McMaster University

MICHAEL G. DEGROOTE SCHOOL OF BUSINESS

RESEARCH AND WORKING PAPER SERIES

A FAST ALGORITHM TO MINIMIZE MAXIMUM LATENESS FOR THE TWO-MACHINE FLOW-SHOP PROBLEM WITH RELEASE TIMES

By

Jinliang Cheng, George Steiner, and **Paul Stephenson**

Michael G. DeGroote School of Business
McMaster University
Hamilton, Ontario

Working Paper # 439

June, 1999

A FAST ALGORITHM TO MINIMIZE MAXIMUM LATENESS FOR THE TWO-MACHINE FLOW-SHOP PROBLEM WITH RELEASE TIMES

By

Jinliang Cheng, George Steiner, and **Paul Stephenson**

Michael G. DeGroote School of Business
McMaster University
Hamilton, Ontario

Working Paper # 439

June, 1999

A Fast Algorithm to Minimize Maximum Lateness for the Two-Machine Flow-Shop Problem with Release Times*

Jinliang Cheng, George Steiner† and Paul Stephenson
Management Science and Information Systems Area
Michael G. DeGroote School of Business
McMaster University, Hamilton, Ontario, Canada

Abstract

We consider the two-machine flow-shop problem with release times where the objective is to minimize the maximum lateness. We derive a new dominance order and incorporate it into an efficient branch and bound algorithm which uses an adaptive branching scheme together with new fuzzy dominance properties for scheduling and searching. The algorithm performed very well. It solved within a few seconds more than 97% of the test problems with up to 200 jobs in a large-scale computational experiment. For the unsolved problems, the average gap between the best solution found and the optimum was less than 0.5%.

Keywords: Scheduling, permutation flow shop, lateness, release times, dominance orders, algorithm.

*This research was supported in part by the Natural Sciences and Engineering Research Council of Canada, under Grant No. OGP0001798.

†Corresponding author. e-mail: steiner@mcmaster.ca

1 Introduction

We consider the two-machine flow-shop problem with release times where the objective is to minimize the maximum lateness for permutation schedules, denoted by $F2\,/r_j, perm/\,L_{\max}$. (We use the standard notation to describe scheduling problems, and refer the reader to [7] and [10] for any terminology not defined here.) This is a well-known strongly NP-hard scheduling problem [8]. We use the delivery-time formulation of the problem, which transforms it into a makespan minimization problem in a four-machine flow shop, where the first and last machines are non-bottleneck machines. We denote this problem by $F2\,/r_j, perm/\,L'_{\max}$. Much of the underlying interest in the problem $F2\,/r_j, perm/\,L'_{\max}$ stems from the fact that it arises naturally in the solution of more complicated scheduling problems, such as the problem $Fm//C_{\max}$ [6]. The complexity result above suggests that in order to optimally solve the problem $F2\,/r_j, perm/\,L'_{\max}$, it is necessary to resort to some efficient enumerative technique, such as branch and bound.

Branch and bound algorithms are characterized by various components: the branching scheme employed, the different types of upper and lower bounds used, and other features. Grabowski [2] and Grabowski et al. [3] presented branch and bound algorithms for the problems $F2\,/r_j, perm/\,L'_{\max}$ and $Fm\,/r_j, perm/\,L'_{\max}$, respectively. These algorithms used a branching scheme that exploited certain dominance properties of a critical path. Tadei et al. [12] tried this type of branching scheme for the problem $F2\,/r_j/\,C_{\max}$, but found it to be less effective than their traditional n-ary branching scheme, that fixes jobs only at the beginning of the schedule. Potts [11] considered an adaptive branching scheme that fixes jobs at both ends of the schedule, and found this branching scheme to be quite effective for the problem $Fm//C_{\max}$. Dominance rules are another common feature of branch and bound algorithms, they are typically used to eliminate nodes (before their bounds are calculated) in order to reduce computation time and storage requirements. Cheng et al. [1] apply approximate dominance rules, in the context of fuzzy inference, for scheduling and searching in flow-shop problems, such as $F3//C_{\max}$ and $Fm\,/perm/\,C_{\max}$. The authors were able to find a nearly optimal initial schedule by repeatedly applying the fuzzy dominance rule to schedule the jobs. This fuzzy schedule also proved very useful for tie-breaking purposes, to decide which way to branch among several nodes with the same lower bound.

In this paper, we consider a new branch and bound algorithm for solving the problem $F2\,/r_j, perm/\,L'_{\max}$, with the following main features. We use an adaptive n-ary branching rule, a variant of the one used by Potts. We derive a new dominance order on the job set, using the new proof technique of subset-restricted interchange, that employs a new interchange operator which we introduce. In addition, we make use of fuzzy dominance properties for initial scheduling and tie-breaking. We also incorporate a simple decomposition procedure that reduces the problem size by fixing

jobs at the beginning of the schedule. We use six very quickly computable lower bounds at each node of the tree. The algorithm represents an effective tool for solving large instances of the strongly NP-hard problem $F2\,/r_j, perm/\,L'_{\max}$. In a large-scale computational experiment, the algorithm has solved, in a matter of a few seconds, 4,384 of the 4,500 randomly generated test problems with up to 200 jobs. Even for the unsolved problems, the best solution found by the algorithm was on average within less than 0.5% of the optimal value.

The rest of the paper is organized as follows. In the next section, we introduce the preliminary definitions and notation for sequencing problems and partial orders. In section 3, we derive several new dominance results for the problem $F2\,/r_j, perm/\,L'_{\max}$. In section 4, we present the details of our branch and bound algorithm, a pseudocode for it is found in the Appendix. In section 5, we discuss the results of a large-scale computational experiment. In the last section, we make our concluding remarks.

2 Preliminary definitions and notation

2.1 Sequencing notation

We call a scheduling problem a *sequencing problem* and its objective a *sequencing function* if any schedule can be completely specified by a single sequence in which the jobs are performed on every machine. As customary, we will restrict our attention to such permutation schedules, which allows us to treat the problem $F2\,/r_j/\,L_{\max}$ as a sequencing problem. Let $J = \{1, 2, \ldots, n\}$ be the set of jobs to be sequenced. Jobs are characterized by a list of parameters, each job j possesses a release time r_j, a due date d_j, and processing times a_j and b_j on machines 1 and 2, respectively. A sequence s on J is a function from $\{1, 2, \ldots, n\}$ to J represented by the n-tuple $(s(1), s(2), \ldots, s(n))$, where $s(i)$ is the i^{th} job in sequence s. The completion time of job $s(i)$ on machines 1 and 2 will be denoted by $C^1_{s(i)}$ and $C^2_{s(i)}$, respectively, and the lateness of job $s(i)$ is defined to be $L_{s(i)} = C^2_{s(i)} - d_{s(i)}$. We will consider the problem in its equivalent delivery-time form, $F2\,/r_j, perm/\,L'_{\max}$, where $q_j = T - d_j$ is the delivery time for job j and T is a constant with $T \geq \max\{d_j \,|j \in J\}$. If we define $L'_j = C^2_j + q_j$, then $L'_j = C^2_j + T - d_j = L_j + T$, and we see that the two objectives $L'_{\max}$ and $L_{\max}$ are equivalent. The delivery-time representation has the important advantage that this problem is completely symmetrical and equivalent to its *reverse problem* in which each job j has 'release time' q_j, and must be processed first on machine 2 for b_j time, followed by processing on machine 1 for a_j and a 'delivery time' of r_j. For our problem, the dominance order will be a partial order defined by the parameters of the jobs. Thus we introduce certain definitions for partially ordered sets (posets).

2.2 Partial orders

By a partially ordered set we mean a pair $P = (X, \leq_P)$ consisting of a set X together with a binary relation $\leq_P$ on $X \times X$ which is reflexive, antisymmetric, and transitive. For $u, v \in X$, $u \leq_P v$ is interpreted as u is less than or equal to v in P. Similarly, $u <_P v$ means that $u \leq_P v$ and $u \neq v$. The usual symbols $\leq$ and $<$ will be reserved for relations between real numbers. An element v is *minimal* in P if there is no u with $u <_P v$. Similarly, u is *maximal* if there is no v such that $u <_P v$. A partial order $P = (X, \leq_P)$ is a *linear order* (or *complete order*) if for every pair $(u, v) \in X \times X$ either $u \leq_P v$ or $v \leq_P u$. Given a pair of partial orders $P = (X, \leq_P)$ and $Q = (X, \leq_Q)$ on the same set X, we call Q an *extension* of P (P a *suborder* of Q) if $u \leq_P v$ implies $u \leq_Q v$ for all $u, v \in X$. A partial order $Q = (X, \leq_Q)$ is a *linear extension* of a partial order $P = (X, \leq_P)$, if Q is a linear order that extends P. Given two partial orders $P_1 = (X, \leq_{P_1})$ and $P_2 = (X, \leq_{P_2})$, we can define the partial order $P_1 \cap P_2 = (X, \leq_{P_1 \cap P_2})$, the *intersection* of P_1 and P_2, where $u \leq_{P_1 \cap P_2} v$ if and only if $u \leq_{P_1} v$ and $u \leq_{P_2} v$ for all $u, v \in X$. A partial order P on the job set of a sequencing problem is called a *dominance order* if there is an optimal sequence that is a linear extension of P.

3 Dominance results

In this section, we present dominance results for the problem $F2/r_i, perm/L'_{\max}$. First we derive a new dominance order $\prec$, using *subset-restricted interchange,* a new technique that incorporates certain restrictions on the subset of intermediate jobs for a new interchange operator which we introduce. Secondly, we consider a fuzzy approximation of dominance that is used in both initial scheduling and searching.

3.1 Subset-restricted interchange

We follow Monma [9] in defining our interchange operators. Let s_1 be a sequence with job m preceding job k. In general, s_1 is of the form $s_1 = (XmYkZ)$, where X, Y and Z are subsequences of J, and let U and V be disjoint subsequences partitioning Y. Three types of interchanges of jobs k and m that leave k preceding m in the resulting sequence s_2, are the following:

1. *Backward Insertion*(BI): $s_2 = (XYkmZ)$
2. *Forward Insertion*(FI): $s_2 = (XkmYZ)$
3. *Shuffle Interchange(SI):* $s_2 = (XUkmVZ)$.

