AIIE Transactions

ISSN: 0569-5554 (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uiie19

Centralized Scheduling and Priority Implementation Heuristics for a Dynamic Job Shop Model

Rosser T. Nelson , Charles A. Holloway & Ruby Mei-Lun Wong

To cite this article: Rosser T. Nelson , Charles A. Holloway & Ruby Mei-Lun Wong (1977) Centralized Scheduling and Priority Implementation Heuristics for a Dynamic Job Shop Model, AIIE Transactions, 9:1, 95-102, DOI: 10.1080/05695557708975127

To link to this article: https://doi.org/10.1080/05695557708975127

Centralized Scheduling and Priority Implementation Heuristics for a Dynamic Job Shop Model

ROSSER T. NELSON
UCLA
Los Angeles, CA 90024

CHARLES A. HOLLOWAY
Stanford University
Stanford, CA 94305

RUBY MEI-LUN WONG
Stanford University
Stanford, CA 94305

Abstract: The problem considered is the scheduling of a job shop with job due dates, intermittent job arrivals, and statistical processing times. Centralized scheduling uses a sequence of static problems for generating priorities at review times. A multi-pass heuristic program, which has proven effective in earlier research, is applied to the up-dated static scheduling problem at each review time. A procedure is proposed for implementing priorities on the shop floor between review times. The procedure is expressly designed to integrate the scheduling of newly arriving jobs to modify the schedule. In simulation experiments using tardiness statistics for evaluation, centralized scheduling and the proposed implementation procedure proved to be an extremely effective combination. Comparison with another procedure that gives the centralized schedule precedence over new arrivals indicates the importance of the implementation procedure when periodic centralized scheduling is used in a dynamic situation.

■ Job shop scheduling has been the focus of a substantial amount of research over the last two decades (for a review of much of this work see [1]). An explanation for this activity is the combination of the theoretical challenge posed by the combinatorial nature of the problem and the wide range of practical applications. Much of the work has concentrated on problems with a static arrival process (all jobs available at the beginning of a scheduling period) and deterministic processing times. The motivation for considering this class of problems is that realistic job shop scheduling problems, characterized by dynamic (intermittent) arrivals and statistical processing times, might, for practical reasons, be modeled as a sequence of static problems.[1] Other research has concentrated on development of scheduling procedures directly applicable to job shops with dynamic arrivals and stochastic processing times. These procedures are characterized by dispatching rules based on local information. The use of only local information allows intermittent arrivals and stochastic variations in processing times to be directly incorporated into the priority calculation.

In [3] the authors investigated the use of schedules generated using static arrivals and deterministic processing times when actual processing times are statistical in nature. The schedules were generated using expected processing times; the simulated implementation of the schedule on the shop floor used actual processing times taken from a specified processing time distribution for each operation. The experiments compared centralized schedule generation with four dispatching rules. From the standpoint of possible application, the key results were that the relative performance of the scheduling procedures on job tardiness criteria was largely independent of the variance of the processing time distributions and that, for a wide variety of conditions, schedules generated centrally using expected processing times outperformed those generated using dispatching rules. The results also indicated the importance of the procedure used for implementing centrally generated priorities on the

[1] Peters [7] has applied this idea to a three machine flow shop, using a branch-and-bound algorithm for minimizing maximum flow time to solve the static problem.

Received May 1975; revised April 1976; revised August 1976.

This research was principally supported by the Office of Naval Research under Contract N00014-69-A-0200-4004, Task NR-047-003, partially by the Western Management Science Institute under a grant from the Ford Foundation, and partially by the Graduate School of Business, Stanford University. Reproduction in whole or in part is permitted for any purpose of the United States Government.

The authors wish to acknowledge the helpful comments of a referee on a previous draft of the paper.

shop floor using actual processing times. Two implementation procedures led to quite different results in the static problem.

This paper focuses on the development and testing of a scheduling procedure for *dynamic* problems with *statistical* processing times. Guided by the results of the earlier research, the proposed centralized priority generation procedure uses expected processing times. In view of the earlier results regarding implementation in the static problems, two implementation procedures are evaluated under varying shop conditions. The implementation options represent two identifiable strategies for using the centralized schedule in a dynamic environment. These options are compared with a dispatching rule suited for direct implementation in a dynamic problem. Based on previous research, the minimum SLACK rule[2] was chosen as representative of an effective due-date oriented priority rule utilizing only local information. The purpose of this paper is to investigate the potential usefulness of centrally generated schedules based on static arrivals and deterministic processing times and the sensitivity of the results to implementation schemes.

