Journal of the Operational Research Society (1996) 47, 666–677 0160-5682/96 $12.00

Machine Criticality Measures and Subproblem Solution Procedures in Shifting Bottleneck Methods: A Computational Study

HARRY H. HOLTSCLAW and REHA UZSOY
School of Industrial Engineering, Purdue University, USA

We examine the effects of different machine criticality measures (MCMs) and subproblem solution procedures (SSPs) on the performance of Shifting Bottleneck (SB) methods for the problem of minimizing maximum lateness in a job shop. Extensive computational experiments show that for problems with balanced workloads and random routeings simple MCMs and SSPs can be used without affecting solution quality. Routeing structure affects the performance of the SB method significantly. We also find that, contrary to some previous studies, the problem of maintaining feasibility in the SB method is significant.

Key words: scheduling, heuristics, computational analysis

INTRODUCTION

The problem of scheduling can be defined as that of assigning scarce resources to competing activities over time to optimize some measure of performance. A complex and widely studied scheduling problem is that of job shop scheduling, which can be described as follows: n jobs are to be processed on m machines. Each job visits each machine exactly once, and the sequence in which jobs visit the machine (i.e., the routeing) is known *a priori*. Each machine can process only one job at a time, pre-emption is not allowed, and there are no sequence dependent setups. A visit by a given job to a given machine is known as an operation. The processing times p_{ij} for operations j of job i, $j = 1, \ldots, m$, $i = 1, \ldots, n$, and job due dates d_i, $i = 1, \ldots, n$ are integer and known *a priori*. All jobs are available simultaneously for processing. A job is considered completed when its last operation has been processed. Let C_i, $i = 1, \ldots, n$ be the completion time of job i in some schedule for the job shop.

Most job shop scheduling research to date has focused on the problem of minimizing makespan (Cmax), where Cmax is defined as $\max_i\{C_i\}$. The problem is denoted by J//Cmax in the notation of Lageweg *et al.*[1] and has been shown[2] to be strongly NP-hard. Hence, efforts to develop optimal solution procedures have focused on branch and bound methods[3,4]. While these guarantee optimal solutions, their rapidly increasing computational requirements render them impractical for lage problems. Hence, considerable effort has been devoted to developing heuristic methods to obtain near-optimal solutions in reasonable computation times.

The most common approaches to job shop scheduling in practice have been dispatching rules[5,6]. Whenever a machine becomes idle, a job currently in the queue is selected for processing. Examples of dispatching rules are Shortest Processing Time (SPT), which selects the available job with shortest processing time on the current machine; and Earliest Due Date (EDD), which selects the job with the earliest due date. Dispatching rules are easy to implement and have low computational burden. However, their myopic nature, due to their considering only the current state of the machine and its immediate surroundings, may result in poor long-term performance. Hence, the use of heuristics which use information on the state of the entire shop to obtain better schedules at the cost of increased computation time may be justified. Examples of such approaches are the simulated annealing approach of Matsuo *et al.*[7], the tabu search approach of Taillard[8], and the Shifting Bottleneck (SB) approach of Adams *et al.*[9] which motivates this study.

The SB approach, designed by Adams *et al.*[9] for J//Cmax, is a heuristic that decomposes the job shop problem into single machine subproblems. The subproblems are solved individually and

Correspondence: R. Uzsoy, School of Industrial Engineering, 1287 Grissom Hall, Purdue University, West Lafayette, IN 47907-1287, USA

their solutions assembled into a solution for the job shop problem. The promising computational results obtained by these authors and others[3] render a more detailed study of this case of methods of considerable interest. One issue is the effect of subproblem solution procedures (SSPs) on the quality of the solutions produced by the SB procedure. One can solve the subproblems to optimality, which may require a substantial amount of time since the subproblems are strongly NP-hard. An alternative is to use heuristics to obtain rapidly near-optimal solutions. Another issue is the effect of the order in which the subproblems are solved. One would like to schedule the most critical machines, i.e., the machines most likely to affect the quality of the schedule, early in the process. This raises the question of how to develop effective machine criticality measures (MCMs) to accurately assess the criticality of a given machine.

In this paper we examine the effects of different SSPs and MCMs on the performance of the SB procedure for the job shop scheduling problem with maximum lateness (Lmax) as the performance measure of interest, denoted by J//Lmax. The lateness L_i of job i is defined as $L_i = C_i - d_i$. Note that L_i can be negative, in contrast to the tardiness $T_i = \max\{0, L_i\}$. Lmax is defined as $\text{Lmax} = \max_i\{L_i\}$. We present a computational study comparing six MCMs and two SSPs across a range of problem instances. Our results show that for problems with random routeings relatively simple MCMs suffice and that heuristics for the subproblems give good results. We also find that, contrary to the results of Adams *et al.*[9], the problem of maintaining feasibility in the SB approach is significant.

In the following section we give an overview of the SB method and review previous related work. The subsequent two sections describe the different MCMs. The next section describes the SSPs used in the study. The design of the computational experiment is given in the next section, before the results are discussed. We conclude with a summary and some directions for future work.

PREVIOUS RELATED WORK

The SB procedure (called SBI by Adams *et al.*[9]) can be stated as follows:

Step 1. Represent the scheduling problem using a disjunctive graph[9].
Step 2. Using some MCM, find the most critical unscheduled machine.
Step 3. Using some SSP, schedule the critical machine.
Step 4. Re-optimize each previously scheduled machine using the chosen SSP.
Step 5. If all machines have been scheduled, stop. Otherwise, go to Step 2.

