# SCHEDULE--CONSTRAINED JOB SCHEDULING IN A MULTIPROGRAMMED COMPUTER SYSTEM

Thomas Murray Cook

The University of Tulsa

## ABSTRACT

The schedule-constrained job scheduling problem is defined as the problem of deciding what jobs should coexist in the memory of a multiprogrammed computer to insure satisfactory schedule performance and adequate resource utilization. At the present time, the job scheduling function in many multiprogrammed computer systems is being accomplished in a suboptimal manner. In computer installations that must pay strict attention to schedule performance, the scheduling module of the operating system, because it does not consider schedule constraints, cannot be allowed to schedule jobs as they beome available for processing. Instead, human judgment must override the operating system by deciding which jobs should be input to the computer. To solve the schedule-constrained job scheduling problem, the author has developed and tested seven scheduling algorithms using a digital simulation model.

## INTRODUCTION

Current methods used for scheduling jobs into a multiprogrammed computer are yielding suboptimal results with respect to schedule performance and resource utilization. This paper addresses this problem and offers a solution which significantly improves current methods of job scheduling.

As depicted in Figure 1, the improvement of computer performance has been approached in two basic ways, better hardware and better software. With each generation of computers, performance has been dramatically increased. Due to new technological advances this trend will likely continue. In addition to design changes, it is possible, given the modular construction of today's computer handware, to mold a hardware configuration to the special needs of individual organizations and, thus, increase efficiency.

There are three fundamental ways in which software can be utilized to attempt to optimize system performance. Firstly, programs written by individual users, i.e., application programs, can be designed more efficiently. Secondly, the operating system can be improved. Finally, improved scheduling of the computer system and its different components can affect performance favorably.

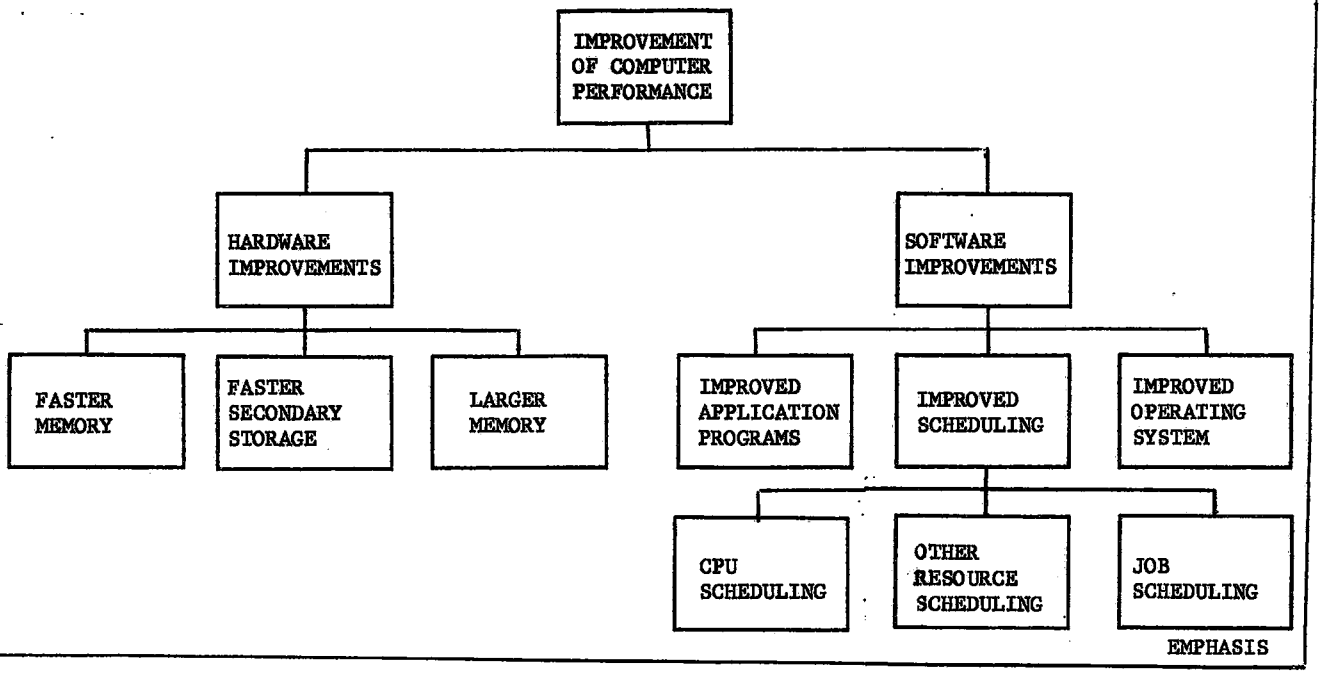
Scheduling can be broken into three classes: 1) CPU Scheduling, 2) Other Resource Scheduling, and 3) Job Scheduling. CPU scheduling, sometimes referred to as dispatching, is defined as the function of the supervisor program of an operating system that decides which job, of those resident in core, will receive CPU time. Scheduling of resources other than the CPU, such as mass storage and peripheral devices, is another area where resource utilization can be enhanced. Job scheduling simply means deciding which jobs should be loaded into memory to contend for CPU time. This research is limited to promoting performance through the betterment of the job scheduling function.

The scheduling function has received a considerable amount of attention from both computer scientists and operations researchers. The great majority of the work which has been done has sought to optimize one of several goals, such as maximizing the number of jobs processed per period, maximizing resource utilization, or minimizing the expected turnaround time. Techniques, such as classical queueing theory, integer programming, chance-constrained linear programming, dynamic programming, and simulation, have been reported in the literature. (2) Most attempts have been concerned with finding the optimal CPU scheduling algorithm. Several researchers have worked on the job scheduling problem, but few have considered the problem to be schedule-constrained.

