Pergamon

PII: S0360-8352(98)00105-3

Computers ind. Engng Vol. 34, No. 4, pp. 793–806, 1998

Printed in Great Britain
0360-8352/98 $ - see front matter

SCHEDULING IDENTICAL JOBS WITH UNEQUAL READY TIMES ON UNIFORM PARALLEL MACHINES TO MINIMIZE THE MAXIMUM LATENESS

MAGED M. DESSOUKY*

Department of Industrial and Systems Engineering, University of Southern California, Los Angeles, CA 90089-0193, U.S.A.

Abstract—We consider the problem of scheduling n identical jobs with unequal ready times on m parallel uniform machines to minimize the maximum lateness. This paper develops a branch-and-bound procedure that optimally solves the problem and introduces six simple single-pass heuristic procedures that approximate the optimal solution. The branch-and-bound procedure uses the heuristics to establish an initial upper bound. On sample problems, the branch-and-bound procedure in most instances was able to find an optimal solution within 100,000 iterations with $n \leq 80$ and $m \leq 3$. For larger values of m, the heuristics provided approximate solutions close to the optimal values. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords—parallel machine scheduling, maximum lateness, heuristics, branch-and-bound

1. INTRODUCTION

We consider the problem of scheduling n identical jobs with unequal ready times on m uniform parallel machines. In this paper, the term *identical jobs* means that all jobs have the same processing time on a given machine and the term *uniform machines* means that the processing time of a job on a particular machine is the ratio of the processing time of the job on a machine with a standard speed to the speed of the particular machine [1]. With identical jobs and uniform parallel machines, the processing time is only a function of the machine working on the job.

This configuration is applicable to capital intensive industries such as semiconductor manufacturing where it is common to find newer, more modern machines running side by side with older, less efficient machines which are kept in operation because of high replacement cost. In this case, the different machines could be processing identical products. This problem is also relevant in the garment industry where it is common to have seamstresses working at different speeds in parallel. Our objective will be to minimize the maximum lateness. This objective is representative of the cases where the impact of a single large delay is more harmful to a manufacturer than many small delays. Furthermore, its solution can be applied to solve the feasibility problem of identifying a schedule if one exists in which each job cannot be started before its ready time and must not be completed after its due date.

Each job has a distinct ready time r_j and due date d_j for $j = 1, \ldots, n$. The processing time of a job on machine i is p_i for $i = 1, \ldots, m$. The problem is to determine the job completion times C_j, $j = 1, \ldots, n$, that minimizes the maximum lateness, $L_{\max}$, where $L_j = C_j - d_j$ and $L_{\max} = \max_{j=1,\ldots,n} L_j$. We use the classification scheme as presented by Graham *et al.* [5] to formally state the problem as $Q|r_j, d_j, p_j = 1|L_{\max}$. The first field specifies the job and machine configuration as the n-job uniform parallel machine case. The second field states the job characteristics as the identical jobs case with unequal ready times and distinct due dates. The designation $p_j = 1$ implies that all jobs take the same time to process on a given machine. The third field specifies the objective function as minimizing the maximum lateness.

The majority of the literature on parallel machine scheduling considers the case of non-identical jobs and identical machines. For an excellent review of these types of problems the reader is referred to Ref. [1]. More recent research with non-identical jobs with equal ready times has

*Corresponding author. Tel.: +1-213-740-4891; Fax: +1-213-740-1120; E-mail: maged@rcf.usc.edu.

focused on the non-identical machines case. For unrelated parallel machines, Suresh and Chaudhuri [12] develop an exchange heuristic to minimize the maximum tardiness and Suresh and Chaudhuri [13] extend the heuristic for the bicriteria of minimizing the makespan and maximum tardiness. For uniform parallel machines, Guinet [6] develops a heuristic based on a transportation algorithm to also minimize the maximum tardiness.

For a single machine and identical jobs with unequal ready times, Lageweg *et al.* [7] present an $O(n^2)$ procedure to minimize the maximum lateness. The procedure guarantees an optimal solution only when all the ready times and due dates are integer multiples of the processing time. Simons [11] presents a recursive polynomial-time algorithm that guarantees optimality even when the ready times and due dates are not integer multiples of the processing time. With uniform parallel machines, Dessouky *et al.* [3] propose an $O(n \log m)$ algorithm to minimize the maximum lateness when the jobs have equal ready times. The algorithm uses the earliest completion time (ECT) rule to determine a set of minimum completion times. Then, the due dates are matched to the completion times in non-decreasing order of both.

We present six simple approximate heuristic solution procedures for the problem. A theoretical as well as an experimental comparison of the heuristics is performed. We also present an efficient branch-and-bound procedure that makes use of the heuristics to optimally solve the problem. In Section 3 we compare the performance of the heuristics to the optimal solution procedure on sample problems. The evaluation criteria for the optimal solution procedure (branch-and-bound) are (1) the percentage of times an optimal solution is found given a prespecified time limit in the search and (2) if an optimal solution is found the length of time required to derive the solution. The evaluation criteria for the heuristic procedures are (1) the departure from optimality measured by the minimum maximum lateness given by the heuristic sequence minus the optimal maximum lateness and (2) the percentage of times that the heuristic found an optimal solution.

2. PROBLEM FORMULATION

We present an integer programming model formulation for problem $Q|r_j,\ d_j,\ p_j=1|L_{\max}$. The following formulation is similar to the model presented by Suresh and Chaudhuri [12] for the scheduling of n non-identical jobs with equal ready times on m unrelated parallel machines. Let $x_{ijk}=1$ if job j is the kth job in the sequence on machine i and zero otherwise. Define c_{ik} as the completion time of the kth job in the sequence on machine i, $k = 1, \ldots, n$ and $i = 1, \ldots, m$.

Minimize $L_{\max}$ subject to:

$$\sum_{i=1}^{m}\sum_{k=1}^{n} x_{ijk} = 1 \quad j = 1, \ldots, n \tag{1}$$

$$\sum_{j=1}^{n} x_{ijk} \le 1 \quad i = 1, \ldots, m;\ k = 1, \ldots, n \tag{2}$$

$$p_i + c_{i,k-1} \le c_{i,k} \quad i = 1, \ldots, m;\ k = 1, \ldots, n \tag{3}$$

$$p_i + \sum_{j=1}^{n} r_j x_{ijk} \le c_{i,k} \quad i = 1, \ldots, m;\ k = 1, \ldots, n \tag{4}$$

$$\sum_{i=1}^{m}\sum_{k=1}^{n} c_{ik} x_{ijk} = C_j \quad j = 1, \ldots, n \tag{5}$$

$$C_j - d_j \le L_{\max} \quad j = 1, \ldots, n \tag{6}$$

$$C_j \ge 0, \quad c_{ik} \ge 0, \quad x_{ijk} \in (0, 1) \quad i = 1, \ldots, m;\ k = 1, \ldots, n;\ j = 1, \ldots, n \tag{7}$$

Equation (1) ensures that each job is assigned to only one position in the sequence for some particular machine. Equation (2) guarantees that not more than one job is assigned to any position in the sequence for any machine. Equation (3) restricts the kth job in the sequence of machine i from starting until the $k-1$th job in the sequence finishes processing and Equation (4) prevents a job from processing before its ready time. Equation (5) limits the completion time for each job and Equation (6) defines the maximum lateness, $L_{\max}$.

