Simulation Studies of Predictive Scheduling

A. S. Noetzel

This work was supported by
National Science Foundation Grant GJ-39658

Technical Report No. 37

Department of Computer Sciences

The University of Texas at Austin

Austin, Texas 78712

July, 1974

TABLE OF CONTENTS

ABSTRACT

This report presents the results of simulation studies of a class of resource scheduling algorithms called predictive algorithms. These algorithms base their decisions on available data predicting the resource utilization characteristics of the user jobs. The predictive memory scheduling algorithms are BAL (which selects jobs to balance processor and I/O requirements) and Shortest-Time-First (STF). The predictive CPU scheduling algorithms are Shortest-Burst-Time (SBT), Longest-Burst-Time (LBT), and Shortest-Burst-to-I/O (SBIO). Each combination of the predictive memory algorithms, and the predictive CPU scheduling algorithms plus three non-predictive algorithms was run in simulation. The results of all thirty combined scheduling algorithms are presented for comparison in terms of CPU and I/O utilization, degree of multiprogramming, and throughput.

The results show the superiority of the predictive algorithms; and the particular effectiveness of the predictive CPU scheduling algorithm in maintaining high CPU utilization.

The best CPU shceduling algorithm (SBIO) was run with the predictive data available to it being only partially correct. This algorithm is superior to the best of the standard CPU shceduling algorithms as long as the predictive data is greater than approximately seventy per cent correct.

I. Introduction

Scheduling disciplines for computer systems have been studied both analytically and by simulation modeling. The analytical studies, although restricted to assumptions that are considerable approximations to the complexities of real computer systems, serve to elucidate relationships and tradeoffs between design parameters and provide order-of-magnitude estimates of significant variables. They are useful guides to the more detailed (simulation and experimental) studies necessary in the design of complex systems.

In this report, the expected benefit of a particular class of scheduling algorithms--called predictive algorithms--is investigated through simulation modeling. A predictive algorithm is one which bases its decisions on available (complete or partial) information about the future characteristics of the job being scheduled. These algorithms have not been studied extensively in queueing theory, and in fact, the general case of scheduling from partially correct data appears quite difficult. The usefulness of this class of algorithms is therefore investigated in simulation before the complete analysis is attempted.

The simulation model is not overly detailed, since its purpose is to obtain an indication of benefit of predictive data in scheduling, and not to fine-tune the system. Hence, although it is based on the CDC 6600 system (because extensive measurements of that system were available for the job-characteristic distributions, and for validation of the model) the results should be applicable to other (non-virtual memory) systems as well. Comparisons of predictive and nonpredictive algorithms were made for both CPU and memory-scheduling algorithms. Also, a class

of memory-scheduling algorithms in which the predictability of the jobs processing time was known with probability $p<1$, was studied.

The results show the predictive algorithms are superior to non-predictive algorithms in most cases, and their advantage is greater in CPU scheduling algorithm than in memory scheduling, at least in terms of the measure of CPU utilization. Furthermore, memory-scheduling algorithms that use partially complete predictive data require data that is at least sixty to seventy per cent reliable in order to achieve superiority over similar non-predictive algorithms.

II. The Simulation Model

The simulation model used to study predictive scheduling algorithms has been adapted from a model previously used to study standard (nonpredictive) scheduling algorithms. (See reference 5).

The system used as a basis for the simulation study is the CDC 6600 computer system. It is briefly described as follows: The 6600 has one main processor, which executes user jobs in main memory, and ten peripheral processors (PP's) which perform operating system service functions, including I/O, for the user programs. Main memory is 128K of sixty-bit words; each PP has a 4K memory of twelve-bit words. The main processor can be interrupted, but not the PP's. Each PP can gain access to each of the twelve I/O channels, but no I/O is possible directly from a data channel to main memory. Each user job is loaded into a continuous set of addresses in main memory called the field length
base re-
of the job. Memory addressing is performed through an automatic base register, enabling memory management by core compaction and preemption from main memory. The backup memory is a 512K-word extended core storage (ECS), with a cycle time that is effectively that of main memory.

