JOB SHOP SCHEDULING BY MEANS OF SIMULATION AND AN OPTIMUM-SEEKING SEARCH

Dr. James C. Emery
Professor of Industry
Wharton School of Finance and Commerce
University of Pennsylvania
Philadelphia, Pennsylvania 19104

Abstract

The paper discusses a job shop simulator designed for use in the scheduling of a set of jobs. The simulator has been implemented in the form of a Fortran program. The simulator can be used in conjunction with a search program that adjusts priority rule parameters in seeking an improved schedule.

1. INTRODUCTION

Job shop scheduling has been discussed at length elsewhere and need not be elaborated here. (2, 3, 4, 6, 7) Suffice it to say that job shop scheduling represents an enormously complex combinatorial problem that has not been successfully tackled either by traditional methods or by optimizing methods.

The traditional method is to assign a scheduled time for each operation on the basis of a standard lead time that includes both processing and queue time. This approach ignores actual capacities and the interactions among jobs competing for the same production facilities. Accordingly, a schedule determined in this way will be infeasible to some degree. Production control personnel in the shop are then left with the responsibility for making detailed sequencing decisions using the schedule as a loose guide. The resulting performance is (not surprisingly) often unsatisfactory due to excessive job tardiness, low capacity utilization, and high work-in-process inventory.

Some interesting optimizing models have been developed to deal with job shop scheduling. For example, Bowman developed an integer programming model to minimize the total time to complete a set of jobs. (1) His model suffers, however, from being entirely infeasible from a computational standpoint. This becomes all the more true as one enriches the model to include such refinements as order splitting, alternate routing, machine substitution, and combining setups.

The work reported herein was supported in part by a grant from the General Electric Foundation

In job shop scheduling one is thus faced with a problem that is not handled satisfactorily by either traditional methods or by optimizing methods. And yet the payoff from even a modest improvement in scheduling can be very great in terms of better delivery performance, greater throughput, and lower inventory. Scheduling through simulation offers an attractive way of dealing with this problem. (5, 8)

2. SCHEDULING THROUGH SIMULATION

Suppose there exists a set of jobs that must be scheduled. Each job may have one or more operations remaining before the completion of the order. The sequence of work centers ("machines") through which a job flows constitutes the job's routing. The routings for various jobs will, in general, vary widely. For example, one job may go first to a lathe, then a milling machine, and finally a drill press, while another job may go to a bender and then to a punch press. The operation required at each machine normally includes a machine setup and actual run time. A given job may involve work on a single piece or on multiple pieces that are processed in a single batch.

The shop can be scheduled by means of a simulator. Beginning with the existing state of the shop, the flow of work through the shop can be simulated. Upon completion of all jobs in simulated time, the simulated results can be analyzed. Results may be measured in such terms as the total hours of job tardiness and the total time jobs are in the shop (flow time). If the results appear satisfactory, the sequence of events in simulated time can be taken as the scheduled events. If results are not satisfactory, the shop can be simulated again

using different machine capacities or different decision rules. This can continue until a satisfactory schedule is found, or until it is concluded that further search is unwarranted.

The basic simulation cycle is triggered with the completion of an operation. It consists of the following steps:

(1) Determine which machine next finishes an operation.

(2) Assign to the now free machine the highest priority job in the queue.

(3) Move the job that just completed an operation to its next machine, or, if all operations have been completed, remove the job from the shop.

The complete simulation consists of a series of discrete events of this sort. Figure 1 shows a more detailed description of the simulator logic.

3. CHARACTERISTICS OF THE SIMULATED SHOP

The present simulator is based on a shop having the following characteristics:

(1) The shop consists of various machine groups, each of which contains one or more machines. All machines in the group are identical and draw from the same queue of jobs.

(2) "Lap-phasing" is not permitted, i.e., an operation on one machine must be completed for all pieces in a batch before the next operation can proceed.

(3) Once begun, an operation is continued until it is completed.

(4) A transit time may be required to move jobs from a given machine group to any other group. If so, any job processed in such a group cannot begin its next operation until the expiration of the transit time.

(5) Estimated setup and run times are available for each operation of each job. The sum is termed "processing time."

(6) A given job may be released at any time. Until its release, a job is not eligible for assignment to a machine.