The number of constraints for the above formulation is $3n + 3nm$. The number of continuous and integer variables are $nm + n + 1$ and n^2m, respectively. For even relatively modest problem sizes, there are too many integer variables in the above formulation for the problem to be solved by a standard integer program solver. For example, for $m = 5$ and $n = 80$, there are 19200 integer variables. Hence, specialized algorithms that make use of the problem structure are required to solve the problem.

The complexity of problem $Q|r_j, d_j, p_j=1|L_{\max}$ is an open question in the literature. That is, it has not been shown that a known NP-hard problem is a special case of our problem. However, there are indications which are usually associated with hard problems. We note that the objective function of the problem can have local minimums. That is to say, certain scheduling strategies may be minimum in their immediate neighborhood without being globally optimal. Furthermore, Dessouky *et al.* [4] show that a similar problem $Q|r_j, d_j$ assign, $p_j=1|L_{\max}$ is strongly NP-hard. In their problem, the due dates may be *assigned* to any job. In the *assignable* due date problem, the idea is to match a due date from $(d_1, \ldots, d_n)$ to a particular job with ready time r_j, $j = 1, \ldots, n$, in order to minimize the maximum job lateness. It is easy to see that the assignable due date problem provides the scheduler more flexibility than our problem $Q|r_j, d_j, p_j=1|L_{\max}$ where the due dates are designated to a particular job with ready time r_j, $j = 1, \ldots, n$.

3. HEURISTIC PROCEDURES

We consider six single-pass heuristic procedures to find approximate solutions to problem $Q|r_j, d_j, p_j=1|L_{\max}$. Some of these heuristics provide an optimal solution to special cases of the problem. These procedures are based on similar heuristics developed by McMahon and Florian [10] and Larson and Dessouky [8] to solve the single machine sequencing problem to minimize the maximum lateness with *non-identical* jobs given ready times and designated due dates ($1|r_j, d_j, p_j|L_{\max}$). Problem $1|r_j, d_j, p_j|L_{\max}$ is NP-hard [9]. We generalize the heuristics to consider parallel machines.

In all heuristic procedures considered, a job is scheduled on a machine when either a job becomes ready for processing while a machine is free or the machine becomes available after completing another job while the job to be scheduled is waiting. Note that a job can be released at the same time a machine becomes available. In this case, there will be no job waiting time or idle machine time.

The heuristics differ in their selection of job and machine. The six procedures presented here fall into three categories, depending on the choice of the machine for a given job to schedule next. The first category selects the *fastest available machine* (FAM) and the next category selects the machine that gives the *earliest completion time* (ECT). These two categories proceed in a forward manner. The third category proceeds backward from an arbitrary distant point in time and selects the machine that gives the *latest start time* (LST). Within each machine category, two rules for choosing the job are considered. The first depends on the original given job parameters and the second depends on an updated version of these parameters. In the forward procedures the parameter in question is the ready times and in the backward procedure it is the due date. In a forward procedure the selected job is added at the end of the sequence on the machine selected. In a backward procedure, it is added at the beginning. We next present the heuristics.

3.1. Fastest available machine procedures

The first heuristic procedure selects the job to be scheduled next based on the ready times and schedules the selected job on the fastest available machine. This heuristic is referred to as pro-

cedure FAMR. That is, the job to be scheduled next is the job that has been waiting the longest among the unscheduled jobs and ties are broken by selecting the job with the minimum due date. The job is scheduled on the fastest available (idle) machine. If no machine is currently idle, wait for the first available machine. We now formally present heuristic FAMR. Let N be the set of all jobs and G_i the set of jobs that have been assigned to machine i, $i = 1, \ldots, m$, ordered based on the sequence of processing. The heuristic is a forward procedure that adds one job at a time to the end of set G_i.

3.1.1. Procedure FAMR

1. Order jobs in set N in non-decreasing order of ready times. Break ties by selecting the job with the minimum due date.
2. Let job j be the first job in set n. Let $s_{i,j}$ be the earliest start time of job j on machine i, i.e. $s_{i,j} = \max(r_j, a_i)$ for $i = 1, \ldots, m$ where a_i is the completion time of the last job scheduled on machine i. Let g be the machine that minimizes $s_{i,j}$, $i = 1, \ldots, m$. Break ties by selecting the machine with the smallest processing time (fastest). Add job j to the end of the sequence in set G_g and let $C_j = s_{g,j} + p_g$ and $a_g = C_j$. Remove job j from set N.
3. If N is an empty set, stop; otherwise, go to step 2.

3.1.2. Procedure FAMS

For due date performance measures, an obvious improvement to heuristic FAMR is to rank the jobs based on their earliest start times instead of simply on their ready times. In this manner, if two or more jobs are ready for processing when a machine becomes idle, the job with the minimum due date is selected instead of the job that has been waiting the longest (i.e. the job with the minimum ready time). The new heuristic referred to as FAMS may be formally stated as follows.

1. Set $s_{i,j} = r_j$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$.
2. Let h and g be the job and machine, respectively, that minimize $s_{i,j}$, $i = 1, \ldots, m$ and $j \in N$. Break ties in j by selecting the job with the minimum due date and break ties in i by selecting the machine with the smallest processing time. Add job h to the end of the sequence in set G_g and let $C_h = s_{g,h} + p_g$. Remove job h from set N. Update $s_{g,l} = \max(r_l, C_h)$ for all $l \in N$.
3. If N is an empty set, stop; otherwise, go to step 2.

3.2. Earliest completion time procedures

The above two heuristics are myopic in the sense that an unscheduled job is scheduled on the fastest machine currently idle. It might be better for the job to wait for a faster machine that is currently busy but will be available in the near term. We now present two forward sequencing procedures that look ahead in time. The heuristic procedures make use of the earliest completion time (ECT) rule to select the machine for processing and are referred to as procedures ECTR and ECTS, respectively. If more than one machine can complete a job at the same time, the machine with the longest processing time is selected so that the faster processing machines can be saved for later jobs. As before, in procedure ECTR the job is selected based on the ready times and in procedure ECTS the job is selected based on the earliest start times, and ties in both procedures are broken by selecting the job with the smallest due date. The heuristics can be formally presented as follows.

3.2.1. Procedure ECTR

1. Order jobs in set N in non-decreasing order of ready times. Break ties by selecting the job with the minimum due date.
2. Let job j be the first job in set N. Let $t_{i,j}$ be the completion time of job j if scheduled on machine i, i.e. $t_{i,j} = \max(r_j, a_i) + p_i$ for $i = 1, \ldots, m$ where a_i is completion time of the last job in the sequence in set G_i. Let g be the machine that minimizes $t_{i,j}$, $i = 1, \ldots, m$. Break ties by

selecting the machine with the longest processing time (the slowest machine). Add job j to the end of the sequence in set G_g and let $C_j = t_{g,j}$ and $a_g = C_j$. Remove job j from set N.
3. If N is an empty set, stop; otherwise, go to step 2.