The system being simulated is represented in the form of a queueing network in figure one. Omitted from this representation of the 6600 computer system are the ten peripheral processors, as well as all the data channels except the four channels that have the system disk units. This is because detailed measurements of system utilization have shown that it is the four system disk channels that are the primary I/O bottleneck. (Reference 4). The PP's and the remaining channels will have sufficient availability so that under every scheduling algorithm, queueing for these resources will not impede progress of the user jobs.

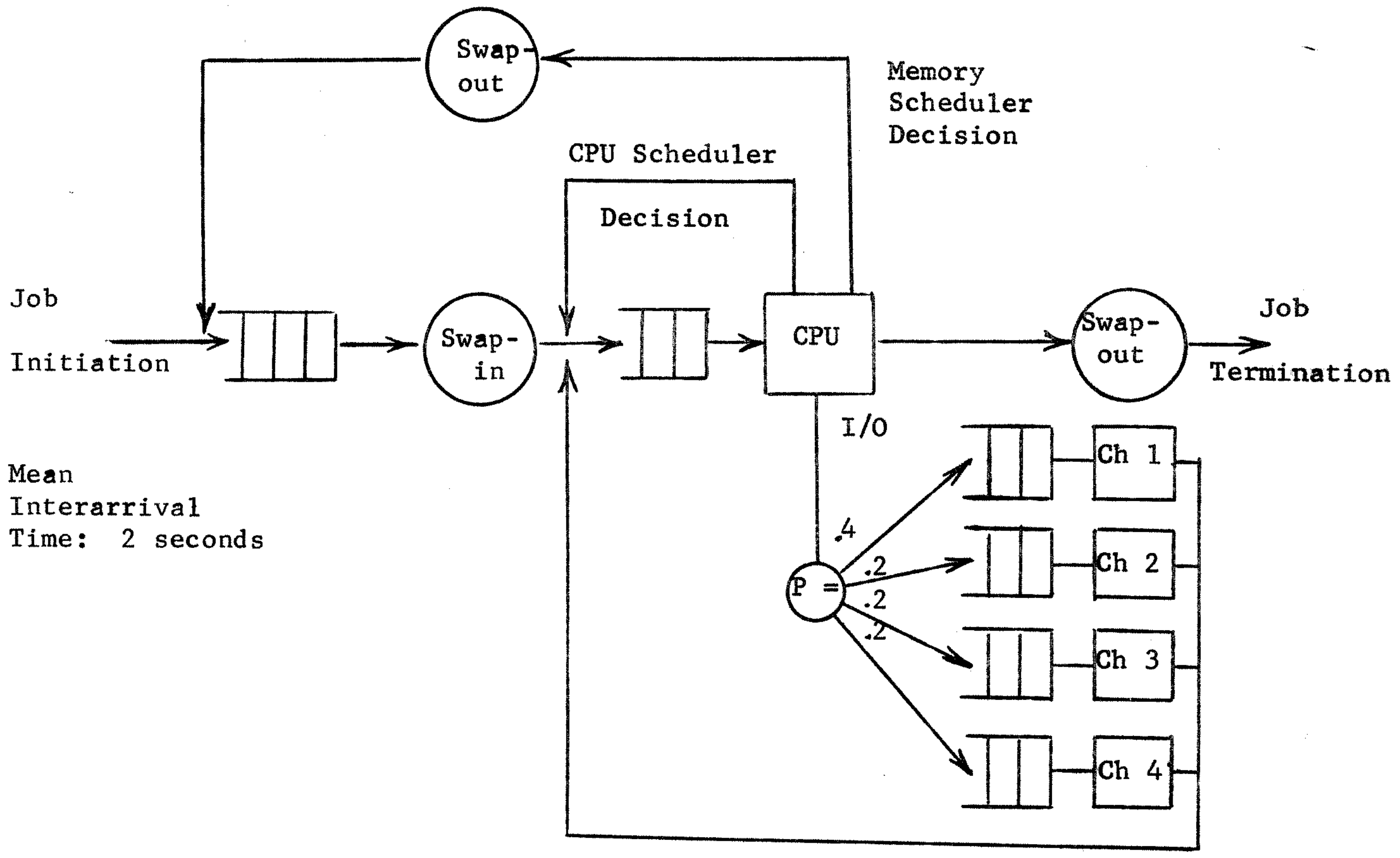


Figure 1

Queueing Model of CDC 6600 System.