To define more precisely the new SI operator, consider again sequence $s_1 = (XmYkZ)$ and a partition of Y into $Y = U \cup V$, where, for example, $U = (u_1u_2u_3)$ and $V = (v_1v_2v_3v_4)$, and Y is of the form $Y = (u_1u_2v_1v_2u_3v_3v_4)$. Then SI applied to sequence

s_1, for this particular choice of U and V, gives sequence $s_2 = (Xu_1u_2u_3kmv_1v_2v_3v_4Z)$. We call it shuffle interchange because the interchange of jobs has the resulting net effect of 'shuffling' sequence Y into subsequences U and V, and placing jobs k and m between them. Notice that SI may change the relative order of some u's and v's after interchange, but it never changes the relative order of two u's or two v's. Thus SI depends on the choice of subsequences U and V. SI generalizes BI and FI: If we let V and U be the sets of jobs in sequences V and U (for the sake of brevity, we do not distinguish between sets and sequences) when $V = \varnothing$ or $U = \varnothing$, then SI reduces to BI or FI, respectively. Further, these interchanges all reduce to adjacent pairwise interchange in the case when $Y = \varnothing$. In summary, the SI operator generalizes all these operators and at the same time allows a unified treatment for them.

Recall that the problem $F2//C_{\max}$ is solved by the *Johnson order*: first order the jobs with $a_j \leq b_j$ in *nondecreasing* a order followed by the jobs with $a_j > b_j$ in *nonincreasing* b order [5]. To emphasize the partitioning of the jobs in the Johnson order we define $J_{a\leq b} = \{j \,|a_j \leq b_j\}$ and $J_{a>b} = \{j \,|a_j > b_j\}$. This ordering of the jobs is an adjacent interchange order for the problem $F2//C_{\max}$, and since it also completely orders every pair of jobs, it is an optimal ordering (i.e., an optimal sequence). An adjacent interchange order for the problem $F2\,/r_j, perm/\,L'_{\max}$ is the *intersection* of the nondecreasing r order, the nonincreasing q order and the Johnson order, as defined formally below. Note that this order is no longer a complete order, rather it is only a partial order.

Definition 1 *Adjacent Interchange Order* $\lessdot$: $k \lessdot m$ *if* $r_k \leq r_m$, $q_k \geq q_m$ *and*
(i) $k, m \in J_{a\leq b}$ *and* $a_k \leq a_m$ *or,*
(ii) $k \in J_{a\leq b}$ *and* $m \in J_{a>b}$ *or,*
(iii) $k, m \in J_{a>b}$ *and* $b_k \geq b_m$.

In general, if an adjacent interchange order is only a partial order (and not a complete order), it need not be a dominance order. This is the case for $\lessdot$ defined above, if we consider the instance of $F2\,/r_j, perm/\,L'_{\max}$ in Example 1. Here we have $3 \lessdot 1$ (since $r_3 = r_1 = 10$, $q_3 = 10 \geq 0 = q_1$, and $3, 1 \in J_{a>b}$ with $b_3 = 25 > 15 = b_1$), however, the unique optimal sequence is $(1,2,3,4)$ with $L'_{\max} = 125$. Thus we see that $\lessdot$ is not a dominance order, since there is no optimal sequence that is a linear extension of $\lessdot$. We will use subset-restricted interchange to find a suborder of $\lessdot$ that is a dominance order.

Example 1 *A 4 job problem to illustrate that* $\lessdot$ *is not necessarily a dominance order.*

j	1	2	3	4
r_j	10	20	10	30
a_j	20	20	30	25
b_j	15	30	25	20
q_j	0	10	10	0

Recall, that the SI operator above interchanges k and m 'around' sequence Y, for some particular choice of U and V. Intuitively, whether or not such an interchange leads to a reduction in cost (for a given sequencing function f and adjacent interchange order $\lessdot$), should depend on the composition of Y, as well as on the choice of U and V. We consider interchanges that are restricted by conditions on Y and define the subset-restricted interchange condition for SI.

Definition 2 *An adjacent interchange order $\lessdot$ together with the collection of subsets $\left\{U_{k\lessdot m} \cup V_{k\lessdot m} \,|k\lessdot m\right\}$ satisfies the Shuffle Interchange Condition for a sequencing function f if*
for all jobs k, m and sequences X, Y, Z there exist a partition U and V into disjoint subsequences of Y such that
$k\lessdot m$, $U \subseteq U_{k\lessdot m}$, and $V \subseteq V_{k\lessdot m}$ imply that $f\,(XUkmVZ) \leq f\,(XmYkZ)$.

Next we examine the structure of $U_{k\lessdot m}$ and $V_{k\lessdot m}$ for the different types of pairs $k\lessdot m$ for the problem $F2\,/r_j, perm/\,L'_{\max}$.

Theorem 1 *If $k \in J_{a\leq b}$ and $a_k \leq a_m$, then $k\lessdot m$ together with the sets $U_{k\lessdot m} = \{u \,|u \in J_{a\leq b}, r_u \leq r_m, a_u \leq a_m\}$ and $V_{k\lessdot m} = \{v \,|q_v \leq q_k\}$ satisfy the Shuffle Interchange Condition.*

Proof. Given a sequence s we construct the directed graph $G(s)$ (the conjunctive graph) to evaluate $L'_{\max}(s)$. Each job $s(j)$ is represented by four nodes with weights $r_{s(j)}$, $a_{s(j)}$, $b_{s(j)}$, and $q_{s(j)}$, respectively. $L'_{\max}(s)$ is the length of the longest 'node-weighted' path from the start-node to the finish node. These paths can be uniquely identified by the triples $(s(i), s(j), s(k))$, for $1 \leq i \leq j \leq k \leq n$, representing the end points of the three horizontal segments on the path (see Figure 1). By definition,

$$L'_{\max}\,(s) = \max_{\substack{(s(i),s(j),s(k)) \\ 1\leq i\leq j\leq k\leq n}} \left(r_{s(i)} + \sum_{l=i}^{j} a_{s(l)} + \sum_{l=j}^{k} b_{s(l)} + q_{s(k)} \right),$$

and we can evaluate $L'_{\max}\,(s)$ as the maximum over all such paths in $G\,(s)$.

Let $s_1 = (XmYkZ)$ be a sequence for pair $k\lessdot m$ with $k \in J_{a\leq b}$ and $a_k \leq a_m$. Let $U \subseteq U_{k\lessdot m}$ and $V \subseteq V_{k\lessdot m}$ be disjoint subsequences of Y, with $Y = U \cup V$. We apply shuffle interchange to s_1, with this U and V, and obtain sequence $s_2 = (XUkmVZ)$. (We use lower case letters x, u, v, and z to refer to arbitrary generic elements of subsequences X, U, V, or Z, respectively.) We can demonstrate that s_2 is not worse than s_1, by exhibiting for every path in $G(s_2)$ a dominating path in $G(s_1)$ (see Figure 2). For example, consider the path (u, k, v) in $G(s_2)$, then the corresponding dominating path in $G(s_1)$ is (m, m, k), and the following inequality states that path (u, k, v) in $G(s_2)$ is not longer than the path (m, m, k) is in $G(s_1)$.

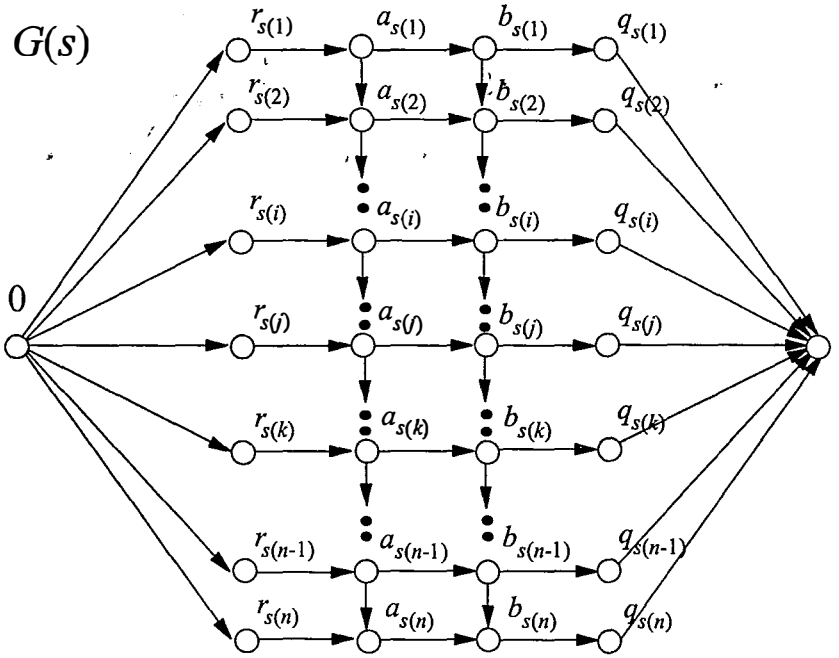

Figure 1: Directed graph $G(s)$ for problem $F2/r_j, perm/ L'_{\max}$.

$$r_u + \sum_{i \in U_{[u,\cdot]}} a_i + a_k + b_k + b_m + \sum_{i \in V_{[\cdot,v]}} b_i + q_v \leq r_m + a_m + b_m + \sum_{i \in U} b_i + \sum_{i \in V} b_i + b_k + q_k,$$

The inequality holds, since $r_u \leq r_m$, $U \subseteq J_{a \leq b}$, $a_k \leq a_m$, and $q_k \geq q_v$. (Note that we use $U_{[u,\cdot]}$ to represent the subsequence of U starting with u and ending with the last job in U; and $V_{[\cdot,v]}$ to represent the subsequence of V from its beginning to job v.)