(7) A worker is assigned to a unique machine group. Thus, if no jobs are available in a given group its workers cannot be transferred to some alternate group. (The simulator is currently being modified to allow such flexibility.)

(8) If a machine becomes free, the highest priority job waiting in its queue is assigned to it; "hold-off," in which a machine is kept vacant while waiting for a high priority job, is not permitted.

(9) A job to be scheduled may or may not have a previously assigned due date. If a due date is given, a penalty is incurred for each hour of scheduled tardiness (and, if desired, for each hour of earliness). Jobs without due dates are assigned one equal to the simulated completion time (or perhaps with some extra slack for safety). By this means new orders can be introduced into the shop and assigned realistic due dates that recognize existing loads and delivery commitments.

4. PRIORITY RULE

For a given shop capacity (defined in terms of the number of manned machines available within each group over various periods of the day), the only type of decision required during the course of the simulation is the choice of job to assign to a free machine. The resulting schedule is simply the composite of these individual decisions. The effectiveness of the schedule, therefore, depends upon the rule used to evaluate the relative priority of all jobs waiting in the queue of the group to which an available machine belongs.

A great deal of research has been devoted to the choice of priority rule. (Buffa (2) gives an excellent summary.) Most of the rules studied assign a priority value to a given job as a function of such variables as its current estimated processing time, the due date of the job, and the remaining processing time following the current operation. Assigning the job with the shortest current processing time tends to result in good machine utilization at the expense of a relatively large number of late jobs. Carroll finds that his "C/T" rule (roughly, cost per unit delay ÷ current processing time) achieves both good machine utilization and delivery performance. (3)

The priority function used in my simulator is a composite rule. The determination of priority is done in two stages. The first stage performs some preliminary screening to eliminate from further consideration (i.e., for assignment to the currently available machine) those jobs in the queue that are not "critical" according to at least one of six screening criteria (discussed below). The second stage computes a priority value for all jobs surviving the stage-one screening. If, during the first stage, all jobs but one are screened out, the remaining job is assigned to the free machine. If two or more jobs remain for stage two, the job having the highest priority value is selected.

4.1 STAGE-ONE SCREENING

The six screening criteria are:

(1) External priority class

(2) Carroll's C/T (discussed below)