Interactions between jobs and machines are captured by a disjunctive graph. Define an operation to be a visit by a job to a machine. The disjunctive graph contains a node ij for each operation j of job i, as well as a source node 0 and a sink node i^* for each job i representing the completion of that job. Precedence constraints between operations of a job are represented by conjunctive (directed) arcs. Competition for capacity among operations requiring the same machine is represented by pairs of disjunctive arcs. A pair of disjunctive arcs consists of a pair of arcs with opposite orientations such that any path in the graph can contain at most one of the two. The cost of arc (ij, kl) is given by p_{ij}, the processing time of operation ij. The source node 0 is linked to the nodes corresponding to the first operation of each job by an arc with cost 0, while each sink node i^* is connected to the node im corresponding to the last operation of job i by an arc with cost p_{im}. Scheduling a machine corresponds to fixing the disjunctive arcs between operations on that machine in one of their two possible orientations.

The SB procedure schedules one machine at each iteration. Hence at an intermediate iteration, we have a partial schedule where some machines have been scheduled (i.e., have all their disjunctive arcs fixed) and some not. The scheduling decisions already made constrain those remaining, in terms of when jobs become available for processing and how long after completing a given operation the job can be completed. These release times and due dates can be estimated using longest path calculations in the directed graph corresponding to a partial schedule as follows. Let $L(ij, kl)$ denote the length of the longest path from node ij to node kl in the disjunctive graph. Then the release time r_{ij} of operation is given by $r_{ij} = L(0, ij)$. The due date d_{ij} of operation ij is given by $d_{ij} = d_i - L(ij, i^*) + \text{LMAX} + p_{ij}$, where LMAX is the Lmax value of the current partial schedule, d_i the due date of job i and p_{ij} the processing time of operation ij. Since we are minimizing Lmax,

it is important to take the Lmax value of the current partial schedule into account while setting operation due dates to avoid overconstraining the subproblems. Thus, LMAX is added to the operation due dates at each iteration of the SB process. The Lmax value relative to these due dates represents the increase in the Lmax value of the partial schedule at the current iteration. Further details of the disjunctive graph and its use in SB methods are given by Adams *et al.*[9] and Ovacik and Uzsoy[10].

The SB approach does not guarantee feasibility at intermediate iterations. As new machines are scheduled a cycle may be created in the directed graph corresponding to the partial schedule. Adams *et al.*[9] point out this possibility, and suggest modifying the subproblems giving rise to the cycle and resolving them. However, no infeasibilities arose in the course of their experiments. Dauzere-Peres and Lasserre[11] identify a set of start-to-start precedence constraints ignored by the SB method[9] which may lead to its generating infeasible solutions. These authors and Balas *et al.*[12] suggest modifications to the subproblems to avoid this situation. In this paper we follow Ovacik and Uzsoy[10]'s approach for restoring feasibility at intermediate iterations. Whenever an infeasibility occurs, the operation causing it is identified. The release time of this operation is modified to regain feasibility and the machine on which the operation is processed is resolved. This procedure is repeated until no infeasibilities remain.

A number of authors have studied the SB method and extended it in different ways. Morton[18] discusses extending SB to more general problems, such as project scheduling, and different performance measures. Applegate and Cook[3] modify its selection of critical machines. In this method, after the first k machines have been scheduled, each unscheduled machine has its schedule tentatively fixed and appended to the partial schedule. The unscheduled machine which produces the partial schedule with the lowest Cmax becomes the $k + 1$'st machine scheduled. This produces better solutions at the cost of more computation time.

The scheduling problem solved in Step 3 is that of minimizing Lmax on a single machine with release times and due dates. This problem is denoted by $1/r_j/\text{Lmax}$[1], and is strongly NP-hard[2]. Carlier[13], Hall and Shmoys[14] and Potts[15] show that scheduling the available job with the earliest due date whenever the machine becomes free (the Earliest Due Date or Extended Jackson algorithm) has excellent worst-case performance. Branch and bound algorithms have been developed by Baker and Su[16] and Carlier[13]. The latter algorithm is used by SB[9] to schedule all unscheduled machines at each iteration. The machine with the highest Lmax value is chosen as the critical machine, and the schedule obtained by the Carlier algorithm adopted for it.

The effects of different MCMs have not been studied extensively to date. Applegate and Cook[3] suggest an algorithm that tries to select a subset of the most critical machines to schedule. They point out that it is not clear how to identify critical machines, and that random selection does not yield good results. Morton[18] suggests solving the pre-emptive relaxation of the single machine problem to optimality and using the Lmax of this problem as a MCM. In addition, most computational studies have used exact solution techniques for the single-machine subproblems. The tradeoffs involved in using faster heuristic procedures to solve the single machine problems have not been examined. Both these issues are addressed in this study. In addition, computational studies to date[3,9] have focused on a relatively small set of problem instances for the J//Cmax problem. For example, Adams *et al.*[9] compare the performance of SB to that of dispatching rules on a set of 59 test problems. They note that it obtains the optimal solution to the well-known 10 jobs/10 machines problem given by Muth and Thompson[17], and obtain modest improvements over dispatching rules, the maximum improvement being of the order of 10%. There has been no large-scale experimentation with the SB method except the work of Ovacik and Uzsoy[10] which focuses on problems motivated by semiconductor testing facilities. The specialized nature of these problems, with sequence-dependent setup times and re-entrant product flows, makes their results difficult to extend to other problems.

MACHINE CRITICALITY MEASURES

In this section, we describe five MCMs ranging from simple static measures of machine workload to more sophisticated techniques taking into account the distribution of workload over time.

A more involved MCM is discussed in detail in the next section. In all cases except the random approach, the machine with the highest value of a MCM is the most critical.