To complete the proof, we present Table 4 in the Appendix, which gives the dominating paths in $G(s_1)$ for each path in $G(s_2)$. The last column contains the argument why each path is a dominating path, i.e., why the corresponding inequality holds. ■

Theorem 2 *If $k \in J_{a \leq b}$ and $m \in J_{a>b}$, then $k \lessdot m$ together with the sets $U_{k \lessdot m} = \{u \,|\, u \in J_{a \leq b}, r_u \leq r_m\}$ and $V_{k \lessdot m} = \{v \,|\, v \in J_{a>b}, q_v \leq q_k\}$ satisfies the Shuffle Interchange Condition.*

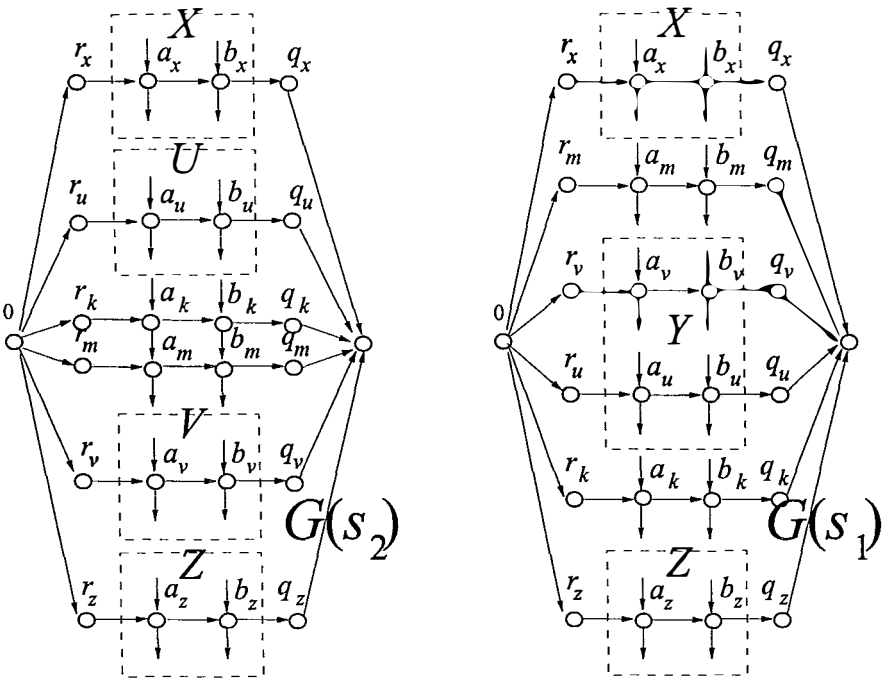

Figure 2: Directed graphs $G(s_2)$ and $G(s_1)$ for SI.

Proof. Similarly, Table 5 in the Appendix contains the appropriate dominating paths. ■

Theorem 3 *If $m \in J_{a>b}$ and $b_k \geq b_m$, then $k \lessdot m$ together with the sets $U_{k \lessdot m} = \{u \,|r_u \leq r_m\}$ and $V_{k \lessdot m} = \{v \,|v \in J_{a>b}, q_v \leq q_k, b_v \leq b_k\}$ satisfies the Shuffle Interchange Condition.*

Proof. Symmetric to Theorem 1, i.e., Theorem 3 is equivalent to Theorem 1 applied to the reverse problem ■

Theorems 1 to 3 represent new dominance conditions applicable to *sequences* of jobs. Their application, however, would require elaborate data structures and complex procedures for checking that the intermediate sequences Y meet the conditions of the theorems. In the following section, we show that there is a much better way to exploit these dominance results, by deriving from them a new dominance order between *pairs of jobs*, irrespective of where they are in a sequence.

3.2 New dominance order for $F2/r_j, perm/L'_{\max}$

We use subset-restricted interchange to derive a new dominance order on the jobs, denoted $\prec$, for the problem $F2/r_i, perm/L'_{\max}$. We define the following sets of jobs for every pair $k \lessdot m$. (Note that a pair $k \lessdot m$ may satisfy more than one of the following three conditions, so more than one may be applicable to a pair.)

1. If $a_k \leq b_k$ and $a_k \leq a_m$, then let $U^1_{k\lessdot m} = \{u \,|u \in J_{a\leq b}, r_u \leq r_m, a_u \leq a_m, q_u > q_k\}$
   $V^1_{k\lessdot m} = \{v \,|q_v \leq q_k\}$.
2. If $a_k \leq b_k$ and $a_m > b_m$, then let $U^2_{k\lessdot m} = \{u \,|u \in J_{a\leq b}, r_u \leq r_m\}$
   $V^2_{k\lessdot m} = \{v \,|v \in J_{a>b}, q_v \leq q_k\}$.
3. If $a_m > b_m$ and $b_k \geq b_m$, then let $U^3_{k\lessdot m} = \{u \,|r_u \leq r_m\}$
   $V^3_{k\lessdot m} = \{v \,|v \in J_{a>b}, q_v \leq q_k, b_v \leq b_k, r_v > r_m\}$.

These are just the sets $U_{k\lessdot m}$ and $V_{k\lessdot m}$ from the previous three theorems with additional conditions in cases 1 and 3, to ensure that $U^i_{k\lessdot m} \cap V^i_{k\lessdot m} = \varnothing$ and maintain that $v_i \not\lessdot u_i$ for $u_i \in U^i_{k\lessdot m}$ and $v_i \in V^i_{k\lessdot m}$. In case 1, $q_u > q_k$ has been added and $r_v > r_m$ has been added in case 3. For a given pair $k \lessdot m$, $k \lessdot m$ is included in the dominance order $\prec$ exactly when any of the applicable sets $U^i_{k\lessdot m} \cup V^i_{k\lessdot m}$ is the *entire* job set. Thus, the dominance order $\prec$ is defined as the suborder of $\lessdot$ consisting of pairs $k \lessdot m$, that can be interchanged around *any* intermediate sequence using the SI operator for the appropriate choice of U and V.

Definition 3 *If $k \lessdot m$ and $U^i_{k\lessdot m} \cup V^i_{k\lessdot m} = J$ for some i applicable to k and m, then $k \prec m$.*

To illustrate the definition of $\prec$, consider again Example 1. The adjacent interchange order $\lessdot$ consists of the pairs $2 \lessdot 4$, $3 \lessdot 4$, and (as we saw previously) $3 \lessdot 1$. For pair $2 \lessdot 4$ both cases 1 and 2 apply, and checking the conditions for these we see that both $U^1_{2\lessdot 4} \cup V^1_{2\lessdot 4}$ and $U^2_{2\lessdot 4} \cup V^2_{2\lessdot 4}$ are equal to J, thus we have that $2 \prec 4$. For pairs $3 \lessdot 4$ and $3 \lessdot 1$ only case 3 applies. We also have $U^3_{3\lessdot 4} \cup V^3_{3\lessdot 4} = J$, thus $3 \prec 4$. However, $U^3_{3\lessdot 1} \cup V^3_{3\lessdot 1} \neq J$ because job $2 \notin V^3_{3\lessdot 1}$ since $2 \in J_{a\leq b}$, and $2 \notin U^3_{3\lessdot 1}$ since $r_2 > r_1$, thus $3 \not\prec 1$. Therefore, we have that the dominance order $\prec$ is the suborder of $\lessdot$ consisting of the pairs $2 \prec 4$ and $3 \prec 4$, and we see that the unique optimal sequence $(1,2,3,4)$ is indeed a linear extension of $\prec$, as we would expect.

We define the following suborders of $\prec$ for the different types of pairs in $\lessdot$

$$\begin{aligned} \prec_1 &= \prec \bigcap (J_{a\leq b} \times J_{a\leq b}) \\ \prec_2 &= \prec \bigcap (J_{a\leq b} \times J_{a>b}) \\ \prec_3 &= \prec \bigcap (J_{a>b} \times J_{a>b}), \end{aligned}$$

where $A \times B$ represents all pairs with first element from A and second element from B. Note that for the pairs in $\prec_1$ and $\prec_3$, cases 1 and 3 must hold, respectively.

Whereas, for the pairs in $\prec_2$ at least one of cases 1-3 must hold. These suborders will be used first to demonstrate that $\prec$ is indeed a partial order, and then to prove that $\prec$ is a dominance order for problem $F2/r_i, perm/ L'_{\max}$.

Lemma 1 *Suborder $\prec$ is a partial order.*

Proof. Since $\prec$ is a suborder of $\leqslant$, it only remains to show that $\prec$ is transitive, i.e., we want to show that for any jobs k, l, and m if $k \prec l$ and $l \prec m$, then $k \prec m$. First we consider the case where all of the jobs k, l, and m are in $J_{a\leq b}$. In this case, we have that $k \prec_1 l$ and $l \prec_1 m$, and we want to demonstrate that $k \prec_1 m$. By Definition 3 $k \prec_1 m$ if $U^1_{k\leqslant m} \cup V^1_{k\leqslant m} = J$. Since $k \leqslant l \leqslant m$ and k, l, and m are in $J_{a\leq b}$, we have that $r_k \leq r_l \leq r_m$ and $a_k \leq a_l \leq a_m$. These imply that $U^1_{k\leqslant m} \supseteq U^1_{k\leqslant l}$, and since $V^1_{k\leqslant m} = V^1_{k\leqslant l}$, we have that $U^1_{k\leqslant m} \cup V^1_{k\leqslant m}$ contains $U^1_{k\leqslant l} \cup V^1_{k\leqslant l}$. However, $k \prec_1 l$ implies that $U^1_{k\leqslant l} \cup V^1_{k\leqslant l} = J$, which by the above observation and Definition 3 implies that $k \prec_1 m$. Next we examine the case where k, $l \in J_{a\leq b}$ and $m \in J_{a>b}$, in which $k \prec_1 l$ and $l \prec_2 m$. We consider separately the different cases of Definition 3 that may apply for pair $l \prec_2 m$. If case 1 applies for $l \prec_2 m$, then $(a_k \leq) a_l \leq a_m$ and the previous argument carries over exactly as above. For case 2, we assume that $U^2_{l\leqslant m} \cup V^2_{l\leqslant m} = J$. Since $k \leqslant l \leqslant m$, we have that $r_k \leq r_l \leq r_m$ and $q_k \geq q_l \geq q_m$. These imply, however, that $U^2_{k\leqslant m} = U^2_{l\leqslant m}$ and $V^2_{k\leqslant m} \supseteq V^2_{l\leqslant m}$, so we also have that $U^2_{k\leqslant m} \cup V^2_{k\leqslant m} = J$. Thus, by Definition 3 we have that $k \prec_2 m$. Finally, if case 3 applies to $l \prec_2 m$, then $b_l \geq b_m$ and $U^3_{l\leqslant m} \cup V^3_{l\leqslant m} = J$. We notice that $(U^3_{l\leqslant m} \cup V^3_{l\leqslant m}) \cap J_{a\leq b}$ equals $U^3_{l\leqslant m} \cap J_{a\leq b}$, which equals $U^2_{k\leqslant m}$. Since $V^3_{l\leqslant m} \cap J_{a\leq b} = \varnothing$ and $U^3_{l\leqslant m} \cup V^3_{l\leqslant m} = J$, this implies that $U^2_{k\leqslant m}$ contains all of the jobs in $J_{a\leq b}$. We also have that $U^1_{k\leqslant l} \cup V^1_{k\leqslant l} = J$ since $k \prec_1 l$. Similarly, $(U^1_{k\leqslant l} \cup V^1_{k\leqslant l}) \cap J_{a>b}$ equals $V^1_{k\leqslant l} \cap J_{a>b}$, which equals $V^2_{k\leqslant m}$. Since $U^1_{k\leqslant l} \cap J_{a>b} = \varnothing$ and $U^1_{k\leqslant l} \cup V^1_{k\leqslant l} = J$, this implies that $V^2_{k\leqslant m}$ contains all of the jobs in $J_{a>b}$. Combining these two results we have that $U^2_{k\leqslant m} \cup V^2_{k\leqslant m}$ contains all of the jobs in J, from which it follows by Definition 3 that $k \prec_2 m$. The remaining cases, where k, l, and m are in $J_{a>b}$ $k \prec_3 l$ and $l \prec_3 m$, or $k \in J_{a\leq b}$ and l, $m \in J_{a>b}$ with $k \prec_2 l$ and $l \prec_3 m$, follow by symmetry. Thus we see that $\prec$ is a transitive order indeed. ■

