McMaster University

MICHAEL G. DEGROOTE SCHOOL OF BUSINESS

RESEARCH AND WORKING PAPER SERIES

A FAST ALGORITHM TO MINIMIZE MAKESPAN 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 # 438

June, 1999

A FAST ALGORITHM TO MINIMIZE MAKESPAN 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 # 438

June, 1999

A Fast Algorithm to Minimize Makespan 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 makespan. We derive a new dominance order and incorporate it into an efficient branch and bound algorithm which uses an adaptive branching scheme. The algorithm performed very well. It solved within a few seconds 95% of the test problems with up to 500 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%. Experiments also indicate that the speed of the algorithm is largely due to the use of the dominance order which cut solution times roughly in half.

1 Introduction

Minimizing the makespan in a flow-shop environment is a classical scheduling problem. The simplest case of this problem, the two-machine flow-shop problem $F2//C_{\max}$ on n jobs, is solved in $O(n \log n)$ time by Johnson's rule [5]. (We use the standard notation to describe scheduling problems, and refer the reader to [7] and [10] for any terminology not defined here.) When arbitrary release times are added, the problem $F2/r_j/C_{\max}$ becomes strongly NP-hard [8]. This result has led researchers to consider different approximation and branch

*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

and bound algorithms. Potts [12] developed several approximation algorithms and analyzed their worst case performance. The best of these algorithms has a worst case performance ratio of 5/3 with time complexity $O(n^3 \log n)$. Later, polynomial approximation schemes were developed for the problem by Hall [4] and Kovalyov and Werner [6]. Grabowski [2] presented a branch and bound algorithm for the problem $F2\,/r_j/\,L_{\max}$, which can also be used for the problem $F2\,/r_j/\,C_{\max}$. Grabowski's algorithm used a new type of branching scheme that exploited certain dominance properties of a critical path. Tadei et al. [13] tested several branch and bound algorithms for the problem $F2\,/r_j/\,C_{\max}$. They tested the effectiveness of various lower bounds and branching schemes. They have classified instances as "easy" or "hard" according to the distribution of the release times. From the easy class they have solved problems with up to 200 jobs. While from the hard category they were able to solve instances with up to 80 jobs within 300 seconds. They have also found Grabowski's dichotomic branching scheme less effective for the problem $F2\,/r_j/\,C_{\max}$ than their n-ary branching scheme.

In this paper, we consider a new branch and bound algorithm for the problem $F2\,/r_j/\,C_{\max}$, with the following main features. We use an adaptive n-ary branching rule, that fixes jobs at both ends of the schedule, similar to the one used by Potts [11] for the problem $Fm//C_{\max}$. We present a new dominance order on the job set, which is derived by using a new proof technique we call "subset-restricted interchange". Although the proof of validity for the dominance order is quite elaborate, its application requires only a very fast and simple procedure at the root of the branch and bound tree. We also incorporate a simple decomposition procedure which proved to be especially effective for problems with large job release times. We use four very quickly computable lower bounds at each node of the tree. The algorithm represents an effective and very fast tool for solving large instances of the strongly NP-hard problem $F2\,/r_j/\,C_{\max}$. In a large scale computational experiment, the algorithm has solved in a few seconds 1,714 of the 1,800 randomly generated test problems with up to 500 jobs. We have also gained the insight that the relatively few problems which were left unsolved had their parameter which determines the range of release times concentrated in a very narrow interval. Even for the unsolved problems, the best solution found by the algorithm was on average within less than 0.5% of the minimum schedule length.

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 $F2/r_j/C_{\max}$. 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 the sequence in which the jobs are performed. It is well known that the problem $F2/r_j/C_{\max}$ always has an optimal permutation schedule, i.e., it is a sequencing problem [1]. 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 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. For the sequencing problem that we study, 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. A job v is *minimal* if there is no job u with $u <_P v$. Similarly, a job u is *maximal* if there is no job 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 subset $I \subseteq X$ is an *ideal* of P if for every $v \in I$ and $u \in X$ such that $u \leq_P v$ we have $u \in I$. Similarly, $F \subseteq X$ is a *filter* of P if for every $u \in F$ and $v \in X$ such that $u \leq_P v$ we have $v \in F$. For every $v \in X$ the *principal ideal* $I(v)$ is defined by $I(v) = \{u \in X \,|u \leq_P v\}$ and the *principal filter* $F(v)$ is defined by $F(v) = \{u \in X \,|v \leq_P u\}$. 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 a new dominance order $\prec$ for problem $F2/r_j/C_{\max}$. We derive $\prec$ using *subset-restricted interchange*, which is a new technique that incorporates certain restrictions on the subset of intermediate jobs for the different interchange operators.

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. We present two types of interchanges of jobs k and m that leave k preceding m in the resulting sequence s_2.

1. *Backward Insertion*(BI)

   $s_2 = (XYkmZ)$

2. *Forward Insertion*(FI)

   $s_2 = (XkmYZ)$

If we let Y be the set of jobs in sequence Y (we do not distinguish between these), then these interchanges reduce to the traditional adjacent pairwise interchange in the case when $Y = \emptyset$. This leads to the definition of the *adjacent interchange order.*

Definition 1 *A partial order $\preccurlyeq$ is an Adjacent Interchange Order for a sequencing function f if*
for all jobs k, m and sequences X, Z
$k \preccurlyeq m$ implies $f(XkmZ) \leq f(XmkZ)$.