The random approach. Machines are selected for scheduling in random order, with all unscheduled machines having equal probability of selection. This approach serves as a benchmark for all other MCMs.

The Simple Workload (SW) approach. The total workload on a given machine, given by the sum of all operation times on that machine, is used as an MCM.

The Pre-emptive Earliest Due Date (PEDD) approach. This approach uses the Pre-emptive Earliest Due Date (PEDD) rule to solve the pre-emptive relaxation of the single machine problem optimally[19]. The MCM is the Lmax value obtained from this relaxation.

The Baker and Su (BS) approach. This MCM is the Lmax value obtained when the branch and bound algorithm of Baker and Su[16] is used to solve the $1/r_j$/Lmax problem to optimality. While this is not the most efficent exact procedure for this problem, it is the simplest to implement and is used in this study both as a MCM and as a SSP.

The Extended Jackson (EJL) approach. This approach uses the Extended Jackson Algorithm with local improvements (EJL) procedure[10], which supplements the Extended Jackson algorithm with a local improvement procedure based on adjacent pairwise interchanges, to solve the $1/r_j$/Lmax problem. The Lmax value obtained by EJL is used as a MCM. This procedure is described in detail in Ovacik and Uzsoy[10].

We now describe the most involved of the MCMs, the Aggregate Demand approach.

THE AGGREGATE DEMAND (DAGGR) APPROACH

This approach is based on the MICRO-BOSS scheduling system developed by Sadeh[20] for the job shop scheduling problem with earliness and tardiness costs. MICRO-BOSS uses constraint-guided heuristic search to build partial schedules one operation at a time. Considering the number of operations requiring a given machine in a given time interval, the system develops a demand profile over time for each machine. The machine with the greatest demand conflict, i.e., the most operations competing for the same time interval, is determined and the operation ij contributing most to this demand conflict is scheduled next. This operation ij is called the critical operation. A particular time interval in which to schedule this operation is also chosen. This identification and scheduling of a critical operation is repeated until all operations are scheduled. Recall that all problem data, such as operation processing times and due dates, are assumed to be known and integer.

The demand profile is based on the costs $c_{ij}(t)$ attributed to various possible starting times t (called reservations) for each operation. For each reservation t, $c_{ij}(t)$ is used to estimate a probability $\sigma_{ij}(t)$ that the operation ij will begin at time t. This probability is calculated such that reservations with low $c_{ij}(t)$ have high probability and those with high $c_{ij}(t)$ have low probability. The demand $D_{ij}(t)$ for a given reservation t by operation ij on a given machine is given by the sum of the $\sigma_{ij}(t)$s for those reservations which result in the machine being busy at time t. Finally, the demands $D_{ij}(t)$ are aggregated for each machine at each reservation to yield an aggregate demand profile for each machine over time. A flowchart of the Aggregate Demand approach is given in Figure 1.

To use the DAGGR approach within the SB methodology, several modifications must be made. The costs $c_{ij}(t)$ of assigning operation ij to reservation t must be modified to reflect the performance measure of Lmax, and the procedure must be modified so it can be used when all operations on a given machine are scheduled at once, rather than one operation at a time. In the following we focus on the modifications we have made to the basic procedure, referring the reader to Sadeh[20] for details.

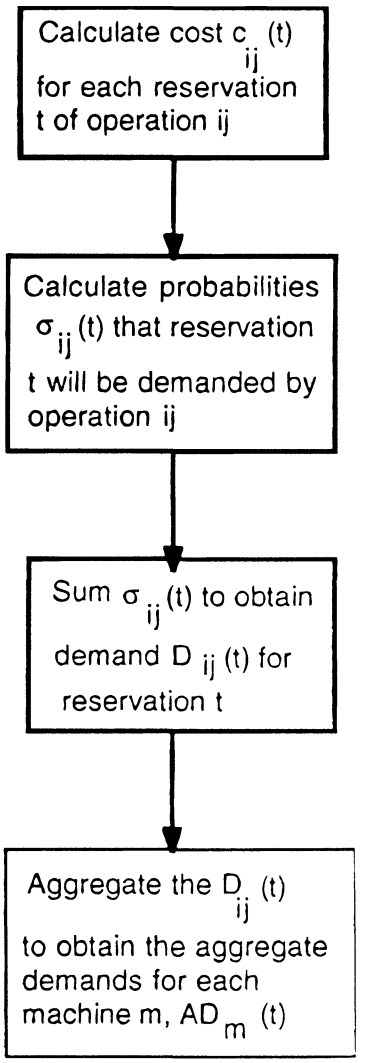

FIG. 1. *Flowchart of DAGGR approach.*

To find the cost $c_{ij}(t)$ of a given reservation t one determines the latest time t^* at which the operation can start and still minimize the amount by which the operation due date is exceeded, i.e., the tardiness of that operation. Recalling that the lateness L_{ij} of operation ij is defined as $C_{ij} - d_{ij}$, and the tardiness $T_{ij} = \max\{0, L_{ij}\}$, we see that there is no penalty for each completion of a job and that tardiness is a linear increasing function of t. Thus we obtain

$$c_{ij}(t) = \begin{cases} 0, & \text{if } t \leqslant t^* \\ t - t^*, & \text{otherwise.} \end{cases}$$

Reservations are considered until t reaches a chosen upper bound UB. Sadeh[20] appears to base this upper bound on a case-by-case assessment of how late a job can be and still be acceptable. In our implementation, we use the sum of all operation processing times as an upper bound, with a maximum UB of 5000. To illustrate, suppose we have an operation ij with release time $r_{ij} = 30$, processing time $p_{ij} = 116$, and due date $d_{ij} = 369$. The earliest start time for the operation is 30, and the operation can start any time between 30 and $t^* = 253$ and not be tardy. Hence we have $c_{ij}(t) = 0$ for $30 \leqslant t \leqslant 253$. The costs beyond 253 increase linearly until the UB is reached. A similar cost profile is generated for each operation in the problem.