Theorem 4 *Partial order $\prec$ is a dominance order for problem $F2/r_i, perm/ L'_{\max}$.*

Proof. Let s be an optimal sequence. We find an optimal linear extension of $\prec$ by repeatedly applying the SI operator to sequence s without increasing the length of the longest path. We demonstrate the comparabilities in the order $\prec_3$, $\prec_1$ and $\prec_2$, respectively.

Assume that s is not a linear extension of $\prec_3$. Let $m \in J_{a>b}$ be the last job in s with some job $j \in J_{a>b}$ with $j \prec_3 m$, such that j is after m in s. Let k be the

first such job j in s, then s is of the form $s = (XmYkZ)$. Since $k \prec m$, k and m can be interchanged around any intermediate sequence using SI, in particular around subsequence Y. Since $k \prec_3 m$, $U_{k \leq m} \cup V_{k \leq m} = U^3_{k \leq m} \cup V^3_{k \leq m} = J$. Applying SI to s with $U = U^3_{k \leq m} \cap Y$ and $V = V^3_{k \leq m} \cap Y$, we obtain sequence $(XUkmVZ)$. This sequence now orders k and m properly, and it does not introduce any new violations of $\prec_3$: To see this, consider any jobs $u \in U$ and $v \in V$. Since $u \in U^3_{k \leq m}$ and $v \in V^3_{k \leq m}$, $v \not\leq u$, so $v \not\prec u$, which implies that $v \not\prec_3 u$. Also, $m \not\prec_3 u$ follows by the choice of m, and $v \not\prec_3 k$ follows since $r_v > r_m \geq r_k$. By repeatedly applying the above procedure until there is no such job m, we obtain an optimal sequence s' which is a linear extension of $\prec_3$.

For $\prec_1$, we proceed in exactly the same way. Assume that sequence s' is not a linear extension of $\prec_1$. Let $m \in J_{a \leq b}$ be the last job in s' with some job $j \in J_{a \leq b}$ with $j \prec_1 m$, such that j is after m in s. Let k be the first such job j in s', then s' is of the form $s' = (XmYkZ)$. Applying SI to s' with $U = U^1_{k \leq m} \cap Y$ and $V = V^1_{k \leq m} \cap Y$, we obtain sequence $(XUkmVZ)$. This sequence now orders k and m properly, and it does not introduce any new violations of $\prec_1$ or $\prec_3$: For $\prec_1$, once again consider any jobs $u \in U$ and $v \in V$. Since, $v \not\leq u$, then $v \not\prec_1 u$. Also $m \not\prec_1 u$ since $q_u > q_k \geq q_m$ from the definition of $U^1_{k \leq m}$, and $v \not\prec_1 k$ follows by the choice of k. To see that this doesn't introduce any new violations of $\prec_3$ either, note that all jobs in $Y \cap J_{a>b}$ must be in V, and for these jobs their relative order is unchanged. Repeatedly applying SI we obtain an optimal sequence s'' that is a linear extension of $\prec_1$ and $\prec_3$.

Finally, assume that sequence s'' is not a linear extension of $\prec_2$. Let $m \in J_{a>b}$ be the last job in s with some job $j \in J_{a \leq b}$ with $j \prec_2 m$, such that j is after m in s. Let k be the first such job j in s'', then s'' is of the form $s'' = (XmYkZ)$. Since $k \prec_2 m$, we know for some i that $J = U^i_{k \leq m} \cup V^i_{k \leq m}$, applying SI for this choice of i with $U = U^i_{k \leq m} \cap Y$ and $V = V^i_{k \leq m} \cap Y$, we obtain sequence $(XUkmVZ)$. This sequence now orders k and m properly, and it does not introduce any new violations of $\prec$: As above, for any jobs $u \in U$ and $v \in V$ we have that $v \not\leq u$ which implies that $v \not\prec u$. In addition, we have $m \not\prec_3 u$ ($\Rightarrow m \not\prec u$) and $v \not\prec_1 k$ ($\Rightarrow v \not\prec k$) as above. Thus, we see that this interchange does not introduce any new violations of $\prec$. Continuing to apply SI until there is no such job m, we obtain an optimal sequence s^* that is also a linear extension of $\prec$. ■

Remark 1 *Since SI is a generalization of the BI and FI operators, the theorem and its proof also apply to these when $U^i_{k \leq m} = J$ or $V^i_{k \leq m} = J$ for some applicable i, i.e., in such special situations the dominance order $\prec$ could be derived using only one of these operators.*

3.3 Fuzzy approximation of dominance

Dominance rules on sequences are usually used to specify whether a node can be eliminated before its lower bound is calculated in a branch and bound algorithm. However, they also can be used heuristically in finding a good initial solution, or directing the search in case of ties [1].

Theorem 5 *Consider a partial sequence σ for the problem $F2/r_j/L'_{\max}$. If there are unsequenced jobs i and j such that*

$$C^l(\sigma ij) \leq C^l(\sigma ji), 1 \leq l \leq 2 \quad (1)$$

$$L'_{\max}(\sigma ij) \leq L'_{\max}(\sigma ji), \quad (2)$$

then the partial sequence σij dominates σji, i.e., any completion of σij has an $L'_{\max}$ which is not larger than the $L'_{\max}$ for the same completion of σji.

Proof. Consider an arbitrary completion sequence w for the remaining unsequenced jobs. If Eq.(1) holds, then on both machines every job in w will be able to start no later in the sequence (σijw) than in the sequence (σjiw). This implies that no job in w can have a larger L' value in the sequence (σijw) than in the sequence (σjiw). Combining this with Eq.(2) completes the proof. ■

If there exists a job i for which Eq.(1) and Eq.(2) hold for *all* unsequenced jobs j, then sequence σ has an optimal completion with job i sequenced next. Such a job i very rarely exists, however. This suggests that if they only *approximately* hold, i.e., they hold for job i with *almost all* unsequenced jobs j, then job i may precede another job j in an optimal completion of sequence σ with high probability. We measure the closeness of this approximation by a fuzzy membership function. We use this fuzzy inference to find a good initial sequence, and to break ties between nodes with the same lower bound when branching. These techniques proved very useful for the problem $Fm//C_{\max}$ [1]. Let

$$\begin{aligned} D^l(\sigma ij) &= C^l(\sigma ij) - C^l(\sigma ji), \quad 1 \leq l \leq 2 \\ D^3(\sigma ij) &= L'_{\max}(\sigma ij) - L'_{\max}(\sigma ji). \end{aligned}$$

The *fuzzy membership function* that represents the likelihood that job i precedes job j in an optimal completion of σ is given by

$$\mu_\sigma(i,j) = 0.5 - \frac{D(\sigma ij)}{2D_{max}(\sigma)},$$

where $D(\sigma ij) = \sum_{l=1}^{3} \alpha_l D^l(\sigma ij)$, $D_{max}(\sigma) = \max_{i,j} |D(\sigma ij)|$ and $\alpha_1, \alpha_2, \alpha_3$ ($0 \leq \alpha_1, \alpha_2, \alpha_3 \leq 1$ and $\sum_{l=1}^{3} \alpha_l = 1$) are real numbers. (Note that this definition ensures that $0 \leq \mu_\sigma(i,j) \leq 1$.) Let S be the set of jobs not scheduled in σ. Then,

the likelihood of job $i \in S$ dominating the remaining jobs after partial sequence σ is measured by

$$\mu^*_\sigma(i) = \min_{j \in S \setminus \{i\}} \mu_\sigma(i,j),$$

and job i^* satisfying

$$\mu^*_\sigma(i^*) = \max_{i \in S} \mu^*_\sigma(i)$$

is identified as the job that immediately follows σ.

The rule determining i^* in this way is referred to as the *fuzzy rule* and the schedule obtained by repeatedly applying the fuzzy rule is referred to as the *fuzzy schedule* (or *fuzzy sequence)*. To obtain our fuzzy sequence we apply the fuzzy rule with $\alpha_l = 1/3$ for $l = 1, 2, 3$. This requires $O(n^2)$ time.

4 Branch and bound

In this section, we outline the basic components of our proposed branch and bound algorithm to solve the problem $F2/r_j, perm/L'_{\max}$. A pseudocode for it is given in the Appendix.

4.1 Branching rule

In order to exploit the symmetry of the problem, we consider a variant of Potts' adaptive branching rule that fixes jobs at both ends of the schedule [11]. More precisely, each node of the search tree is represented by a pair (σ_1, σ_2), where σ_1 and σ_2 are the initial and final partial sequences, respectively. Let S_i denote the set of jobs in σ_i for $i = 1, 2$, then the set of unfixed jobs is $S = J \setminus (S_1 \cup S_2)$. We use $\prec|_S$ to refer to the restriction of $\prec$ to the set S. An immediate successor of (σ_1, σ_2) in the tree is either of the form $(\sigma_1 i, \sigma_2)$ for a *type 1* branching; or $(\sigma_1, i\sigma_2)$ for a *type 2* branching, where i is a minimal or maximal job in $\prec|_S$, respectively. The types of the branchings are all the same within a level of the tree. The type for a given level k is fixed on the very first visit to level k according to the following rule: branch in the direction of the fewest number of ties at the minimum lower bound. Let n_1 and n_2 be the number of ties at the minimum lower bound for potential type 1 and type 2 branchings, at level k. If $n_1 < n_2$ the next branching is of type 1, while if $n_2 < n_1$ then the branching is of type 2. If $n_1 = n_2$ then the branching is the same type as at the previous level.