3.2.2. Procedure ECTS

1. Set $s_{i,j} = r_j$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$.
2. Let job j_i be the job with the minimum $s_{i,j}$ on machine i for $j \in N$ and $i = 1, \ldots, m$. Among ties in $s_{i,j}$, select the job with the minimum due date. Let t_{i,j_i} be the completion time of job j_i if scheduled on machine i, i.e. $t_{i,j_i} = \max(r_{j_i}, a_i) + p_i$ for $i = 1, \ldots, m$ where a_i is the completion time of the last job in the sequence in set G_i. Let g be the machine that minimizes t_{i,j_i}, $i = 1, \ldots, m$. Break ties by selecting the slowest machine. Let h be job j_g. Add job h to the end of the sequence in set G_g and let $C_h = t_{g,h}$ and $a_g = C_h$. Remove job h from set N. Update $s_{g,l} = \max(r_l, C_h)$ for all $l \in N$.
3. If N is an empty set, stop; otherwise, go to step 2.

We now prove that the maximum lateness given by heuristic ECTS will be no worse than the maximum lateness given by heuristic ECTR.

Proposition 1. *Let L^1_{max} be the maximum lateness given by heuristic ECTS and L^2_{max} be the maximum lateness given by heuristic ECTR. Then, $L^1_{max} \leq L^2_{max}$.*

Proof. The two heuristics generate a schedule with identical busy periods for each machine i, $i = 1, \ldots, m$. What may differ in the solution between the two heuristics is the job allocation to the busy periods. Given a known busy period with earliest completion time and two or more jobs with ready times less than the start time of the busy period, choosing the job with the minimum due date gives a no worse maximum lateness than any other selection rule. Note this is the selection criterion for heuristic ECTS. □

We remark that a similar argument cannot be made for heuristics FAMR and FAMS. That is, we cannot ensure that the solution from heuristic FAMS will be no worse than the solution from heuristic FAMR. However, in Section 5 we show on average that heuristic FAMS outperforms FAMR.

3.3. Latest start time procedures

We now present two backward sequencing heuristics that are analogous to heuristic procedures ECTR and ECTS but in reverse time. The heuristics make use of the latest start time (LST) rule to select the machine for processing and are referred to as LSTD and LSTF, respectively. That is, an unscheduled job is assigned to a machine that can start processing it the latest. Let S_j, $j = 1, \ldots, n$, be the latest start time of job j. In procedure LSTD the job is selected based on the due dates and in procedure LSTF the job is selected based on the possible finish times and in both procedures ties are broken by selecting the job with the largest ready time. In the backward sequencing heuristics the unscheduled jobs are added to the front of set G_i.

3.3.1. Procedure LSTD

1. Order jobs in set N in non-increasing order of due dates. Break ties by selecting the job with the maximum ready time.
2. Let job j be the first job in set n. Let $e_{i,j}$ be the start time of job j if scheduled on machine i, i.e. $e_{i,j} = \min(d_j, q_i) - p_i$ for $i = 1, \ldots, m$ where q_i is the start time of the first job placed in the sequence in set G_i. Let g be the machine that maximizes $e_{i,j}$, $i = 1, \ldots, m$. Add job j to the beginning of the sequence in set G_g and let $S_j = e_{g,j}$ and $q_g = S_j$. Remove job j from set N.
3. If N is an empty set, stop; otherwise, go to step 2.

3.3.2. Procedure LSTF

1. Set the latest finish times $f_{i,j} = d_j$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$.
2. Let job j_i be the job with the maximum $f_{i,j}$ on machine i for $j \in N$ and $i = 1, \ldots, m$. Among ties in $f_{i,j}$, select the job with the maximum ready time. Let e_{i,j_i} be the start time of job j_i if

scheduled on machine i, i.e. $e_{i,j_i} = \min(d_{j_i}, q_i) - p_i$ for $i = 1, \ldots, m$ where q_i is the start time of the first job placed in the sequence in set G_i. Let g be the machine that maximizes e_{i,j_i}, $i = 1, \ldots, m$, and let h be job j_g. Add job h to the beginning of the sequence in set G_g and let $S_h = e_{g,h}$ and $q_g = S_h$. Remove job h from set N. Update $f_{g,l} = \min(d_l, S_h)$ for all $l \in N$.

3. If N is an empty set, stop; otherwise, go to step 2.

We now discuss the symmetry properties between procedures ECTS and LSTF. These properties are extensions to the properties developed by Dessouky and Larson [2] for the single machine case. First note that with the objective to minimize the maximum lateness, constant values can be added to or subtracted from all the ready times or all the due dates without changing the optimal sequence. Given a vector of ready times $\mathbf{r} = (r_1, \ldots, r_n)$ and a vector of due dates $\mathbf{d} = (d_1, \ldots, d_n)$, heuristic ECTS with ready times and due dates $(\mathbf{r}, \mathbf{d})$ gives the reverse job sequence on each machine as heuristic LSTF with ready times and due dates $(\mathbf{K}_1 - \mathbf{d}, \mathbf{K}_2 - \mathbf{r})$ where $\mathbf{K}_1$ and $\mathbf{K}_2$ are vectors of large positive constants. This symmetrical property also holds between procedures ECTR and LSTD. Therefore, analogous to the dominance condition for the forward procedures, the maximum lateness given by heuristic LSTF will be no worse than the maximum lateness given by heuristic LSTD. We summarize this result in the following proposition.

Proposition 2. *Let L^1_{max} be the maximum lateness given by heuristic LSTF and L^2_{max} be the maximum lateness given by heuristic LSTD. Then, $L^1_{max} \leq L^2_{max}$.*

When all jobs have identical ready times or identical due dates, we note that the above procedures are identical to the ones presented by Dessouky *et al.* [3]. Hence, for these cases optimal solutions may be found in polynomial-time. We summarize the results in the following propositions.

Proposition 3. *Procedures ECTR and ECTS optimally solve problem* $Q|\mathrm{r_j} = \mathrm{r}, \mathrm{d_j}, \mathrm{p_j} = 1|\mathrm{L_{max}}$.

Proposition 4. *Procedures LSTD and LSTF optimally solve problem* $Q|\mathrm{r_j}, \mathrm{d_j} = \mathrm{d}, \mathrm{p_j} = 1|\mathrm{L_{max}}$.

4. BRANCH-AND-BOUND PROCEDURE

For problem $Q|r_j, d_j, p_j = 1|L_{\max}$, the single-pass heuristics presented in Section 3 do not guarantee an optimal solution even when the order of jobs according to ready times is identical to that of their due dates. Consider the following example: $n = 2$, $m = 2$, $r_1 = 0$, $r_2 = 8$, $d_1 = 10$, $d_2 = 12$, $p_1 = 10$, $p_2 = 11$. Procedures ECTR and ECTS schedule job 1 on machine 1 and job 2 on machine 2, resulting in $L_{\max} = 7$, while the optimal schedule is job 1 on machine 2 and job 2 on machine 1 resulting in $L_{\max} = 6$. Similarly, an alternative example can be easily developed to show that procedures LSTD and LSTF are also sub-optimal.