Scheduler activity takes place at two levels: memory scheduling and CPU scheduling. The memory scheduler selects tasks to be activated whenever there is a change in demand for memory: when a job enters or leaves the system or when a job in memory completes a job step. Preempting jobs in memory is always a possible strategy for the memory scheduler. The CPU scheduler selects the task to be run on the processor: it is called each time there is a change in the availability of jobs for the processor. It may also preempt the job on the processor, either to give the processor to a job just joining the queue for the processor, or because the job on the CPU has completed a scheduler-allocated time slice.

Job Descriptions

Jobs are characterized as sequences of processing periods (CPU bursts) and I/O periods. Their characteristics were chosen from distributions reflecting empirical data (References 4, 10). The total job run time is extracted from the numerically specified distribution shown in figure 2. The CPU burst times are selected from a hyperexponential distribution with mean value of 40 ms. The I/O run times are selected from an exponential distribution with a mean value of 46 ms. The memory requirements for each job are selected from an approximate normal distribution with a mean of 20K and standard deviation of 8K, except that the memory requirement was limited to the range zero to 73K. Since a job changes its memory requirements during its lifetime, the field length of each job is recalculated after every five seconds of the job's run time. Memory scheduling is called with every change of a job's field length in simulation.

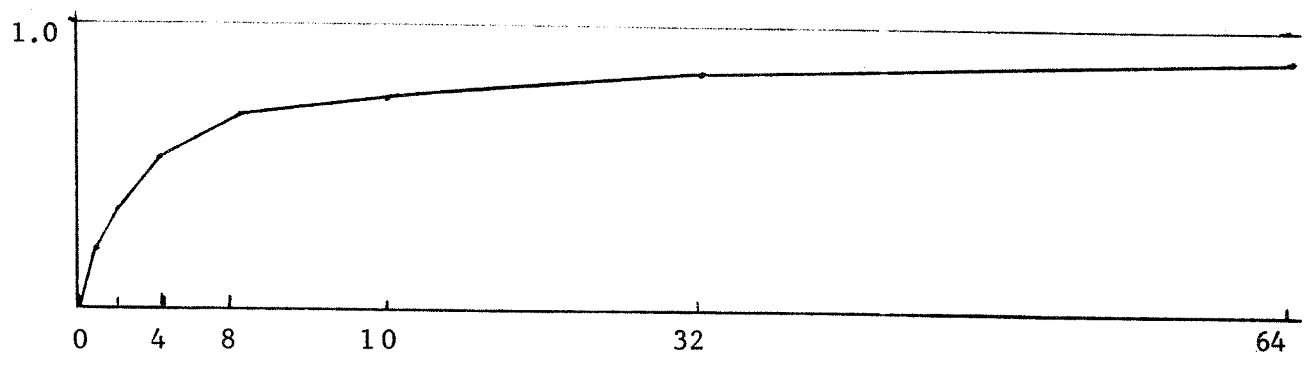


Figure 2.

Empirical Probability Distribution
Function for Total Job Run Time.

Each I/O request generated was directed to channel one with probability 0.4, and to each of the remaining three channels with a probability 0.2.

The interrarival time for the jobs entering the system is exponentially distributed, with a mean value of two seconds. The total length of the simulation runs was 200 seconds.

Scheduling Algorithms

The specific scheduling algorithms investigated in this model include a few standard algorithms and algorithms which use detailed predictive information of the user job's demand for system resources. The standard memory scheduling algorithms are:

1. FCFS. First come, first served. No attempt is made to place into memory a job other than the first job on the queue for memory.
2. SMF. Smallest memory first. Jobs both in memory and on the queue for memory are allocated in the order of the memory requirement (field length) of the job, to the extent that the jobs fit in memory.
3. FCSM: First come/smallest memory. A combined FCFS and SMF algorithm. Memory is allocated on a SMF basis, except that the first task on queue for memroy on an FCFS basis is not passed over more than once for memory allocation by a job arriving later, but with a smaller memory requirement.