In the past, adjacent interchange orders have been used only locally, for interchanging neighbouring jobs. In the following development, we show that adjacent interchange orders may contain *globally usable* information too, which allows us to derive globally valid dominance orders on the job set. Note that the interchanges BI and FI involve interchanging k or m around sequence Y. Intuitively, whether or not an interchange leads to a reduction in cost (for a given sequencing function f and adjacent interchange order $\preccurlyeq$), should depend on the composition of Y. We consider interchanges that are restricted by conditions on Y and define subset-restricted interchange conditions resulting in cost reductions as follows.

Definition 2 *An adjacent interchange order $\preccurlyeq$ together with the collection of subsets $R^{BI} = \left\{R^{BI}_{k\preccurlyeq m} | k \preccurlyeq m\right\}$ satisfies the Restricted Backward Insertion Condition for a sequencing function f if*
for all jobs k, m and sequences X, Y, Z
$k \preccurlyeq m$ and $Y \subseteq R^{BI}_{k\preccurlyeq m}$ imply $f(XYkmZ) \leq f(XmYkZ)$.

Definition 3 *An adjacent interchange order $\preccurlyeq$ together with the collection of subsets $R^{FI} = \left\{R^{FI}_{k\preccurlyeq m} | k \preccurlyeq m\right\}$ satisfies the Restricted Forward Insertion Condition for a sequencing function f if*
for all jobs k, m and sequences X, Y, Z
$k \preccurlyeq m$ and $Y \subseteq R^{FI}_{k\preccurlyeq m}$ imply $f(XkmYZ) \leq f(XmYkZ)$.

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]. 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 $F2/r_j/C_{\max}$ is the *intersection* of the *nondecreasing* r order and the Johnson order. Note that this order is no longer a complete order, rather it is only a partial order. To emphasize the partitioning of the jobs in the Johnson order we define $J_{a\leq b} = \{j \,|a_j \leq b_j\}$ and $J_{a>b} = \{j \,|a_j > b_j\}$. Next, we formally define the adjacent interchange order $\lessdot$ for the problem $F2/r_j/C_{\max}$.

Definition 4 *Adjacent Interchange Order:*

$k \lessdot m$ *if* $r_k \leq r_m$ *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 $F2/r_j/C_{\max}$ in Example 1. Here we have $3 \lessdot 1$($r_3 = r_1 = 10$, $3,1 \in J_{a>b}$ and $b_3 = 25 > 15 = b_1$), however, the unique optimal sequence is (1,2,3,4) with $C_{\max} = 125$. Thus we see that $\lessdot$ is not a dominance order since there is no optimal sequence that is a linear extension of $\lessdot$. We 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

Next we distinguish between different types of pairs in $\lessdot$ that have different interchange properties.

Theorem 1 *If* $k \in J_{a\leq b}$ *and* $a_k \leq a_m$, *then* $k \lessdot m$ *together with the set* $R^{FI}_{k\lessdot m} = J$ *satisfies the Restricted Forward Insertion Condition.*

Proof. Given a sequence s we construct a directed graph $G(s)$ to evaluate $C_{\max}(s)$. Each job $s(j)$ is represented by three nodes with weights $r_{s(j)}$, $a_{s(j)}$, and $b_{s(j)}$ respectively. $G(s)$ has the property that $C_{\max}(s)$ is the length of the longest 'node-weighted' path from 0 to $s(n)$. Each of these paths can be identified by the pair $(s(i), s(j))$, for some $i, j \in [1, n]$, which are the endpoints of the horizontal segments of the path (see Figure 1). Then by definition

$$C_{\max}(s) = \max_{\substack{(s(i),s(j)) \\ 1 \le i \le j \le n}} \left(r_{s(i)} + \sum_{l=i}^{j} a_{s(l)} + \sum_{l=j}^{n} b_{s(l)} \right),$$

and we can evaluate $C_{\max}(s)$ as the maximum over all such pairs $(s(i), s(j))$ in $G(s)$.

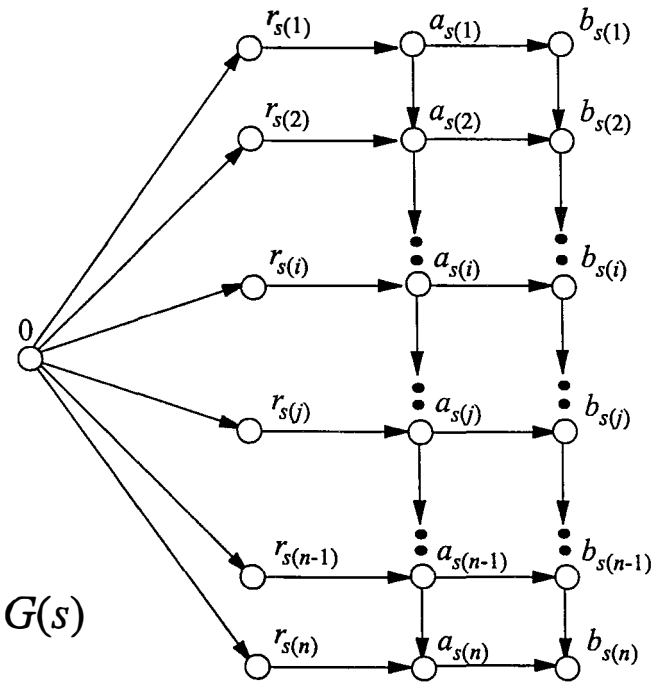

Figure 1: Directed graph $G(s)$.