The Static Arrivals, Deterministic Processing Times Problem

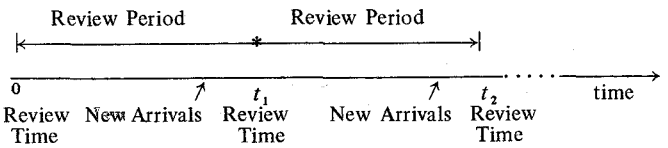
A prerequisite to such an investigation is a procedure for obtaining good solutions to the problem with static arrivals, deterministic processing times and due dates. Although the problem can be formulated as an integer programming problem with a variety of performance measures for the objective function, its size precludes solutions for even relatively small job shop models. Reference [2] describes a procedure, HSP, which has proven to be computationally effective for job shops with static arrivals, deterministic processing times and due dates. (See Appendix A for a brief overview of the procedure.)

The application of HSP requires that provision be made for the case where the specified problem is infeasible (i.e., it is not possible to meet the due dates with the given processing times and job precedence requirements). Previous computational experience has indicated that an efficient strategy is to relax the problem constraints to obtain a problem that is known to be feasible, and then iteratively reapply violated constraints to seek improved solutions. The strategy of relaxation can be applied in a number of ways with the choices taking into consideration the criteria used by management for schedule evaluation. References [5] and [6] describe relaxation strategies for situations in which the resource constraints or job precedence constraints can be relaxed, as through the application of overtime or multiple transports of partial job lots. Reference [4] describes an interactive program that can be used when implicit tradeoffs exist among a number of criteria. For the purpose of the experiments in this paper, we assume that the goal is to minimize maximum tardiness. The relaxation procedure uses the completion times of jobs in the schedule obtained from the SLACK priority rule to establish a set of relaxed due dates for initial use with HSP. These relaxed due dates are then tightened on each iteration to guarantee that maximum tardiness is decreased each time HSP finds a feasible solution. A claim of optimality for the procedure is precluded by the fact that HSP is not guaranteed to find a feasible solution if one exists. This requires that we rely on empirical comparisons with other alternatives in evaluating performance. In addition to the evidence in [2] on the effectiveness of HSP, the iterative procedure used here has previously been applied [5] to eight test problems with an average of 48% improvement in maximum tardiness over the SLACK schedules.

[2]SLACK chooses from jobs in queue that job with the smallest value of [job due date–current time–remaining processing time].

Priority Generation and Implementation for the Dynamic Problem

Use of solutions to static problems in a dynamic problem offers two primary options. The first is to update the system and reschedule each time a new job arrives. In instances where jobs only arrive infrequently, this may be a good strategy and the dynamic aspects of implementation disappear since every new arrival signals a new static problem. In this case schedules for the current static problem are implemented. The second option is to use periodic review times as suggested by the diagram below.



A sequence of review times t_i determine a sequence of review periods which may be fixed or variable in length. At each review time the actual state of the system is updated to reflect current shop status. This would include any impending jobs with known arrival times during the subsequent review period. The current shop status, which includes expected processing times for the operations, is the information assumed to be available for making scheduling decisions for the next review period.

The procedures investigated in this paper use the solution to the static arrivals, deterministic processing times problem to develop a priority list of the jobs known at the review time for each machine. Each priority list includes the sequence of operations scheduled on the machine and the expected arrival time of each operation at the machine. Operations that arrive at the machine during the period that are not on the priority list generated at the beginning of the period are placed on a new job list for the machine. The priority order within this new job list is based on expected SLACK, i.e., highest priority to the operation with the smallest value of [job due date–current time–expected remaining processing time on job].

We consider two procedures for implementing these priorities:

Implementation IP1

When a scheduling decision is to be made on a machine, select from among the operations that are available the operation that is highest on that machine's priority list. If there is no operation available from the priority list, schedule the available operation that is highest in priority on the new job list. IP1 confines attention to operations that are available and selects among these in accordance with (a) the priority list, or (b) the new job list. Thus, IP1 does not permit enforced idle time in the schedule, i.e., a machine is never held idle to wait for a nonavailable operation when there is any work available. This corresponds to the nondelay implementation strategy investigated in [3].

Implementation IP2

Let t = the time at which a scheduling decision is to be made.

Let t_e = the expected arrival time of the highest (unfinished) priority operation on the machine's priority list.

At time t, compute the expected slack S_p of the highest (unfinished) operation on the machine's priority list and the expected slack S_N of the highest (unfinished) operation on the machine's new job list.

Case 1 ($S_p \leqslant S_N$ *or no operations on new job list*)

Schedule the highest operation on the machine's priority list if it is available at time t.

