not only about the level of queuing in the foreground queue, but

also in the other queues. The model indicates that the dispatching of new inputs into service was also substantially affected by the level of congestion in the queues other than the foreground queue. It is worth noting that D had a significant effect on all the performance measures analyzed [5].

Simulation model

Theoretical computer scheduling and performance evaluation results indicated that system responsiveness could be improved but provided no insight about the magnitude of the improvement. A simulation model of the system was implemented to investigate whether significant performance improvements could be achieved by changing the swap scheduling algorithm. The time slicing strategy, interpreter dispatching algorithm, and swap scheduling algorithm were modeled as described in the section on system architecture.

The input traffic to the system was modeled as being generated by a finite population of users. The input insertion times were assumed to have an exponential distribution. The type of input entered by a user was either a compute transaction, an output-bound transaction, or a command, with probabilities 0.833, 0.053, and 0.114, respectively. The service time of an input was sampled from the appropriate c.d.f. In the simulation experiments, performance was measured in terms of both reaction times and the response times of compute transactions requiring ≤ 2 s of service (response time).

The model validation consisted of comparing performance degradation resulting from congestion observed in the real system with that observed in the simulation model. The effects of D, $\overline{L}_F$, and $\overline{L}_B$ on $\overline{R}$ and R_{max}, and on the average value of response $\overline{W}$, were compared. In making these comparisons over the wide range of values for which empirical data were available, six experiments with seven replications each were conducted. In the experiments, the number of active users and the average input insertion times were assigned the following values: 26 and 35 users and 20, 25, and 30 s, respectively.

An inherent difficulty in validating the model was that system performance varied widely for a given number of active users. Statistical analysis previously established that this was due to the wide variation in the workload generated by different user populations of the same size. This caused the model validation methodology used to be more graphical (or descriptive) than mathematically rigorous [12]. The emphasis was on establishing that the same general relationships existed between the performance measures and the performance factors in the model and in the real system.

Figures 7 to 10 summarize graphically the results of the validation. The performance degradation, evidenced in the model in terms of reaction time and response time,

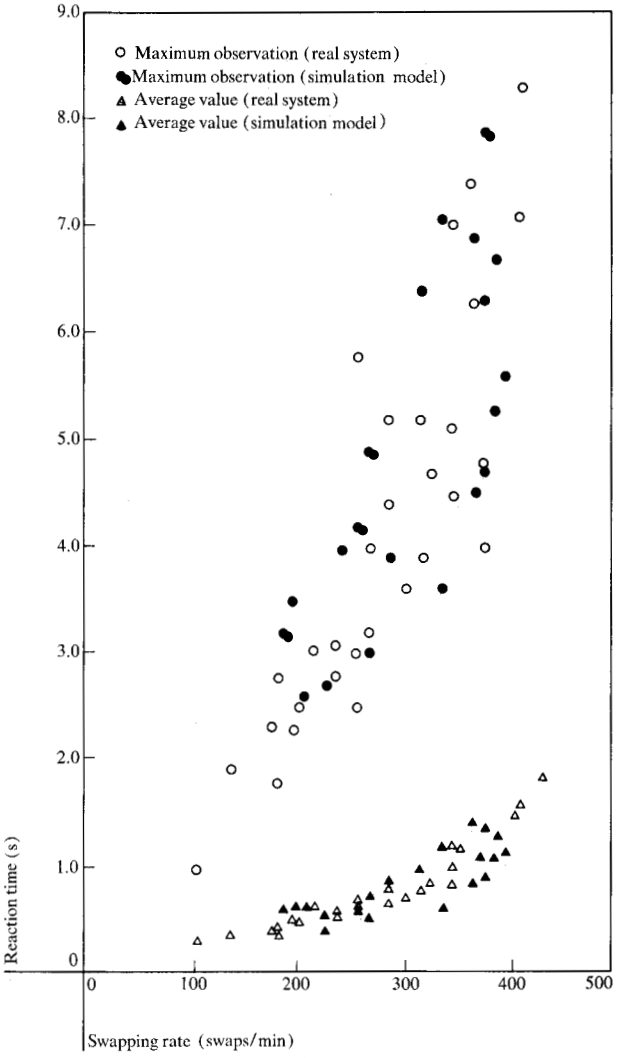


Figure 7 Average and maximum observed values of reaction time versus swapping rate.

was close to that observed for the system. Average reaction time $\overline{R}$ and R_{max} were related to D and $\overline{L}_F$ (Figs. 7 and 8). The fact that R_{max} was very sensitive to changes in D indicated that, during periods of high system utilization, the model lost information, as did the real system about the identity of workspaces in the foreground queue. Average response time $\overline{W}$ was found to be related to D and $\overline{L}_B$ (Figs. 9 and 10).