Let $s_1 = (XmYkZ)$ be a sequence for pair $k \lessdot m$ with $k \in J_{a \le b}$ and $a_k \le a_m$. We apply forward insertion to s_1 and obtain sequence $s_2 = (XkmYZ)$. We demonstrate that s_2 is not worse than s_1 by showing that for *every* pair of jobs in s_2 there exists a *dominating pair* and

s_2	s_1			
$(XkmYZ)$	$(XmYkZ)$	*proof*		
(x,k)	(x,m)		$a_k \leq a_m$	
(x,m)	(x,m)			$k \in J_{a \leq b}$
(x,y)	(x,y)			$k \in J_{a \leq b}$
(x,z)	(x,z)			
(k,k)	(m,m)	$r_k \leq r_m$	$a_k \leq a_m$	
(k,m)	(m,m)	$r_k \leq r_m$		$k \in J_{a \leq b}$
(k,y)	(m,y)	$r_k \leq r_m$		$k \in J_{a \leq b}$
(k,z)	(m,z)	$r_k \leq r_m$		
(m,m)	(m,m)			
(m,y)	(m,y)			
(m,z)	(m,z)			
(y,z)	(y,z)			

Table 1: Dominating pairs for Theorem 1.

corresponding path in s_1 with a not smaller $C_{\max}$ value. Moreover, we show that the choice of the dominating pairs is independent of the intermediate sequence Y, thus we can take the interchange region to be the entire job set i.e., $R^{FI}_{k \lessdot m} = J$. For example consider pair (k,k) in s_2, then the corresponding dominating pair in s_1 is (m,m). That is,

$$r_k + a_k + b_k + b_m + \sum_{y \in Y} b_y + \sum_{z \in Z} b_z \leq r_m + a_m + b_m + \sum_{y \in Y} b_y + b_k + \sum_{z \in Z} b_z,$$

since $r_k \leq r_m$ and $a_k \leq a_m$.

Table 1 gives a dominating pair in s_1 for each pair in s_2. (We use lower case letters x,y or z to refer to arbitrary generic elements of the subsequences X,Y or Z, respectively). The last three columns contain the arguments why they are dominating pairs. ■

Note that $R^{FI}_{k \lessdot m} = J$ in Theorem 1 means that in fact there are no restrictions on the intermediate set Y, i.e., k can be inserted forward before m around *any* subsequence $Y \subseteq J$. The next two theorems show cases when the intermediate set Y must satisfy certain real restrictions for the interchange operators to be applicable.

Theorem 2 *If $m \in J_{a>b}$ and $b_k \geq b_m$, then $k \lessdot m$ together with the set $R^{BI}_{k \lessdot m} = \{j \,|r_j \leq r_m\}$ satisfies the Restricted Backward Insertion Condition.*

s_2	s_1			
$(XYkmZ)$	$(XmYkZ)$	*proof*		
(x,y)	(x,y)			$m \in J_{a>b}$
(x,k)	(x,k)			$m \in J_{a>b}$
(x,m)	(x,k)		$b_k \geq b_m$	
(x,z)	(x,z)			
(y_i,y_j)	(m,y_j)	$r_{y_i} \leq r_m$		$m \in J_{a>b}$
(y,k)	(m,k)	$r_y \leq r_m$		$m \in J_{a>b}$
(y,m)	(m,k)	$r_y \leq r_m$	$b_k \geq b_m$	
(y,z)	(m,z)	$r_y \leq r_m$		
(k,k)	(m,k)	$r_k \leq r_m$		$m \in J_{a>b}$
(k,m)	(m,k)	$r_k \leq r_m$	$b_k \geq b_m$	
(k,z)	(m,z)	$r_k \leq r_m$		
(m,z)	(m,z)			

Table 2: Dominating pairs for Theorem 2.

Proof. Similarly, Table 2 gives a dominating pair in s_1 for each pair in s_2. ■

Theorem 3 *$k \in J_{a\leq b}$ and $m \in J_{a>b}$ then $k \lessdot m$ together with the set $R^{FI}_{k\lessdot m} = J_{a>b}$ satisfies the Restricted Forward Insertion Condition.*

Proof. Table 3 gives the dominating pair in s_1 for each pair in s_2. ■

3.2 New dominance order

Tadei et al. [13] have established a dominance order for problem $F2/r_j/C_{\max}$, which can be interpreted as a corollary of Theorem 1.