With the same ordering of ready times and due dates, the single-pass heuristics guarantee the generation of optimal sequences only when $m = 1$. For $m = 1$ and a general ordering of ready times and due dates, the single-pass heuristics may again generate sub-optimal schedules. When $m = 1$, Simons [11] presents a polynomial-time algorithm that recursively uses a procedure similar to ECTS which optimally solves the problem for a general ordering of ready times and due dates. The main differences between Simon's algorithm and ECTS are that jobs may be removed from the partial schedule and that sets of jobs may be collected together to form scheduling subproblems [11]. Refer to Simons's procedure as algorithm Opt_m1. This algorithm forms the basis for our branch-and-bound procedure.

For this problem, a total schedule is defined by an allocation of jobs to machines, $\boldsymbol{\sigma}^t$, and a sequence of the jobs allocated to each machine, $\boldsymbol{\theta}^t$. The number of possible allocations of jobs to machines for this problem is $O(m^n)$, and for each machine i, $i = 1, \ldots, m$, the number of possible sequences of jobs assigned to it is $n_i!$ where n_i is the number of jobs on machine i with $\Sigma_{i=1}^{m} n_i = n$. The number of possible sequences on all machines for a particular allocation, $\boldsymbol{\sigma}^t$, is $\prod_{i=1}^{m} n_i!$ with an upper bound $n!$. This makes the upper bound on the number of possible

sequences, $\boldsymbol{\theta}^t$, $O((m^n)n!)$. A branch-and-bound procedure that searches in this complete space will thus have a worst case node complexity of $O(mn^n)$. However, given a particular job to machine allocation, an optimal solution can be found by using algorithm Opt_m1 to sequence each machine. Therefore, the branch-and-bound procedure focuses only on the allocation of jobs to machines. This observation is summarized in the following proposition.

Proposition 5. There exists an optimal solution to problem $Q|r_j, d_j, p_j = 1|L_{max}$ where each machine's job sequence is determined by solving problem $1|r_j, d_j, p_j = 1|L_{max}$.

4.1. Overview of procedure

The solution method follows a branch-and-bound procedure, with each node in the branching tree representing a partial assignment of jobs to machines $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)$, where σ_i ($i = 1, \ldots, m$) is the partial allocation on machine i. Let the job sequence on all machines given the partial allocation, $\boldsymbol{\sigma}$, be $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$, where θ_i ($i = 1, \ldots, m$) is the sequence on machine i. The root of the tree represents $\sigma = \varnothing$. A node at level l in the branching tree represents the allocation of jobs $1, \ldots, l$. Let h be the set of jobs not in $\boldsymbol{\sigma}$. Branching from a node represents the allocation of a job $j \in H$ to some machine σ_i, $i = 1, \ldots, m$. The choice of job j from h is based on a non-decreasing order of r_j. Hence, we index jobs in h in a non-decreasing order of r_j. Given the partial allocation of jobs to machines $\boldsymbol{\sigma}$, we now present a procedure that determines a lower bound, $\underline{L}_{max}(\boldsymbol{\sigma})$. The first part of the procedure determines a lower bound for the jobs in the allocated set $\boldsymbol{\sigma}$ and the second part for the unscheduled jobs in set h.

4.2. Procedure LB

1. Sequence the jobs allocated to machine i, σ_i, using the single machine algorithm Opt_m1 by Simons [11] to get θ_i, for $i = 1, \ldots, m$.
2. For the sequence θ_i, compute the lateness $L_j = C_j - d_j$ for $j \in \sigma_i$.
3. Set $r_{min} = \min_{j \in H} r_j$. Set $r_j' = r_{min}$ for $j \in H$.
4. Solve the sequencing problem of jobs in h as the equal ready time problem of $Q|r_j = r_{min}, d_j, p_j = 1|L_{max}$ using procedure ECTS. Let the resulting job to machine allocation and sequence be $\boldsymbol{\sigma}'$ and $\boldsymbol{\theta}'$, respectively.
5. For each sequence θ_i', $i = 1, \ldots, m$, compute the completion times C_j using the ready times r_j' and compute the lateness $L_j = C_j - d_j$ for $j \in H$.
6. Set the lower bound $\underline{L}_{max}(\boldsymbol{\sigma}) = \max_{j = 1, \ldots, n} L_j$.

The validity of the proposed branch-and-bound procedure is derived from the use of algorithm Opt_m1 in both branching and bounding. The rationale of the branching procedure is to limit the search to all possible job-to-machine allocations rather than schedules, since the optimum sequence for a given allocation can be obtained in polynomial time by applying Opt_m1 to sequence jobs on each machine. This gives rise to a complexity of $O(m^n)$ instead of $O(mn^n)$ in the number of nodes in the search tree. The validity of the proposed lower bound on L_{max} given a particular partial allocation is based on the way the sets of jobs, those already allocated and those unallocated, are scheduled. The allocated set is scheduled using Opt_m1, which provides the minimum L_{max} for that set. The unallocated set is scheduled using ECT, assuming the ready times of all jobs in the set to be equal to the minimum in the set. This yields a lower bound on L_{max} for all unallocated jobs, *regardless of the sequencing of allocated jobs*.

It is also because of the application of algorithm Opt_m1 that the order of selecting jobs to add to a partial allocation does not affect the final solution, since jobs are later resequenced using that algorithm. However, selecting the job with minimum ready time helps to improve the efficiency of the algorithm, since it allows the partitioning of the problem, as explained in the following discussion. Let a_i ($i = 1, \ldots, m$), be the completion time of the last job sequenced on machine i given by σ_i. The quantity a_i represents the availability time of machine i to process an additional unscheduled job. If $a_i \leq r_{min}$ for all i, $i = 1, \ldots, m$, then the two sets $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ are disjoint and a complete job to machine allocation, $\boldsymbol{\sigma}^t$, can be given by $\boldsymbol{\sigma}^t = \boldsymbol{\sigma} \cup \boldsymbol{\sigma}'$ and a complete job sequence, $\boldsymbol{\theta}^t$, can be given by $\boldsymbol{\theta}^t = \boldsymbol{\theta} \cup \boldsymbol{\theta}'$.

4.3. Complete branch-and-bound procedure

Let $\mathbf{\Omega}$ be the list of open nodes. An open node is closed and removed from $\mathbf{\Omega}$ if all its branches have been generated or its lower bound exceeds an existing feasible solution. Note that the maximum number of nodes that the branch-and-bound procedure will evaluate is $O(m^n)$. The complete branch-and-bound procedure is as follows.