Modified APL swapping algorithm

After validation of the model, the minimum change to the swap scheduling algorithm that would significantly improve performance was evaluated. This modification

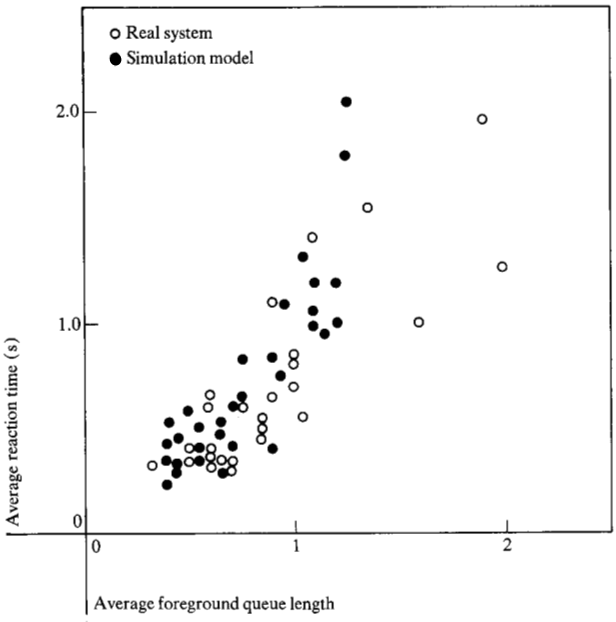


Figure 8 Average value of reaction time versus average length of foreground queue.

consisted of the following changes: 1) The disk-resident workspaces in the foreground queue were swapped into main memory in order of the arrival of their respective user's input. The disk-resident workspaces in the foreground queue were always swapped into the main memory before the disk-resident workspaces in the background queue. 2) The length of a time slice was 0.2 s and was *not* terminated by the completion of an input/output event.

Two experiments with eight replications each were conducted to investigate the modified swapping algorithm. The number of active users for the two experiments were 26 and 35. The main conclusion derived from these experiments was that the statistical properties of reaction time were substantially improved. The mean, the standard deviation, and the maximum observed reaction time were only affected by the congestion in the foreground queue. Figure 11 illustrates that, for the modified swapping algorithm, $\overline{R}$ and $R_{\max}$ were not noticeably affected by D. The reason for this was directly attributable to modification 1). Workspaces in the foreground queue were no longer overlooked by the swap scheduling algorithm, nor were they swapped in some random order. Modification 2) prevented the premature termination of a time slice and resulted in decreasing the swapping rate, since the swap scheduler was executed less often.

In summary, the results in Table 4 for the modified swapping algorithm indicate that the estimated statistics of reaction time did *not* increase noticeably with increased congestion within the system. This resulted in compute transactions that could be serviced in one time slice not being affected by increased congestion, a desirable objective for an interactive computing system. It was therefore concluded that modifications 1) and 2) substantially improved system performance.

Table 4 Comparison of performance of real system with simulation model of original swapping algorithm and modified swapping algorithm for 35 active users.

Empirical data for 35 active users on the real system

			Reaction Time			Response Time
D	$\overline{L}_F$	$\overline{L}_B$	Avg.	Max.	Std. Dev.	Avg.
2560	0.84	1.55	0.46*	2.4	0.42	0.92
2681	0.86	2.06	0.48*	3.2	0.43	1.64
2713	0.96	3.02	0.60	4.7	0.49	0.89
4149	1.96	6.63	1.57	8.3	1.76	4.39
4082	2.02	7.81	1.47	7.1	0.96	4.08
2344	0.83	1.31	0.42*	2.0	0.18	0.57
3809	1.61	4.13	1.17	8.5	0.65	2.22
3109	1.09	2.42	0.63	9.8	0.38	1.06