Corollary 1 *[13] If $r_k \leq r_m$, $k \in J_{a\leq b}$ and $a_k \leq a_m$ then job k precedes m in an optimal solution.*

Corollary 1 is indeed a consequence of Theorem 1, since its conditions imply $k \lessdot m$, and so k and m satisfy all the conditions of Theorem 1. Therefore, FI can be applied to any sequence of the form $(XmYkZ)$ to insert k before m around any intermediate sequence Y. Since $k \lessdot m$, it is clear that the dominance order of Corollary 1 is a suborder of the adjacent interchange order $\lessdot$. Note that this suborder does not contain any pairs with *both* k and

s_2	s_1			
$(XkmYZ)$	$(XmYkZ)$	*proof*		
(x,k)	(x,k)		$m \in J_{a>b}$	$Y \subseteq J_{a>b}$
(x,m)	(x,m)		$k \in J_{a\leq b}$	
(x,y)	(x,y)		$k \in J_{a\leq b}$	
(x,z)	(x,z)			
(k,k)	(m,k)	$r_k \leq r_m$	$m \in J_{a>b}$	$Y \subseteq J_{a>b}$
(k,m)	(m,m)	$r_k \leq r_m$	$k \in J_{a\leq b}$	
(k,y)	(m,y)	$r_k \leq r_m$	$k \in J_{a\leq b}$	
(k,z)	(m,z)	$r_k \leq r_m$		
(m,m)	(m,m)			
(m,y)	(m,y)			
(m,z)	(m,z)			
(y,z)	(y,z)			

Table 3: Dominating pairs for Theorem 3.

$m \in J_{a>b}$. Our dominance order will enrich this suborder by extending it to pairs with $k, m \in J_{a>b}$ too.

Before we derive our dominance order, we introduce a planar representation of the adjacent interchange order $\lessdot$. We represent $\lessdot$ in the plane with the x and y axes replaced by the r and Johnson orders. A Job j is represented by the point (r_j, a_j) if $j \in J_{a\leq b}$ or (r_j, b_j) if $j \in J_{a>b}$. Then $k \lessdot m$ in this representation exactly if k is not to the right or above m. This is demonstrated for the 4-job problem of Example 1 in Figure 2. The planar representation here implies $2 \lessdot 4$, $3 \lessdot 1$ and $3 \lessdot 4$.

The principal ideals and filters in $\lessdot$ are obtained by dividing the plane into quadrants using the line $r = r_j$ and the line $a = a_j$ if $j \in J_{a\leq b}$ or $b = b_j$ for $j \in J_{a>b}$. Then the SW and NE quadrants correspond to $I(j)$ and $F(j)$, the principal ideal and filter in $\lessdot$ for job j (see Figure 3). We add new comparabilities for the jobs in $J_{a>b}$ using subset-restricted interchange. These new comparabilities are defined using certain 'extreme jobs' in the planar representation of $\lessdot$. These are the corner-point boundary jobs $M = \{M_i \,|i = 1, 2, \ldots, H+1\}$, i.e., the jobs with empty SE quadrants that form a descending staircase, and the set of jobs on their inscribed diagonal, the *diagonal jobs* $\Delta = \{D_1, D_2, \ldots, D_H\}$. Note that the diagonal jobs are not necessarily real jobs because of the way they are constructed, rather they may

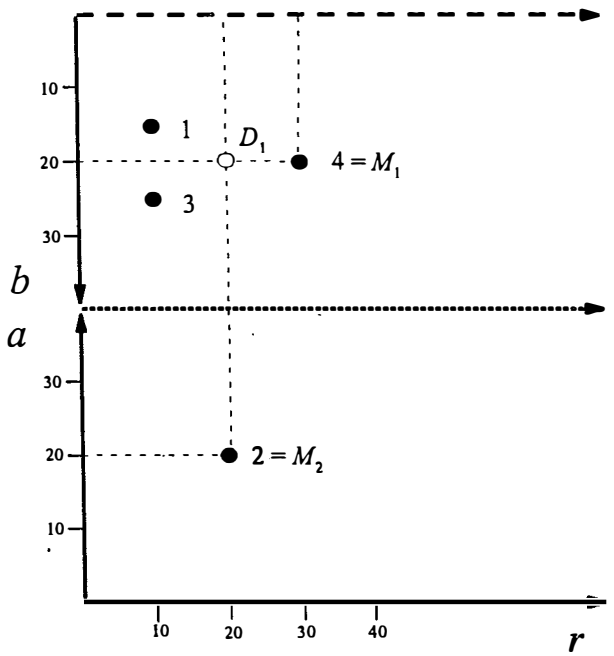

Figure 2: Planar representation for Example 1.

be 'artificial jobs' and are used only for ordering purposes. We augment the partial order $P = (J, \leqslant)$ by these diagonal jobs and call it $P_\Delta = (J_\Delta, \leqslant_\Delta)$, where $J_\Delta = J \cup \Delta$ and $\leqslant_\Delta$ is the planar order with these diagonal jobs included. Jobs $k \leqslant m$ ($k,m \in J$) are *separated by* Δ (are Δ-separated) if there exists a D_i ($i = 1, 2, \ldots, H$) such that k is in its principal ideal and m is in its principal filter, i.e., $k \in I(D_i)$ and $m \in F(D_i)$ in P_Δ. Δ induces a *partition* of P into separable and nonseparable pairs. Notice that the separable pairs can be of the three types in Definition 4, and are not restricted to pairs $k \leqslant m$ with $k \in J_{a \leq b}$. The new pairs we add are precisely the separable pairs in P. We now formally define our dominance order $\prec$.