The schedule-constrained problem is concerned with maximizing some function, such as resource utilization, subject to due-in and due-out times for individual jobs. At present, all attempts at job scheduling use the operating system to load jobs into memory from the jobs available in the queue. Jobs are queued as they are read into the system; thus, jobs due into the system but not yet queued cannot be considered by the operating system job scheduler. Therefore, scheduling personnel must supplement the job scheduling function of the operating system by exercising a considerable amount of discretion when placing jobs into the system. In other words, since the operating system at present does not consider schedule constraints,

FIGURE 1

Improvement of Computer Performance




people must. Even the most experienced of these schedulers cannot be expected to accomplish a high degree of efficiency in scheduling a multiprogrammed computer. There are simply too many variables, such as resource requirements and due-out times, with which to contend. Therefore, the best job scheduling algorithm might be one that can forecast jobs due into the system. The due-in time and job characteristics for recurring production jobs then can be read into the system prior to arrival of the job.

## METHODOLOGY

In order to improve the job scheduling function, the following methodology has been used. Firstly, a num- of job scheduling algorithms have been developed and coded. Secondly, a digital simulation model has been designed and implemented. Thirdly, to determine which algorithm performs most efficiently under various conditions, simulation experiments have been run.

In designing the experiments, it is necessary to limit the schedule-constrained job scheduling problem. It is assumed that the general environment of the computer system is one in which strict attention must be paid to schedule performance. Scheduling methods which neglect to consider deadlines are unrealistic. It is further assumed that the computer system is operated in a batch mode; that is, no on-line systems are active during the scheduling period. This assumption permits the treatment of resource availability as a deterministic variable. In addition, a non-paged environment is assumed. Consequently, the data used in the experiments has been gathered from a large facilities management firm. Because on-line systems are active during the prime shift, only data from the last two shifts has been used.

## JOB SCHEDULING ALGORITHMS

Algorithms for scheduling jobs in a multiprogrammed computer can be classified as either static or dynamic. Static algorithms utilize the job due-in/due-out schedule and the characteristics of each job, such as expected run time and resource requirements, to generate schedules which are based on these expected values. The static algorithm has the disadvantage of treating the stochastic variables involved, such as run time and due-in time, as deterministic. Therefore, if a job, whose expected run time is thirty minutes, aborts after thirty seconds of residence in memory, a serious degradation in resource utilization could occur if the schedule remains unchanged. It is for this reason that the overhead (i.e., computer system resources) involved in generating a static schedule is crucial to the success or failure of such an algorithm. If a job scheduling algorithm requires a large amount of resources, including CPU time, it would be infeasible to use that algorithm often without defeating its very purpose. As the variance in the relevant stochastic variables decreases, the allowable amount of overhead connected with generating a static schedule increases. Therefore, if a computer installation has a job mix that varies only slightly from week to week, and the

characteristics of each job can be accurately estimated, a static algorithm might yield the best results.

The algorithms classified as dynamic in this research are dynamic for the following reason. In lieu of a static schedule, every time a new job arrives in the queue or a job in execution is completed, the dynamic algorithm must decide which job of the queued jobs can and should be loaded.

Two static job scheduling algorithms have been developed. Both are inspired by rather unsophisticated techniques that, given sufficient amount of time, will yield an optimal schedule. One way of optimizing the job scheduling function is to generate all feasible job loading combinations and simply choose the best. Unfortunately, given a facility of even modest size, this solution is not feasible due to the number of possible combinations that would have to be generated and evaluated. Therefore, instead of generating all combinations, the first static algorithm which has been developed and tested is one that takes a sample of the possible combinations and chooses the best job mix. This algorithm has been named Random. The second static algorithm (Backward) is based on a dynamic programming type model which minimizes resource waste subject to due-in and due-out schedule constraints.

The literature concerned with job scheduling or job shop scheduling provides an almost inexhaustible set of dynamic scheduling algorithms. For this research, however, only three dynamic algorithms have been investigated. All three of these algorithms have been used to schedule both critical (jobs that have a definite schedule) and non-critical (unscheduled jobs) jobs, and two have been used in conjunction with the two static algorithms. The static algorithms load critical jobs and the dynamic algorithms schedule non-critical jobs. The first dynamic algorithm has been advocated by Baskett. (2) This algorithm assigns loading priorities according to a job's space-time product. Jobs with low kilobyte second requirements are loaded first. The second algorithm seeks to maximize the degree of multiprogramming. The third dynamic algorithm orders jobs according to slack time.

Baskett has contended that giving higher priorities to jobs with small space-time requirements tends to maximize resource utilization. (2) Therefore, a priority index defined in the following manner has been calculated:

$$p_i = t_i \cdot m_i \qquad (1)$$

where $p_i$ = priority index of job i
$t_i$ = estimated run time of job i
$m_i$ = memory requirement of job i

The algorithm sorts the input queue in ascending order using $p_i$. After the input queue has been sorted, the algorithm attempts to load the first job in the queue. If successful, the next job is examined to see if it can be loaded. If a job cannot be loaded, the algorithm merely looks at the next job in the queue for possible loading. The process is continues until all jobs have been examined. Whenever a new job arrives, or an old job leaves the system, the input queue is again sorted on $p_i$, and the loading process is repeated.

Penny has shown that if an algorithm maximizes the degree of multiprogramming, it tends to maximize resource utilization. (41) Therefore, an algorithm designed to give priority to jobs with small resource requirements should be investigated. The priority index is defined in the following manner:

$$p_i = \sum_{k=1}^{N} \frac{r_{ik}}{R_k} \qquad (2)$$

where $p_i$ = priority index of job i
$r_{ik}$ = requirement of job i for resource k
$R_k$ = available amount of resource k
N = number of resources contended for

With the exception of how the priority index is computed, this algorithm is identical to the previously described Space-Time Algorithm.