The search strategy is to branch to the newest active node with the smallest lower bound, breaking ties by the appropriate fuzzy rule. For type 1 branchings we use fuzzy sequence 1 to break ties, this is the sequence obtained by repeatedly applying the fuzzy rule to the forward problem. For type 2 branchings we use fuzzy sequence 2 to break ties, this is the analogous sequence for the reverse problem.

4.2 Bounds

Upper bounds are calculated only for the first n nodes, afterward the upper bound is evaluated only at leaf nodes. For the first n nodes we calculate four separate upper bounds. The first two upper bounds are obtained by sequencing the unfixed jobs in the fuzzy sequence for the forward and reverse problems, respectively. Each of these requires only $O(n)$ time working from the previously established fuzzy sequences. The two remaining upper bounds are obtained by sequencing the unfixed jobs in ready $q+b$ order (ordering the ready jobs in nondecreasing $q+b$ order) for the forward problem, and ready $r+a$ order for the reverse problem. Each of these upper bounds requires $O(n \log n)$ time to compute. There are no unfixed jobs at leaf nodes, and the upper bound is just the length of the sequence obtained by concatenating σ_1 and σ_2.

Since the branch and bound tree may require the computation of lower bounds for a potentially very large number of nodes, it is important that we use lower bounds which require only $O(n)$ time per bound. (We have also experimented with some potentially better lower bounds, requiring $O(n \log n)$ time, but they have noticeably slowed down the algorithm without any substantial increase in the number of problems solved.) We consider six lower bounds for each node (σ_1, σ_2). We calculate lower bounds on the lengths of different paths for the unfixed jobs S, and we combine these in various ways with the actual lengths of the fixed sequences σ_1 and σ_2. For σ_1 we look at the forward problem and we let $C^1(\sigma_1)$ and $C^2(\sigma_1)$ be the completion times on machines 1 and 2, respectively. For S we consider the forward problem, and assume that these jobs cannot start on machines 1 and 2 before $C^1(\sigma_1)$ and $C^2(\sigma_1)$, respectively. Ignoring release times for the jobs in S, let $L_1(S)$ be the completion time on machine 2 of the Johnson sequence on S. Let $L_2(S)$ be the completion time on machine 1 of the jobs in S sequenced in nondecreasing r order, that is, $L_2(S)$ is the earliest time at which the jobs in S can be completed on machine 1. Similarly, if we relax the capacity constraint on machine 1 and sequence the jobs in nondecreasing $r+a$ order, then the length of this schedule on machine 2, $L_3(S)$, is a lower bound on when the jobs in S can complete on machine 2. For σ_2 we define the completion times $C^1(\sigma_2)$ and $C^2(\sigma_2)$ on machines 1 and 2, respectively, for the reverse problem. Once again, we assume that the jobs in S cannot start before these times. We let $L_4(S)$ be the completion time on machine 1 of the reverse Johnson sequence. As for the forward problem, the earliest that the jobs in S can complete on machine 2 in the reverse problem is to sequence them in nondecreasing q order, we denote the length of this schedule on machine 2 by $L_5(S)$. Relaxing the capacity constraint on machine 2, and sequencing the jobs on machine 1 in nondecreasing $q+b$ order, the length of this schedule denoted by $L_6(S)$, is a lower bound on when the jobs in S finish on machine 1 in the reverse problem. We compute the following for the node (σ_1, σ_2):

$$LB_1 = L_1(S) + C^2(\sigma_2)$$

$$
\begin{aligned}
LB_2 &= L_2(S) + C^1(\sigma_2) \\
LB_3 &= L_3(S) + C^2(\sigma_2) \\
LB_4 &= C^1(\sigma_1) + L_4(S) \\
LB_5 &= C^2(\sigma_1) + L_5(S) \\
LB_6 &= C^1(\sigma_1) + L_6(S).
\end{aligned}
$$

Finally, the lower bound is the maximum of the above six lower bounds and the lengths of the fixed sequences σ_1 and σ_2 (in the forward and reverse problems), $L'_{\max}(\sigma_1)$ and $L'_{\max}(\sigma_2)$, respectively.

At the root node we evaluate two additional lower bounds, these are single machine preemptive bounds obtained by relaxing the capacity constraints on each of machines 1 and 2 in the forward problem, respectively. It is well known that these are solved by the preemptive, ready-Jackson rule, which requires $O(n \log n)$ time to compute.

4.3 Decomposition and dominance

We find a starting sequence σ_1 by applying the following simple decomposition procedure, which is a generalization of the one used in [12] for the problem $F2/r_i/C_{max}$. Given a sequence s, then partial sequence $s^{k-1} = (s(1), \ldots, s(k-1))$ is an optimal initial sequence if there exists a $k \in [2, n]$ such that $\min_{k \le i \le n} r_{s(i)} \ge C^1_{s(k-1)}$, $\min_{k \le i \le n} [r_{s(i)} + a_{s(i)}] \ge C^2_{s(k-1)}$, and we have $LB(J \backslash (s^{k-1})) \ge L'_{\max}(s^{k-1})$, where $LB(J \backslash (s^{k-1}))$ is computed as follows. We apply the decomposition procedure for the jobs sequenced in nondecreasing r order, and we use the two preemptive single machine bounds mentioned in the last paragraph of the previous section to determine $LB(J \backslash (s^{k-1}))$. These conditions mean that the jobs in the partial sequence s^{k-1} have no effect on the optimal schedule for the remaining jobs. Then the initial partial sequence σ_1 is chosen as s^{k-1} for the largest k value for which all the conditions hold.

After we have fixed σ_1, we determine the dominance order $\prec$ on the remaining jobs in $S = J \backslash S_1$. In addition, it is possible to dynamically update the dominance order $\prec$ at each node (σ_1, σ_2), to see whether fixing jobs in σ_1 and σ_2 can add new comparabilities, thus further reducing the amount of branching.

5 Computational experiment

5.1 Test problems

For each problem with n jobs $4n$ integer data (r_i, a_i, b_i, q_i) were generated. The processing times a_i and b_i were both uniformly distributed between $[1, 100]$. Release times r_i and delivery times q_i were uniformly distributed in the range $[0, n \cdot 101 \cdot R]$

and $[0, n \cdot 101 \cdot Q]$, respectively, following the technique used by Hariri and Potts [4]. Different R and Q values were tested for values $Q \leq R$ and $R \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$. For each individual R, Q, and n combination, 50 problems were generated. We used the random number generator of Taillard [13] to generate these problems. This means that all of our test problems are reproducible by running the problem generation procedure from the same seeds. (The seeds used have been saved and are available from the authors on request.)

5.2 Results

The branch and bound algorithm was coded in Sun Pascal 4.2 and run on a Sun Sparc5. Three separate versions of the algorithm were run. Algorithm A_1 has the dominance order $\prec$ 'turned off'. Algorithm A_2 has the dominance order 'turned on', and it dynamically updates $\prec$ at each node. Finally, algorithm A_3 also has the dominance order 'turned on', but $\prec$ is only calculated once at the root node. Each version of the algorithm was run until either it obtained the optimal solution or the number of nodes branched from in the tree reached one million for the problem. In the latter case, the problem was declared unsolved.

Tables 1, 2, and 3 contain the results of the computational experiment for the different R and Q values. For each group of problems we report: the fraction of problems solved (denoted by *solved*); for each of the unsolved problems we calculate the gap between the best solution obtained and the smallest lower bound among the left-over nodes in the tree, and the maximum gap (denoted *max gap*) is the largest of these over all the problems in the group; the total number of nodes in all of the trees (denoted by *total nodes*); the average CPU time for the solved problems in the group (denoted by *avg CPU*); and the total CPU time required for all of the problems in the group (denoted by *total CPU*).

Recall, a problem was 'unsolved' when the number of nodes branched from reached a million. Thus the total number of nodes for a group of 50 problems, contains a million nodes for each unsolved problem. As can be seen, the algorithms typically generated much fewer nodes for the solved problems. In those groups where there were unsolved problems, the solution obtained was always nearly optimal as measured by the maximum gap. The largest such gap was on the order of 3 percent, meaning that, in the worst case, we were within this range of the optimal solution. The average gap was much smaller, less than 0.5 percent.

As the results indicate, all three versions of the new branch and bound algorithm proved very effective in solving the test problems. In total, each of the three solved 4,384 of the 4500 randomly generated test problems with up to 200 jobs. There were differences in the time-performance of the three versions, however. In general, algorithms A_2 and A_3 (with the dominance order) required less time than version A_1 (without the dominance order). Algorithms A_2 and A_3 needed roughly 5 percent

and 15 percent less total CPU time, respectively, than A_1 for the 4500 test problems. For the 4384 solved problems, the savings in total CPU time amounted to 15 percent and 20 percent, respectively, when compared to A_1. Thus, both A_2 and A_3 made the search through the tree faster, but the substantially larger overhead of A_2 (i.e., of updating $\prec$ at every node of the tree) offset a large part of the speed-up. In summary, algorithm A_3 was the most effective method, it was extremely fast on all of the solved problems, never requiring more than a few seconds.

The very extensive computational experiment has led to further insights. Our results suggest that the difficulty in solving a particular problem depends *much more* on the values of R and Q, than it does on the number of jobs. Given this observation, the difficult problems are those with intermediate R values, with the most difficult being the problems with $R = 0.4$. This means that the most difficult problems occur when the maximum range for the release times ($n \cdot 101 \cdot R$) is close to the total expected processing time on the first machine ($n \cdot 50.5$). We have also found that for a given R, the problems with small Q are most difficult. Recall, that the actual size of the problem for the branch and bound algorithm is determined by how many jobs are fixed by the decomposition procedure. We noticed some trends regarding the effectiveness of the decomposition procedure. First, and most importantly, we noticed that as R increases the percentage of jobs fixed tends to increase. We also noticed that for a given R value, the percentage of jobs fixed decreases for increasing Q values, where recall that $Q \leq R$. For small R values with $R \leq 0.4$, the decomposition procedure is ineffective and fixes at most 1 and 6 percent of the jobs for $R = 0.2$ and 0.4, respectively. Moreover, for these values the decomposition procedure tends to become less effective as the number of jobs increases. For larger R values with $R \geq 0.6$, the decomposition procedure is quite effective, and it performs even better as the number of jobs increases. For $R = 0.6$, the decomposition procedure fixes around 40 percent of the jobs for the problems with $n = 20$, this increases to around 80 percent for $n = 200$. For $R = 1.0$, the percentage of jobs fixed increases to around 75 percent and 90 percent for problems with $n = 20$ and $n = 200$, respectively.