Definition 5 *Dominance order $\prec$: For $k \leqslant m$, we define $k \prec m$ if*

(i) $k \in J_{a \leq b}$ *and* $a_k \leq a_m$ *or,*
(ii) *there exists a* D_i *in* P_Δ *such that* $k \in I(D_i)$ *and* $m \in F(D_i)$.

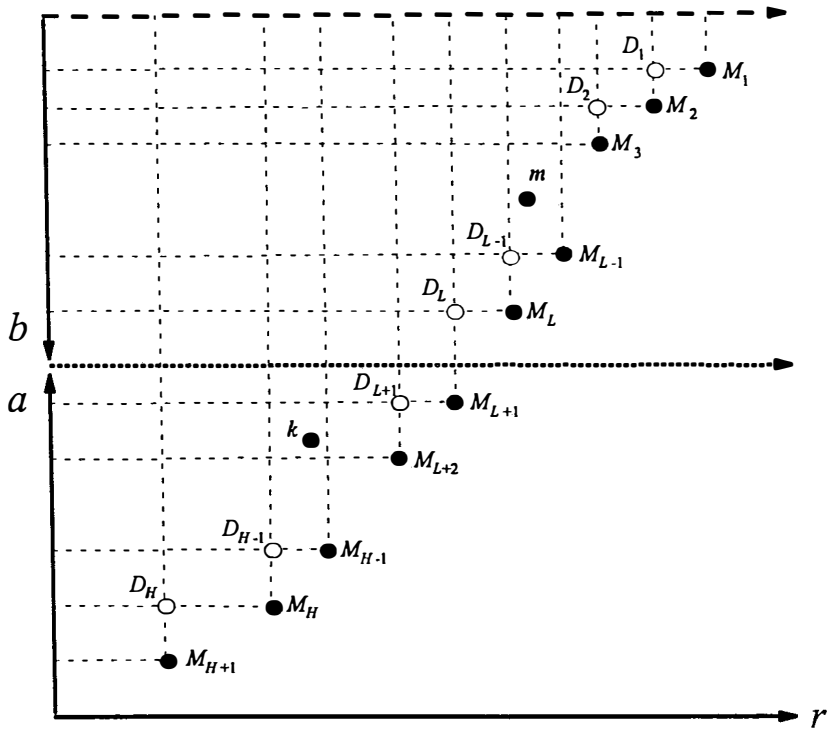

Figure 3: Representation of the diagonal jobs Δ.

To illustrate the definition of $\prec$ consider the planar representation for Example 1 in Figure 2. Applying condition (i) to pairs $k \lessdot m$ with $k \in J_{a \leq b}$ we see that $2 \lessdot 4$ is the only such pair and since $a_2 = 20 \leq 25 = a_4$ we have $2 \prec 4$. For condition (ii) here there are 2 corner-point boundary jobs $M_1 = 4$, $M_2 = 2$, and a single diagonal job D_1 in $J_{a>b}$ with coordinates $(20, 20)$. We see that D_1 separates jobs 2 and 3 from job 4, which gives us (again) $2 \prec 4$ together with the new pair $3 \prec 4$. Finally combining these types of pairs we get that, jobs 2 and 3 precede job 4 in $\prec$.

Theorem 4 *Partial order $\prec$ is a dominance order for $F2/r_j/C_{\max}$.*

Proof. Let s be an optimal sequence, we can assume by Theorem 1, that s is a linear extension of the suborder for condition (i) of the theorem. Thus we need to prove the theorem only for pairs satisfying condition (ii).

The following observations immediately follow from the construction of Δ:

$$\begin{array}{ccccccc} D_1 & \succcurlyeq_\Delta & D_2 & \succcurlyeq_\Delta & \cdots & \succcurlyeq_\Delta & D_H \\ F(D_1) & \subset & F(D_2) & \subset & \cdots & \subset & F(D_H) \\ I(D_1) & \supset & I(D_2) & \supset & \cdots & \supset & I(D_H). \end{array}$$

Furthermore, we have the following crucial property: for all $y \in J \setminus F(D_i)$ and $m \in F(D_i)$ we have $r_y \leq r_{D_i} \leq r_m$ for any i. Consider a pair k and m satisfying condition (ii). If both $k, m \in J_{a \leq b}$, then they also satisfy condition (i), and k is already before m in s. Next we deal with pairs with k and m satisfying condition (ii) and for which both $k, m \in J_{a>b}$.