The Slack-Time Priority Algorithm orders jobs in the job queue in ascending sequence according to slack time. (33) Slack-time is defined in the following manner:

$$p_i = d_i - (t + s_i) \qquad (3)$$

where $p_i$ = priority index for job i = slack time
$t$ = present time
$s_i$ = expected service time for job i
$d_i$ = due-out time for job i

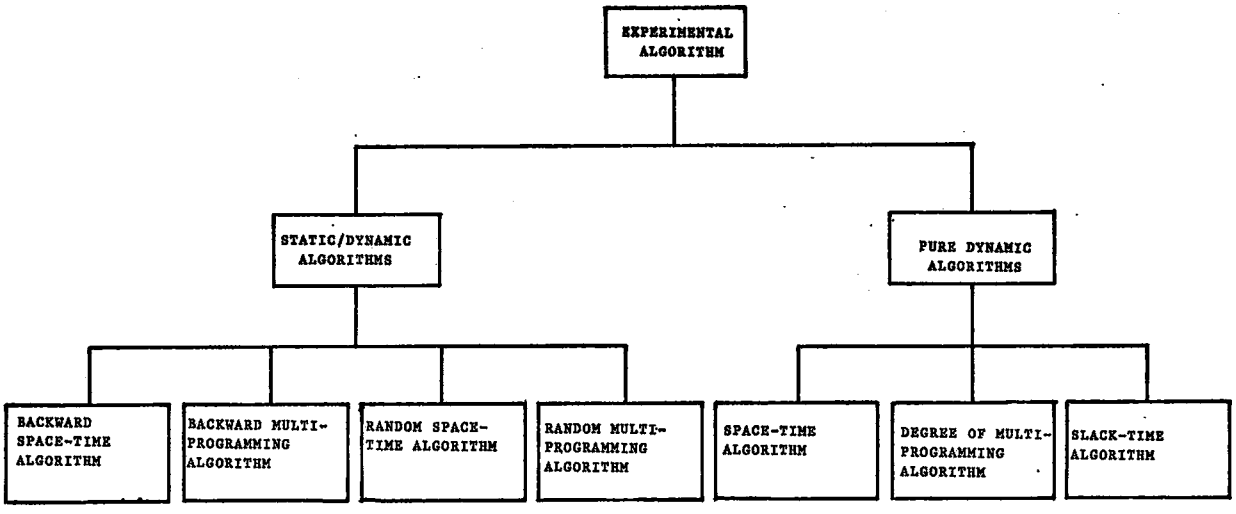
Again, the loading logic for this algorithm is the same as for the other two dynamic algorithms.

Two static and three dynamic job scheduling algorithms have been defined. The experiments use seven algorithms, four of which are a combination of the two static algorithms and two dynamic loading algorithms. The remaining three experimental algorithms are the three dynamic algorithms used independently of the static algorithms.

By combining the static and dynamic algorithms, the following interaction occurs. At the beginning of the simulation run and at specified periods, a static schedule is generated for all critical, schedule-constrained jobs. Non-critical jobs are scheduled by a dynamic algorithm. These non-critical jobs act as filler jobs to use resources not being used by critical jobs. When a critical job is scheduled in at a specific time, and the necessary resources are being used by a non-critical job, this less important job is rolled out (returned to the queue), and the critical job is loaded. However, if the preempting of non-critical jobs would still not produce the adequate resources, then the critical job is forced to wait until another critical job is completed and releases the necessary resources. Critical jobs are not allowed to be preempted. Figure 2 summarizes the experimental algorithms used in this research.

FIGURE 2

Summary of Experimental Algorithms



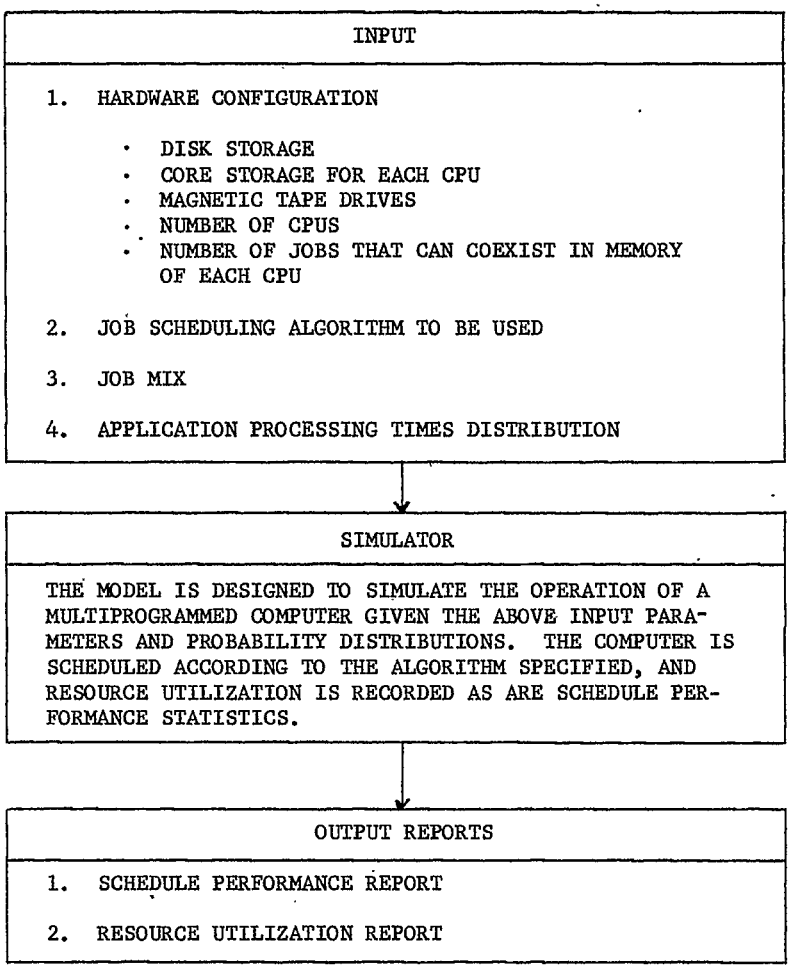

## SIMULATION MODEL

As depicted in Figure 3, the simulation model is highly parameterized and allows such exogeneous variables as the hardware configuration, the job scheduling algorithm, and the job mix profiles to be input by the researcher. Given these controllable variables, the model simulates the operation of a multiprogrammed/multiprocessor computer system. At the end of the simulation, two types of reports are printed: a schedule performance report and a resource utilization report.