The memory scheduling technique that uses the predictive data is based on the concept of attempting to balance the short-term demand for both the CPU and the I/O devices. Therefore, a ratio of processing time to total run time (processing and I/O) for the next 300 ms. of a job's operation is computed, whenever necessary. The scheduling policy is:

4. BAL: balanced processing requirements. If the set of tasks selected for memory by the SMF algorithm has an average processing ratio of less than a threshold value (chosen after optimizing the algorithm; in this case, the value was .42) the task with the largest compute ratio is chosen to replace a task with a compute ratio below the threshold.

Another predictive memory scheduling technique attempts to maximize throughput by choosing jobs with small processing requirements. This is an exact version of the standard technique of scheduling jobs on the basis of control-card estimates of job run time.

5. STF: smallest time first. The tasks with the smallest total run time to completion are selected for memory.

The CPU scheduling algorithms are likewise divided into standard and predictive algorithms. The standard algorithms are:

1. RR: Round-robin. Each ready job takes the processor in turn for a small time slice.

2. FCFS: First-come, first served. The first job on the processor queue is given the processor and not preempted.

3. LMF: Largest memory first. The job having the largest field length is given the processor and not preempted.

The predictive CPU scheduling algorithms are:

4: SBT. Shortest burst time. The task that will use the CPU for the shortest time until it completes or waits for an I/O operation is selected. This algorithm attempts to ensure that the CPU does not become idle by trying to achieve fast turn-around in using the I/O subsystem.

5. LBT. Longest burst time. An attempt to keep the processor busy

by doing the largest processing load on queue for the processor.

6. SBIO: Shortest burst time to an idle I/O device. A modification of the SBT, this algorithm selects for the CPU the job with the smallest burst time that is followed by either job completion or an I/O operation to a device that is currently idle.

Scheduling in the computer system requires both a CPU and a memory scheduler, which operate independently of each other. Thus, the total class of algorithms to be investigated is summarized in the following table:

CPU Scheduler	Memory Scheduler				
	FCFS	SMF	FCSM	BAL	STF
RR	Standard Algorithms			Predictive Memory Standard CPU	
FCFS					
LMF					
SBT	Standard Memory Predictive CPU			Predictive Memory and CPU	
LBT					
SBIO					

III. Results of the Simulation Studies

Simulation runs were made for each combination of the five memory scheduling algorithms and six CPU scheduling algorithms. The results in terms of CPU and I/O utilization are shown in figure 3. A few simple interrelationships can be noted from this data. First, the CPU utilization is directly related to I/O utilization for all memory scheduling algorithms except the STF algorithm. This is expected, since each job is represented by a sequence of CPU bursts and I/O utilization periods. The utilization of all the devices increases as the rate at which jobs cycle through the CPU-I/O subsystem increases. The memory scheduling algorithm does not interrupt this pattern, since swap-outs are made to ECS, contributing small amounts of overhead, but no additional I/O.

In the case of the STF algorithm, the I/O utilization was high because the short jobs tended to be more I/O bound than the longer jobs. The CPU burst times were hyperexponentially distributed, and the short-run time jobs had no long processing bursts selected from the long tail of the hyperexponential distribution.