Otherwise:

i. If the shortest expected processing time among available operations is $\leqslant (t_e - t)$, schedule the available operation with the largest expected processing time $\leqslant (t_e - t)$.

ii. If the shortest expected processing time among available operations is $> (t_e - t)$, schedule the operation on the new job list with smallest expected processing time. If the new job list is empty, do not schedule an operation to begin at time t.

Case 2 ($S_N < S_p$ *or no operations on priority list*)

Schedule the highest operation on the machine's new job list at time t.

IP2 differs from IP1 in two respects. First, it is designed to integrate the scheduling of old and new jobs. Second, it allows for delays and selective use of enforced idle time in the schedule. The first step is to use expected slack as a criterion for deciding whether the priority list generated at the beginning of the period or the new job list should receive primary attention in the scheduling decision being made. The former case (Case 1) assigns the highest priority job from the priority list if it is available at time t. If it is not available, the expected time until that operation's arrival ($t_e - t$) is used to attempt to schedule an available operation to reduce the enforced idle time that would be incurred if the machine were left idle until the arrival of the highest priority operation. Step i attempts to make an assignment such that the expected time that the machine will be available is before, but as close as possible to, the expected arrival time of the highest priority operation. If such an assignment is impossible, step ii attempts to assign the operation from the new job list such that the expected time that the machine will be available is as close as possible to the expected arrival time of the highest priority operation. If there is no operation on the new job list, enforced idle time is incurred on the machine until the next operation arrives and a new scheduling decision is made. In Case 2 where the initial step indicates that the new job list should receive primary attention at time t, the highest priority operation from the new job list is scheduled to begin processing at time t. If the scheduling has proceeded to a point where there are no operations remaining on a machine's priority list from the original jobs, operations are scheduled for new jobs on a minimum expected slack basis.

Although there are several details of the two implementation schemes for which alternative choices are readily identified, they are designed to represent the "natural" implementation of two extreme strategies. IP1 corresponds to a non-delay strategy in which centrally scheduled operations always have precedence over new arrivals. IP2 corresponds to a delay strategy in which priority between centrally scheduled operations and new arrivals depends upon the SLACK of the highest priority operation in each category. Results with these strategies as well as suggestions for future research are discussed in upcoming sections.

Illustrative Example

We conclude this section with a detailed example to illustrate the procedure. The data for the example is as follows:

Job	Arrival Time	Routing and Processing Times	Due Date
1	0	D(7,9) A(3,3) B(2,2) E(1,1)	26
2	0	C(20,25) A(2,2) B(2,3)	48
3	0	C(4,3) F(3,4) E(10,10)	34
4	0	A(10,10) F(3,4) B(1,1) D(8,8)	44
5	0	E(1,1) B(3,2) C(6,5) D(1,1) E(1,1)	24
6	0	E(22,25) A(3,2)	50
7	0	E(5,5) D(7,6) A(13,11)	50
8	5	B(9,11) F(1,1) A(6,6) D(1,1)	39
9	15	B(1,1) A(1,1) C(6,4) F(8,8)	47
10	25	F(9,5) B(3,5)	49
11	35	F(3,2) B(11,10) D(10,12)	83
12	45	F(8,10) B(3,4)	67

Letters A through F are machine identifiers. For each operation, the first number in parentheses is the expected processing time. The second number is the actual processing time sampled from a binomial distribution with mean (np) equal to the expected processing time and with the binomial parameter $p = 0.5$. The due date of each job was set equal to arrival time plus two times the total expected processing time of the job. Jobs 1 through 7 are available at time 0 and are assumed to be the only jobs known of at the beginning of the scheduling period.

We summarize the results of applying the priority generation procedure to the example at time 0:

Priority Generation at t = *0*

The SLACK schedule obtained using the due dates, routings, and expected processing times for jobs 1 through 7 results in maximum tardiness of 10. Relaxed due dates were established from the SLACK schedule to initiate the iterative procedure with HSP for seeking improved solutions for maximum tardiness. This led to a schedule with maximum tardiness of 0.

The Gantt chart for the HSP schedule is given in Fig. 1, together with the corresponding machine priority lists. The sequence of job numbers on each machine is simply the sequence from the Gantt chart. The expected arrival time t_e for each operation is the completion time of the previous operation on the job in the Gantt chart.