*Low levels of congestion

Simulation results for 35 active users on system with original swapping algorithm

					Std.	
D	$\overline{L}_F$	$\overline{L}_B$	Avg.	Max.	Dev.	Avg.
3967	1.15	7.07	1.47	5.6	1.06	2.15
2984	0.64	4.06	0.47	5.1	0.41	1.15
2626	0.53	4.20	0.66	2.5	0.58	1.24
4065	1.09	4.76	1.19	7.5	0.86	1.59
3888	1.06	6.26	1.32	6.7	0.99	1.98
2747	0.61	2.17	0.70	4.9	0.66	0.83
3342	0.68	3.97	0.70	4.5	0.59	1.61
3834	1.10	6.55	1.07	7.6	0.70	1.45

Simulation results for 35 active users, a time slice length of 0.2 second, and a modified swapping algorithm

			Reaction time			Response time
D	$\overline{L}_F$	$\overline{L}_B$	Avg.	Max.	Std. Dev.	Avg.
2519	0.60	3.93	0.54	2.6	0.20	1.92
2643	0.70	4.89	0.58	2.2	0.28	1.75
2472	0.52	6.80	0.63	2.4	0.45	0.84
2584	0.64	4.26	0.67	2.4	0.45	0.79
2618	0.78	3.17	0.68	4.0	0.45	1.19
2519	0.60	3.93	0.54	2.6	0.20	1.92
2643	0.70	4.89	0.58	2.2	0.28	1.75
2475	0.53	7.77	0.50	2.1	0.15	0.68

3 × 2 × 2 factorial design

In the next set of simulation experiments, the statistical methodology of factorial design was used to investigate

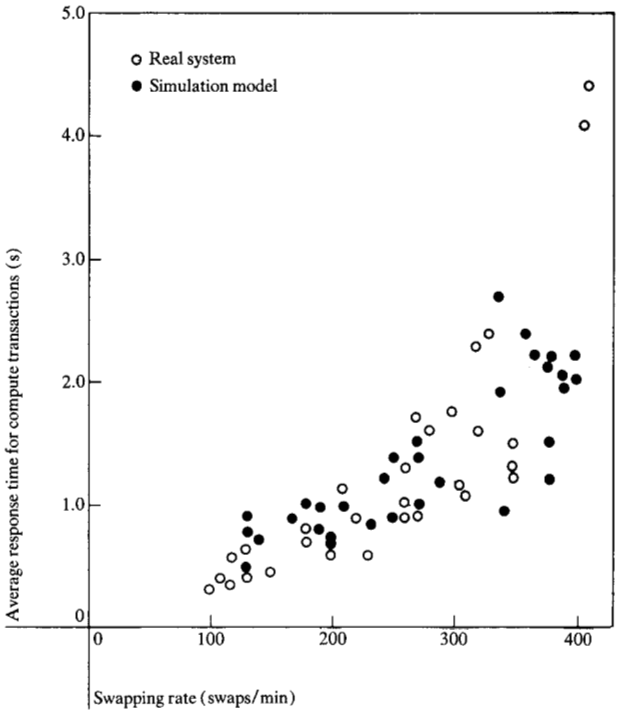


Figure 9 Average value of response time versus swapping rate.