(3) Time the job has been in the current queue

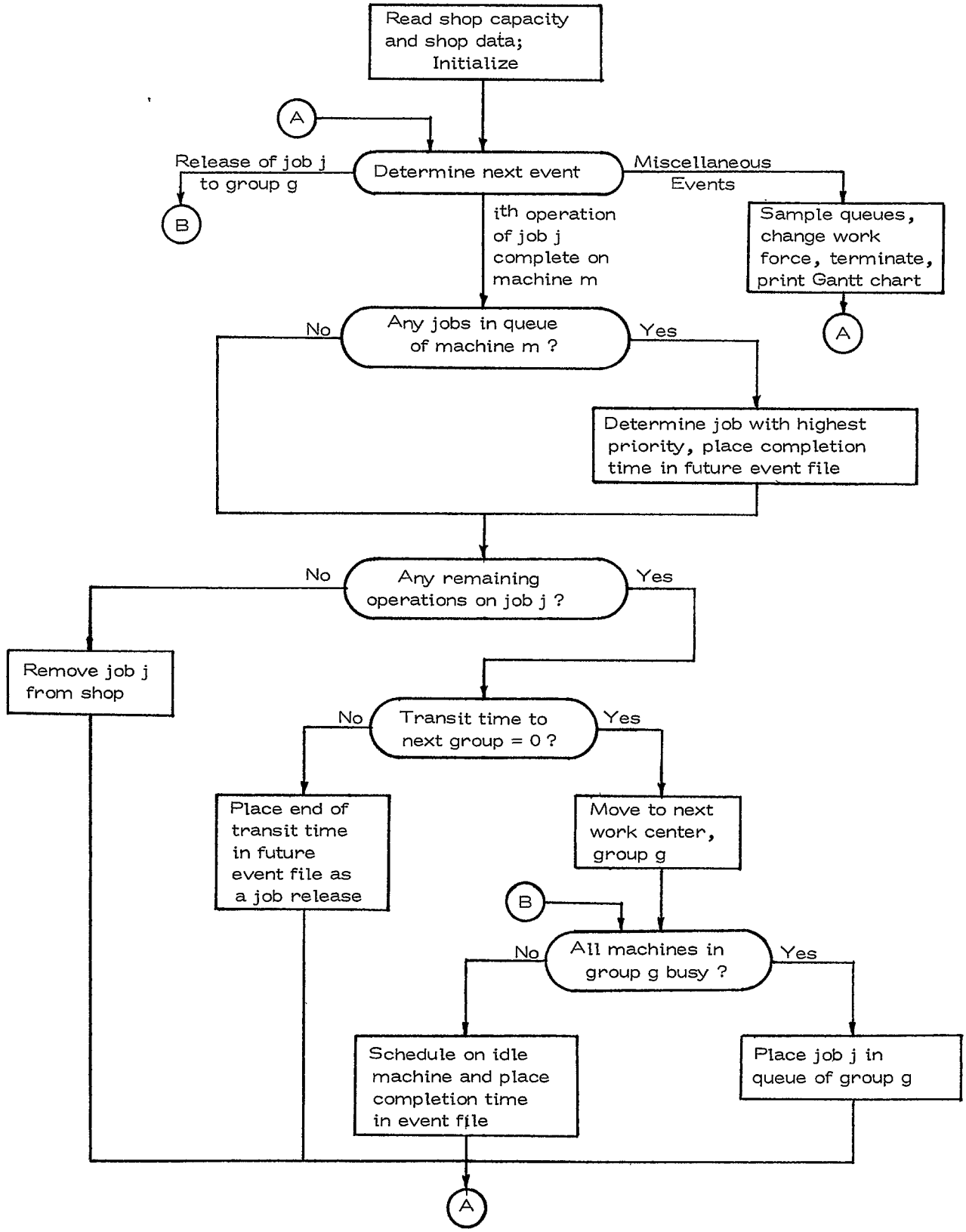


Figure 1. Flow Diagram of Simulator

(4) Ratio of remaining machine and transit time to the processing time of the current operation

(5) Processing time of current operation

(6) Size of queue (in terms of processing time per machine) at the next group to which the job will go upon completion of the current operation

The screening process takes place sequentially in the order shown. Jobs that pass one screen are then screened by the next one. The survivors of the sixth screening then pass to the stage-two priority function. Figure 2 illustrates the approach.

Each screening criterion has two parameters associated with it. The first is a threshold parameter which determines whether the criterion is relevant for a particular set of jobs. Consider, for example, the threshold for the time in queue (TIQ) criterion. The longer a job sits in its current queue, the more critical it becomes. If one or more job has been in the queue longer than the TIQ threshold, then this criterion becomes relevant for screening.

The second screening parameter establishes the tolerance from the most critical job that any other job must fall within, if it is to be eligible for further consideration. Continuing with the TIQ criterion, suppose that the TIQ threshold is 10 hours. Suppose also that the maximum time that any job has been in the current queue is 20 hours. Since this maximum exceeds the threshold of 10 hours, the TIQ criterion thus becomes relevant for screening out jobs that fall within a specified tolerance from the critical value of 20 hours. The tolerance is expressed as a decimal fraction of the most critical value. If the TIQ tolerance is, say, .8, any job with a TIQ of 16 hours or more (.8 x 20) is eligible for further consideration (i.e., it is passed on to the next criterion, remaining time/current processing time). All other jobs are eliminated and cannot be assigned to the free machine. Figure 3 illustrates this example. All of the other criteria work in a similar manner (except in the case of current processing time and next queue criteria, the lower the value, the higher the priority).

It may be useful to describe the six screening criteria in more detail:

External priority class. Each job can be assigned a priority code that identifies its external priority class. The higher the number, the higher the priority. For jobs having previously assigned due dates, their C/T ratio is multipled by 2^p, where p is the priority code.