As shown in Figure 4, the logic of the simulation model can be broken into four major categories. Firstly, run parameters and job profiles must be input to the model. Secondly, if the job scheduling algorithm includes a static algorithm, the appropriate subroutine is called to generate a static schedule. A static job scheduling algorithm utilizes the job due-in/due-out schedule and the characteristics of each job, such

FIGURE 3

Simulation Model Schematic




as expected run time and resource requirements, to generate a loading schedule. If only a dynamic algorithm is to be used, the necessity to generate a static schedule is deleted. Thirdly, the simulation model is run with a next most imminent event logic. Events, such as job departures and arrivals, are generated and handled by the model. Fourthly, after the last job has departed the modeled system, two reports are generated and printed, and the simulation is terminated.

The first report is concerned with schedule performance and reflects the following measurements: 1) The number of late jobs reflects only critical jobs whose departure time is later than its due-out time. 2) Mean lateness is computed in the following manner:

$$x = \frac{\sum_{i=1}^{N} X_i}{N} \qquad (4)$$

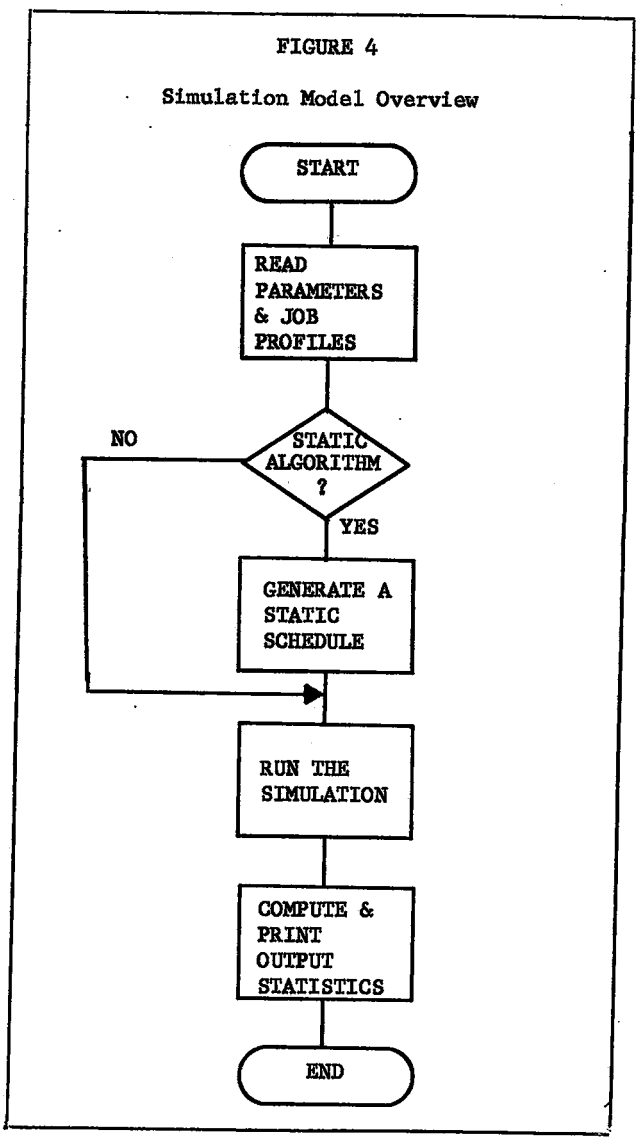
where $x$ = mean lateness
$x_i$ = lateness in minutes of job i
N - number of late jobs.

3) Mean turnaround time refers to the average time necessary for a non-critical job to be processed by the computer system. A job's turnaround time is the time it is resident in memory plus the time it is forced to wait in the input queue. 4) Variance in turnaround time is calculated such that an analysis of variance concerning the mean turnaround time can be performed. 5) Standard deviation for turnaround time gives a good indication of the dispersion in the distribution of turnaround times. 6) Mean wait time reflects the average amount of time (in minutes) that a non-critical job spends in the queue waiting to be loaded. 7) Variance in wait time allows the analysis of variance to be performed on the mean wait time. 8) Standard deviation for wait time indicates the degree of variability in the distribution of the wait times.

The second report produced by the simulation model is concerned with resource utilization. Since the job

FIGURE 4

Simulation Model Overview




mix remains constant among the different experiments, resource utilization can be measured by one statistic, namely, the amount of time required to process completely the given amount of work. Other statistics, such as the percentage of core utilization, tape drive utilization, and disk pack utilization, are calculated to indicate which resources appear to be the most constraining.

## MODEL VERIFICATION AND VALIDATION

Simulation model verification and validation is perhaps the most difficult and frequently overlooked task involved in accomplishing research using a digital simulator. Unless a researcher has some degree of confidence that his model accurately represents the "real world" system, he cannot advocate its use in making decisions concerning the actual system.

In order to verify the simulation model used in this research, three types of tests have been made. Firstly, the logic of the driver program and all the subroutines have been thoroughly tested to insure that the model is performing as designed. A set of ten jobs and over one hundred individual print statements have revealed numerous errors which subsequently have been corrected.

After verification of the simulator's logic, two types of statistical tests have been run. The first test is a non-parametric goodness of fit test which determines if the run time generating subroutine is functioning properly. A job's run time is generated by the following function:

$$t_i = \bar{y}_i + x \cdot \bar{y}_i \qquad (5)$$

where $t_i$ = generated run time for job i

$x$ = generated deviation of the run time from the expected value of the run time for job i (generated using an empirical probability distribution)

$x = (y_{ik} - \bar{y}_i) / \bar{y}_i$

$\bar{y}_i$ = the expected run time for job i.