Machine A		Machine B		Machine C	
Job Number	t_e	Job Number	t_e	Job Number	t_e
4	0	5	1	3	0
1	7	4	13	5	4
2	30	1	13	2	0
7	14	2	32		
6	41				

Machine E		Machine E		Machine F	
Job Number	t_e	Job Number	t_e	Job Number	t_e
1	0	5_1	0	3	4
7	6	7	0	4	10
5	10	3	7		
4	14	5_2	15		
		1	16		
		6	0		

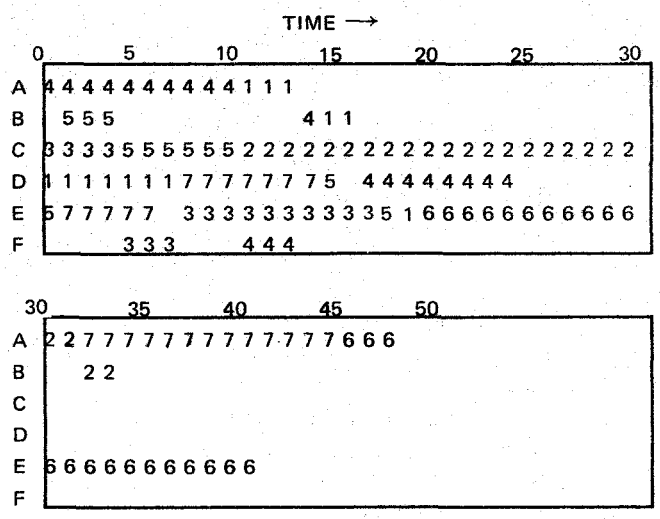


Fig. 1. Gantt Chart for Schedule to be Implemented.

Priority Implementation:

The priorities were implemented with actual processing times using procedures IP1 and IP2. Scheduling with the minimum SLACK single-pass priority procedure[3] was included for comparison. The resulting tardiness statistics are given in Table 1. The IP2 Gantt chart that resulted in zero tardiness is included as Appendix B.

Table 1: Tardiness statistics.

Tardiness Measure	Implementation Procedure		
	SLACK	IP1	IP2
Percent of jobs tardy	33%	33%	0%
Total tardiness	28	44	0
Maximum tardiness	12	18	0

Evaluative Experiments

In this section we summarize the design and results of experiments used for initial evaluation of the procedure described in Section II. Flow charts, listing, and documentation of the (GPSSV) simulation program are given in reference [8].

Simulated Environment. The problem employed for evaluation consisted of 6 machines with 7 jobs available at the beginning of the scheduling period. The data for these jobs is given in Appendix C. One replication of the simulation consisted of one review period and continued until all original jobs and all new jobs that arrived during the review period were completed. New arrivals were generated throughout to prevent a bias in the tardiness statistics due to deteriorating load near the end of the run. The characteristics of new jobs were generated internally by the simulator in accordance with prescribed parameters and distributions (see Appendix C for details).

Experiments. The experimental variables employed were (1) the number of new job arrivals during the review period, (2) the processing time distributions, and (3) the priority implementation procedure.

(1) The ratio N = number of new jobs/number of original jobs was used to explore the sensitivity of the performance of the implementation procedures to the number of new arrivals during the review period. This is viewed as an important factor because it relates to the spacing of the review times in a dynamic problem. The three values of N employed are 0, 3/7, and 1; the

[3]The SLACK priority was computed with expected processing times as would be necessary on the shop floor.

first case being the static problem with only the original jobs. These values span the range from 100% to 50% of the jobs being known by HSP in centralized scheduling.

(2) Actual processing times were generated as the operations were assigned to machines in the simulation. Three distributions were employed:
   (i) a uniform distribution over the range 80% to 120% of expected processing time for the operation.
   (ii) a binomial distribution with $p = 0.5$, symmetric about the mean (np) equal to expected processing time for the operation.
   (iii) a binomial distribution with $p = 0.1$, mean (np) equal to expected processing time, and skewed toward longer processing times.

(3) The priority implementation procedures used were SLACK, IP1 and IP2. SLACK was chosen since it can easily be applied on the shop floor and because it exhibited the best over-all tardiness performance among the single-pass dispatching rules in earlier experiments [3] with static problems and statistical processing times. The HSP program was used with the data for the original jobs to generate the machine priority lists for IP1 and IP2. These priorities were input data to the simulator.

A complete factorial design of the 27 factor combinations was employed. Since the processing times were random variables, 50 independent replications were made for each factor combination.[4]

Results. The tardiness statistics reported were the percentage of jobs tardy and the mean, variance, and maximum tardiness. The complete data, as tabulated in [8], is summarized in Figs. 2 through 5. Figure 3 includes 95% confidence intervals for mean tardiness.

The most notable result is the superior performance of IP2 under all conditions tested and for all four tardiness criteria. This confirms that the implementation strategy is important when using solutions to static problems in a dynamic environment. Previous research on problems with $N = 0$ (see [3]) demonstrated that centrally generated schedules, properly implemented, can be effective over a range of problems and processing time distributions. This research indicates that centrally generated schedules, properly implemented (e.g., IP2), can be effective over a range of $0 \leqslant N \leqslant 1$. This means that centrally generated schedules with review times in which as many as 50% of the jobs arrive during the review period may significantly outperform local dispatching rules. Information similar to that contained in Figs. 2 through 5 could be used by a manager to decide on the proper length of time between reviews by trading-off costs of updating and generating the centralized schedule with tardiness costs.