Once we have obtained $c_{ij}(t)$ for all reservations t and operations ij, we need to find the probablity $\sigma_{ij}(t)$ that operation ij will start at time t. The $\sigma_{ij}(t)$ are calculated such that reservations with

high $c_{ij}(t)$ have low $\sigma_{ij}(t)$ and *vice versa*, and $\Sigma_t \sigma_{ij}(t) = 1$. Thus we have

$$\sigma_{ij}(t) = N\left[1 - B\left(\frac{c_{ij}(t)}{\text{maxcost}}\right)\right].$$

Here N is a normalization factor ensuring that $\Sigma_t \sigma_{ij}(t) = 1$ and B a bias factor specifying the degree to which $\sigma_{ij}(t)$ is biased towards the value of the cost ($0 \leqslant B < 1$). If $B = 0$, then the $c_{ij}(t)$ does not affect the $\sigma_{ij}(t)$. For all calculations, we let $B = 0.9$ (following Sadeh[20]) so that the $\sigma_{ij}(t)$ are biased towards the value of the costs. Maxcost are given by $\max_t\{c_{ij}(t)\}$. Once again, this is done for all operations ij.

Now we have calculated the $\sigma_{ij}(t)$, we need to know the probability that an operation will require its machine at each reservation. An operation requires its machine at a particular reservation if it either starts at that reservation or has started prior to that reservation but has not yet completed. Thus, if we seek the demand of operation ij at time t (with processing time $p_{ij} = 116$), we start with $D_{ij}(t) = \sigma_{ij}(t)$. However, since operation ij would still be running if it started at any time from $t - 1$ to $t - 115$, our $D_{ij}(t) = \sigma_{ij}(t - 115) + \sigma_{ij}(t - 114) + \cdots + \sigma_{ij}(t - 1) + \sigma_{ij}(t)$. In general, we have

$$D_{ij}(t) = \sum_{t - p_{ij} < k \leqslant t} \sigma_{ij}(k).$$

A similar demand is generated for each operation ij in the problem.

Having calculated the demand profiles $D_{ij}(t)$ for all operations, we now aggregate them to evaluate the total demand on each machine at reservation t, which is given by

$$AD_m(t) = \sum_{ij \in s(m)} D_{ij}(t)$$

where $s(m)$ is the set of all operations ij using machine m.

In MICRO-BOSS[20], once one has obtained these aggregate demand profiles for each machine, the maximum value of $AD_m(t)$ over all machines m determines the machine with the critical operation, the operation causing the most demand during the reservation in which this maximum value occurs. This operation is scheduled at the time reservation at which it creates the most demand.

Our version proceeds differently since we are scheduling one machine at a time, not one operation at a time. A critical machine and not a critical operation is selected. All operations on this critical machine are scheduled using the selected SSP at Step 3 of the SB procedure. This is repeated for each machine until all are scheduled.

An operation in a job that is scheduled to start at a particular time may have unscheduled operations preceding it in the same job. Suppose operation ij has been scheduled to start at time 170, but has an unscheduled operation ih preceding it. When this happens, operation ih no longer affects the lateness of job i. No matter when this operation starts, it cannot feasibly make ij start any later than time 170. Even though it can no longer affect the lateness of job, operation ih still exerts demand on the machine on which it is processed. It is merely the case that all its possible start times have the same cost of zero, since ih could start at any of them and not delay the completion of the overall job. The calculations of $\sigma_{ij}(t)$, $D_{ij}(t)$ and $AD_{ij}(t)$ for this case remains the same.

SUBPROBLEM SOLUTION PROCEDURES

In order to explore the effect of different SSPs on the quality of solutions obtained by the SB approach, we use the following two SSPs. Of particular interest is the tradeoff between using a more time-consuming exact solution procedure and a faster heuristic procedure.

The Baker and Su (BS) algorithm. This is a branch and bound algorithm which solves the $1/r_j/\text{Lmax}$ problem exactly. It is also used as a MCM as discussed in the third section.

The Extended Jackson algorithm with local improvements (EJL) algorithm. This heuristic combines the Extended Jackson algorithm for $1/r_j/\text{Lmax}$ with a local improvement procedure. It is also used as a MCM and was described in the third section. EJL does not guarantee optimality, but given the excellent worst-case performance of the Extended Jackson algorithm, we would expect it to produce good solutions in short CPU times.

EXPERIMENTAL DESIGN

In this section we describe the design of the computational experiments used to compare the different MCMs and SSPs. As described previously, we use six different MCMs when implementing the SB methodology. With each of these MCMs, two different SSPs, one exact and one heuristic, are used, making 12 versions of SB that are tested across a number of job shop problem instances.

The problem instances themselves are divided into two types of job shop problems. One type is the classical job shop problem where each job visits each machine in the job shop in a predetermined order where the routeing of a job is a random permutation of the machines. The second type divides the m machines into two sets of $m/2$ machines each. In each job, the first $m/2$ operations take place on the first set of machines in some random order and the second $m/2$ operations on the second set of machines. Storer *et al.*[21] found this second type of problem to be substantially more difficult for SB. We shall refer to the first family of problems as the 'Easy' problems, and the second family as the 'Hard' problems. For both problem types, all jobs are available at time zero and the operation processing times are discrete and uniformly distributed over the interval [1,200].