$y_{ik}$ = actual run time for job i the kth run.

A random variable, x, has been defined as the amount of deviation between a job's expected run time and its actual run time. In order to generate this random variable, a subroutine, which uses an empirical cumulative probability distribution to generate x, has been coded. To test this function, 100,000 x's have been generated and grouped into a frequency distribution. This distribution has been compared to the empirical distribution using the Kolmogorov-Smirnov goodness of fit test. The generating function easily passed the test.

The final test in the verification procedure has required the replication of the experiments and one way analysis of variance. The simulation model has been run nine times with all controllable variables, except the random number seed, held constant. Before each run, the random number seed has been replaced by another random number. Two of the statistics, mean turnaround time and mean wait time, have been chosen for one way analysis of variance tests. The null hypotheses are that all nine runs have produced mean turnaround times and mean wait times which have been taken from the same population. Using an $\alpha$ of .05 these null hypotheses must be accepted.

It is recognized that the verification procedures described above do not provide absolute model validation. To further test the validity of the model, simulation results will be compared to results produced by an actual computer system.

## DESIGN OF SIMULATION EXPERIMENTS

The purpose of the experiments is to evaluate the performance of several job scheduling algorithms operating in an environment where schedule constraints exist. To accomplish this goal, a somewhat constant environment is necessary. As seen in Figure 5, the hardware configuration and job set are held constant for each simulation. The only variables which are allowed to vary are the scheduling algorithms and the schedule constraints.

Two basic sets of experiments have been run. The first set treats run time as a deterministic variable which can be estimated perfectly. The reason for this is to establish an upper bound on the performance of the static-dynamic combination algorithms. If these combination algorithms perform poorly with perfect information, their use cannot be advocated. The second set of simulations treats run time as a stochastic variable which is generated in accordance with an empirical distribution.

Each set of experiments consists of twenty-one separate simulations. The seven algorithms have been tested under loose, moderate, and heavy schedule constraints. In order to simulate the various degrees of tightness in the schedule constraints, the due-in and due-out times of the jobs have been set arbitrarily. Figure 6 identifies the simulations which have been run.

## SUMMARY AND RESULTS

A comprehensive literature search reveals that the great majority of research concerning the scheduling of multiprogrammed computers concentrates on the scheduling of the CPU and not on job scheduling. The existing job scheduling research has sought to maximize efficient use of computer resources without regard to schedule performance. If one relies solely upon the literature, he will find it difficult even to perceive the schedule-constrained problem. If, however, one ventures into the "real world" and communicates with executives responsible for computer installation performance, the problem is transformed from an obscure one to one of paramount importance. All installations questioned have emphasized that algorithms currently being used perform efficiently with regard to computer resource utilization. However, these installations have also indicated that it is impossible to use these algorithms without manually overriding them to insure schedule performance. There is a need for a job scheduling algorithm which seeks to insure schedule performance and, at the same time, efficiently utilize computer resources.

To resolve the schedule-constrained job scheduling problem, seven experimental algorithms have been developed and tested. The experimental results as summarized in Figure 6 indicate two major conclusions. 1) Due to excessive overhead required and the stochastic nature of job run times, static job scheduling algorithms do not provide an adequate solution to the job scheduling problem. 2) A simple Slack-Time scheduling algorithm promises to perform quite well with respect to schedule performance and resource utilization. See Figures 7 and 8. Based upon the results of this research, this author advises that implementation of the Slack-Time algorithm into the operating systems of installations which have a job scheduling problem with schedule constraints. It is believed that, in addition to performing in a manner superior to methods now being utilized, the Slack-Time algorithm will provide automatic job scheduling. No longer will

scheduling personnel have to interfere with the job scheduling function of the operating system in order to insure satisfactory schedule performance.