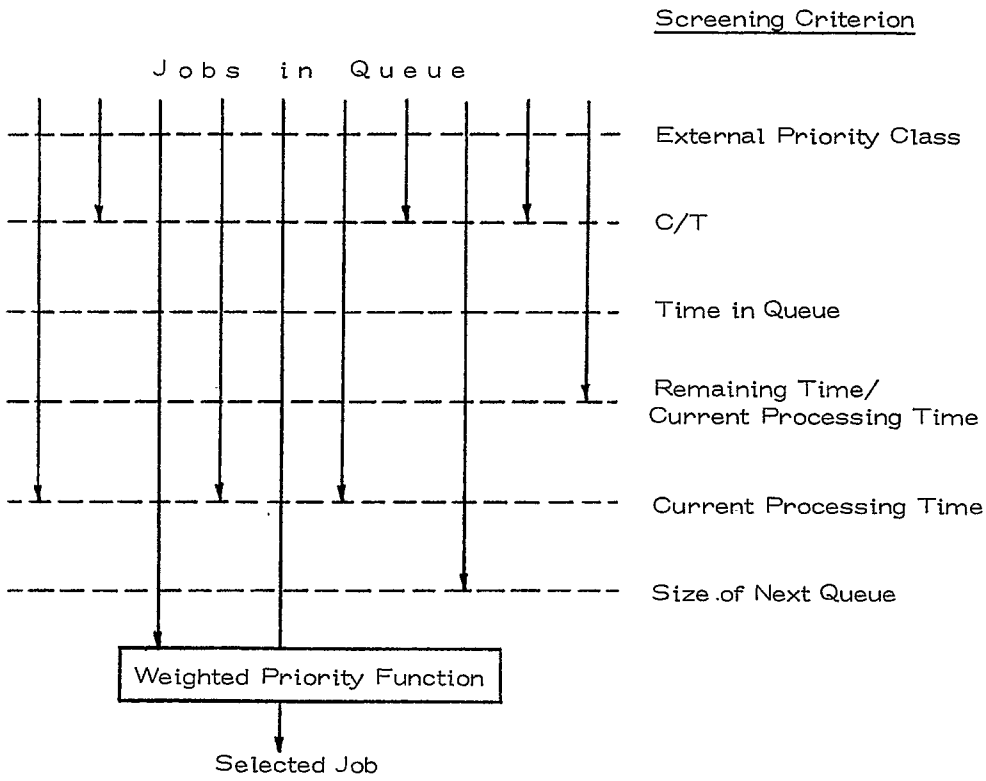
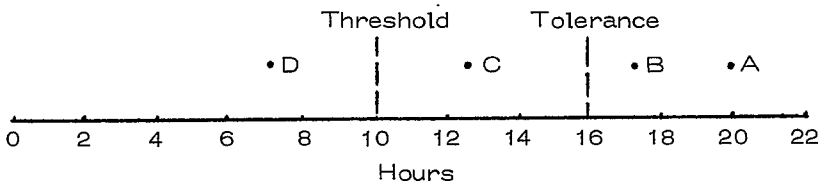


Figure 2. Job Priority Determination



Threshold to be relevant = 10 hours
Maximum TIQ = 20 hours
Tolerance limit = 16 hours

Therefore, Jobs A and B remain in competition for assignment to the free machine, while Jobs C and D do not.

Figure 3. Screening Technique

If the maximum ratio is less than the C/T threshold value, jobs with due dates are treated the same as those without. In this event the only eligible jobs in a queue are those falling within the highest priority class. For example, if class 3 is the highest class represented in a queue, only a class 3 job can be assigned. (Of course once class 3 jobs have been scheduled, lower class jobs can then be scheduled.) High external priorities must be assigned judiciously, or else they cease to have any meaning. However, a high priority class may be justified for jobs using especially expensive material (in order to reduce lead time and work-in process inventory) or a particularly important job. Conversely, a low priority might be used for batches being produced for inventory in order to level production loads.

C/T. Carroll's C/T rule is used:

$$\text{Priority} = \frac{c_i}{t_i}$$

where t_i is the current processing time of the i^{th} job

$$\text{and } c_i = \begin{cases} \frac{w_i - s_i}{w_i} & \text{when } w_i > s_i > 0 \\ 1 & \text{when } s_i < 0 \\ 0 & \text{when } s_i > w_i \end{cases}$$

where s_i is the slack time of the i^{th} job and w_i is its wait time.

Slack time equals the job's due date minus the sum of the current time and all remaining processing and transit times for the job. In other words it is the maximum amount of time the job can sit idle in queues without experiencing tardiness. Expected wait time is the amount of idle time that the job is expected to experience in its remaining queues. Wait time is estimated dynamically as the simulation proceeds, and is based upon the processing time currently in each of the job's remaining queues.