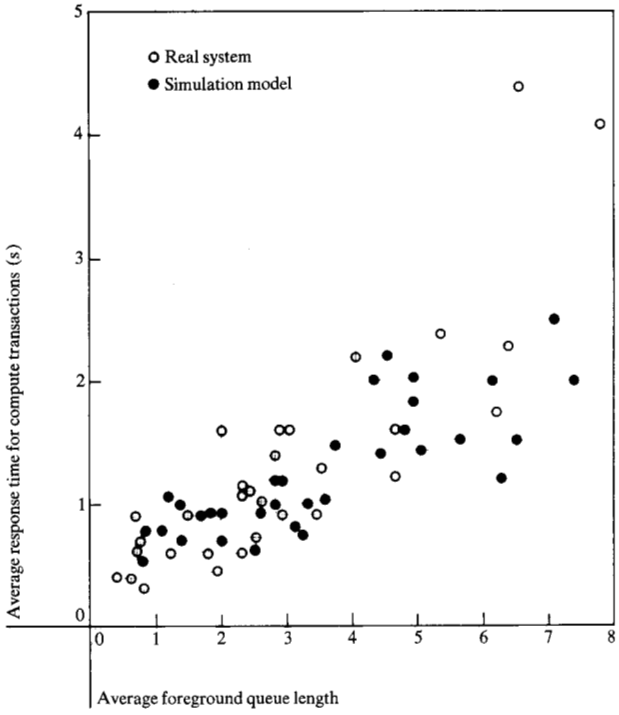


Figure 10 Average value of response time versus average length of background queue.

further scheduling improvements. In these experiments, the performance measures reaction time, response time, average number of queued workspaces, and number of inputs serviced were analyzed. Each one measured a different aspect of performance. The results of the experiments were used to compute a separate ANOVA for each performance measure.

The $3 \times 2 \times 2$ factorial design [9] was used to investigate the effects on performance of the three factors: A, the swap scheduling algorithm; B, the length of a time slice; and C, the number of resident workspaces in the main memory. The three levels of factor A represented the following swap scheduling algorithms: the modified algorithm, the FB_2 algorithm, and the $FB\infty$ algorithm. The two levels of factor B were 0.1 and 0.2 s. The two levels of factor C were 3 and 4 resident workspaces. The modified algorithm was used because it had provided better performance than the original algorithm. The FB_2 algorithm swapped workspaces in the background queue in a round robin fashion into main memory. Therefore, it treated all workspaces in that queue "fairly." The $FB\infty$ algorithm was investigated because the computer service times had increasing m.r.l., and it was expected to minimize the mean response time. The 0.1-s time slice was commensurate with the average time needed to perform a disk access. A shorter time slice would increase the overhead of the system. A time slice longer than 0.2 s would have resulted in more compute transactions being completed in less than one time slice. However, this

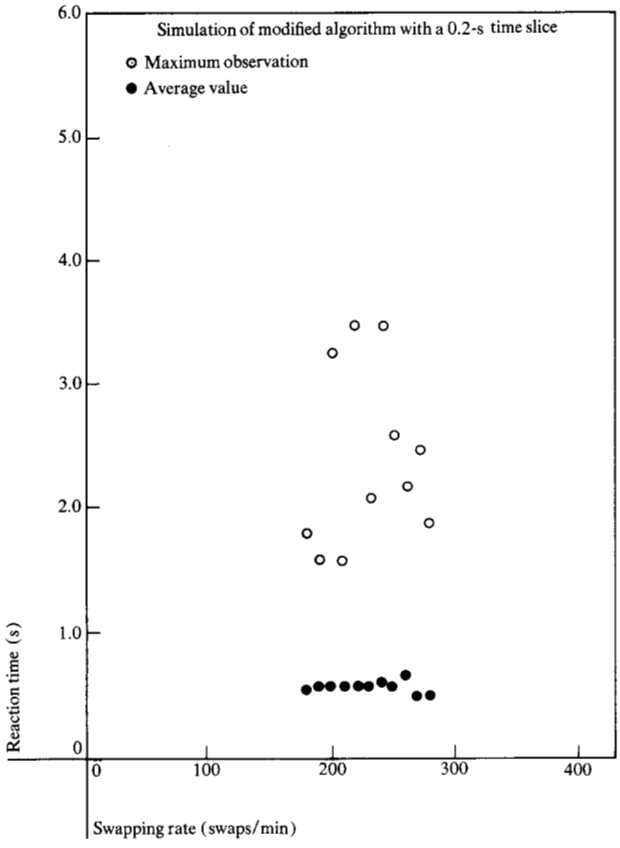


Figure 11 Average and maximum observed value of reaction time versus swapping rate for modified swapping algorithm.

Table 5 Summary of significant effects on average number of queued workspaces.