In the following, the notation $S|_{J_{a>b}}$ is used to refer to $S \cap J_{a>b}$ for any $S \subseteq J$. If every job in $I(D_1)|_{J_{a>b}}$ is before every job in $F(D_1)$, then all jobs separated by D_1 in $J_{a>b}$ are already in $\prec$ order, and we consider $I(D_2)|_{J_{a>b}}$ and $F(D_2)$. Otherwise, let $k_1 \in I(D_1)|_{J_{a>b}}$ be the last job in s that is after some job from $F(D_1)$, and let m_1 be the last such job from $F(D_1)$ before k_1. That is $s = (X_1 m_1 Y_1 k_1 Z_1)$, where $Z_1 \cap I(D_1)|_{J_{a>b}} = \emptyset$ and $Y_1 \subset J \setminus F(D_1)$. By the above property, we have $r_{y_1} \leq r_{D_1} \leq r_{m_1}$ for all $y_1 \in Y_1$, which implies that $Y_1 \subseteq R^{BI}_{k_1 \lessdot m_1}$. Thus, by Theorem 2, we can insert m_1 *backward* just after k_1 to obtain the alternative optimal sequence $(X_1 Y_1 k_1 m_1 Z_1)$. Note that this interchange cannot violate condition (i), because $m_1 \in J_{a>b}$ and condition (i) could never require it to precede any job. Following in this way, inserting the last job in $F(D_1)$ *backward* after k_1 until there are no such jobs, we obtain sequence s_1 which is an optimal sequence with the property that $I(D_1)|_{J_{a>b}}$ is before $F(D_1)$. Continuing with a similar argument, we obtain s_i for $i = 2, 3, \ldots, L$, where D_L is the last diagonal job in $J_{a>b}$, and s_L is an optimal sequence satisfying condition (i) with the additional property that $I(D_i)|_{J_{a>b}}$ is before $F(D_i)$ for $i = 1, 2, \ldots, L$. This takes care of the pairs with k, $m \in J_{a>b}$.

Now we prove $\prec$ is a dominance order for the remaining pairs with $k \in J_{a \leq b}$ and $m \in J_{a>b}$. Let m_L be the first job in s_L from $F(D_L)$. If there are no jobs from $J_{a \leq b}$ after m_L, then s_L also satisfies the property that $I(D_i)$ is before $F(D_i)$ for $i = 1, 2, \ldots, L$. Otherwise, let $k_L \in J_{a \leq b}$ be the first such job after m_L. By the definition of D_L we have $k_L \lessdot m_L$. Then $s_L = (X_L m_L Y_L k_L Z_L)$ and by the choice of k_L we have that $Y_L \subset J_{a>b} = R^{FI}_{k_L \lessdot m_L}$. Thus, by Theorem 3, we can insert k_L *forward* just before m_L to obtain the sequence $(X_L k_L m_L Y_L Z_L)$. Note that this interchange does not violate condition (i), because $Y_L \subset J_{a>b}$ and condition

(i) would never require that k_L follow any job from Y_L. Repeating, until there is no such job k_L, we obtain the sequence $s_L^{'}$ satisfying condition (i) with the property that $I(D_i)$ is before $F(D_i)$ for $i = 1, 2, \ldots, L$. We want to demonstrate that $I(D_i)$ is before $F(D_i)$ for $i = L+1, \ldots, H$. Let k_{L+1} be the last job in $I(D_{L+1})$ that is after some job in $F(D_{L+1})|_{J_{a>b}}$, and let m_{L+1} be the last such job before k_{L+1}. Then $s_L^{'} = (X_{L+1} m_{L+1} Y_{L+1} k_{L+1} Z_{L+1})$ where $Z_{L+1} \cap I(D_{L+1}) = \emptyset$ and $Y_{L+1} \subset J \setminus F(D_{L+1})$. Note that Y_{L+1} may contain jobs from both parts of the Johnson partition. Let y_i be the first job in $J_{a \leq b} \cap Y_{L+1}$ after m_{L+1}. Then we have by our crucial property that $r_{y_i} \leq r_{D_{L+1}} \leq r_{m_{L+1}}$, and since $y_i \in J_{a \leq b}$ and $m_{L+1} \in J_{a>b}$ that $y_i \lessdot m_{L+1}$. Furthermore, all the jobs between m_{L+1} and y_i are in $J_{a>b} = R^{FI}_{y_i \lessdot m_{L+1}}$, and thus by Theorem 3, we can insert y_i *forward* before m_{L+1}. By repeatedly doing this, we end up with a sequence where there are only jobs from $J_{a>b}$ between m_{L+1} and k_{L+1}. Finally, again using Theorem 3, we can insert k_{L+1} *forward* before m_{L+1} to obtain a sequence with k_{L+1} before m_{L+1}. As for the previous case, these interchanges do not violate condition (i). We continue until there is no such k_{L+1} and m_{L+1} and then we obtain the sequence $s_{L+1}^{'}$ that satisfies the property that $I(D_i)$ is before $F(D_i)$ for $i = 1, 2, \ldots, L+1$. We proceed in exactly the same way for the remaining cases for $L+1 < j \leq H$, and we end up with a sequence $s_H^{'}$ satisfying both conditions (i) and (ii). ■

4 Branch and Bound

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

4.1 Branching rule

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 $i = 1, 2$ and let S be the set of unfixed jobs i.e., $S = J \setminus (S_1 \cup S_2)$. 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. We consider two rules to break ties between nodes with the same lower bound. We assume that the jobs are indexed in increasing Johnson order. The rule T_2 breaks ties for type 1 or type 2 branchings by choosing the job with the smallest or largest index, respectively. The other rule T_1 breaks ties for type 1 branchings by choosing the job with the largest principal filter in $\prec|_S$, the smallest index is used as a further tie-breaker. Similarly, for type 2 branchings the rule is to break ties by choosing the job with the largest principal ideal in $\prec|_S$, with the largest index as a further tie-breaker.