Time in queue. This criterion is used in order to insure that jobs having long operation times will not be unduely discriminated against in favor of jobs with short times. This is not necessary in the case of jobs with due dates, since the reduction in slack as a job sits in a queue will insure that it eventually is given a high priority. However, for a job being scheduled in order to assign a due date, the priority rule should guard against an excessively long lead time.

Remaining time/current processing time. In a queue network it is important to guard against bottleneck operations that prevent work flowing to downstream operations. The work load flowing downstream is maximized when one chooses the job having the greatest downstream load per unit of current processing time.

Size of next queue. In order to level the load on various machine groups, it is desirable to schedule a job that will have its next operation performed in a machine group that currently has a light load in its queue.

4.2 STAGE-TWO WEIGHTED PRIORITY FUNCTION

Jobs surviving the screening process are evaluated in terms of a weighted priority function. The job having the highest value is assigned to the free machine. The weighted function is:

Priority value job i =

$$w_1 q_i^2 + (w_2 r_i + w_3)\frac{1}{t_i} + w_4 \frac{1}{(1 + p_i)} + w_5 \frac{c_i}{t_i}$$

where w's are weights for the various criteria

q_i is the time spent in current queue by the i^{th} job

r_i is remaining processing plus transit time for the i^{th} job

t_i is the processing time of the i^{th} job on the current machine

p_i is the processing time per machine in

the group that performs the next operation on the i^{th} job

c_i is the value used in the C/T screening (see above)

5. SEARCH FOR IMPROVED SCHEDULES

The key to the effectiveness of the priority rule is, of course, the value of the various parameters. There are a total of 15 parameters, three for each of five criteria. Two parameters --the threshold and the tolerance--are associated with each criterion used in the screening process, and a weight parameter is associated with each criterion in the weighted priority function.

The priority rule can be made a pure one with respect to any one of the criteria simply by a suitable selection of parameter values. For example, a first-come, first-serve rule will result if the screening threshold and tolerance for the time in queue criterion are set equal to 0 and 1, respectively, while all other screening thresholds are set so that they cause no screening. Thus by suitable choice of the parameter values the priority rule can vary from a simple rule that considers only one criterion to a composite rule that considers all criteria.

When searching for an improved schedule, a planner can run the simulator using different sets of parameter values. This can continue until he judges that any improvement from further simulations cannot justify the incremental cost of the search. If the best schedule found with a given shop capacity is not satisfactory, a search can be made with a different capacity.

This is fairly awkward for the planner if very many simulation trials are to be made. Therefore, the simulator includes an optimum-seeking search program that automatically varies parameter values.

An obvious requirement for such a program is an objective function that provides a single index of the relative merit of each alternative schedule generated by the simulator. The one used is a weighted sum of the total tardiness and earliness of jobs having assigned due dates and the total flow time of jobs not having due dates (where flow time is the sum of the time the jobs are in the shop, from their release until their completion. The search program seeks to minimize this function. This tends to reward on-time completion of jobs with due dates, high capacity utilization, and a reduction in lead time for jobs being assigned a due date. The objective function used in a particular shop depends, of course, on the specific nature of the shop--the penalty for tardiness (and earliness), the unit cost of production capacity, the value of inventory, and so forth.

Each parameter being searched is adjusted over a specified interval and with a specified mesh. A beginning value is given that falls within the search interval. The search proceeds toward the lower point of the interval. The search continues in this direction until the lower point is reached or until the search is aborted because an improvement in the schedule is not found within a specified number of simulation runs. Additional runs may then proceed from the beginning point toward the upper bound of the search interval.

For example, the time in queue threshold could be varied from 5 hours to 25 hours, beginning with a value of 10 hours. The search mesh could be specified as, say, .8. This is multiplied by the previous parameter value in order to determine the value for the next trial. Therefore, the threshold parameter would have a value on successive runs of 10, 8, 6.4, and 5.12 (if not aborted earlier due to a failure to improve the schedule within a specified number of tries). Upon termination of the search in the downward direction, the search may continue with the values of 12.5 (10/.8), 15.625, etc. The search in the upward direction will not take place if the number of improvements found in the downward search exceeds a specified limit.