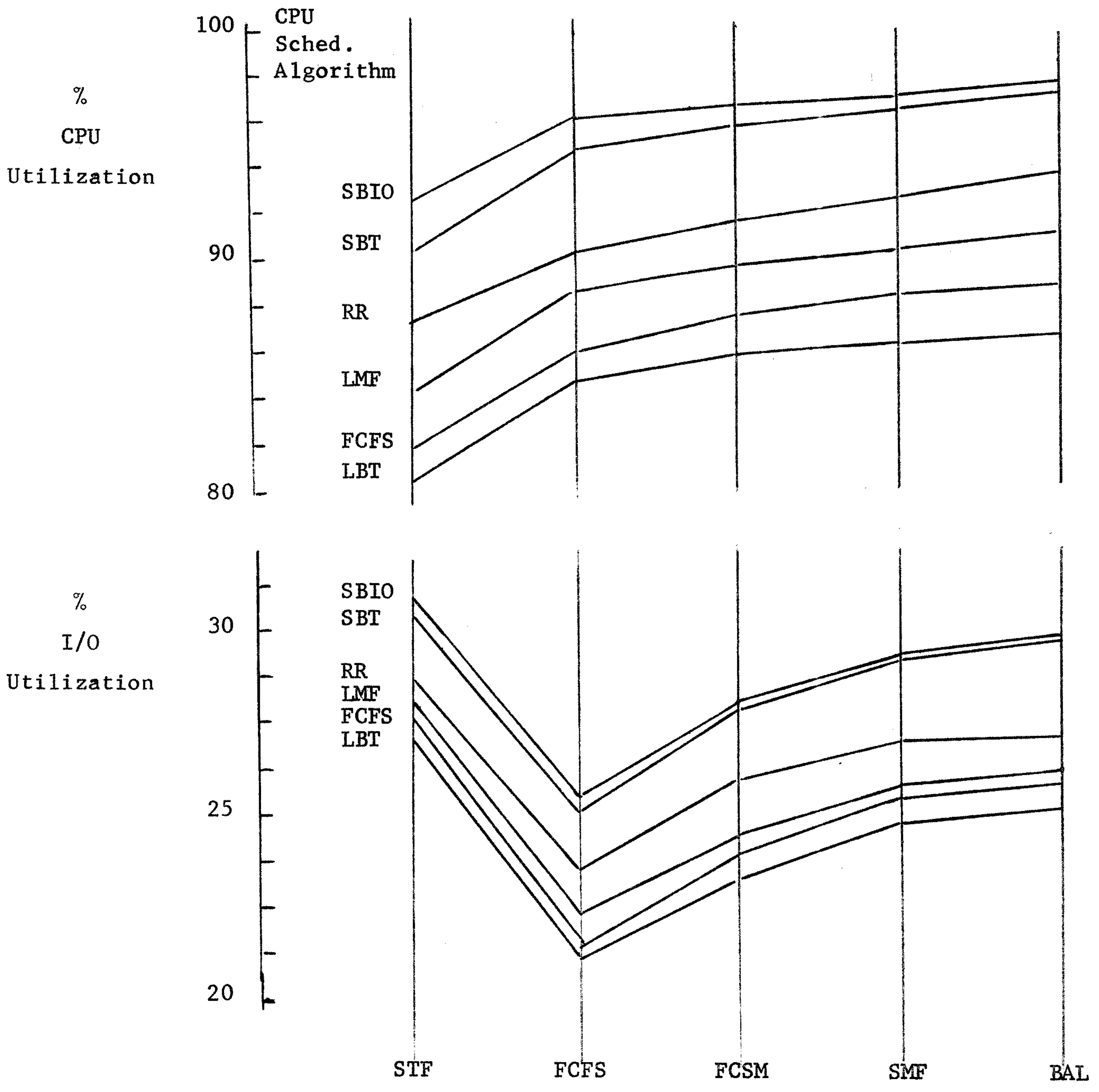
Second, the parallelism of the characteristics of the various CPU scheduling algorithms as different memory schedulers are applied, indicates that the relative effectiveness of the algorithms used to schedule each resource are independent of the scheduling of the other. The relative effectiveness of CPU scheduling algorithms could be expected to vary with the degree of multiprogramming, but in this case, as is shown in figure 4, the degree of multiprogramming did not vary greatly. The independence of CPU of CPU and I/O schedulers is also a consequence of the similarity of both the CPU and I/O requirements among the various jobs, and their independence of the job's memory requirement. This assumption is probably not well justified in reality. The design of memory schedulers that provide a mix of jobs that allow the CPU scheduler to achieve optimum values requires good knowledge of the total resource demand of jobs on queue for memory.

The throughput achieved by the scheduling algorithms is shown in figure 5. Throughput is optimized by the STF memory-scheduling algorithm.

The results show that the predictive BAL memory scheduling algorithm is marginally better than the SMF algorithm, which is the best of the standard memory scheduling algorithms. The other predictive memory scheduler, STF, is worse than FCFS in terms of CPU efficiency. It achieves the highest throughput for the length of the simulation run by advancing the jobs that can get done in thepperiod, but the superior throughput could not be sustained indefinitely with a high job input rate. The STF algorithm is a misapplication of the predictive data.