[4]The simulation program was validated manually using the example in Section II. X^2 tests were used to validate the processing time generator separately.

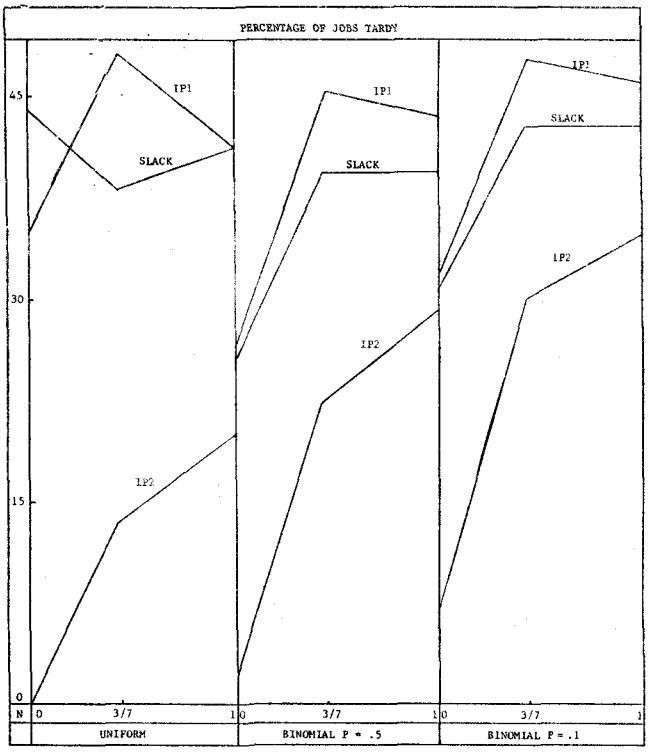


Fig. 2. Percentage of Jobs Tardy

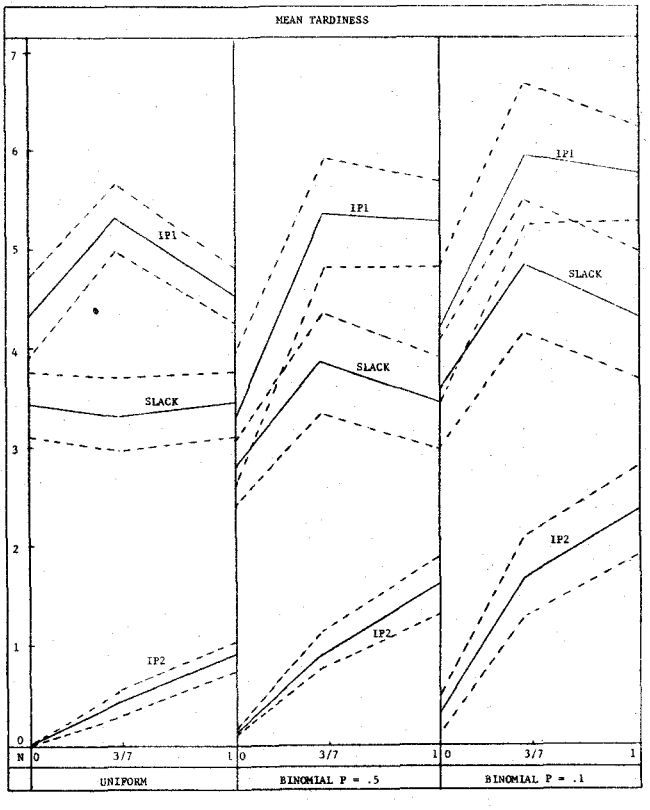


Fig. 3. Mean Tardiness Statistics

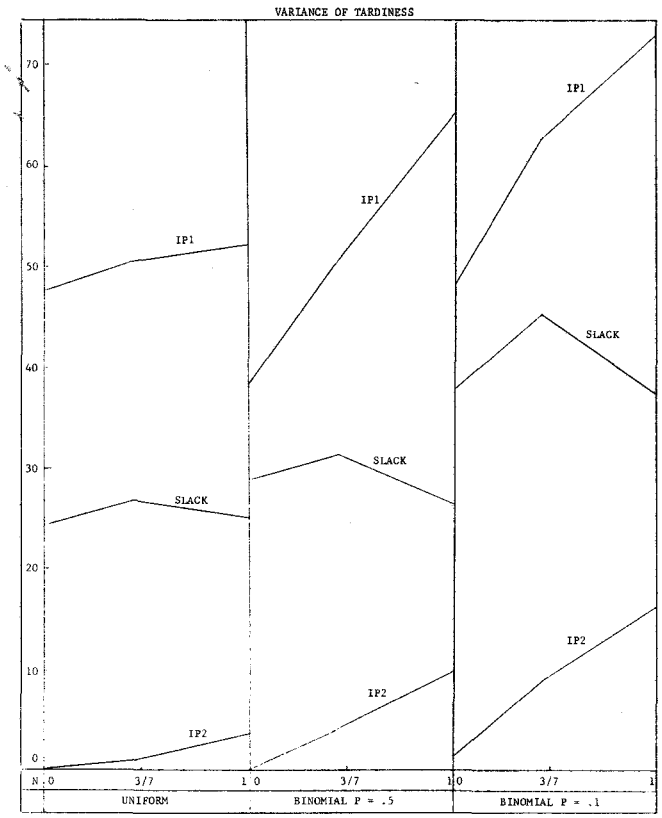


Fig. 4. Variance of Tardiness.

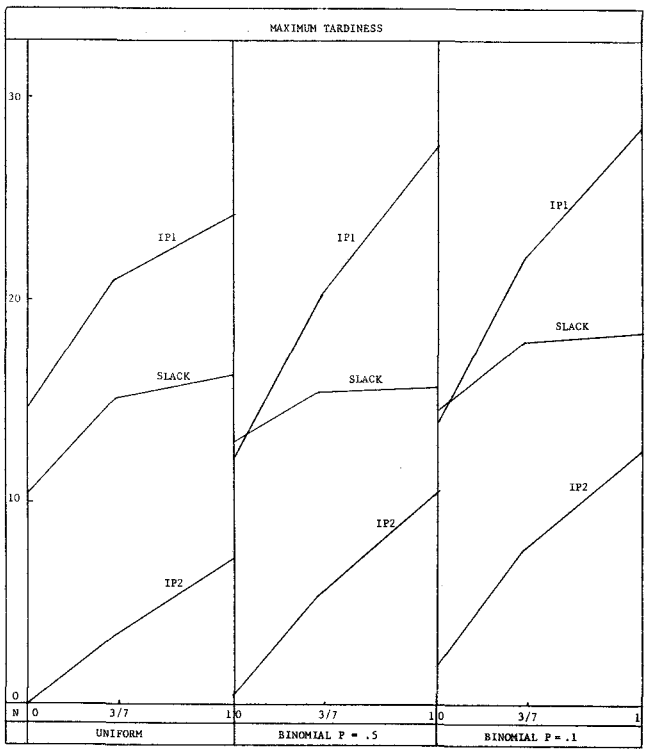


Fig. 5. Maximum Tardiness

The costs of data collection and computation can be divided into those relating to generation of the centralized schedule at review times and those relating to calculation of priorities on the shop floor. The computational times for generating the centralized schedule will depend upon the problem size and characteristics. The problems used in this paper required approximately 1 second per HSP run on an IBM 360-91. With an average of two iterations for the step sizes used in tightening due dates for HSP, the total computation time was approximately 2 seconds per problem. No general results are available but previous research (see [2]) indicates that, for up to 100 operations, the computing time in seconds per HSP iteration on an IBM 360-91 is approximately (number of operations/30). Two problems with 400 operations took 14 and 24 seconds of computer time each. Data required for each review is only that normally available in production control systems.

Priority calculation on the floor requires the centrally generated schedule for each machine, estimated arrival time, due-dates, estimate processing time on the given machine, and estimated remaining processing time on the job. All of this information is either required for the normal SLACK calculation or available from the output of centralized scheduling procedure. The number of calculations on the shop floor using IP2 can be fewer than with SLACK since the amount of slack need only be calculated for new jobs and for centrally scheduled operations when new jobs are available.

The results also provide some insights into the two implementation strategies. The nondelay strategy with preference given to centrally scheduled jobs (IP1) appears to be inferior to a strategy based on delays and integration of new jobs (IP2). Previous research [3] indicates that the relatively poor performance of IP1 compared to SLACK for $N = 0$ may be due to the characteristics of the problem used. However, we speculate that the increase in superiority of SLACK over IP1 for higher values of N reflects the lack of a means for integrating new jobs with those centrally scheduled. These findings and speculations indicate that continued research along the lines suggested in the next section may be valuable.