The search continues in this manner for all parameters being searched. (Any parameter value can be held constant if desired.) Upon completion of a pass over all parameters, additional passes will be performed if so specified. If changes in a parameter value fail to result in improved schedules, the parameter can be held constant during subsequent passes.

This "sectioning" search strategy is obviously rather simple minded. Such a strategy may be inefficient or incapable of moving toward the optimum of certain functions. (9) However, the unusual nature of the function being optimized makes it difficult to use some of the more sophisticated search strategies.

The function remains constant over ranges of parameter values. Gradient search techniques, therefore, cannot be used. Furthermore, it is not well behaved with respect to a change in a parameter value; increasing the value for a given parameter may, for example, lead first to an improvement, then a worsening, and again an improvement in the schedule. As a consequence, previous trials supply relatively little information about the direction in which parameters should be changed.

The vastness of the parameter space and the irregular nature of the function being optimized make it impossible to search more than a minute fraction of the alternatives. But suppose it would be feasible to run, say, 100 simulations each time the shop is scheduled. Obviously, the evaluation of 100 alternative sets of parameter values, out of the very large number that exist, will not reveal the true optimum schedule. But the best schedule out of a sample of 100 is likely to be a fairly good one, not too far from the optimum. Furthermore, if, as seems to be the case, the best set of parameter values tends to be rela-

tively stable with respect to changes in job mix and shop conditions, then the process tends to converge over time toward better and better parameter values. Once a good set of parameter values has been found, routine scheduling runs can be made without any search at all, based on the parameter values found during the last search. The parameters could be re-evaluated relatively infrequently--for example, when some idle computer capacity is available--in order to adapt to changes in job or shop characteristics.

The feasibility of running the simulator for anything like 100 trials depends critically on execution time. Considerable attention has been paid to make the simulator fast enough to permit multiple trials. On an IBM 360/75 it schedules at the rate of about 300-600 operations per second (depending mainly on average queue size). A shop with 10,000 operations to be scheduled (for example, 1,000 jobs with an average of 10 operations each) would take perhaps about 25 seconds per run. One hundred runs would therefore take about 40 minutes. This would not be infeasible if the search program were run relatively infrequently. The computational cost of scheduling without search is likely to be negligible--in the order of $5.00 to $10.00--and therefore can be run quite frequently.*

One of the reasons that the simulator is relatively fast is that all data are kept in core storage. Job data (i.e., release time, due date, external priority, and routing) are read at initialization and remain in core throughout a search. Frequent accesses to secondary storage for these data would obviously add greatly to execution time.

The objective of keeping all data core resident places a high premium on storing the data in as compact a form as possible. The largest internal file is for job data. These data are stored contiguously with a pointer to the beginning of the data pertaining to a given job. Job queues are stored as lists in order to avoid having to reserve contiguous storage for each queue sufficient for the maximum size that it might reach at any point during a simulation.

Despite such compression, the core storage requirements for even a moderate-sized shop remain formidable. A shop with 100 machine groups, 200 machines, 1000 jobs, and 10,000 operations requires about 123K bytes of storage.** The program adds another 44K bytes (although the use of program overlays could reduce this by about a third without increasing processing time significantly). Scheduling by means of the simulation program obviously calls for a large computer, but only for a relatively short time. A remote batch entry system that allows the sharing of a large computer appears to be an entirely feasible way of providing the necessary capacity.

The simulator is written in Fortran instead of a simulation language in order to increase speed. Fortran was also used because of its generality in handling the complex priority rule and other complicated logic involved in the simulator. Finally, the widespread availability of Fortran translators permits the simulator to be run on a variety of machines.

6. OUTPUT FROM THE SIMULATOR

A number of output options exist. Among them are the following:

(1) A Gantt chart of the schedule generated by the simulator (see Figure 4). When a search is made, normally only the best schedule is displayed.