The real benefit of predictive data is apparent in CPU scheduling. Whereas the non-predictive algorithms could achieve CPU utilization in the range of 80.5 to 92.6 per cent for the various memory scheduling policies, the predictive scheduling algorithms for these same memory policies were in the range of 90.3 to 97.9 per cent. The SBT algorithm was able to achieve a large gain in efficiency because, when several jobs are on queue for the CPU, (which represents an imbalance in demand in the CPU-I/O sybsystem) the decision that most quickly places a job in the I/O subsystem so that it may return to the processor queue before the queue is depleted is always taken.

The SBT algorithm may be improved by choosing for the CPU not just the job that can generate the next I/O operation most quickly, but the



Memory Scheduling Algorithm

Figure 3

CPU and I/O Utilization as a Function
of CPU and Memory Scheduling Algorithms

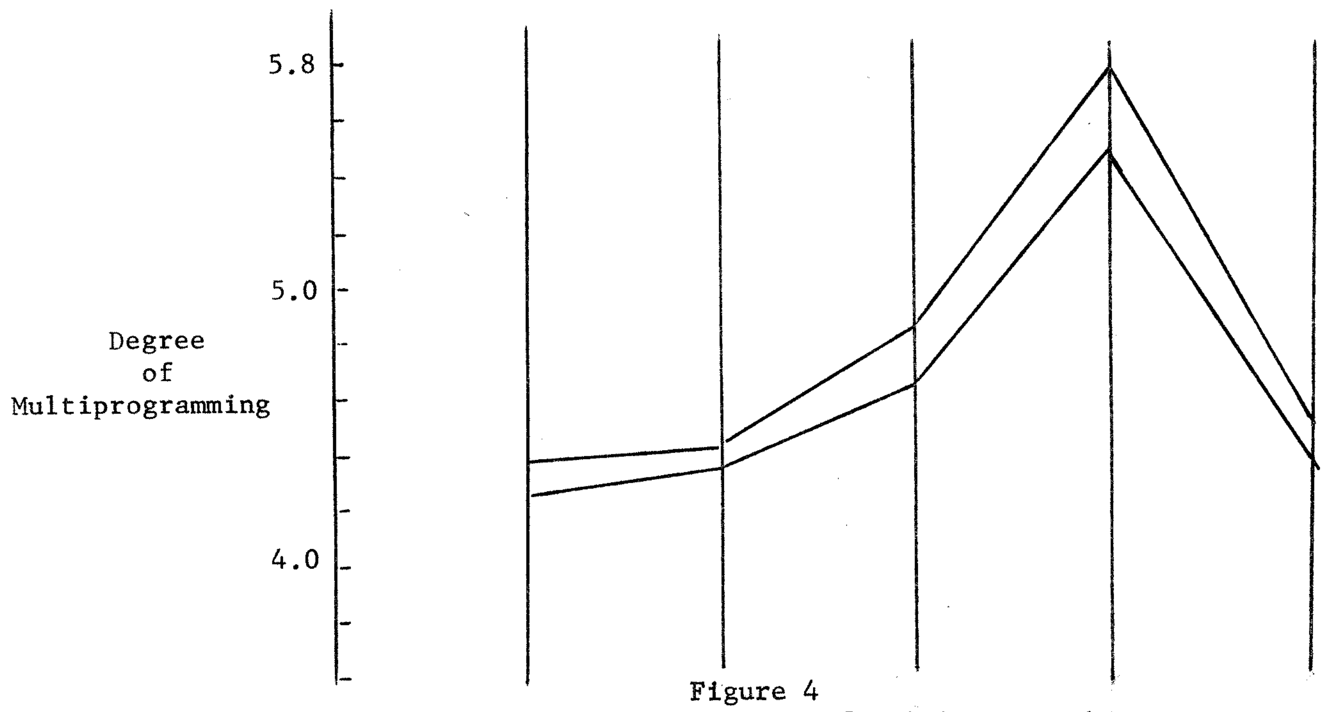
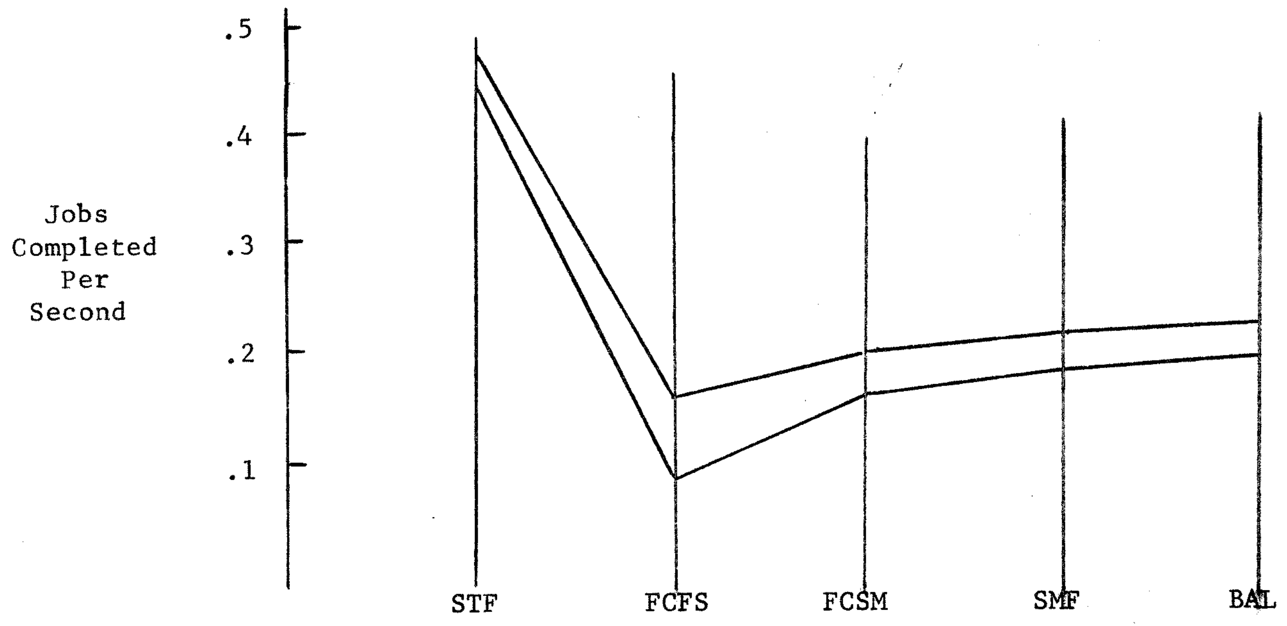