Job due dates are uniformly distributed over an interval determined by the expected makespan of the instance and two parameters τ and R. τ is a tightness parameter defining the expected percentage of tardy jobs, while the range parameter R specifies the range of the due dates. The mean μ of the due dates is given by

$$\mu = (1 - \tau) \times (\text{Expected Makespan}).$$

The interval that the due dates are generated from is thus

$$\mu \pm (\mu \times R)/2.$$

The expected makespan is calculated by estimating the total processing time for all operations and dividing this by the number of machines available. In our experiment, we selected 0.3 and 0.6 as values for τ, and 0.5 and 2.5 as values for R, yielding four due date configurations. Problem instances with 10 and 20 jobs and six and 10 machines were solved, yielding four combinations of problem sizes.

The overall experimental design is summarized in Table 1. There are 32 different problem types, for each of which we solve 10 randomly generated instances, giving a total of 320 instances. Each instance is solved by 12 different SB procedures and the EDD dispatching rule, totaling 13 problem solution methods. Whenever the Baker and Su algorithm is used, either as a MCM or a SSP or both, we have limited both the maximum number of active nodes and the total number of nodes examined to 500 in order to ensure reasonable computation time. Preliminary experimentation revealed that increasing this limit made no significant difference to solution quality but

TABLE 1. *Summary of experimental design*

	Levels used		No. of levels
Problem type	Hard	Easy	2
Due date range	0.5	2.5	2
% Tardy jobs	0.3	0.6	2
Number of machine	6	10	2
Number of jobs	10	20	2
Total problem configurations			32
Problem instances/configuration			10
Total number of problem instances			320

considerably increased computation time. The Earliest Due Date (EDD) dispatching rule is motivated by the fact that it is optimal for the static single machine problem[19] and has excellent worst-case performance for the dynamic single machine problem[13–15]. The different variants of the SB methodology compared and their abbreviations are summarized in Table 2. All algorithms were coded in C and run on a Sun SPARC workstation. The results of the experiments are described in the following section.

TABLE 2. *SB methods compared in experiments*

Machine criticality measure	Subproblem solution method	Abbreviation
Baker and Su	Baker and Su	BB
Baker and Su	EJL	BE
DAGGR	Baker and Su	DB
DAGGR	EJL	DE
EJL	Baker and Su	EB
EJL	EJL	EE
PEDD	Baker and Su	PB
PEDD	EJL	PE
Random	Baker and Su	RB
Random	EJL	RE
Simple workload	Baker and Su	SWB
Simple workload	EJL	SWE

RESULTS

We first discuss the results from the point of view of solution quality, and then address the issue of computation time. To compare the quality of the solutions obtained by the different algorithms, we use the ratio of the value of the solution found by each algorithm to that of the best solution found by any algorithm for that particular problem instance. To avoid problems caused by non-positive Lmax values, we convert the Lmax values into makespan values for the problem with delivery times as described by Lenstra[22]. This is done by adding a constant at least as large as the maximum due date for that problem instance to the Lmax value. Hence the ratio upon which our comparison is based is

$$\frac{\text{Lmax} + \text{dmax}}{\text{Best} + \text{dmax}}$$

where Lmax denotes the value of the solution found by the algorithm being considered, Best the lowest Lmax value found for that problem instance by any of the algorithms being compared, and dmax the maximum due date in the problem instance.

Figures 2 and 3 show the average ratio over all instances with the same size (number of jobs and machines) for the two different problem configurations (Easy and Hard), The first observation is that EDD outperforms most of the SB procedures for the Hard problems, while the SB methods

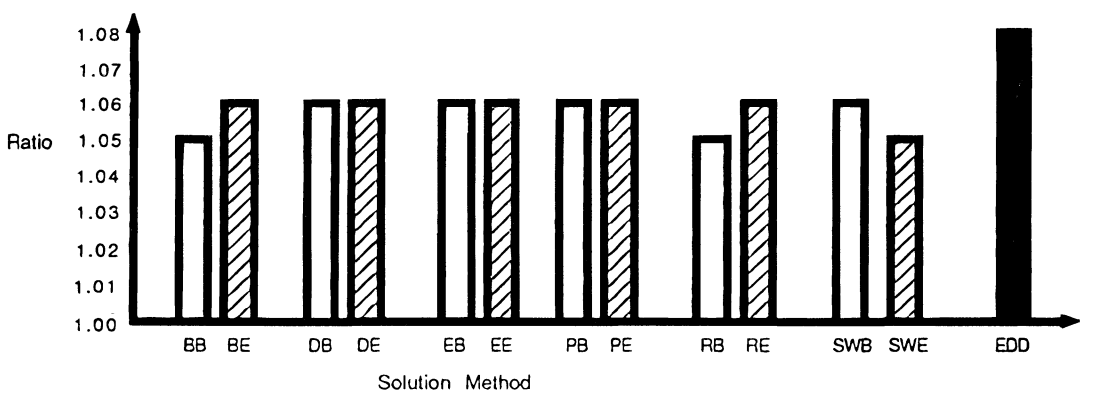

FIG. 2. *Average (Lmax + dmax)/(Best + dmax) ratios for each solution method across all problems of the Easy configuration.*