(2) A tabular schedule arranged in both machine and job sequence

(3) Summary data about each machine group (e.g., number of operations scheduled, total processing time scheduled, machine utilization, average wait time, and size of the maximum queue)

(4) Queue status at a specified sample interval

(5) Graphical display of the results from each trial of a search process (see Figure 5)

7. APPLICATION OF THE SIMULATOR

Only very tentative results are presently available from the simulator. The skimpy evidence that is available, however, suggests that the compound priority rule leads to a worthwhile--if not spectacular--improvement over the best available simple rule. In the one, extensive analysis that has been made, a sample of 500 jobs having 2,400 operations led to the comparative results shown in Table 1.***

The real test of the simulator will come when it has been applied in scheduling an actual shop. An attempt is currently being made to do this for a modest-sized shop. The principal difficulties have been in collecting data and working out suit-

* Other costs, such as data collection, may not be negligible. Ideally, the scheduling program should have access to an on-going data base that is maintained--perhaps in an on-line system--for such purposes as payroll and job location.

** The incremental core storage requirements are 66 bytes per group, 6 bytes per machine, 52 bytes per job, and 6 bytes per operation.

*** Of the 500 jobs, 367 had due dates and the remainder did not. The shop consisted of eight machine groups and 14 machines.

GANTT CHART OF GROUP-MACHINE

Day/sft/hr	1- 1	2- 1	2- 2	2- 3	3- 1	3- 2	4- 1	5- 1	5- 2	5- 3	6- 1
1 1 0.	1811	111			2411	311	5011	1611			2211
	I	I	5012		I	I	2811	I			I
	I	I	I	711	I	I	I	I			2011
	2212	I	I	I	I	I	1711	I			I
	I	I	I	I	I	I	I	I	2711		3611
	I	2812	1712	I	2111	I	411	I	I		I
	I	I	I		I	1211	I	I	I	312	I
	I	I	I		I	I	I	I	I	I	I
1 1 4.	2412		I	412	I	I	6011		I	I	1511
	2813	2413	I	I	I	I	6211	1512	I	I	4011
	I	I	6012	I		I	I	I	I	I	I
	1713	I	I	2814	2712	I	I	I	2911	I	113
	3011	I	I	I	I	3311	I	I	I	1212	I
	I	2112	I	I	I	I	4711	I	I	I	1011
	3711	I	1911	I	I	I	I	I	I	I	I
	I	I	I		I	2414	I	I	I	I	3312
1 2 0.	I	I	I		I	I	3012	I	I	I	I
	4411		I		I	4211	I	I	I	I	4712
	I				3911	I	I	3111	I	I	I
	I	3013			I	I	1912	I	2713	I	I
	4012	I	1913	2511	I	I	2415	I	I	I	I
	I	I	I	I	I	4511	812	I	I	4212	1213
	I	I	I	I	I	I	I	I	I	I	I
	I	3912	I	I	4811	I	I	I	I	I	712
1 2 4.	1513	I	2416	I	I	I	I	I	I	I	I
	I	I	I	I		I	I	114	4611	I	I
	713		I			I	I	I	I	I	1411
	I						I	I	I	4812	I
	I						2714	I	I	I	I
	I	2715					1914	I	I	I	I
	5711	I	3511		5611		I	I	I	I	I
	I	I	I		I	1915	7011	I	I	I	I
1 3 0.	I	I	I	7012	I	I	2113	2417	5511	I	I
	I			I	I	I	I	I	I	I	I
	I	2114		I	I	5811	2716	I	I		2816
	I	I		I	6111	I	I	I	I	2817	4911
	I	I		I	I	I	4813	I	I	I	I
	I	I			I	I	I	I	I	I	5411
	7013	I	4814		I	2717	6511	5812	I	I	5612
	I	I	I	6512	I	I	4412	I		I	I
1 3 4.	1916		I	I	5412	I	I	I	6112	I	I
	I		I	I	I		I	I	I	I	116
	I	4111	I	I			I	I	I	I	I
	1412	I	I	I	1917		413	I	I	5613	I
	I	I		I	I		I	I	I	I	I
	I	I			I		I		I	I	5413
	6711				I		I		I	I	I
	3513				I		I		I	I	5311
2 1 0.	I	414			7211		4112	6611		I	I
	I	I	4113		I		3014	I			I
	I	I	I		I		I	I			I
	7111		I		I		I	I			6113
	I	3015	5111		I		4013	I			I