Figure 4
Range of Degree of Multiprogramming for Memory Scheduling Algorithms



Memory Scheduling Algorithm

Figure 5

Range of Throughput (Jobs/Sec.) for Memory Scheduling Algorithms job whose next I/O operation can be started soonest. For example, if the I/O subsystem is imbalanced (as it is in the system being simulated), it is likely that the next I/O of the job selected for the CPU will be directed to a device for which a queue already exists. This is not the optimal decision. The SBIO algorithm achieves greater efficiency by activating a job whose run time to an I/O operation on a currently idle device is the smallest. This reasoning can be extended to develop more complex algorithms. If the total future resource utilization data for a job is available, there is no limit to the possible complexity of computation for scheduling, and one hundred per cent CPU utilization (including overhead, of course) is attainable.

IV. Prediction from Inexact Data

The preceding predictive schedulers were evaluated in simulation under the assumption that the predictive information is complete and correct. This is an unrealistic assumption. Therefore, the scheduling models were also evaluated with the predictive data being correct with a probability of only $P<1$, where P was varied. The predictive schedulers recognized two classes of jobs: predictable jobs, in which the predictive data was assumed to be valid, and unpredictable jobs, in which the predictive data was known to be incorrect. The schedulers were modified to check at each step to be sure a predictable job was behaving as expected: if it didn't, it was flagged as an unpredictable job. The schedulers were also modified for a smooth transition between predictable and unpredictable job scheduling. For example, in the case of SBT and SBIO schedulers, a cyclic counter, modulo the dynamically-varying number of predictable jobs in memory, is incremented each time the scheduler works on the basis of predictable information. When the counter reaches zero, the scheduler swaps to RR for scheduling of the unpredictable task.