FIGURE 5

Hardware Configuration

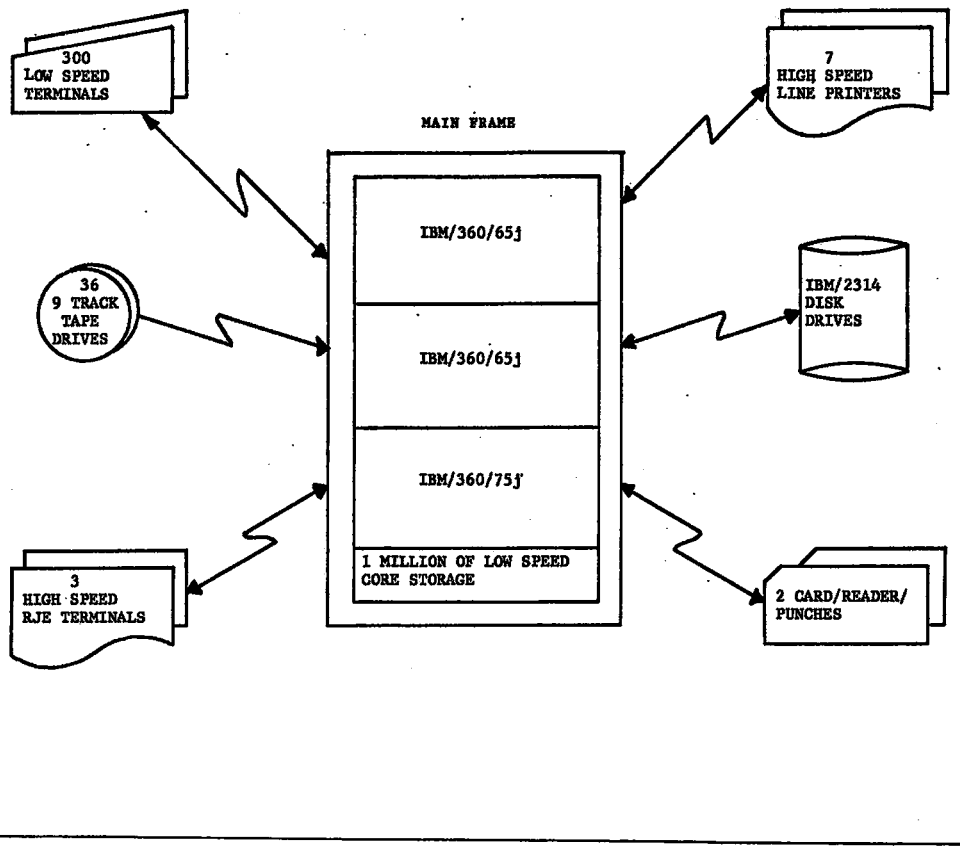


FIGURE 6

Summary of Experimental Results

| EXPERIMENT NUMBER | NATURE OF RUN TIME | | JOB SCHEDULING ALGORITHM | | | | | | | | SCHEDULE CONSTRAINTS | | | OUTPUT STATISTICS | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DETERMINISTIC | STOCHASTIC | BACKWARD/ SPACE-TIME | SPACE-TIME | RANDOM/ SPACE-TIME | RANDOM/MULTI-PROGRAMMING | RANDOM/MULTI-PROGRAMMING | SPACE-TIME | DEGREE OF MULTI-PROGRAMMING | SLACK TIME | LOOSE | MODERATE | TIGHT | NUMBER OF LATE JOBS | MEAN LATENESS (IN MINUTES) | MEAN TURNAROUND (IN MINUTES) | STANDARD DEVIATION ON TURN-AROUND (IN MIN.) | MEAN WAIT TIME (IN MINUTES) | STANDARD DEVIATION WAIT TIME (IN MINUTES) | TIME PERIOD IN MINUTES | PERCENT CORE UTILIZATION | PERCENT TAPE UTILIZATION | PERCENT DISK UTILIZATION | PERCENT INITIATOR/TERMINATOR UTILIZATION |
| 1 | | | | | | | | | | | | | | 0 | 0 | 250.89 | 116.14 | 240.97 | 110.90 | 456.50 | 85.84 | 55.91 | 23.26 | 87.95 |
| 2 | | | | | | | | | | | | | | 0 | 0 | 224.10 | 88.41 | 214.19 | 88.12 | 446.75 | 87.71 | 57.13 | 23.72 | 89.87 |
| 3 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 4 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 5 | | | | | | | | | | | | | | 1 | 21.77 | 206.54 | 107.34 | 96.61 | 101.15 | 508.77 | 77.01 | 50.13 | 20.87 | 78.87 |
| 6 | | | | | | | | | | | | | | 3 | 7.25 | 240.14 | 115.23 | 230.40 | 114.70 | 493.97 | 79.31 | 51.63 | 21.50 | 81.23 |
| 7 | | | | | | | | | | | | | | 0 | 0 | 334.62 | 80.90 | 324.67 | 90.95 | 432.8 | 90.52 | 58.93 | 24.53 | 92.71 |
| 8 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 9 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 10 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 11 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 12 | | | | | | | | | | | | | | 21 | 89.64 | 92.55 | 91.07 | 82.61 | 91.91 | 496.42 | 79.92 | 51.38 | 21.39 | 80.83 |
| 13 | | | | | | | | | | | | | | 31 | 67.88 | 177.03 | 110.33 | 167.09 | 113.31 | 494.42 | 80.86 | 52.65 | 21.92 | 82.83 |
| 14 | | | | | | | | | | | | | | 4 | 35.05 | 257.95 | 108.27 | 248.07 | 114.58 | 449.72 | 87.26 | 56.81 | 23.65 | 89.37 |
| 15 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 16 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 17 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 18 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 19 | | | | | | | | | | | | | | 31 | 105.25 | 92.55 | 93.07 | 82.61 | 91.91 | 496.42 | 78.92 | 51.38 | 21.39 | 80.83 |
| 20 | | | | | | | | | | | | | | 51 | 93.12 | 177.03 | 110.33 | 167.09 | 113.31 | 484.42 | 80.88 | 52.65 | 21.92 | 82.83 |
| 21 | | | | | | | | | | | | | | 2 | 15.69 | 265.38 | 115.77 | 255.44 | 121.80 | 448.65 | 87.32 | 56.85 | 23.67 | 89.44 |
| 22 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 23 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 24 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 25 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 26 | | | | | | | | | | | | | | 2 | 24.20 | 105.67 | 108.60 | 95.69 | 101.85 | 505.43 | 79.86 | 49.93 | 19.35 | 77.78 |
| 27 | | | | | | | | | | | | | | 2 | 50.95 | 223.25 | 115.33 | 212.88 | 115.04 | 573.27 | 67.26 | 39.51 | 17.89 | 69.02 |
| 28 | | | | | | | | | | | | | | 0 | 0 | 120.30 | 82.42 | 110.40 | 91.04 | 467.93 | 82.39 | 48.41 | 21.91 | 84.55 |
| 29 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 30 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 31 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 32 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 33 | | | | | | | | | | | | | | 20 | 73.76 | 94.17 | 87.79 | 83.77 | 88.22 | 566.25 | 68.09 | 40.00 | 18.11 | 69.87 |
| 34 | | | | | | | | | | | | | | 20 | 53.55 | 163.87 | 102.78 | 153.47 | 107.19 | 566.42 | 68.07 | 39.99 | 18.10 | 69.85 |
| 35 | | | | | | | | | | | | | | 4 | 56.42 | 232.04 | 96.74 | 221.64 | 104.72 | 566.08 | 68.11 | 40.02 | 18.12 | 69.89 |
| 36 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 37 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 38 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 39 | | | | | | | | | | | | | | * | * | * | * | * | * | * | * | * | * | * |
| 40 | | | | | | | | | | | | | | 33 | 89.14 | 94.17 | 87.79 | 83.77 | 88.22 | 566.25 | 68.09 | 40.00 | 18.11 | 69.87 |
| 41 | | | | | | | | | | | | | | 47 | 84.05 | 163.87 | 102.78 | 153.47 | 107.19 | 566.42 | 68.07 | 39.99 | 18.10 | 69.85 |
| 42 | | | | | | | | | | | | | | 4 | 56.83 | 235.47 | 108.49 | 225.07 | 115.09 | 566.09 | 68.11 | 40.02 | 18.12 | 69.89 |

*Indicates the algorithm has proved infeasible due to excessive overhead requirements.