Job No. Operation

Figure 4. Schedule Printed in Gantt Chart Form

```
SEARCH FOR IMPROVED SCHEDULE (EACH LINE REPRESENTS A COMPLETE SIMULATION)

PARAMETER   OBJ FUNC  25.00     26.00     27.00     28.00     29.00     30.00
NO.  VALUE    1000.    .----.----.----.----.----.----.----.----.----.----.----.
              27.8890                                 *
 .
 .
 .
 4   5.834    29.7066                                                  .
THRT   L

 4  12.618    27.8898                                 .
 4  18.555    26.9398                       *
THRT   U

 5   0.500    27.3287                           .
DLRT   L

DLRT   U

 6   0.001    27.0404                        .
 6   0.001    27.0327                        .
  W2   -

 6   0.003    26.9561                        .
 6   0.004    26.8744                       *
 6   0.006    26.9369                       .
 6   0.009    26.7642                      *
  W2   U

 7   0.680    26.7338                     *
 7   0.500    27.0826                         .
THPT   L
 .
 .
 .
13   0.435    26.3257              *
13   0.473    26.3795              .
13   0.514    27.2453                         .
THTC   -

14   0.408    26.7948                      .
14   0.300    26.9746                        .
DLTC   -

14   0.882    26.7185                     .
DLTC   U

NO. SIMULATIONS =    56
NO. OPS. SCHED. =     135296
BEST VALUE = 26325.7

PARAMETERS AND THEIR VALUES
      1  THTQ    8.400      2  DLTQ    0.600      3   W1    0.001      4  THRT
      7  THPT    0.680      8  DLPT    1.500      9   W3    2.059     10  THNQ
     13  THTC    0.435     14  DLTC    0.600
```

Figure 5. Output From a Search Program

	Priority Rule	Objective Function* (000 omitted)
(1)	Random selection (the best of 2 runs)	58.8
(2)	Minimum processing time of current operation	56.1
(3)	First-come, first served	45.3
(4)	Carroll's C/T rule (with jobs having no due date selected on the basis of minimum processing time)	29.9
(5)	Compound rule with best set of parameters	26.3

Table 1. Results from Simulations Using Different Priority Rules

able operating procedures. Some modifications are also being made in the simulator itself to tailor it to the specific characteristics of the shop. For example, it is necessary to permit some flexibility in shifting workers from one machine group to another. The program is therefore being modified to recognize both machine and labor constraints.

REFERENCES

(1) Bowman, E. H., The schedule sequence problem, Operations Research, Vol. 7 (September 1959), pp. 621-24.

(2) Buffa, E. S., Production-inventory Systems--Planning and Control, Irwin, Homewood, Illinois, 1968.

(3) Carroll, D. C., Heuristic Sequencing of Single and Multiple Component Jobs, unpublished Ph.D. dissertation, Sloan School of Management, M.I.T., Cambridge, Massachusetts, 1965.

(4) Emery, J. C., An approach to job shop scheduling using a large-scale computer, Industrial Management Review, Vol. 3 (Fall 1961), pp. 78-96.

(5) Reiter, S., A system for managing job-shop production, Journal of Business, Vol. 34, No. 3 (July 1966), pp. 371-93.

(6) Rowe, A. J., Application of computer simulation to sequential decision rules in production scheduling, Proceedings of the Eleventh Annual Industrial Engineering Institute, University of California, 1959.

(7) Rowe, A. J., Toward a theory of scheduling, Journal of Industrial Engineering, Vol. 11, (March 1960), pp. 125-36.

(8) Steinhoff, Jr., H. W., Daily system for sequencing orders in a large-scale job shop, Chapter 10 in Readings in Production and Operations Management, E. S. Buffa (ed.), Wiley, New York, 1966.

(9) Wilde, W. J. and C. S. Beightler, Foundations of Optimization, Prentice-Hall, Englewood Cliffs, New Jersey, 1967, pp. 296-97.

BIOGRAPHICAL DATA

Dr. James C. Emery is a Professor of Industry at the Wharton School of Finance and Commerce, University of Pennsylvania. Prior to his current affiliation, he was an Assistant Professor at M.I.T., where he earned a Ph.D. in Industrial Management in 1965. He currently teaches primarily in the field of computer technology and management information systems. His research interests include data management and man-machine planning systems. He has extensive industrial experience as a consultant and former full-time employee of Westinghouse Electric Corporation. Much of this experience is in the field of production and inventory control.

* The objective function used was 5•(sum of hours tardy) + .5•(sum of hours early) + 1•(total flow time of jobs without a due date).