The results of the predictive algorithms with inexact data are shown in table 1 for the SBIO processor scheduler. It is seen that there is a linear decline in the advantage of predictive algorithms as P is reduced from 1 to 0.6, at which time the advantage of the predictive algorithm disappears. These results seem to indicate that the reliability of the predictive data should be at least sixty-five or seventy per cent before the design of such schedulers is seriously considered.

Table 1

CPU Efficiency Showing
Effectiveness of Predictive Scheduling Algorithms
With Inexact Data

CPU Sched. Algorithm \ Memory Sched. Algorithm	STF	FCFS	FCSM	SMF	BAL
RR	87.2	90.3	91.8	92.9	94.0
SBIO,P=1.0	92.4	96.2	97.0	97.4	97.9
SBIO,P=0.9	91.1	94.5	94.6	95.8	95.8
SBIO,P=0.8	89.0	93.0	92.9	93.9	93.9
SBIO,P=0.7	87.8	91.7	92.1	93.5	94.0
SBIO,P=0.6	86.9	90.1	91.0	92.3	93.3

(P is probability of correctness of predictive data.)

V. Conclusions

The results of the simulation runs with inexact predictive data are the most relevant to the design of scheduling algorithms for computer systems. Knowledge of what conditions might be obtained if complete predictive data were available is interesting but irrelevant, since it is inconceivable that this will ever be the case. The results indicate that seventy per cent of the jobs in the system must be totally predictable before the predictive scheduler will have an advantage over nonpredictive schedulers.

Although it is not expected that such a large portion of the jobs in the system will be found to be totally predictable, it is conceivable that the resource utilization patterns of more than seventy per cent of the jobs may be found to be predictable for a large part of their run time; during the compilation and loading of programs to be compiled and run, for example. Furthermore, the predictive algorithms investigated here represent the simplest possible use of predictive data. Once the resource utilization patterns of jobs are recorded and made available, scheduling techniques to further exploit the data will surely be developed. For example, in this relatively simple model (and also in many advanced operating systems), scheduling in the I/O subsystem is strictly on a first come, first served basis. Another increment in utilization and throughput may well be possible if decisions in the I/O subsystem are made to achieve balance globally, with the aid of predictive data.

The implementation of the predictive technique would be considerably more complex in a virtual memory system, but it has potential for great benefit there also; for example, in look-ahead paging.

The results of the simulation study encourage measurement and recording of resource utilization patterns by user jobs to determine whether indicated degree of predictability required to make the implementation of predictive schedulers feasible, can in fact be attained.

REFERENCES

1. Baskett, F.
   Mathematical Models of Computer Systems. Ph.D. Dissertation, The University of Texas at Austin, 1970.

2. Baskett, F., Browne, J. C., and Raike, W. M.
   "The Management of a Multi-Level Non-Paged Memory System," Proc. AFIPS 1970 SJCC, Vol. 36, Montvale, N.J., pp. 459-465.

3. Estrin, G., Muntz. R. R., and Uzgalis, R. C.
   Modelling, Measurement, and Computer Power, Proc. SJCC, 1972 pp. 725-738.

4. Johnson, Douglas S.
   A Process-Oriented Model of Resource Demands in Large Multiprocessing Computer Utilities, University of Texas Computation Center Report TSN-33, Austin, Texas.

5. Lan, J. C.
   A Study of Job Scheduling and Its Interaction with CPU Scheduling Computation Center Report TSN-24, University of Texas at Austin (Dec. 1971).

6. MacDougall, M. H.,
   "Simulation of an ECS-based Operating System," Proc., AFIPS 1967 SJCC Vol. 30, pp. 735-741.

7. Noe, J. D. and Nutt, G. D.
   Validation of a Trace-driven CDC 6400 Simulation, Proc. SJCC (1972) 749-757.

8. Noetzel, A. S.
   The Design of a Meta-System, Proc. SJCC, 1971, 415-424.

9. Schwetman, H. D.
   A Study of Resource Utilization and Performance Evaluation of Large-Scale Computer Systems, University of Texas Computation Center Report TSN-12 (July 1970), Austin, Texas.

10. Sherman, S., Baskett, F., and Browne, J. C.
    "Trace Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System," The University of Texas at Austin, 1970.