FIGURE 7

Resource Utilization Statistics

| Schedule Constraints | Job Scheduling Algorithm | Time Period in Minutes | Per Cent Core Utilization | Per Cent Tape Utilization | Per Cent Disk Utilization | Initiator/ Terminator Utilization |
|---|---|---|---|---|---|---|
| LOOSE | SPACE-TIME | 508.77 | 77.01 | 50.13 | 20.87 | 78.87 |
| LOOSE | MULTIPRO-GRAMMING | 493.97 | 79.31 | 51.63 | 21.50 | 81.23 |
| LOOSE | SLACK-TIME | 432.8 | 90.52 | 58.93 | 24.53 | 92.71 |
| MODERATE | SPACE-TIME | 496.42 | 78.92 | 51.38 | 21.39 | 80.83 |
| MODERATE | MULTIPRO-GRAMMING | 484.42 | 80.88 | 52.65 | 21.92 | 82.83 |
| MODERATE | SLACK-TIME | 449.72 | 87.26 | 56.81 | 23.65 | 89.37 |
| TIGHT | SPACE-TIME | 496.42 | 78.92 | 51.38 | 21.39 | 80.83 |
| TIGHT | MULTIPRO-GRAMMING | 484.42 | 80.88 | 52.65 | 21.92 | 82.83 |
| TIGHT | SLACK-TIME | 448.65 | 87.32 | 56.85 | 23.67 | 89.44 |

FIGURE 8

Schedule Performance Statistics

| Job Scheduling Algorithm | Number of Late Jobs | Mean Lateness in Minutes | Mean Turnaround in Minutes | Standard Deviation Turnaround in Minutes | Schedule Constraints |
|---|---|---|---|---|---|
| SPACE-TIME | 2 | 24.20 | 105.67 | 108.60 | LOOSE |
| MULTIPROGRAMMING | 2 | 50.95 | 223.25 | 115.31 | LOOSE |
| SLACK-TIME | 0 | 0 | 320.80 | 82.42 | LOOSE |
| SPACE-TIME | 20 | 73.76 | 94.17 | 87.79 | MODERATE |
| MULTIPROGRAMMING | 29 | 53.55 | 163.87 | 102.78 | MODERATE |
| SLACK-TIME | 4 | 56.42 | 232.04 | 96.74 | MODERATE |
| SPACE-TIME | 33 | 89.18 | 94.17 | 87.79 | TIGHT |
| MULTIPROGRAMMING | 47 | 84.05 | 163.87 | 102.78 | TIGHT |
| SLACK-TIME | 4 | 56.83 | 235.47 | 108.49 | TIGHT |

BIBLIOGRAPHY

1. Baskett, Forest, III; Browne, J. C.; and Raike, W. M. "The Management of a Multi-Level Non-Paged Memory System," in Proceedings of the AFIPS 1970 Spring Joint Computer Conference, Vol. 36.
2. Baskett, Forest, III. "Mathematical Models of Multiprogrammed Computer Systems," Unpublished Ph.D. dissertation, The University of Texas at Austin, December, 1970.
3. Block, Andrew F. "Automated Computer Scheduling," in Share XXXI Guide 27 Proceedings, Vol. 2, Sec. 5, October-November, 1968.
4. Codd, E. F. "Multiprogram Scheduling," in Communications of the ACM, Parts I and II, Vol. 3, No. 6, June, 1960.
5. _____. "Multiprogram Scheduling," in Communications of the ACM, Parts III and IV, Vol. 3, No. 7, July, 1960.
6. Coffman, E. F., Jr. "Analysis of Two Time-Sharing Algorithms Designed for Limited Swapping," in Journal of the ACM, Vol. 15, No. 3, July, 1968.
7. _____. "A Simple Probability Model Yielding Performance Bounds," in Institute of Electrical and Electronic Engineers Transactions, C-17, No. 1, January, 1968.
8. _____, and Kleinrock, L. "Computer Scheduling Methods and Their Countermeasures," in Proceedings of the AFIPS 1968 Spring Joint Computer Conference.
9. Conway, R. W. and Maxwell, W. L. "Network Scheduling by the Shortest Operation Discipline," in Operations Research, Vol. 10, No. 1, 1962.
10. Denning, Peter J. "Effects of Scheduling on File Memory Operations," in Proceedings of the AFIPS 1967 Spring Joint Computer Conference.
11. Denning, Peter J. "Resource Allocation in Multiprocess Computer Systems." Unpublished Ph.D. dissertation at Massachusetts Institute of Technology, 1968.
12. Edelman, Paul R. "Scheduling the Computer in a Manufacturing Environment," in Data Processing, Vol. XIII, 1968.
13. Estrin, G. and Kleinrock, L. "Measures, Models, and Measurements for Time-Shared Computer Utilities," in Proceedings of the ACM 22nd National Conference.
14. Fenichel, Robert R. and Grossman, Adrian J. "An Analytic Model of Multiprogrammed Computing," in Proceedings of the AFIPS 1969 Spring Joint Computer Conference.
15. Fife, Dennis W. "An Optimization Model for Time-Sharing," in Proceedings of the AFIPS 1966 Spring Joint Computer Conference.
16. _____ and Rosenberg, R. "Queueing in a Memory-Shared Computer," in Proceedings of the ACM 19th National Conference.
17. Fine, Gerald H. and McIssac, Paul V. "Simulation of a Time-Sharing System," in Management Science, Vol. 12, No. 6, February, 1966.
18. _____, _____, and Jackson, C. W. "Dynamic Program Behavior Under Paging," in Proceedings of the ACM 21st National Conference, Washington, D. C., 1966.
19. Fineberg, M. S. and Serlin, Omri. "Multiprogramming for Hybrid Computation," in Proceedings of the AFIPS 1967 Fall Joint Computer Conference.
20. Gabor, D. "Associative Holographic Memories," in IBM Journal of Research and Development, Vol. 13, No. 2, March, 1969.
21. Gaver, D. P., Jr. "Probability Models for Multiprogramming Computer Systems," in Journal of the ACM, Vol. 14, No. 3, July, 1967.