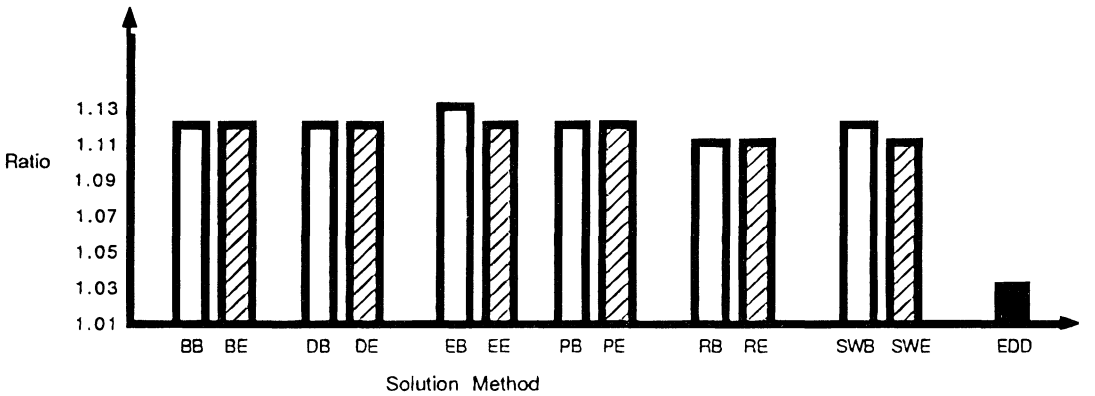

FIG. 3. *Average (Lmax + dmax)/(Best + dmax) ratios for each solution method across all problems of the Hard configuration.*

perform slightly better for the Easy problems. The SB methods perform markedly worse for Hard problems than for Easy problems, supporting the conjecture that the performance of SB methods is affected by the structure of the job routeings, as suggested by Storer *et al.*[21].

There are no significant differences in performance between the different SB methods for the Easy problems, and only slight ones for the Hard problems. The more complex MCMs such as BS and DAGGR perform slightly worse than the Random or Simple Workload approaches. The reason for this is probably the nature of the problem instances being solved. Since all operations on all machines have the same processing time distributions, over a number of randomly generated instances all machines are equally likely to become a bottleneck due to heavy workload. In addition, since the routeings are random, a given machine is equally likely to be at different positions in the routeings for different jobs, thus preventing the development of a bottleneck due to demand for a machine being concentrated in a narrow time interval. Thus, at least for the types of problems used in these experiments, a simple MCM such as Simple Workload is sufficient.

There does not seem to be any significant effect due to the choice of SSP. This is probably because EJL performs very well for the subproblems in this study which do not have sequence-dependent setup times. Hence on average, there is no advantage to using the more time-consuming BS procedure.

Figures 4 and 5 show the maximum ratios for the various problem configurations. There is a marked decline in worst-case performance for the SB methods on the Hard problems compared to the Easy problems. As was the case for the average performance, EDD outperforms the SB methods for the Hard problems, and is outperformed by them on the Easy problems. The best worst-case performance is obtained by the BE and PE methods for the Easy problems, and DE for the Hard problems. This indicates that more complex MCMs may lead to slightly better worst-case performance, but evidence is not conclusive. In terms of SSPs, EJL yields better results for the Easy problems, and there is no significant difference for the Hard problems.

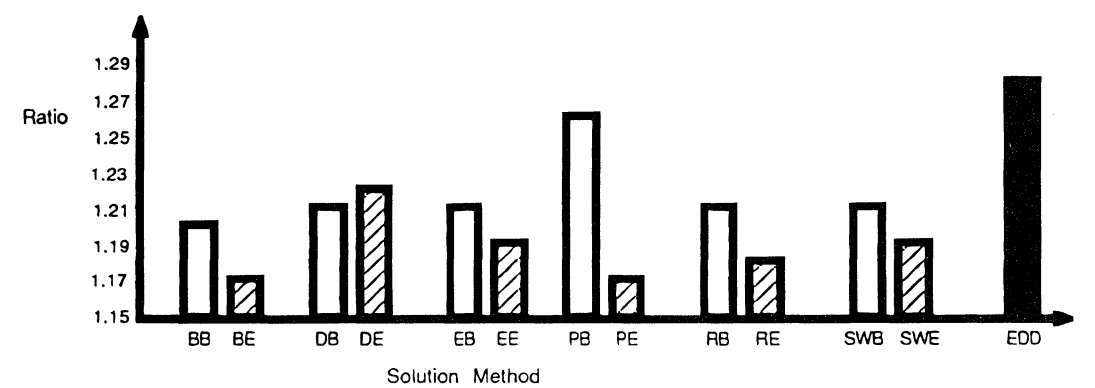

FIG. 4. *Maximum (Lmax + dmax)/(Best + dmax) ratios for each solution method across all problems of the Easy configuration.*

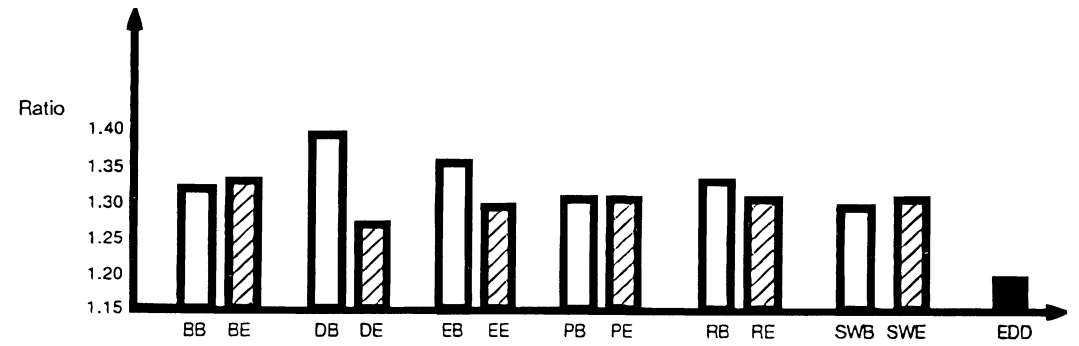

FIG. 5. *Maximum (Lmax + dmax)/(Best + dmax) ratios for each solution method across all problems of the Hard configuration.*