1. Set the upper bound $L^{\mathrm{u}}_{\max}$ to the minimum maximum lateness given by the procedures FAMR, FAMS, ECTS and LSTF and let $\boldsymbol{\theta}^{\mathrm{u}}$ be the resulting job sequence. Initialize the list of open nodes $\mathbf{\Omega}$ as the root node $\boldsymbol{\sigma} = \varnothing$. Let the set of unscheduled jobs h be the set of all jobs N.
2. Remove an open node from $\mathbf{\Omega}$ and let the partial job to machine allocation given by that node be $\boldsymbol{\sigma}$ and the sequence of jobs given by $\boldsymbol{\sigma}$ be $\boldsymbol{\theta}$. Let h be the set of jobs not in $\boldsymbol{\sigma}$. Let the relaxed ready times be r_j' and the actual ready times be r_j for $j = 1, \ldots, n$. Set $r_{\min} = \min_{l \in H} r_l$, $r_j' = r_j$ for $j \in \boldsymbol{\sigma}$, and $r_j' = r_{\min}$ for $j \in H$. Let the machine availability time a_i, $i = 1, \ldots, m$, be the completion time of the last job in the sequence θ_i.
3. Use procedure LB to obtain the lower bound $\underline{L}_{\max}(\boldsymbol{\sigma})$. If $\underline{L}_{\max}(\boldsymbol{\sigma}) \geq L^{\mathrm{u}}_{\max}$, go to step 8. Otherwise go to step 4.
4. If $a_i \leq r_{\min}$ for all i, set the complete job to machine allocation, $\boldsymbol{\sigma}^{\mathrm{t}}$, to $\boldsymbol{\sigma}^{\mathrm{t}} = \boldsymbol{\sigma} \cup \boldsymbol{\sigma}'$ and set the complete job sequence, $\boldsymbol{\theta}^{\mathrm{t}}$, to $\boldsymbol{\theta}^{\mathrm{t}} = \boldsymbol{\theta} \cup \boldsymbol{\theta}'$, and go to step 5. Otherwise, go to step 6.
5. Check if the complete allocation $\boldsymbol{\sigma}^{\mathrm{t}}$ is optimal given the partial allocation $\boldsymbol{\sigma}$. If $C_j - p_{b(j)} \geq r_j$ for each $j \in H$ and $i = 1, \ldots, m$, where $b(j)$ is the machine allocated to job j in $\boldsymbol{\sigma}^{\mathrm{t}}$, then the complete job to machine allocation $\boldsymbol{\sigma}^{\mathrm{t}}$ is an optimal allocation given the partial allocation $\boldsymbol{\sigma}$. Update $L^{\mathrm{u}}_{\max} = \underline{L}_{\max}(\boldsymbol{\sigma})$ and $\boldsymbol{\theta}^{\mathrm{u}} = \boldsymbol{\theta}^{\mathrm{t}}$ and go to step 8. Otherwise, go to step 6.
6. Generate a feasible schedule for the complete allocation $\boldsymbol{\sigma}^{\mathrm{t}}$. For the sequence θ_i^{t}, forward schedule on machine i, $i = 1, \ldots, m$, to compute the new completion times C_j using the actual ready times r_j and compute the lateness $L_j = C_j - d_j$ for $j \in \sigma_i^{\mathrm{t}}$. If $L^{\mathrm{u}}_{\max} > \max_{j = 1, \ldots, n} L_j$, update $L^{\mathrm{u}}_{\max} = \max_{j = 1, \ldots, n} L_j$ and $\boldsymbol{\theta}^{\mathrm{u}} = \boldsymbol{\theta}^{\mathrm{t}}$.
7. Let j be the job with the minimum ready time in set h and break ties by selecting the job with the minimum due date. For each machine i, $i = 1, \ldots, m$, generate a new branch node that represents the allocation of job j to machine i. The generation of the branch node is as follows. Set $\sigma_y = \sigma_i$, $y \neq i$. Set $\sigma_i = \sigma_i \cup \{j\}$ and perform the following test. If $r_j \geq a_i$, add j to the end of the partial sequence θ_i; otherwise, recompute a new sequence θ_i with the jobs in allocation σ_i using algorithm Opt_m1. Refer to this allocation and sequence as $\boldsymbol{\sigma}^i$ and $\boldsymbol{\theta}^i$. Add the allocation $\boldsymbol{\sigma}^i$ and resulting sequence $\boldsymbol{\theta}^i$ as an open node in $\mathbf{\Omega}$. Note that the set of offspring nodes to $\boldsymbol{\sigma}$ is the nodes $\boldsymbol{\sigma}^i$, $i = 1, \ldots, m$.
8. If there are open nodes in the list $\mathbf{\Omega}$, go to step 2; otherwise, the optimal sequence is given by $\boldsymbol{\theta}^{\mathrm{u}}$ and the optimal maximum lateness is given by $L^{\mathrm{u}}_{\max}$.

In Section 5, we discuss the most efficient branching search strategy from nodes in $\mathbf{\Omega}$. An overview of the branch-and-bound algorithm is provided in Fig. 1.

5. NUMERICAL EXPERIMENTS

We experimentally compared the effectiveness of solutions generated by the heuristic procedures with the optimal solutions generated by the branch-and-bound procedure. The purpose of the experimentation was two-fold. First, it was desired to find how large the problem sizes in terms of m and n the branch-and-bound procedure can efficiently solve. Second, for cases where the branch-and-bound procedure cannot find an optimal solution it was desired to identify the best performing heuristic for different scenarios of the ready times and due dates.

Since the number of nodes to evaluate in the branch-and-bound procedure is $O(m^n)$, we tested the effectiveness of the solution procedures on different combinations of the number of machines and the number of jobs. The values tested for the number of jobs were 10, 20, 40 and 80 and for the number of machines were 2, 3 and 5. Integer data of the processing time on each machine and the ready time and due date of each job were generated from uniform distributions between 1 and $P_{\max}$, $R_{\max}$ and $D_{\max}$, respectively. Note that adding a constant to all values of