Average number of queued workspaces for simulation experiments

Scheduling algorithm	*Resident workspaces*	*Length of a time slice (s)* 0.1	0.2
Modified algorithm	3	15.2	15.6
	4	14.9	14.5
FB_2 algorithm	3	14.8	14.8
	4	14.4	14.1
FB∞ algorithm	3	13.2	13.1
	4	12.8	12.9

ANOVA for effects of
A′ – Linear Effect Scheduling Algorithm
A″ – Quadratic Effect of Scheduling Algorithm
B – Linear Effect of Time Slice Length
C – Linear Effect of Number of Resident Workspace

Sum of squares due to	*Mean sum of squares*	*F-ratio*
A′	64.28	135.9
A″	5.21	11.0
B	0.07	0.15
C	6.36	13.5
Error	0.47	

would have caused a decrease in the utilization of the main memory, a critical resource. The system at the time it was monitored permitted four workspaces to reside concurrently in main memory. This number was sufficient to keep the computer processor at a high level of utilization. It was conjectured that, with the appropriate swap scheduling algorithm, three resident workspaces would permit satisfactory performance to be achieved. Hellerman and Ron [3] had observed in their simulation experiments of the system that going from three to four resident workspaces did not result in an appreciable gain in the number of users the system could support.

Randomized blocking [9] of the factorial design was achieved by generating the same set of eight workloads in each experiment. These eight workloads were generated for each of the twelve experiments. Therefore, there were eight replications of each experiment, i.e., eight simulated ten-minute time intervals of system operation. In the simulation experiments, a particular workload was modeled as being that of a population of 35 active users, each having an exponentially distributed input insertion time with an average value of 20 s. The 35 active users were selected because the system frequently serviced that many users with degraded performance. We conjectured that it was possible to significantly improve system performance when servicing 35 users.

Results of the 3 × 2 × 2 factorial design

The FB∞ algorithm performed as theoretically predicted. It had the smallest average number of workspaces queued in the system. This is shown in Table 5, where the average number of queued workspaces was 13, 14.5, and 15 for the FB∞, FB_2, and modified algorithms, respectively. From the ANOVA (Table 5), it was concluded that the effect of factor C on the average number of queued workspaces was not as significant (F-ratio of 13.5) as factor A (F-ratio of 135.9). However, four resident workspaces consistently resulted in fewer queued workspaces. In the ANOVA, the computed F-ratios were used to judge which effects and interactions were significant, albeit the normal assumption for the error term for each observation was not appropriate. An effect or an interaction was *judged* to be significant if it had a mean sum of squares term that was at least an order of magnitude larger than the value of the mean sum of squares due to error.

The number of inputs serviced over a ten-minute interval from a fixed size population of active users is a measure of the throughput of the system. Again, the same effects and interactions were significant for this performance measure (Table 6). As the average queue length decreased, throughput increased. An increase in throughput from approximately 560 to 580 inputs resulted when the modified algorithm was replaced by the FB_2 algorithm. An increase from approximately 580 to 620 inputs resulted when the FB_2 algorithm was replaced by the FB∞ algorithm. The FB∞ algorithm permitted users to accomplish more computing that required "reasonable" amounts of servicing than did either of the other two algorithms.

The ANOVA (Table 7) indicated that the average value of reaction time was not significantly affected by factor A. This was expected, since all three algorithms treated workspaces in the foreground queue equivalently in allocating main memory. The small differences that did occur were attributed to the differences in computer and disk utilization. Factor B had a very significant effect on the statistics of reaction time (an F-ratio of 1811), because it affected the number of disk accesses; i.e., the smaller the time slice, the higher the disk accessing rate. When the length of a time slice was 0.1 s, $\overline{R}$ was approximately 0.40 s. When it was 0.2 s, $\overline{R}$ was approximately 0.55 s. Both factor C and its interaction with factor B had a significant effect on $\overline{R}$. They had F-ratios of 88.5 and 25.8, respectively. Average reaction time $\overline{R}$ was typically 0.02 to 0.05 s longer in the experiments involving four resident workspaces than it was for the other experiments. There was a small but consistent difference due to the increased computer utilization that occurred when there were four resident workspaces. Thus, based on the ANOVA for $\overline{R}$ it was concluded that

Table 6 Summary of significant effects on average number of inputs serviced in a sample interval (throughput). Average number of inputs serviced for simulation experiments (a) and ANOVA for most significant effects (b).