Summary and Indicated Extensions

Summary. Building upon the results of earlier research, a procedure is proposed for utilizing solutions to a job shop scheduling problem with due dates, static arrivals, and deterministic processing times in a dynamic environment with statistical processing times. Application of the procedure requires that a description of current shop status be available at points in time designated as review times. Two options are considered for implementing the priorities on the shop floor. The paper includes a detailed illustrative example and the results of experiments with an evaluative example. The principal conclusion of the research is that periodic centralized generation of schedules, implemented using option IP2, merits consideration as an effective sched-

uling procedure in dynamic job shop processes with due dates.

The two strategies for implementation evaluated could obviously be extended to consider other implementation procedures. For instance, the interrelationship between the delay and integration features combined in IP2 could be investigated as well as different methods for accomplishing each. In addition, more extensive evaluation of IP2 may be valuable.

Another area which appears fruitful for further research is the interactive use of the procedure. The iterative procedure used in this research was based on the single criterion, minimize maximum tardiness, and a simple search procedure. There is reason to believe that an individual faced with a scheduling problem may be interested in more than maximum tardiness. Moreover, an individual's intuition may prove superior to any prescribed iterative procedure such as used here for seeking improved solutions. This suggests the central thesis that in an interactive mode the machine is more useful as a source of information for problem identification than as a problem solver.

Guided by this thesis, the authors [4] have developed an interactive version of HSP. That program (ISP) includes provisions for modification of goals and information displays to aid the scheduler in applying ad hoc resource adjustments. ISP provides a mechanism for extending the use of the scheduling procedure described in this paper into the realm of interactive scheduling research.

Appendix A

The HSP procedure is designed to solve the following formulation of the static job shop problem.

Minimize [extent of precedence violations]
subject to:
1. All job due dates are satisfied
2. Fixed machine capacity

The basic structure of the multi-pass procedure is outlined below. For a more detailed description see [2].

Constraints

Because it is based upon the above formulation, the procedure schedules each machine subject to firm constraints imposed by the fixed machine capacity and job due dates.

(a) *Machine capacity constraints.* These constraints are imposed by taking the given processing time for each operation as fixed, by requiring that only one operation can be in process on a machine at any given time, and by not allowing any operation to be scheduled to begin before its earliest possible start time (EPST) given by the total processing time of the preceding operations on the job.

(b) *Due date constraints.* These constraints are imposed by not allowing any operation to be scheduled to begin after its latest possible start time (LPST) given by job due date minus remaining processing time for the job.

Criterion Function

The criterion function for the formulation is to minimize the extent of precedence violations among the operations in the schedule. The schedule requirements imposed on the operations for a given machine by precedence requirements and the current schedule on other machines are transmitted by computing an earliest start time (EST) and latest start time (LST) for each operation. These EST's and LST's change as the schedule changes and provide a means for measuring the value of the criterion function at any time. Operationally, the scheduling procedure tries to honor the EST's and LST's, but they are treated as soft constraints that can be violated to the extent necessary to schedule a machine. The essence of the scheduling procedure is a heuristic adjusting process designed to minimize the value of the criterion function as measured by the extent of the EST and LST violations.

Forces and Pseudo Constraints

The violation of an EST or LST indicates the need to reschedule an adjacent operation on the same job. When an operation is forced in this way, special attention is given to the need to reschedule the operation by using a set of pseudo constraints designated as EPST-P and LPST-P. The scheduling procedure treats these as intermediate in rigidity between the soft (EST and LST) and firm (EPST and LPST) constraints. These pseudo constraints provide the mechanisms for adjusting a machine's schedule to respond to forces that have resulted from the scheduling of other machines.

Classes of Forces

The scheduling procedure recognizes different classes of forces. A hierarchy of force classes ranging from most desirable to least desirable is established. When scheduling a machine, the operations in the class involving no forces are used first. A priority rule is used to select among the operations in that class. If all operations on the machine can be scheduled within that class, scheduling is begun on the next machine. If unscheduled operations remain, a choice must be made between (1) revising the current partial schedule on the machine in a renewed attempt to complete the scheduling of the machine without causing any forces; or (2) allowing operations to be scheduled from less desirable classes involving forces. The choice centers upon the question of how hard to try to schedule a machine without creating a force. A mode structure allows this question to be addressed operationally.