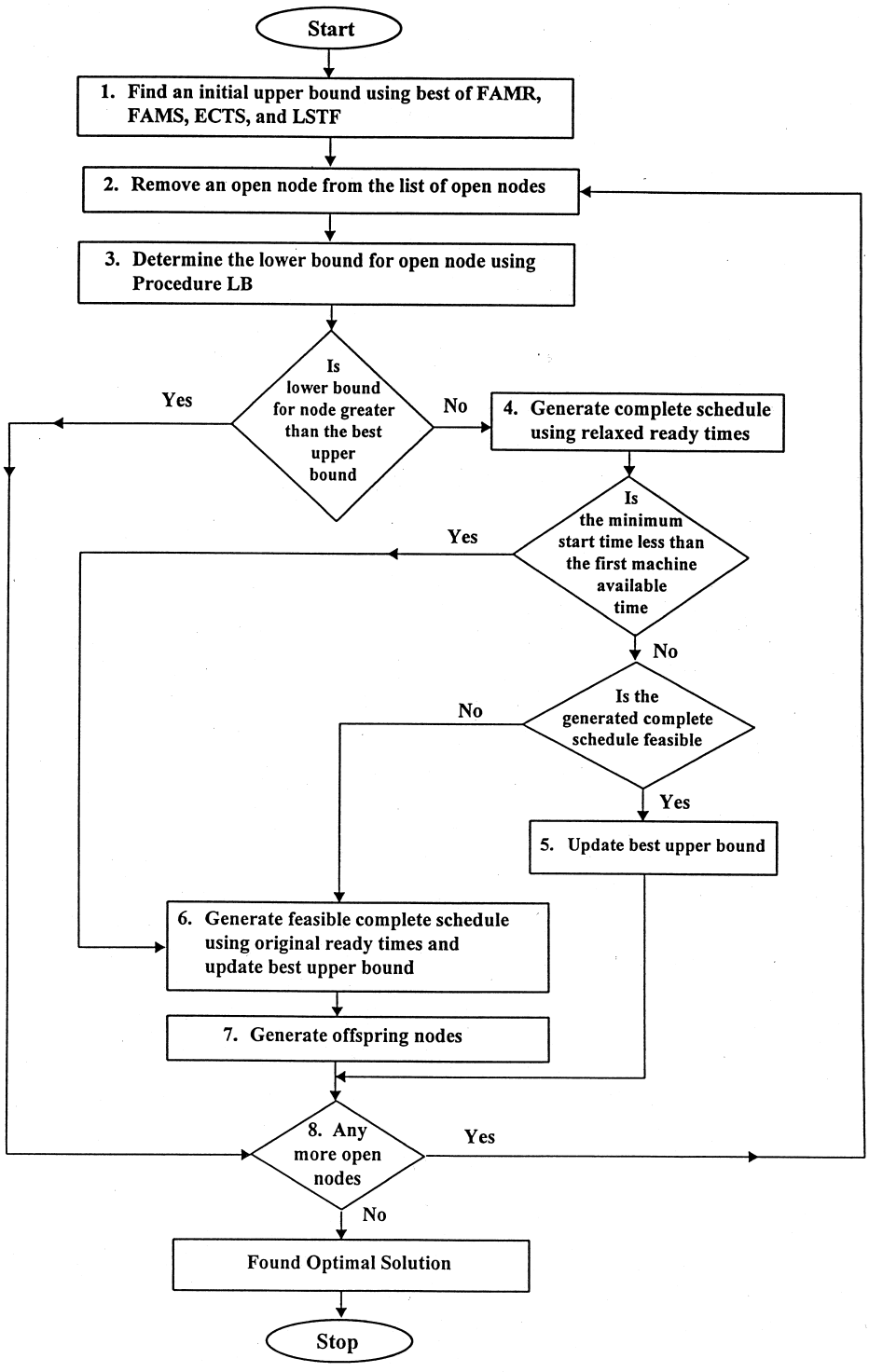

Fig. 1. Overview of branch-and-bound procedure.

ready times or due dates does not change the optimal sequence, though it will affect the lateness of jobs and thus $L_{\max}$ [7, 8]. Hence, it is the relative values of the ready times and due dates, rather than the absolute values, that affect the optimum sequence. Accordingly, we designed experiments to cover different ranges of ready times and due dates, expressed as [1, $R_{\max}$] and [1, $D_{\max}$], respectively. Also, the range of processing times is defined as [1, $P_{\max}$]. Tested ranges for both $R_{\max}$ and $D_{\max}$ were 10, 50 and 100 and for $P_{\max}$ were 10 and 50.

A depth-first branching strategy was used for the branch-and-bound procedure with the offspring nodes generated in order of the machine's processing time. This branching strategy was used because of its minimal storage requirements and it consistently outperformed other branching strategies (e.g. breadth-first and smallest lower bound). We stopped the branch-and-bound procedure after 100,000 nodes were evaluated so that many experiments could be performed. For each scenario, 30 experiments were run. The run-time for evaluating one node in the branch-and-bound procedure on a Hewlett-Packard Series 9000 Workstation was on the average 0.02 CPU second.

To gain insight and understand the relationship between the various problem parameters, we first held the number of jobs fixed to a small number, $n = 10$, and varied the other parameters of the model. Then, we tested the sensitivity of the results for larger values of the number of

Table 1. Number of runs and optimal solution is reached out of 30 runs, with $n = 10$, for branch-and-bound and heuristic procedures

				Branch-and-bound			Heuristics			
							number of times optimal			
m	P_{max}	R_{max}	D_{max}	number optimal	average # nodes	maximum # nodes	FAMR	FAMS	ECTS	LSTF
2	10	10	10	30	5.5	75	5	16	28	27
2	10	10	50	30	23.3	261	10	20	21	28
2	10	10	100	30	13.6	201	13	23	23	29
2	10	50	10	30	8.9	93	26	26	30	21
2	10	50	50	30	42.2	661	24	24	26	25
2	10	50	100	30	14.8	177	24	24	26	26
2	10	100	10	30	62.0	1031	27	27	30	22
2	10	100	50	30	91.9	983	28	28	29	25
2	10	100	100	30	38.8	907	27	27	29	26
2	50	10	10	30	1.0	1	4	13	26	28
2	50	10	50	30	32.1	663	0	10	23	20
2	50	10	100	30	62.2	403	0	10	15	19
2	50	50	10	30	4.7	95	2	7	16	22
2	50	50	50	30	19.0	185	0	9	16	20
2	50	50	100	30	26.6	225	2	8	11	21
2	50	100	10	30	4.9	33	13	16	21	14
2	50	100	50	30	13.1	235	9	14	19	15
2	50	100	100	30	26.2	315	6	8	18	16
3	10	10	10	30	435.9	8509	6	13	22	26
3	10	10	50	30	738.2	11524	11	15	19	28
3	10	10	100	30	1101.0	20887	16	19	19	30
3	10	50	10	30	2703.2	55696	27	27	29	22
3	10	50	50	30	2907.9	49957	27	27	28	27
3	10	50	100	30	634.8	15529	28	28	28	29
3	10	100	10	30	3553.3	55810	30	30	30	25
3	10	100	50	30	2335.1	50749	30	30	30	26
3	10	100	100	30	2067.9	44542	30	30	30	29
3	50	10	10	30	346.4	9490	1	8	22	23
3	50	10	50	30	306.8	2632	1	7	17	19
3	50	10	100	30	1232.4	8137	2	4	8	13
3	50	50	10	30	20.2	334	5	12	20	19
3	50	50	50	30	624.5	7495	5	9	15	13
3	50	50	100	30	1473.8	26590	8	11	11	13
3	50	100	10	30	724.2	11149	12	13	25	11
3	50	100	50	30	1527.2	18388	15	16	23	11
3	50	100	100	30	3233.2	53383	15	15	22	8
5	10	10	10	28	3706.6	73836	19	21	26	25
5	10	10	50	27	1432.1	19531	25	25	27	26
5	10	10	100	27	1417.9	19531	27	27	27	26
5	10	50	10	26	4573.9	96591	24	24	26	23
5	10	50	50	27	1679.9	21656	27	27	27	24
5	10	50	100	29	1232.5	19531	29	29	29	28
5	10	100	10	24	4938.0	97656	23	23	23	22
5	10	100	50	26	1635.4	19531	26	26	26	23
5	10	100	100	27	1628.2	19531	27	27	27	27
5	50	10	10	30	1263.0	29141	0	0	23	16
5	50	10	50	28	2350.6	14436	1	2	9	12
5	50	10	100	27	3442.4	22176	2	3	7	19
5	50	50	10	29	1009.1	15651	8	8	19	12
5	50	50	50	26	1445.5	12896	6	6	14	13
5	50	50	100	25	1065.8	17601	8	10	13	15
5	50	100	10	28	4527.8	68671	16	16	23	13
5	50	100	50	28	3274.9	41401	19	19	21	13
5	50	100	100	25	1698.3	24511	21	21	18	16

jobs, n. Table 1 shows the results of the branch-and-bound procedure. For each scenario, Table 1 lists the number of times an optimal solution was found out of the 30 runs and the average and the maximum number of nodes evaluated in the branch-and-bound procedure to find an optimal solution. Table 1 also lists the number of times each heuristic gave the optimal solution out of the 30 runs. Table 2 reports for each heuristic the average departure from optimality measured by the maximum lateness given by the heuristic sequence minus the optimal maximum lateness. The general findings of these experiments are summarized as follows:

(1) Comparing the heuristics: As Tables 1 and 2 show, the most myopic of the heuristic rules, FAMR, is consistently outperformed by some other heuristic in all the experiments. The two best performing heuristics were ECTS and LSTF. Heuristic ECTS performed best on average when $R_{\max}$ is greater than $D_{\max}$ while LSTF performed best on average when $D_{\max}$ is greater than $R_{\max}$. These results suggest a forwarding sequencing heuristic based on ready times be used

Table 2. Average deviation from the optimal solution for the various heuristics with $n = 10$

				Average deviation from optimality			
m	$P_{\max}$	$R_{\max}$	$D_{\max}$	FAMR	FAMS	ECTS	LSTF
2	10	10	10	2.967	0.933	0.067	0.100
2	10	10	50	5.367	1.000	0.700	0.100
2	10	10	100	4.100	0.733	0.700	0.033
2	10	50	10	0.500	0.500	0.000	0.567
2	10	50	50	0.367	0.367	0.200	0.367
2	10	50	100	0.433	0.433	0.200	0.367
2	10	100	10	0.300	0.300	0.000	0.600
2	10	100	50	0.133	0.133	0.033	0.400
2	10	100	100	0.233	0.233	0.033	0.367
2	50	10	10	5.533	3.600	0.167	0.200
2	50	10	50	20.500	6.833	1.100	1.000
2	50	10	100	37.100	8.067	3.933	1.867
2	50	50	10	7.500	5.967	1.733	1.067
2	50	50	50	17.400	6.300	1.800	1.233
2	50	50	100	27.667	5.300	2.433	1.600
2	50	100	10	5.967	4.933	1.033	2.833
2	50	100	50	11.933	5.633	1.333	2.000
2	50	100	100	15.500	5.667	2.300	2.300
3	10	10	10	3.033	1.400	0.367	0.333
3	10	10	50	2.867	1.200	0.633	0.100
3	10	10	100	2.300	0.933	0.633	0.000
3	10	50	10	0.267	0.267	0.033	0.433
3	10	50	50	0.233	0.233	0.100	0.133
3	10	50	100	0.233	0.233	0.167	0.033
3	10	100	10	0.000	0.000	0.000	0.367
3	10	100	50	0.000	0.000	0.000	0.233
3	10	100	100	0.000	0.000	0.000	0.033
3	50	10	10	8.100	6.033	0.533	0.400
3	50	10	50	20.733	8.600	2.700	1.667
3	50	10	100	30.633	10.933	6.933	2.700
3	50	50	10	7.000	5.800	0.733	1.000
3	50	50	50	15.600	8.400	3.200	2.567
3	50	50	100	19.400	7.633	4.200	3.267
3	50	100	10	6.933	6.467	0.433	3.633
3	50	100	50	9.167	6.833	1.600	3.400
3	50	100	100	10.033	7.167	2.700	4.100
5	10	10	10	0.867	0.633	0.067	0.133
5	10	10	50	0.300	0.167	0.000	0.067
5	10	10	100	0.133	0.033	0.033	0.033
5	10	50	10	0.133	0.133	0.000	0.233
5	10	50	50	0.000	0.000	0.000	0.133
5	10	50	100	0.000	0.000	0.000	0.033
5	10	100	10	0.033	0.033	0.033	0.233
5	10	100	50	0.000	0.000	0.000	0.133
5	10	100	100	0.000	0.000	0.000	0.033
5	50	10	10	13.733	12.433	1.067	1.500
5	50	10	50	21.000	15.167	4.800	3.267
5	50	10	100	18.900	12.133	4.067	1.033
5	50	50	10	11.033	10.533	1.667	3.000
5	50	50	50	12.567	10.767	1.733	2.700
5	50	50	100	11.633	9.167	2.033	1.067
5	50	100	10	3.400	3.400	0.233	4.000
5	50	100	50	2.700	2.700	0.667	2.600
5	50	100	100	2.433	2.433	1.133	1.333

when the variability in ready times is the greatest and a backward sequencing heuristic based on due dates be used when the variability in due dates is the greatest.

(2) Sensitivity to R_{max} and D_{max}: The average time to compute an optimal solution in most of the scenarios is greater for the case when $R_{max} > D_{max}$ than for the case when $R_{max} < D_{max}$. For example, the average number of nodes evaluated is 1101.0 when $m = 3$, $P_{max} = 10$, $R_{max} = 10$ and $D_{max} = 100$ while it increases to 3553.3 when R_{max} and D_{max} are reversed. These findings are not surprising since the proposed branch-and-bound procedure is a forward sequencing procedure and as R_{max} increases the lower bounds in the procedure become less tight. These results suggest that a backward sequencing branch-and-bound procedure be used when $R_{max} > D_{max}$. Because of the symmetry of the problem, it might be easier to apply the same forward sequencing branch-and-bound procedure using the reverse data set ($\mathbf{K}_1-\mathbf{d}$, $\mathbf{K}_2-\mathbf{r}$), where $\mathbf{K}_1$ and $\mathbf{K}_2$ are vectors of large positive constants, with the objective to minimize E_{max}, where $E_{max} = \min_{j=1,\ldots,n}(r_j - S_j)$. The procedure will give the reverse of the optimal sequence on each machine given by a backward branch-and-bound procedure on the original data set (**r**, **d**).

(3) Sensitivity to m: Since the number of feasible sequences increases exponentially with m, the time it takes to find an optimal solution also increases as m increases. As Table 1 shows, the branch-and-bound procedure consistently found the optimal solution when there were 3 or less uniform parallel machines. For $m = 5$, the branch-and-bound procedure was stopped before finding an optimal solution in some of the scenarios. For example, with $P_{max} = 50$, $R_{max} = 50$ and $D_{max} = 50$, the procedure was able to find the optimal solution in 26 out of the 30 runs. For this scenario, a heuristic may be required to approximate the optimal solution. Fortunately, as m increases the heuristics on average provided a better approximation in terms of the absolute deviation from optimality. The heuristics performed better when m increases while n is constant because with more machines it is less likely that any interchange in the job sequence on a particular machine given by the best single-pass heuristic will improve the lateness criteria since there are fewer jobs scheduled on each machine.