Table 3 shows the results for problems with different due date configurations. All methods except EB and RE in the Hard problems perform worse under a narrow due date range ($R = 0.5$). The SB methods perform best for loose due dates with a wide range ($R = 2.5$, $\tau = 0.3$), outperforming EDD for the Easy problems and matching it on the Hard problems. EDD performs better for other due date configurations for the Hard problems. This is due to the fact that when we have loose due dates and a wide range, there is considerable difference between the due dates of individual jobs, making the global information in the SB methods more useful, for example to introduce idle time on a machine to wait for an urgent job instead of processing an available job with a later due date. When a large proportion of the jobs are tardy or the due date range is narrow, it becomes more important to keep the machine busy and maximize throughput, which the EDD rule will do.

TABLE 3. *Average ratios over different due date combinations*

Diff.	Due date		BB	BE	EB	EE	DB	DE	PB	PE	RB	RE	SWB	SWE	EDD
	R	t													
E	0.5	0.3	1.07	1.07	1.09	1.08	1.07	1.06	1.08	1.07	1.07	1.07	1.08	1.06	1.09
E	0.5	0.6	1.07	1.08	1.07	1.09	1.09	1.08	1.09	1.07	1.06	1.06	1.07	1.06	1.11
E	2.5	0.3	1.03	1.03	1.03	1.03	1.03	1.04	1.02	1.03	1.02	1.03	1.03	1.04	1.06
E	2.5	0.6	1.05	1.05	1.06	1.06	1.06	1.05	1.06	1.05	1.05	1.06	1.05	1.05	1.07
Avg.			1.05	1.06	1.06	1.06	1.06	1.06	1.06	1.06	1.05	1.06	1.06	1.05	1.08
H	0.5	0.3	1.15	1.14	1.15	1.14	1.16	1.14	1.14	1.14	1.13	1.12	1.14	1.12	1.02
H	0.5	0.6	1.14	1.13	1.13	1.14	1.16	1.13	1.15	1.13	1.15	1.11	1.13	1.13	1.01
H	2.5	0.3	1.08	1.1	1.05	1.09	1.09	1.1	1.09	1.1	1.08	1.09	1.1	1.08	1.08
H	2.5	0.6	1.1	1.1	1.17	1.1	1.12	1.1	1.1	1.09	1.09	1.1	1.11	1.09	1.02
Avg.			1.12	1.12	1.12	1.12	1.13	1.12	1.12	1.12	1.11	1.11	1.12	1.1	1.03

In summary, our results indicate that for the types of problem instances used in this study there is no advantage in terms of average performance to using a sophisticated MCM such as DAGGR, since the results obtained do not differ significantly from those of much simpler and faster methods like Simple Workload. The SB methods outperform EDD both on average and in the worst case for the Easy problems, and are outperformed by it for the Hard problems. The due date configuration affects solution quality significantly. SB methods do well on problems with loose due dates and a wide range and not so well on problems with other configurations. The choice of SSP does not affect results significantly, rendering the use of the faster EJL procedure more advantageous.

An important observation throughout the experiment concerned infeasibilities. Whenever infeasibilities (representing cycles in the graph corresponding to the partial solution at a given iteration of the SB approaches) occur, the subproblems have to be re-optimized. Infeasibilities occurred often. Most problem instances had numbers of infeasibilities in the hundreds. This could also have affected the quality of the SB solutions. It is curious that previous studies[3,9] do not mention this problem.

The CPU time requirements of the different SB procedures are shown in Table 4. The approaches using the BS algorithm as MCM or SSP are by far the most computationally demanding. All SB procedures require considerably more CPU time than EDD. However, if we restrict our attention to the procedures which do not use the BS procedure, the maximum computation time for a 200 operation problem was 142 seconds on a Sun SPARC workstation, which is reasonable for a problem of this size.

TABLE 4. *Average CPU times (in seconds) for algorithms*

Diff.	Due date		BB	BE	EB	EE	DB	DE	PB	PE	RB	RE	SWB	SWE	EDD
	R	*t*													
E	0.5	0.3	107	28.2	64	46.1	97.9	26.1	95.9	16.6	62.8	16.3	19.8	16.1	0.21
E	0.5	0.6	96.2	28.7	66.6	45.9	93.8	26.5	94.5	17	62.4	16.7	20.5	16.6	0.21
E	2.5	0.3	88.8	25.8	61.2	41.2	73.1	23.1	79.2	14.8	49	14.9	18.1	14.5	0.22
E	2.5	0.6	123	27.4	65.6	43.2	116	25	122	15.6	76.8	16	22.6	15.3	0.21
H	0.5	0.3	145	33	34.6	50.3	159	30.6	164	30.9	126	18.4	22.5	18.9	0.21
H	0.5	0.6	142	33.2	34.6	51	156	30.8	112	31.2	133	18.9	22.3	18.8	0.22
H	2.5	0.3	84.1	26.1	13.3	42.2	92.7	23.5	50.8	23.7	81.2	14.5	17.6	14.1	0.22
H	2.5	0.6	136	30.5	36.9	46.3	160	26.6	89	27.1	123	16.4	21.1	16.4	0.21

CONCLUSIONS AND FUTURE DIRECTIONS

In this study we have examined the effects of different SSPs and MCMs on the performance of SB methods for J//Lmax. Our results show that for the problem configurations studied in our experiments, simple MCMs are sufficient. This is due to the fact that the random nature of the job routeings and the identical distribution of the processing times at each machine make the formation of well-defined bottlenecks unlikely. We also find that an effective heuristic method for solving the subproblems results in performance comparable to, and often superior to, a more demanding exact solution method. The SB methods consistently outperform EDD for the Easy problems, and are consistently outperformed by it, both on average and in the worst case, for the Hard problems. Even for the Easy problems, the improvement obtained over EDD is relatively small, of the order of 3% on average. These results are consistent with those obtained by a number of other researchers, such as Ovacik and Uzsoy[10] for the problem with sequence-dependent setup times and Adams *et al.*[9] for the J//Cmax problem.