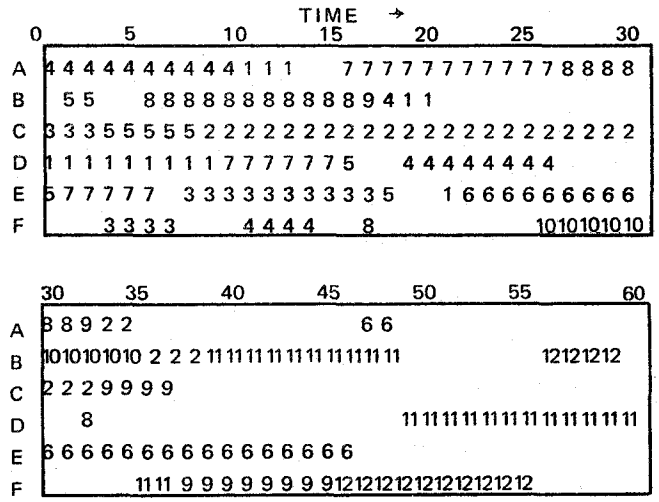
Scheduling Modes

A hierarchy of scheduling modes is established, paralleling the class structure. Each scheduling mode allows operations to be scheduled from a progressively larger subset of the force classes. Within each mode the preferred class structure is maintained. Before changing from one scheduling mode to the next, a back tracking routing can be used

to attempt to reschedule the machine within the current scheduling mode.

Appendix B

The Gantt chart for the illustrative example with IP2 priority implementation is given below. An example of the use of enforced idle time occurs at time $t = 6$ on machine E. At that time, job 6 is available for processing on the machine but IP2 calls for the machine to be held idle awaiting job 3, due to arrive from machine F.



Appendix C

The data for the *original jobs* in the experiments reported in Section 2 is given below:

Job	Arrival Time	Job Routing:Machine (Expected Processing Time)	Due Date
1	0	D(7) A(3) B(2) E(1)	26
2	0	C(20) A(2) B(2)	48
3	0	C(4) F(3) E(10)	34
4	0	A(10) F(3) B(1) D(8)	44
5	0	E(1) B(3) C(6) D(1) E(1)	24
6	0	E(22) A(3)	50
7	0	E(5) D(7) A(13)	50

The data for *new jobs* arriving during the scheduling period was generated as follows:

Poisson arrival process with mean arrival rate dependent on N.

Number of operations on job equals random integer from 1 to 6.

Each machine is equally likely for any operation (subject to the constraint that consecutive operations on a job be on different machines).

Expected processing time for each operation equals random integer between 1 and 10.

Job due date equal job arrival time plus two times total expected processing time of job.

References

[1] Conway, R. W., Maxwell, W. L., and Miller, L. W., *Theory of Scheduling*, Addison-Wesley (1967).

[2] Holloway, Charles A., and Nelson, Rosser T., "An Alternative Formulation of the Job Shop Problem with Due Dates," *Management Science*, **20** 1 (September 1973).

[3] Holloway, Charles A., and Nelson, Rosser T., "Job Shop Scheduling with Due Dates and Variable Processing Times," *Management Science*, **20** 9 (May 1974).

[4] Holloway, Charles A., and Nelson, Rosser T., "An Interactive Program for the Job Shop Scheduling Problem with Due Dates," Stanford University Graduate School of Business, Technical Report No. 22 (December 1972).

[5] Holloway, Charles A., and Nelson, Rosser T., "Job Shop Scheduling with Due Dates and Overtime Capability," *Management Science*, **21** 1 (September 1974).

[6] Holloway, Charles A., and Nelson, Rosser T., "Job Shop Scheduling with Due Dates and Operation Overlap Feasibility," *AIIE Transactions*, **7** 1 (March 1975).

[7] Peters, Michael H., "Optimal Static Job Shop Scheduling under Dynamic Conditions," PhD Dissertation, Indiana University (1971).

[8] Wong, Ruby Mei-Lun, "An Evaluation of Priority Implementation Procedures in a Job Shop Model with Due Dates, Dynamic Arrival Process, and Statistical Processing Times," Master's Thesis, UCLA (1973).

Ruby Mei-Lun Wong holds the BA and MS Degrees from UCLA. She is presently a PhD candidate in the Graduate School of Business, Stanford University. She was enrolled in the DBA Program at Harvard University and is a member of Phi Beta Kappa, Pi Mu Epsilon and Beta Gamma Sigma honorary fraternities.

Dr. Rosser T. Nelson is a Professor in the Graduate School of Management at UCLA. His current research interests are in job shop and resource constrained project scheduling, decomposition approaches in scheduling and man-machine interactive scheduling. Dr. Nelson received a BS Degree from Syracuse University and MS and PhD Degrees from UCLA. He is a member of TIMS and ORSA.

Dr. Charles A. Holloway is Associate Professor of Management Science in the Graduate School of Business, Stanford University. His current research is in job shop and large project scheduling, decomposition approaches in scheduling, mathematical programming with interactive identification and optimization, design of interactive graphic systems for managers and comparability of information systems. Dr. Holloway received the BSEE Degree from the University of California at Berkeley and MS and PhD Degrees from UCLA. He is a member of ORSA and TIMS.