4.2 Bounds

Upper bounds are calculated for the root node and for the first n nodes, afterward the upper bound is evaluated only at leaf nodes. At the root, the upper bound is the result of the improved ready Johnson heuristic due to Potts [12]. This heuristic has time complexity $O(n^3 \log n)$ and a worst case performance ratio of 5/3. For the first n nodes, the upper bound is the length of the sequence obtained by concatenating the ready Johnson sequence between σ_1 and σ_2, which requires $O(n \log n)$ time. For leaf nodes, there are no unfixed jobs 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 huge number of nodes, it is very important that we use lower bounds whose calculation requires 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 four 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 (with release times) and we let $C^1(\sigma_1)$ and $C^2(\sigma_1)$ be the completion times on machines 1 and 2, respectively. For σ_2 we consider the reverse problem and define the completion times $C_r^1(\sigma_2)$ and $C_r^2(\sigma_2)$ analogously. We also define $C_{\max}^r(\sigma_2)$ to be the delivery completion time for σ_2 with release times treated like delivery times. 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 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 this length $L_3(S)$ is a lower bound on when the jobs in S can complete on machine 2. Our lower bounds are the following:

$LB_1 = L_1(S) + C_r^2(\sigma_2)$
$LB_2 = L_2(S) + C_r^1(\sigma_2)$
$LB_3 = L_2(S) + \min_{i \in S} b_i + C_r^2(\sigma_2)$
$LB_4 = L_3(S) + C_r^2(\sigma_2).$

Finally, the best lower bound is the maximum of the above four lower bounds and $C_{\max}^r(\sigma_2)$.

4.3 Decomposition and dominance

We find a starting sequence σ_1 by applying the following simple decomposition procedure, which was also used in [13]. Given a sequence s, if there exists a $2 \leq k \leq n$ such that $\min_{k \leq i \leq n} r_{s(i)} \geq C_{s(k-1)}^1$ and $\min_{k \leq i \leq n} [r_{s(i)} + a_{s(i)}] \geq C_{s(k-1)}^2$, then sequence $(s(1), \ldots, s(k-1))$ is an optimal initial sequence. We apply the decomposition procedure for the jobs sequenced in nondecreasing r order, then $\sigma_1 = (s(1), \ldots, s(k-1))$ for the largest k value found. After we have fixed σ_1, we determine the dominance order $\prec$ on the remaining jobs in $S = J \backslash S_1$. Note that the dominance order is calculated only once at the root node. This eliminates the potentially large overhead of having to update dynamically changing dominance conditions.

5 Computational experiment

5.1 Test problems

For each problem with n jobs $3n$ integer data (r_i, a_i, b_i) were generated. The processing times a_i and b_i were both uniformly distributed between $[1, 100]$. Release times r_i were uniformly distributed in the range $[0, n \cdot 101 \cdot R]$ for $R \in \{0.2, 0.4, 0.5, 0.6, 0.8, 1.0\}$, following the technique used by Hariri and Potts [3]. For each R and n combination, 50 problems were generated. We used the random number generator of Taillard [14] to generate these problems. This means that all 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 workstation. Three separate versions of the algorithm were run, testing the effectiveness of the dominance order and the different tie-breakers T_1 and T_2. Algorithm A_1 has the dominance order $\prec$ 'turned on' and uses tie-breaker T_1. Algorithm A_2 also has the dominance order 'turned on', but it uses T_2 to break ties. Finally, algorithm A_3 has the dominance order 'turned off' with tie-breaker T_2. 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.

Table 4 contains the results of the computational experiment for the different R values. For each group of 50 problems we report: the total CPU time required for the 50 problems (denoted by *total CPU*); the average CPU time for the solved problems in the group (denoted by *avg CPU*); the total number of nodes in all of the trees (denoted by *total nodes*); the number of problems solved (denoted by *solved*); and 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, the maximum gap (denoted *max gap*) is the largest of these over all the problems.

Our results indicate that the difficulty in solving a problem depends *much more* on the

value of R than on the number of jobs. The actual size of the problem for the branch and bound algorithm is determined by how many jobs are fixed by the decomposition procedure. In general, as R increases the percentage of the jobs fixed increases. Recall, in the model $n \cdot 101$ is the expected total processing time of the jobs, and the range of the release times is $[0, n \cdot 101 \cdot R]$. For small R values ($R < 0.4$), the decomposition procedure is ineffective and fixes less than 2 percent of the jobs. However, since all the jobs are ready relatively early, release times play less of a role and the problem is easily solved in just a few nodes. For large R values ($R > 0.6$), the opposite is true, the jobs do not become ready simultaneously, there may be gaps in the schedule, and the decomposition procedure fixes over 90 percent of the jobs on average. The difficult problems are those with R values concentrated close to $R = 0.5$, in the range $R \in [0.4, 0.6]$, with the most difficult being $R = 0.5$ itself. For $R = 0.4$ the decomposition procedure fixes only 3 percent of the jobs, while for $R = 0.6$ it fixes a much larger 70 percent of the jobs on average. In spite of this large difference in the effectiveness of the decomposition procedure for $R = 0.4$ and 0.6, we are still able to solve at least 90 percent of these problems, even those with 500 jobs. Whereas, for $R = 0.5$ the decomposition procedure fixes around 20 percent of the jobs, and the problems appear to be more difficult with a clear decreasing trend in the number of problems solved as n increases.