6 Concluding remarks

We have considered a new branch and bound algorithm for the problem $F2/r_j, perm/L'_{\max}$. The main features of the proposed algorithm are: an adaptive branching rule that fixes jobs at both ends of the schedule, a new dominance order to reduce branching together with new fuzzy dominance properties that are used for scheduling and tie-breaking, and a simple decomposition procedure that reduces the problem size by fixing jobs at the beginning of the schedule. In general, computational results indicated that the algorithm performed very well. It has solved more than 97% of the test problems with up to 200 jobs within a few seconds. Even when

we failed to solve a problem, we were always very close to the optimal solution. We found that the use of the dominance order resulted in a reduction in computation time of 15 to 20 percent. We also found that the use of fuzzy dominance properties aided the search by breaking ties and by often finding a nearly optimal initial solution. In summary, we feel that the proposed algorithm is more effective than previous solution methods for the problem $F2/r_j, perm/ L'_{\max}$, in that it found fewer hard problems and it was also able to solve much larger problems. The fact that most problems were solved to optimum within a few seconds, means that the algorithm has the potential of being used as a callable subroutine for $F2/r_j, perm/ L'_{\max}$-type subproblems generated during the solution of more complex scheduling problems.

Table 1: Results for $(0.2, 0.2)$ and $(0.4, 0.2)$.

n	*alg*	*solved*	*max gap* (%)	*total nodes*	*avg CPU* (sec)	*total CPU* (hrs:min:sec)
$R = 0.2$, $Q = 0.2$						
20	A_1	50/50	-	24,666	0.0840	4.20
	A_2	50/50	-	24,593	0.0958	4.79
	A_3	50/50	-	24,666	0.0828	4.14
40	A_1	50/50	-	3,894	0.1146	5.73
	A_2	50/50	-	3,893	0.1190	5.95
	A_3	50/50	-	3,894	0.1098	5.49
60	A_1	48/50	1.1401	2,001,789	0.2225	30:57.50
	A_2	48/50	1.1401	2,001,791	0.2056	33:17.86
	A_3	48/50	1.1401	2,001,791	0.1954	29:33.53
80	A_1	49/50	0.4968	1,009,292	1.0969	20:16.54
	A_2	49/50	0.4968	1,009,278	1.1967	22:48.84
	A_3	49/50	0.4968	1,009,288	1.0549	19:53.81
100	A_1	48/50	0.6178	2,029,119	4.6633	2:40:15.34
	A_2	48/50	0.6178	2,048,242	6.9856	2:16:42.26
	A_3	48/50	0.6178	2,048,242	5.9825	2:01:24.72
200	A_1	49/50	0.0655	1,063,650	38.7327	8:02:25.27
	A_2	49/50	0.0655	1,063,647	24.1063	7:06:01.35
	A_3	49/50	0.0655	1,063,647	22.0294	6:11:14.99
$R = 0.4$, $Q = 0.2$						
20	A_1	48/50	1.6575	2,947,498	2.9265	7:32.65
	A_2	48/50	1.6575	2,900,048	3.1988	8:11.50
	A_3	48/50	1.6575	2,903,012	2.7614	7:14.97
40	A_1	38/50	1.4881	13,365,909	7.3640	49:02.53
	A_2	38/50	1.4881	13,364,572	7.0429	44:46.62
	A_3	38/50	1.4881	13,365,936	6.7934	42:02.63
60	A_1	37/50	1.0747	13,323,626	1.5924	49:22.47
	A_2	37/50	1.0747	13,323,151	1.7548	43:38.78
	A_3	37/50	1.0747	13,323,293	1.5751	44:37.38
80	A_1	41/50	0.8299	9,758,575	3.8098	30:58.02
	A_2	41/50	0.8299	9,769,357	4.1583	33:56.53
	A_3	41/50	0.8299	9,782,955	3.8634	31:27.20
100	A_1	42/50	0.4259	8,232,036	2.9133	2:03:40.23
	A_2	42/50	0.4259	8,203,264	2.3276	2:20:44.43
	A_3	42/50	0.4259	8,203,030	2.4000	2:00:28.92
200	A_1	39/50	0.2801	12,391,165	15.8108	1:07:04.76
	A_2	39/50	0.2801	12,391,035	16.7367	1:12:45.20
	A_3	39/50	0.2801	12,391,171	15.5379	1:07:01.03

Table 2: Results for $(0.4, 0.4)$ and $R = 0.6$.

n	*alg*	*solved*	*max gap* (%)	*total nodes*	*avg CPU* (sec)	*total CPU* (hrs:min:sec)
$R = 0.4$, $Q = 0.4$						
	A_1	45/50	3.2119	5,505,470	1.9667	13:31.30
20	A_2	45/50	3.2119	5,474,584	1.6951	14:21.46
	A_3	45/50	3.2119	5,468,818	1.5591	12:53.13
	A_1	43/50	1.3497	7,000,924	0.0623	30:22.67
40	A_2	43/50	1.3497	7,000,874	0.0642	30:20.04
	A_3	43/50	1.3497	7,000,879	0.0607	27:27.53
	A_1	46/50	0.2151	4,001,078	0.1924	18:50.55
60	A_2	46/50	0.2151	4,001,105	0.1935	18:55.89
	A_3	46/50	0.2151	4,001,111	0.1820	17:25.11
	A_1	47/50	0.6494	3,000,978	0.3975	15:54.83
80	A_2	47/50	0.6494	3,001,023	0.3966	9:51.74
	A_3	47/50	0.6494	3,001,013	0.3753	11:24.47
	A_1	48/50	0.4598	2,001,628	0.8077	8:10.88
100	A_2	48/50	0.4598	2,001,549	0.7938	6:40.89
	A_3	48/50	0.4598	2,001,582	0.7523	8:14.86
	A_1	49/50	0.1693	1,002,837	6.4861	22:23.65
200	A_2	49/50	0.1693	1,002,824	6.4520	22:36.68
	A_3	49/50	0.1693	1,002,841	6.0212	19:50.06
$R = 0.6$, $Q = 0.2, 0.4, 0.6$						
	A_1	147/150	1.3150	3,120,585	0.0995	6:22.79
20	A_2	147/150	1.3150	3,098,357	0.0852	6:37.17
	A_3	147/150	1.3150	3,099,007	0.0767	6:25.34
	A_1	137/150	1.3754	13,100,696	0.1467	36:56.68
40	A_2	137/150	1.3754	13,103,281	0.1450	36:03.00
	A_3	137/150	1.3754	13,103,391	0.1322	31:22.49
	A_1	147/150	0.7059	3,055,775	0.1070	13:34.11
60	A_2	147/150	0.7059	3,036,978	0.0748	12:32.31
	A_3	147/150	0.7059	3,036982	0.0707	11:38.68
	A_1	145/150	0.3267	5,003,273	0.0682	11:43.19
80	A_2	145/150	0.3267	5,003,245	0.0657	24:34.21
	A_3	145/150	0.3267	5,003,245	0.0635	14:08.36
	A_1	148/150	0.4820	2,006,622	0.0818	4:28.17
100	A_2	148/150	0.4820	2,006,722	0.0845	4:40.68
	A_3	148/150	0.4820	2,006,722	0.0810	3:23.59
	A_1	147/150	0.1533	3,000,729	0.2136	17:11.18
200	A_2	147/150	0.1533	3,000,778	0.2206	18:39.67
	A_3	147/150	0.1533	3,000,732	0.2064	15:32.97

Table 3: Results for $R = 0.8$ and $R = 1.0$.

n	*alg*	*solved*	*max gap* (%)	*total nodes*	*avg CPU* (sec)	*total CPU* (hrs:min:sec)
$R = 0.8$, $Q = 0.2, 0.4, 0.6, 0.8$						
20	A_1	200/200	-	1164	0.0026	0.51
	A_2	200/200	-	1149	0.0029	0.58
	A_3	200/200	-	1149	0.0027	0.54
40	A_1	200/200	-	914	0.0056	1.11
	A_2	200/200	-	912	0.0056	1.11
	A_3	200/200	-	912	0.0053	1.06
60	A_1	199/200	0.2625	1,001,163	0.0098	2:32.97
	A_2	199/200	0.2625	1,001,167	0.0097	6:53.71
	A_3	199/200	0.2625	1,001,167	0.0095	2:33.83
80	A_1	198/200	0.1513	2,000,359	0.0013	6:10.13
	A_2	198/200	0.1513	2,000,357	0.0012	3:35.69
	A_3	198/200	0.1513	2,000,357	0.0012	3:30.68
100	A_1	200/200	-	257	0.0181	3.61
	A_2	200/200	-	261	0.0183	3.65
	A_3	200/200	-	261	0.0180	3.59
200	A_1	200/200	-	259	0.0571	11.42
	A_2	200/200	-	267	0.0569	11.37
	A_3	200/200	-	267	0.0568	11.36
$R = 1.0$, $Q = 0.2, 0.4, 0.6, 0.8, 1.0$						
20	A_1	250/250	-	376	0.0020	0.49
	A_2	250/250	-	543	0.0022	0.54
	A_3	250/250	-	543	0.0023	0.57
40	A_1	250/250	-	332	0.0042	1.04
	A_2	250/250	-	329	0.0043	1.08
	A_3	250/250	-	329	0.0043	1.08
60	A_1	250/250	-	354	0.0076	1.91
	A_2	250/250	-	354	0.0080	1.99
	A_3	250/250	-	354	0.0082	2.06
80	A_1	249/250	0.2736	1,000,309	0.0119	2:36.63
	A_2	249/250	0.2736	1,000,307	0.0121	3:14.71
	A_3	249/250	0.2736	1,000,307	0.0121	3:13.30
100	A_1	250/250	-	273	0.0177	4.43
	A_2	250/250	-	273	0.0178	4.45
	A_3	250/250	-	273	0.0178	4.45
200	A_1	250/250	-	276	0.0619	15.47
	A_2	250/250	-	276	0.0620	15.49
	A_3	250/250	-	276	0.0620	15.50