These results indicate that for job shop problems with random routeings, balanced workloads and sequence-independent setup times, the advantage of this form of SB method over dispatching rules is limited. However, in situations where job routeings are more structured, such as those encountered in semiconductor wafer fabrication or testing facilities, the benefit of using a more sophisticated procedure are greater[10]. An important direction for future reseach is to perform a similar set of experiments over different shop configurations, where well-defined bottlenecks exist and the routeings are more structured. It would also be interesting to repeat this experimentation for the J//Cmax problem, to see how far the results obtained from the limited studies carried out to date extend.

Another important direction for future research is the improvement of SB methods themselves. In particular, the procedures which ensure feasibility of the partial solutions require attention. The inclusion of inter-operation delay times suggested by Dauzere-Peres and Lasserre[11] and Balas *et al.*[12] would resolve these problems. Recent work by Ovacik and Uzsoy[23], who develop specialized decomposition procedures for semiconductor testing facilities with improvements in performance of up to 30% over dispatching rules, indicates that this is indeed the case.

Acknowledgement—This research was supported by the National Science Foundation under Grant No. DDM-9107591.

REFERENCES

1. B. J. LAGEWEG, E. L. LAWLER, J. K. LENSTRA and A. H. G. RINNOOY KAN (1981) *Computer-Aided Complexity Clasification of Deterministic Scheduling Problems.* Report No. BW138, Mathematisch Centrum, Amsterdam.

2. M. R. Garey and D. S. Johnson (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco.
3. D. Applegate and W. Cook (1991) A computational study of the job shop scheduling problem. *ORSA J. Computing* **3**, 149–156.
4. J. Carlier and E. Pinson (1989) An algorithm for solving the job-shop problem. *Mgmt Sci.* **35**, 164–176.
5. K. Bhaskaran and M. Pinedo (1991) Dispatching. In *Handbook of Industrial Engineering* (G. Salvendy, Ed.) pp 2182–2198. J. Wiley, New York.
6. J. H. Blackstone, D. T. Phillips and G. L. Hogg (1982) A state-of-the-art survey of dispatching for manufacturing job shop operations. *Int. J. Prod. Res.* **20**, 27–45.
7. H. Matsuo, C. J. Suh and R. S. Sullivan (1988) *A Controlled Search Simulated Annealing Approach for the General Job Shop Scheduling Problem.* Department of Management, Graduate School of Business, University of Texas at Austin.
8. E. Taillard (1994) Parallel taboo search techniques for the job shop scheduling problem. *ORSA J. Computing* **6**, 108–117.
9. J. Adams, E. Balas and D. Zawack (1988) The shifting bottleneck procedure for job-shop scheduling. *Mgmt Sci.* **34**, 391–401.
10. I. M. Ovacik and R. Uzsoy (1992) A shifting bottleneck algorithm for scheduling semiconductor testing operations. *J. Electronics Mfg* **2**, 119–134.
11. S. Dauzere-Peres and J. B. Lasserre (1993) A modified shifting bottleneck procedure for job shop scheduling. *Int. J. Prod. Res.* **31**, 923–932.
12. E. Balas, J. K. Lenstra and A. Vazacopoulos (1995) One machine scheduling with delayed precedence constraints. *Mgmt Sci.* **41**, 94–109.
13. J. Carlier (1982) The one-machine scheduling problem. *Eur. J. Opl Res.* **11**, 42–47.
14. L. A. Hall and D. B. Shmoys (1992) Jackson's rule for single-machine scheduling: making a good heuristic better. *Math. Opns Res.* **17**, 22–35.
15. C. N. Potts (1980) Analysis of a heuristic for one machine sequencing with release dates and delivery times. *Opns Res.* **28**, 1436–1441.
16. K. R. Baker and Z. S. Su (1974) Sequencing with due dates and early start times to minimize maximum tardiness. *Naval Res. Logist. Q.* **21**, 171–176.
17. J. F. Muth and G. L. Thompson (1963) *Industrial Scheduling.* Prentice Hall, Englewood Cliffs, NJ.
18. T. E. Morton (1990) *Shifting Bottleneck Methods in Job Shop and Project Scheduling: Tutorial and Research Directions.* Graduate School of Industrial Administration, Carnegie Mellon University.
19. K. R. Baker (1974) *Introduction to Sequencing and Scheduling.* John Wiley, New York.
20. N. Sadeh (1991) *Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling.* Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University.
21. R. H. Storer, S. D. Wu and R. Vaccari (1992) New search spaces for sequencing problems with application to job shop scheduling. *Mgmt Sci.* **38**, 1495–1509.
22. J. K. Lenstra (1977) *Sequencing by Enumerative Methods.* Mathematical Centre Tract 69, Mathematisch Centrum, Amsterdam.
23. I. M. Ovacik and R. Uzsoy (1994) *Decomposition Methods for Scheduling Complex Job Shops: An Application to Semiconductor Testing Operations.* Research Report, School of Industrial Engineering, Purdue University.

Received October 1993; accepted September 1995 after two revisions