(4) Sensitivity to P_{max}: It takes a little longer on average to find an optimal solution when $P_{max} = 10$ than when $P_{max} = 50$ because with a smaller processing time and the same R_{max} the more likely there is idle time on a machine between two adjacent jobs in the sequence, thereby reducing the quality of the lower bound in the branch-and-bound procedure. However, as Table 1 shows the heuristics provide a near-optimal approximation with a small P_{max}.

(5) Sensitivity to n: We now focus on testing the sensitivity of the results for larger values of n. Because of the symmetry of the problem, we considered only the cases when $R_{max} \leq D_{max}$ since similar conclusions may be drawn when $R_{max} \geq D_{max}$. These experiments evaluated the performance of the branch-and-bound procedure as n increases as well as the performance of heuristic

Table 3. Performance of branch-and-bound procedure and heuristic LSTF for $P_{max} = 50$

				Branch-and-bound			LSTF	
m	n	R_{max}	D_{max}	number optimal	average # nodes	maximum # nodes	number optimal	deviation optimal
2	20	50	50	30	129	3835	23	0.500
2	20	50	100	30	135	2097	19	1.133
2	40	50	50	30	0.01	1	30	0.000
2	40	50	100	30	0.01	1	27	0.233
2	80	50	50	30	0.01	1	29	0.033
2	80	50	100	30	0.01	1	27	0.133
3	20	50	50	30	1.152	23272	24	0.533
3	20	50	100	30	4.065	81346	16	1.333
3	40	50	50	30	0.01	1	25	0.300
3	40	50	100	30	03.4	988	18	1.033
3	80	50	50	30	0.01	1	26	0.200
3	80	50	100	30	0.01	1	21	0.600
5	20	50	50	20	1.124	33016	12	1.567
5	20	50	100	17	119	1801	7	2.367
5	40	50	50	28	0.01	1	20	0.833
5	40	50	100	22	01.2	336	15	1.100
5	80	50	50	30	0.01	1	25	0.200
5	80	50	100	29	0.01	1	15	0.833

Table 4. Summary of experimental findings

Range on parameters	Suggested procedure
$m \leq 3$ and $R_{max} < D_{max}$	forward sequencing branch-and-bound algorithm
$m \leq 3$ and $D_{max} < R_{max}$	backward sequencing branch-and-bound algorithm
$m > 3$ and $D_{max} < R_{max}$	heuristic ECTS
$m > 3$ and $R_{max} < D_{max}$	heuristic LSTF

LSTF since it was previously shown that this heuristic performs the best for this scenario. For these experiments, P_{max} was set to the higher level of 50 to evaluate LSTF under the worst case scenario since it was previously shown that the performance of the heuristics worsen as P_{max} increases. The results of the experiments are shown on Table 3. Table 3 lists the number of times an optimal solution was found out of the 30 runs, the average and the maximum number of nodes evaluated in the branch-and-bound procedure to find an optimal solution, the number of times heuristic LSTF gave the optimal solution out of the 30 runs, and the average departure from optimality. Although the number of feasible solutions increases as n increases, the results showed that the branch-and-bound procedure is faster on average as n increases for a fixed number of machines because the proposed lower and upper bounds provide a tighter bound as n increases due to reduced machine idle time. The performance of LSTF on average does not worsen as n increases.

6. CONCLUSIONS

This paper develops a branch-and-bound enumerative procedure that optimally solves the problem of scheduling n identical jobs with unequal ready times on m parallel uniform machines to minimize the maximum lateness. The branch-and-bound procedure searches over the assignment of jobs to machines. Given the assignment, each machine's job sequence is determined using Simon's [11] single-machine sequencing algorithm. The number of nodes evaluated in the branch-and-bound procedure increases as the number of parallel machines increases. However, as the number of jobs increases for a fixed number of machines the problem becomes easier to solve optimally because the presented lower and upper bounds give tighter bounds due to reduced machine idle time. For a large number of parallel machines, heuristics may be necessary to find approximate optimal solutions.

Table 4 summarizes our findings from the experimental analysis. The purpose of analyzing the heuristics is to identify which ones provide the closest approximation to the optimal solution under different ranges of the ready times, processing times, and due dates. The worst performing heuristic is FAMR. The better performing heuristics are the ones with a look ahead capability on the completion time of the machines. The forward sequencing procedure ECTS performed the best when the variability in the ready times is greater than the variability in the due dates while the backward sequencing procedure LSTF performed the best when the variability on the due dates is larger.

REFERENCES

1. Cheng, T. C. E. and Sin, C. C. S. A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research,* **47,** 1990, 271–292.
2. Dessouky, M. I. and Larson, R. E. Symmetry and optimality properties of the single machine problem. *AIIE Transactions,* **10,** 1978, 170–175.
3. Dessouky, M. I., Lageweg, B. J., Lenstra, J. K. and van de Velde, S. L. Scheduling identical jobs on uniform parallel machines. *Statistica Neerlandica,* **44,** 1990, 115–123.
4. Dessouky, M. M., Dessouky, M. I. and Verma, S. K., *Flowshop Scheduling with Identical Jobs and Uniform Parallel Machines*. Technical Report #1995-01. University of Southern California, Los Angeles, CA, 1995. *European Journal of Operational Research* (to be published).
5. Graham, R. L., Lawler, E. L., Lenstra, J. K. and Rinnooy Kan, A. H. G. Optimization and approximation in determining sequencing and scheduling: A survey. *Annals of Discrete Mathematics,* **5,** 1979, 287–326.
6. Guinet, A. Scheduling independent jobs on uniform parallel machines to minimize tardiness criteria. *Journal of Intelligent Manufacturing,* **6,** 1995, 95–103.

7. Lageweg, B. J., Lenstra, J. K. and Rinnooy Kan, A. H. G. Minimizing maximum lateness on one machine: Computational experience and some applications. *Statistical Neerlandica,* **39,** 1976, 25–41.
8. Larson, R. E. and Dessouky, M. I. Heuristic procedures for the single machine problem to minimize maximum lateness. *AIIE Transactions,* **10,** 1978, 176–183.
9. Lenstra, J. K., *Sequencing by Enumerative Methods. Mathematical Centre Tracts*, No. 69. Mathematics Centrum, Amsterdam, The Netherlands, 1977.
10. McMahon, G. and Florian, M. On scheduling with ready times and due dates to minimize maximum lateness. *Operations Research,* **23,** 1975, 475–482.
11. Simons, B., A fast algorithm for single processor scheduling. *Proceedings of the Nineteenth Annual Symposium on Foundations of Computer Science*. 1978, pp. 246–252.
12. Suresh, V. and Chaudhuri, D. Minimizing maximum tardiness for unrelated parallel machines. *International Journal of Production Economics,* **34,** 1994, 223–229.
13. Suresh, V. and Chaudhuri, D. Bicriteria scheduling problem for unrelated parallel machines. *Computers and Industrial Engineering,* **30,** 1996, 77–82.