22. Giblin, Tom. "Techniques for Optimizing OS/360 Multiprogramming Installation Performance," in Share XXXI Guide 21, November 1, 1968.
23. Habermann, A. N. "Prevention of System Deadlocks," in Communications of the ACM, Vol. 12, No. 7, July, 1969.
24. Howarth, D. J., Jones, P. D., and Wyld, M. T. "The Atlas Scheduling Systems," in Proceedings of the AFIPS 1963 Spring Joint Computer Conference.
25. Hume, J. N. P. and Rolfson, C. B. "Scheduling for Fast Turnaround in Job-at-a-Time Processing," in Proceedings of the IFIP Congress, 1968, Hardware 2, Booklet E.
26. Hutchinson, I. G. K. "A Computer Center Simulation Project," in Communications of the ACM, Vol. 8, No. 9, 1965.
27. Jackson, J. R. "Queues with Dynamic Priority Disciplines," in Management Science, Vol. 7, No. 1, 1961.
28. Katz, J. H. "Simulation of a Multiprocessor Computer System," in Proceedings of the AFIPS 1966 Spring Joint Computer Conference.
29. Kleinrock, Leonard. "Time-Shared Systems: a Theoretical Treatment," in Journal of the ACM, Vol. 14, No. 2, April, 1967.
30. _____ and Coffman, E. G. "Distribution of Attained Service in Time-Shared Systems," in Journal of Computer and System Sciences, Vol. 1, No. 3, October, 1967.
31. Krishnamoorthi, B. and Wood, Roger C. "Time-Shared Computer Operations with Both Interarrival and Service Times Exponential," in Journal of the ACM, Vol. 13, No. 3, July, 1960.
32. Lan, Jean C. A Study of Job Scheduling and Its Interaction with CPU Scheduling TSN-24. The University of Texas at Austin Computation Center, Austin, December, 1971.
33. LeGrande, E. "The Development of a Factory Simulation System Using Actual Operating Data," in Management Technology, Vol. 3, No. 1, May, 1963, p. 7.
34. Mamelak, J. S. "Multiprogram Scheduling," in Proceedings of the 5th National Conference of the Computer Society of Canada, 1966.
35. Marshall, B. S. "Dynamic Calculation of Dispatching Priorities Under OS/360 MVT," in Datamation, Vol. 15, No. 8, August, 1969.
36. Naylor, Thomas H., Balintfy, Joseph L., and others. Computer Simulation Techniques. John Wiley and Sons, Inc., New York, 1966.
37. Nielsen, Norman R. "An Analysis of Some Time-Sharing Techniques," in Communications of the ACM, Vol. 14, No. 2, February, 1971.
38. _____. "Computer Simulation of Computer System Performance," in Proceedings of the ACM 22nd National Conference.
39. _____. "The Simulation of Time-Sharing Systems," in Communications of the ACM, Vol. 10, No. 7, July, 1967.
40. Oppenheimer, G. and Weizer, N. "Resource Management for a Medium Scale Time-sharing Operating System," in Communications of the ACM, Vol. 11, No. 5, May, 1968.
41. Penny, J. P. "An Analysis, Both Theoretical and By Simulation, of a Time-Shared Computer System," in Computer Journal, Vol. 9, No. 1, May, 1966.
42. Pirow, P. C. and Kaye, R. L. "Multiprogramming," in South African Computer Bulletin, Vol. 9, No. 1, September-October, 1967.
43. Ramamoorthy, C. V. "The Analytic Design of a Dynamic Look Ahead and Program Segmenting System for Multiprogrammed Computers," in Proceedings of the ACM 21st National Conference.
44. Scherr, Allan L. "An Analysis of Time-share Computer Systems." Unpublished dissertation, Massachusetts Institute of Technology, June, 1965.
45. Schmidt, J. W. and Taylor, R. E. Simulation and Analysis of Industrial Systems. Richard D. Irwin, Inc., Homewood, Illinois, 1970.
46. Shemer, Jack E. "Some Mathematical Considerations of Time-sharing Scheduling Algorithms," in Journal of the ACM, Vol. 14, No. 2, April, 1967.
47. Smith, R. L. "The Determination of a Scheduling Rule for a Computing Center." Unpublished Ph.D. dissertation, George Washington University, Clearinghouse Document #665687, November, 1967.
48. _____. "Multiprogramming Under a Page on Demand Strategy," in Communications of the ACM, Vol. 10, No. 10, October, 1967.
49. Stevens, David F. "On Overcoming High-Priority Paralysis in Multiprogramming Systems: A Case History," in Communications of the ACM, Vol. 11, No. 8, August, 1968.
50. Thiess, H. E. "Mathematical Programming Techniques for Optimal Computer Use," in Proceedings of the ACM 20th National Conference.
51. _____, _____, and Rosenberg, R. "Markovian Models ...Analysis of Computer System Behavior," in Proceedings of the AFIPS Spring Joint Computer Conference, Vol. 28.
52. _____. "A Model for Core Space Allocation in a Time-sharing System," in Proceedings of the AFIPS 1969 Spring Joint Computer Conference.
53. Woellner, D. A. "Scheduling the Work Flow in a 3rd Generation Environment," in Data Processing, Vol. XIII.

Share 9 Burn CACM