Appendix

Calculations for root node

- apply decomposition procedure with nondecreasing r sequence to fix σ_1;
- let $S := J \backslash S_1$, $\sigma_2 := \varnothing$; $nodes := 0$;
- construct the dominance order $\prec$;
- find fuzzy sequence 1and fuzzy sequence 2;
- calculate initial UB and LB, and let $BestUB := UB$;
- call recursive procedure Opt that facilitates branching;

procedure $Opt(\sigma_1, S, \sigma_2, BestUB)$;
begin

if $nodes \leq 1000000$ **then**

begin

$nodes := nodes + 1$;{increment the number of nodes branched from}
if $(nodes \leq n)$ **or** $(|S| = 1)$ **then** calculate UB and (possibly) update $BestUB$
if $(LB < BestUB)$ **and** $(|S| \neq 1)$ **then**
begin
find lists of *minimal* and *maximal* jobs in $\prec|_S$
if $type = 1$ **then** calculate lower bounds for *minimal* and sort in nondecreasing lower bound order breaking ties using fuzzy sequence 1
else if $type = 2$ **then** calculate lower bounds for *maximal* and sort in nondecreasing lower bound order breaking ties using fuzzy sequence 2
else (*type* not set)
begin
calculate and sort lower bounds for both *minimal* and *maximal* breaking ties accordingly
find n_1 and n_2 to determine *type* for node
if $n_1 < n_2$ **then** $type := 1$
else if $n_1 > n_2$ **then** $type := 2$
else $type := previous\_type$;

```
        end;
        if type = 1 then list := minimal
        else if type = 2 then list := maximal;
        while (list ≠ ∅) and (BestUB > lower bound from head of list) do
        begin
            remove head of list
            let job be the job that is fixed and let LB be its lower bound
            remove job from ≺
            S := S\{job};
            if type = 1 then add job to σ1
            else if type = 2 then add job to σ2;
            Opt(σ1, S, σ2, BestUB);{recursive call}
            if type = 1 then remove job from σ1
            else if type = 2 then remove job from σ2;
            S := S ∪ {job};
            restore job to ≺
        end;{while}
    end;{if (LB < BestUB) and (|S| ≠ 1)}
end;{if nodes ≤ 1000000}
end;
```

Table 4: Dominating pairs for Theorem 1.

s_2	s_1	
$(XUkmVZ)$	$(XmYkZ)$	$proof$
(x_1, x_2, x_3)	(x_1, x_2, x_3)	
(x_1, x_2, u)	(x_1, x_2, u)	
(x_1, x_2, k)	(x_1, x_2, k)	
(x_1, x_2, m)	(x_1, x_2, k)	$q_k \geq q_m$
(x_1, x_2, v)	(x_1, x_2, k)	$q_k \geq q_v$
(x_1, x_2, z)	(x_1, x_2, z)	
(x, u_1, u_2)	(x, u_1, u_2)	
(x, u, k)	(x, u, k)	
(x, u, m)	(x, m, k)	$U \subseteq J_{a \leq b}, a_u \leq a_m, q_k \geq q_m$
(x, u, v)	(x, m, k)	$U \subseteq J_{a \leq b}, a_u \leq a_m, q_k \geq q_v$
(x, u, z)	(x, m, z)	$U \subseteq J_{a \leq b}, a_u \leq a_m$
(x, k, k)	(x, k, k)	
(x, k, m)	(x, m, k)	$U \subseteq J_{a \leq b}, k \in J_{a \leq b}, q_k \geq q_m$
(x, k, v)	(x, m, k)	$U \subseteq J_{a \leq b}, a_k \leq a_m, q_k \geq q_v$
(x, k, z)	(x, m, z)	$U \subseteq J_{a \leq b}, a_k \leq a_m,$
(x, m, m)	(x, m, k)	$U \subseteq J_{a \leq b}, k \in J_{a \leq b}, q_k \geq q_m$
(x, m, v)	(x, m, k)	$U \subseteq J_{a \leq b}, k \in J_{a \leq b}, q_k \geq q_v$
(x, m, z)	(x, m, z)	$U \subseteq J_{a \leq b}, k \in J_{a \leq b}$
(x, v_1, v_2)	(x, v_1, k)	$U \subseteq J_{a \leq b}, k \in J_{a \leq b}, q_k \geq q_{v_2}$
(x, v, z)	(x, v, z)	$U \subseteq J_{a \leq b}, k \in J_{a \leq b}$
(x, z_1, z_2)	(x, z_1, z_2)	
(u_1, u_2, u_3)	(u_1, u_2, u_3)	
(u_1, u_2, k)	(u_1, u_2, k)	
(u_1, u_2, m)	(m, m, k)	$r_{u_1} \leq r_m, U \subseteq J_{a \leq b}, a_{u_2} \leq a_m, q_k \geq q_m$
(u_1, u_2, v)	(m, m, k)	$r_{u_1} \leq r_m, U \subseteq J_{a \leq b}, a_{u_2} \leq a_m, q_k \geq q_v$
(u_1, u_2, z)	(m, m, z)	$r_{u_1} \leq r_m, U \subseteq J_{a \leq b}, a_{u_2} \leq a_m$
(u, k, k)	(m, k, k)	$r_u \leq r_m$
(u, k, m)	(m, m, k)	$r_u \leq r_m, U \subseteq J_{a \leq b}, a_k \leq a_m, q_k \geq q_m$

Table 4: (continued)

s_2	s_1	
$(XUkmVZ)$	$(XmYkZ)$	*proof*
(u,k,v)	(m,m,k)	$r_u \leq r_m, U \subseteq J_{a\leq b}, a_k \leq a_m, q_k \geq q_v$
(u,k,z)	(m,m,z)	$r_u \leq r_m, U \subseteq J_{a\leq b}, a_k \leq a_m$
(u,m,m)	(m,m,k)	$r_u \leq r_m, U \subseteq J_{a\leq b}, k \in J_{a\leq b}, q_k \geq q_m$
(u,m,v)	(m,m,k)	$r_u \leq r_m, U \subseteq J_{a\leq b}, k \in J_{a\leq b}, q_k \geq q_v$
(u,m,z)	(m,m,z)	$r_u \leq r_m, U \subseteq J_{a\leq b}, k \in J_{a\leq b}$
(u,v_1,v_2)	(m,v_1,k)	$r_u \leq r_m, U \subseteq J_{a\leq b}, k \in J_{a\leq b}, q_k \geq q_{v_2}$
(u,v,z)	(m,v,z)	$r_u \leq r_m, U \subseteq J_{a\leq b}, k \in J_{a\leq b}$
(u,z_1,z_2)	(m,z_1,z_2)	$r_u \leq r_m$
(k,k,k)	(k,k,k)	
(k,k,m)	(m,m,k)	$r_k \leq r_m, a_k \leq a_m, q_k \geq q_m$
(k,k,v)	(m,m,k)	$r_k \leq r_m, a_k \leq a_m, q_k \geq q_v$
(k,k,z)	(m,m,z)	$r_k \leq r_m, a_k \leq a_m$
(k,m,m)	(m,m,k)	$r_k \leq r_m, k \in J_{a\leq b}, q_k \geq q_m$
(k,m,v)	(m,m,k)	$r_k \leq r_m, k \in J_{a\leq b}, q_k \geq q_v$
(k,m,z)	(m,m,z)	$r_k \leq r_m, k \in J_{a\leq b}$
(k,v_1,v_2)	(m,v_1,k)	$r_k \leq r_m, k \in J_{a\leq b}, q_k \geq q_{v_2}$
(k,v,z)	(m,v,z)	$r_k \leq r_m, k \in J_{a\leq b}$
(k,z_1,z_2)	(m,z_1,z_2)	$r_k \leq r_m$
(m,m,m)	(m,m,m)	
(m,m,v)	(m,m,v)	
(m,m,z)	(m,m,z)	
(m,v_1,v_2)	(m,v_1,v_2)	
(m,v,z)	(m,v,z)	
(m,z_1,z_2)	(m,z_1,z_2)	
(v_1,v_2,v_3)	(v_1,v_2,v_3)	
(v_1,v_2,z)	(v_1,v_2,z)	
(v,z_1,z_2)	(v,z_1,z_2)	
(z_1,z_2,z_3)	(z_1,z_2,z_3)	

Table 5: Dominating pairs for Theorem 2.

s_2	s_1	
$(XUkmVZ)$	$(XmYkZ)$	*proof*
(x_1, x_2, x_3)	(x_1, x_2, x_3)	
(x_1, x_2, u)	(x_1, x_2, u)	
(x_1, x_2, k)	(x_1, x_2, k)	
(x_1, x_2, m)	(x_1, x_2, k)	$q_k \geq q_m$
(x_1, x_2, v)	(x_1, x_2, k)	$q_k \geq q_v$
(x_1, x_2, z)	(x_1, x_2, z)	
(x, u_1, u_2)	(x, u_1, u_2)	
(x, u, k)	(x, u, k)	
(x, u, m)	(x, u, k)	$m \in J_{a>b}, q_k \geq q_m$
(x, u, v)	(x, u, k)	$m \in J_{a>b}, V \subseteq J_{a>b}, q_k \geq q_v$
(x, u, z)	(x, u, z)	$m \in J_{a>b}, V \subseteq J_{a>b}$
(x, k, k)	(x, k, k)	
(x, k, m)	(x, k, k)	$m \in J_{a>b}, V \subseteq J_{a>b}, q_k \geq q_m$
(x, k, v)	(x, k, k)	$m \in J_{a>b}, V \subseteq J_{a>b}, q_k \geq q_v$
(x, k, z)	(x, k, z)	$m \in J_{a>b}, V \subseteq J_{a>b}$
(x, m, m)	(x, m, k)	$U \subseteq J_{a\leq b}, k \in J_{a\leq b}, q_k \geq q_m$
(x, m, v)	(x, m, k)	$U \subseteq J_{a\leq b}, k \in J_{a\leq b}, q_k \geq q_v$
(x, m, z)	(x, m, z)	$U \subseteq J_{a\leq b}, k \in J_{a\leq b}$
(x, v_1, v_2)	(x, v_1, k)	$U \subseteq J_{a\leq b}, k \in J_{a\leq b}, q_k \geq q_{v_2}$
(x, v, z)	(x, v, z)	$U \subseteq J_{a\leq b}, k \in J_{a\leq b}$
(x, z_1, z_2)	(x, z_1, z_2)	
(u_1, u_2, u_3)	(u_1, u_2, u_3)	
(u_1, u_2, k)	(u_1, u_2, k)	
(u_1, u_2, m)	(m, u_2, k)	$r_{u_1} \leq r_m, m \in J_{a>b}, q_k \geq q_m$
(u_1, u_2, v)	(m, u_2, k)	$r_{u_1} \leq r_m, m \in J_{a>b}, V \subseteq J_{a>b}, q_k \geq q_v$
(u_1, u_2, z)	(m, u_2, z)	$r_{u_1} \leq r_m, m \in J_{a>b}, V \subseteq J_{a>b}$
(u, k, k)	(u, k, k)	
(u, k, m)	(m, k, k)	$r_u \leq r_m, m \in J_{a>b}, q_k \geq q_m$