(a)

Scheduling algorithm	*Resident workspaces*	*Length of a time slice* 0.1	0.2
Modified algorithm	3	553.5	549.5
	4	571.3	564.9
FB_2 algorithm	3	581.3	574.5
	4	588.9	586.5
FB∞ algorithm	3	623.5	622.0
	4	631.8	627.4

(b)

Sum of squares due to	*Mean sum of squares*	*F-ratio*
A′	70490.0	272.1
A″	2214.1	8.5
C	2937.1	11.3
Error	259.1	

Table 7 Summary of significant effects and interaction on average reaction time.

Average reaction time for simulation experiments

Scheduling algorithm	*Resident workspaces*	*Length of a time slice* 0.1	0.2
Modified algorithm	3	0.40	0.55
	4	0.43	0.61
FB_2 algorithm	3	0.41	0.55
	4	0.43	0.60
FB∞ algorithm	3	0.42	0.56
	4	0.43	0.62

ANOVA for most significant effect and interaction

Sum of squares due to	*Mean sum of squares*	*F-ratio*
B	0.5932	1811.0
C	0.0290	89.0
BC	0.0084	26.0
Error	0.0003	

Table 8 Summary of significant effects and interactions on average response time.

Average response time for simulation experiments

Scheduling algorithm	*Resident workspaces*	*Length of a time slice* 0.1	0.2
Modified algorithm	3	1.35	1.39
	4	1.25	1.30
FB_2 algorithm	3	1.41	1.45
	4	1.34	1.28
FB∞ algorithm	3	0.75	0.85
	4	0.82	0.95

ANOVA for most significant effects and interactions

Sum of squares due to	*Mean sum of squares*	*F-ratio*
A′	3.713	249.7
A″	1.736	116.8
A′C	0.132	8.9
error	0.015	

compute transactions requiring less than 0.1 s of service were serviced with no performance degradation apparent to the users for the three swapping algorithms (Table 7).

Factor A had a dominant effect on average response time, $\overline{W}$ (Table 8). The ANOVA found the linear and quadratic effects, A′ and A″, both to be very significant. They had F-ratios of 249.7 and 116.8, respectively. The experiments involving the FB∞ algorithm had the smallest average values and smallest standard deviations for response time. Average response time $\overline{W}$ was consistently larger for the other two algorithms, a direct consequence of the decrease in the average number of queued workspaces achieved by the FB∞ algorithm. The interaction of factors A and C was significant. The smallest values for $\overline{W}$ were achieved in the experiments with the FB∞ algorithm, three resident workspaces, and a 0.1-s time slice. In fact, the FB∞ algorithm consistently had the smallest $\overline{W}$ when there were three resident workspaces, because the round robin interpreter dispatching algorithm began to affect $\overline{W}$ more significantly when there were four resident workspaces.

To better evaluate the effects of the three factors on the distribution of response time, the total number and the percentage of compute transactions whose service was completed in ≤ 5 s, ≤ 10 s, and ≤ 15 s were tabulated (Table 9). The improvement in performance achieved by the FB_2 and the FB∞ algorithms over the modified algorithm in terms of the distribution of response time is evident in these tables. Both algorithms resulted in a greater number of compute transactions being serviced with response times under 5, 10, and 15 s than the modified algorithm. However, in terms of the percentage of compute transactions with these response

Table 9 Cumulative histogram of response time for simulation experiments.

Total number (percentage) of compute transactions requiring ≤ 2 s of service with response times ≤ 5 s

	Resident workspaces	Length of a time slice 0.1	0.2
Modified Algorithm	3	3433 (0.952)	3454 (0.965)
	4	3571 (0.958)	3571 (0.972)
FB_2 Algorithm	3	3553 (0.937)	3573 (0.954)
	4	3641 (0.946)	3711 (0.968)
FB∞ Algorithm	3	4004 (0.985)	4039 (0.989)
	4	4055 (0.985)	4039 (0.988)

Total number (percentage) of compute transactions requiring ≤ 2 S of service with response times ≤ 10 S

	Resident workspaces	Length of a time slice 0.1	0.2
Modified Algorithm	3	3505 (0.972)	3502 (0.978)
	4	3644 (0.977)	3609 (0.982)
FB_2 Algorithm	3	3634 (0.959)	3598 (0.960)
	4	3710 (0.964)	3732 (0.973)
FB∞ Algorithm	3	4050 (0.998)	4049 (0.997)
	4	4152 (0.997)	4076 (0.997)