Table 5: (continued)

s_2	s_1	
$(XUkmVZ)$	$(XmYkZ)$	*proof*
(u,k,v)	(m,k,k)	$r_u \leq r_m, m \in J_{a>b}, V \subseteq J_{a>b}, q_k \geq q_v$
(u,k,z)	(m,k,z)	$r_u \leq r_m, m \in J_{a>b}, V \subseteq J_{a>b}$
(u,m,m)	(m,m,k)	$r_u \leq r_m, U \subseteq J_{a\leq b}, k \in J_{a\leq b}, q_k \geq q_m$
(u,m,v)	(m,m,k)	$r_u \leq r_m, U \subseteq J_{a\leq b}, k \in J_{a\leq b}, q_k \geq q_v$
(u,m,z)	(m,m,z)	$r_u \leq r_m, U \subseteq J_{a\leq b}, k \in J_{a\leq b}$
(u,v_1,v_2)	(m,v_1,k)	$r_u \leq r_m, U \subseteq J_{a\leq b}, k \in J_{a\leq b}, q_k \geq q_{v_2}$
(u,v,z)	(m,v,z)	$r_u \leq r_m, U \subseteq J_{a\leq b}, k \in J_{a\leq b}$
(u,z_1,z_2)	(m,z_1,z_2)	$r_u \leq r_m$
(k,k,k)	(k,k,k)	
(k,k,m)	(m,k,k)	$r_k \leq r_m, m \in J_{a>b}, q_k \geq q_m$
(k,k,v)	(m,k,k)	$r_k \leq r_m, m \in J_{a>b}, V \subseteq J_{a>b}, q_k \geq q_v$
(k,k,z)	(m,k,z)	$r_k \leq r_m, m \in J_{a>b}, V \subseteq J_{a>b}$
(k,m,m)	(m,m,k)	$r_k \leq r_m, k \in J_{a\leq b}, q_k \geq q_m$
(k,m,v)	(m,m,k)	$r_k \leq r_m, k \in J_{a\leq b}, q_k \geq q_v$
(k,m,z)	(m,m,z)	$r_k \leq r_m, k \in J_{a\leq b}$
(k,v_1,v_2)	(m,v_1,k)	$r_k \leq r_m, k \in J_{a\leq b}, q_k \geq q_{v_2}$
(k,v,z)	(m,v,z)	$r_k \leq r_m, k \in J_{a\leq b}$
(k,z_1,z_2)	(m,z_1,z_2)	$r_k \leq r_m$
(m,m,m)	(m,m,m)	
(m,m,v)	(m,m,v)	
(m,m,z)	(m,m,z)	
(m,v_1,v_2)	(m,v_1,v_2)	
(m,v,z)	(m,v,z)	
(m,z_1,z_2)	(m,z_1,z_2)	
(v_1,v_2,v_3)	(v_1,v_2,v_3)	
(v_1,v_2,z)	(v_1,v_2,z)	
(v,z_1,z_2)	(v,z_1,z_2)	
(z_1,z_2,z_3)	(z_1,z_2,z_3)	

References

[1] Cheng, J., H. Kise, and H. Matsumoto (1997). A branch and bound algorithm with fuzzy inference for a permutation flowshop scheduling problem. *European J. Oper. Res.* 96, 578-590.

[2] Grabowski, J. (1980). On two-machine scheduling with release and due dates to minimize maximum lateness. *Opsearch* 17, 133-154.

[3] Grabowski, J., E. Skubalska, and C. Smutnicki (1983). On flow shop scheduling with release and due dates to minimize maximum lateness. *J. Oper. Res. Soc.* 34, 615-620.

[4] Hariri, A.M., and C.N. Potts (1983). An algorithm for single machine sequencing with release dates to minimize total weighted completion time. *Discrete Appl. Math.* 5, 99-109.

[5] Johnson, S.M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Res. Logist. Quart.* 1, 61-68.

[6] Lageweg, B.J., J.K. Lenstra, and A.H.G. Rinnooy Kan (1978). A general bounding scheme for the permutation flowshop problem. *Oper. Res.* 26, 53-67.

[7] Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys (1989). Sequencing and scheduling: Algorithms and complexity. Report BS-R8909, CWI Amsterdam.

[8] Lenstra, J.K., A.H.G. Rinnooy Kan, and P. Brucker (1977). Complexity of machine scheduling problems. *Annals of Disc. Math.* 1, 343-362.

[9] Monma, C.L. (1978). Properties and efficient algorithms for certain classes of sequencing problems, Ph. D. Thesis, Cornell University, Ithaca, New York.

[10] Pinedo, M. (1995) *SCHEDULING: Theory, Algorithms and Systems*, Prentice Hall, Englewood Cliffs, N.J.

[11] Potts, C.N. (1980). An adaptive branching rule for the permutation flow-shop problem. *European J. Oper. Res.* 5, 19-25.

[12] Tadei, R., J.N.D. Gupta, F. Della Croce, and M. Cortesi (1998). Minimising makespan in the two-machine flow-shop with release times. *J. Oper. Res. Soc.* 49, 77-85.

[13] Taillard, E. (1993). Benchmarks for basic scheduling problems. *European J. Oper. Res.* 64, 278-285.

Faculty of Business
McMaster University

WORKING PAPERS - RECENT RELEASES

411. Harish C. Jain and S. Muthuchidambaram, "Bill 40 Amendments to Ontario Labour Relations Act: An Overview and Evaluation", June, 1995.

412. Jiang Chan and George Steiner, "Discrete Lot Streaming in Three-Machine Flow Shops", July, 1995.

413. J. Brimberg, A. Mehrez and G.O. Wesolowsky, "Allocation of Queueing Facilities Using a Minimax Criterion", January, 1996.

414. Isik Zeytinoglu and Jeanne Norris, "Global Diversity in Employment Relationships: A Typology of Flexible Employment", March, 1996.

415. N. Archer, "Characterizing World Wide Web Search Strategies", April, 1996.

416. J. Rose, "Immediacy and Saliency in Remedying Employer Opposition to Union Organizing Campaigns", July, 1996.

417. Roy J. Adams and Parbudyal Singh, "Worker Rights Under NAFTA: Experience With the North American Agreement on Labor Cooperation", September, 1996.

418. George Steiner and Paul Stephenson, "Subset-Restricted Interchange for Dynamic Min-Max Scheduling Problems", September, 1996.

419. Robert F. Love and Halit Uster, "Comparison of the Properties and the Performance of the Criteria Used to Evaluate the Accuracy of Distance Predicting Functions", November, 1996.

420. Harish C. Jain and Simon Taggar, "The Status of Employment Equity in Canada", December, 1996.

421. Harish C. Jain and Parbudyal Singh, "Beyond The Rhetoric: An Assessment of the Political Arguments and Legal Principles on Strike Replacement Laws in North America", January, 1997.

422. Jason Schwandt, "Electronic Data Interchange: An Overview of Its Origins, Status, and Future", March, 1997.

423. Christopher K. Bart with John C. Tabone, "Mission Statement Rationales and Organizational Alignment in the Not-for-profit Healthcare Sector", November, 1997.

424. Harish C. Jain, Michael Piczak, Işik Urla Zeytinoğlu, "Workplace Substance Testing - An Exploratory Study", November, 1997.

425. S. Suarga, Yufei Yuan, Joseph B. Rose, and Norman Archer, "Web-based Collective Bargaining Support System: A Valid Process Support Tool for Remote Negotiation", January, 1998.

426. Pawan S. Budhwar and Harish C. Jain, "Evaluating Levels of Strategic Integration and Development of Human Resource Management in Britain", March, 1998.

427. Halit Üster and Robert F. Love, "Application of Weighted Sums of Order p to Distance Estimation", April, 1998.

428. Halit Üster and Robert F. Love, "On the Directional Bias of the ℓ_{bp}-norm", April, 1998.

429. Milena Head, Norm Archer, and Yufei Yuan, "MEMOS: A World Wide Web Navigation Aid", October, 1998.

430. Harish C. Jain and Parbudyal Singh, "The Effects of the Use of Strike Replacement Workers on Strike Duration in Canada", February, 1999.

431. Parbudyal Singh and Harish C. Jain, "Strike Replacements in the United States, Canada and Mexico: A Review of the Law and Empirical Research", February, 1999.

432. John W. Medcof and Jeremy Boyko, "Reinforcing, Revising and Reconciling Attributions in the Employment Interview", March, 1999.

433. Norm Archer, "World Wide Web Business Catalogs in Business-to-Business Procurement", March, 1999.

434. Diwakar Gupta and Saifallah Benjaafar, "Make-to-order, Make-to-stock, or Delay Product Differentiation? - A Common Framework for Modeling and Analysis", April, 1999.

435. Harish C. Jain, Parbudyal Singh and Carol Agocs, "Recruitment, Selection and Promotion of Visible Minorities and Aboriginals in Selected Canadian Police Services", April, 1999.

436. Harish C. Jain and Angus Bowmaker-Falconer, "Employment Equity/Affirmative Action Codes of Practice and Best Practices in USA, Britain, Canada and Other Selected Countries", May, 1999.

437. Diwakar Gupta, Yavuz Günalay, and Mandyam M. Srinivasan, "On the Relationship Between Preventive Maintenance and Manufacturing System Performance", June, 1999.

438. Jinliang Cheng, George Steiner, and Paul Stephenson, "A Fast Algorithm to Minimize Makespan for the Two-Machine Flow-Shop Problem with Release Times", June, 1999.