Total number (percentage) of compute transactions requiring ≤ 2 s of service with response times ≤ 15 s

	Resident workspaces	Length of a time slice 0.1	0.2
Modified Algorithm	3	3544 (0.982)	3524 (0.984)
	4	3675 (0.985)	3629 (0.988)
FB_2 Algorithm	3	3657 (0.965)	3662 (0.978)
	4	3782 (0.983)	3782 (0.986)
FB∞ Algorithm	3	4056 (0.999)	3629 (0.988)
	4	4107 (0.998)	4081 (0.999)

times, the FB_2 algorithm did not offer a significant advantage over the modified algorithm. The superiority of the FB∞ algorithm over the other algorithms in terms of its response time characteristics was plainly evident.

Conclusions of the 3 × 2 × 2 factorial design

The following conclusions were derived from the analysis of the results of the 3 × 2 × 2 factorial design simulation experiments:

1. The FB∞ algorithm consistently had the smallest average number of workspaces queued in the system, or equivalently, the smallest average response time. It had the highest throughput. If performance is measured in terms of the response time of compute transactions requiring ≤ 2 s of service, then the FB∞ algorithm, with a time slice of 0.1 s and three resident workspaces, gave the best performance servicing 35 users. Average reaction time and average response time were 0.42 and 0.75 s, respectively. This algorithm under these conditions performed better than either of the other algorithms using four resident workspaces. The FB∞ alrogithm has the additional benefit of degrading the service of an input based on its computer service time demand.
2. The FB_2 algorithm consistently attained a higher throughput than the modified algorithm but at the expense of slightly lower computer processor utilization. The FB_2 algorithm had no other advantage over the modified algorithm.

Other simulation experiments were conducted to determine if the FB∞ algorithm retained its advantages over the modified algorithm (the minimum change) when the system was moderately loaded and when it was heavily loaded with the number of resident workspaces reduced to two. In both cases the FB∞ algorithm was superior. The detailed results are discussed in [12].

References

1. L. Breed and R. Lathwell, "The Implementation of APL/360," *Proc. ACM Symp. on Interactive Systems for Experimental and Applied Mathematics*, Academic Press, p. 390-399 (1968).
2. G. Bryan, "JOSS 20,000 Hours at the Console, A Statistical Summary," *AFIPS Conf. Proc., FJCC 1967* **31,** p. 769-777.
3. H. Hellerman and Y. Ron, *A Time-Sharing System Simulation and Its Validation, IBM New York Scientific Center Technical Report No. 320-2984*, April 1970.
4. W. Denniston, "SIPE a TSS/360 Software Measurement Technique," *Proc. 24th National Conference ACM*, p. 229-245 (1969).
5. H. A. Anderson and R. G. Sargent, "A Statistical Evaluation of the Scheduler of an Experimental Interactive Computing System," *Statistical Computer Performance Evaluation*, Academic Press, New York, 1972, p. 73-98.
6. R. van den Heever, "Computer Time-Sharing Priority Systems," Ph.D. Dissertation, University of California (Berkeley), September 1969.
7. K. Sevick, "The Use of Service Time Distributions in Scheduling," Ph.D. Dissertation, University of Chicago, August 1971.
8. E. Coffman and L. Kleinrock, "Computer Scheduling Methods and Their Countermeasures," *AFIPS Conf. Proc., SJCC* 1968 **32,** p. 11-21.
9. *The Design and Analysis of Industrial Experiments*, O. Davies (Ed.), Oliver and Boyd, London, 1965.
10. N. Draper and H. Smith, *Applied Regression Analysis*, John Wiley & Sons, New York, 1966.
11. T. Naylor, *Computer Simulation Techniques*, John Wiley & Sons, New York, 1966.
12. H. A. Anderson, Jr., *An Empirical Investigation into Foreground Background Scheduling for an Interactive Computing System, IBM Research Report RC 3941*, Yorktown Heights, New York, July 1972.

Received July 20, 1973

Dr. Anderson is located at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598. Dr. Sargent is in the Department of Industrial Engineering and Operations Research, Syracuse University, Syracuse, New York 13210.