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 algorithm typically generated much fewer nodes for the solved problems. In those groups where there were unsolved problems, the solution obtained was nearly optimal as measured by the maximum gap. The largest such gap was on the order of 1 to 2 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.

Comparing the different versions of the algorithm, we found that versions A_1 and A_2 (with the dominance order) always solved as many problems as version A_3 (without the dominance order), and in some cases a few more. The main benefit of using the dominance order is the substantial reduction in the CPU time required to solve a group of problems. This reduction in CPU time is approximately 50 percent over all the groups, for some individual groups, however, it is substantially more. Consider for example the group of problems with $R = 0.4$

and $n = 200$: here A_1 and A_2 both took approximately 37 minutes to complete, while A_3 took over 12.5 hours to do so. We also found that this speed-up factor tends to increase as n increases. This explains why we did not run algorithm A_3 for groups with $n = 500$ jobs, as it would have required excessively long times. The use of the different tie-breaking rules in A_1 and A_2 did not result in any clearly identifiable differences in the relative performance.

6 Concluding remarks

This paper considered a new branch and bound algorithm for the problem $F2\,/r_j/\,C_{\max}$. The main features of the proposed algorithm are: an adaptive branching rule that fixes jobs at both ends of the schedule, and a new dominance order that substantially reduces branching. In general, computational results indicated that the algorithm performed well in solving a large percentage of test problems, some with up to 500 jobs. We found that the difficult problems were those where the range of release times was approximately 1/2 the expected total processing time of the jobs (i.e., $R = 0.5$ in our model). Even when we failed to solve a problem, we were always very close to the optimal solution. We also found that the use of the dominance order resulted in a reduction in computation time of roughly 50 percent, but in some cases the reduction was substantially more than this. In summary, we feel that the proposed algorithm is relatively more effective than previous solution methods for the problem $F2\,/r_j/\,C_{\max}$, in that it found fewer hard problems and it was also able to solve much larger problems than previous algorithms. 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 $F2\,/r_j/\,C_{\max}$-type subproblems generated during the solution of more complex scheduling problems, for example problem $Fm\,/r_j/\,C_{\max}$. This and the extendability of the dominance order to the problem $F2\,/r_j/\,L_{\max}$ will be investigated in future research.

Table 4: Results of the computational experiment

n	*alg*	*total CPU* (hrs:min:sec)	*avg CPU* (min:sec)	*total nodes*	*solved*	*max gap* (%)
$R = 0.2$						
	A_1	0.28	0.0056	107	50	-
40	A_2	0.24	0.0048	107	50	-
	A_3	0.24	0.0048	107	50	-
	A_1	1.28	0.0256	1787	50	-
60	A_2	1.25	0.0250	1787	50	-
	A_3	2.76	0.0552	1789	50	-
	A_1	1.08	0.0216	89	50	-
80	A_2	1.09	0.0218	89	50	-
	A_3	0.61	0.0122	89	50	-
	A_1	1.86	0.0372	111	50	-
100	A_2	1.83	0.0366	111	50	-
	A_3	1.02	0.0204	129	50	-
	A_1	11.95	0.2390	218	50	-
200	A_2	12.13	0.2426	218	50	-
	A_3	5.37	0.1074	218	50	-
	A_1	5:54.83	7.0966	1384	50	-
500	A_2	4:41.84	5.6368	1384	50	-
$R = 0.4$						
	A_1	1:28.11	0.0050	1,001,861	49	0.14
40	A_2	1:04.16	0.0046	1,001,861	49	0.38
	A_3	3:21.83	0.0223	1,002,050	49	0.24
	A_1	6:36.66	0.4769	2,087,118	48	0.74
60	A_2	6:32.20	0.4757	2,087,118	48	0.74
	A_3	43:24.47	21.8289	2,857,484	48	0.74
	A_1	24:25.99	0.7544	3,098,632	47	0.70
80	A_2	23:09.80	0.7507	3,098,632	47	0.70
	A_3	1:51:02.56	6.1055	3,118,415	47	0.70
	A_1	5:44.35	0.0888	1,005,725	49	0.35
100	A_2	5:37.48	0.0910	1,005,725	49	0.29
	A_3	38:30.42	0.6191	1,007,805	49	0.29
	A_1	37:17.79	0.7253	3,018,471	47	0.39
200	A_2	37:01.58	1.0868	3,018,471	47	0.39
	A_3	12:31:32.71	17.2627	3,050,546	47	0.39
	A_1	11:23:54.14	7.0742	5,005,152	45	0.30
500	A_2	10:07:29.49	5.4205	5,005,152	45	0.30

Table 4: (continued)

n	*alg*	*total CPU* (hrs:min:sec)	*avg CPU* (min:sec)	*total nodes*	*solved*	*max gap* (%)
$R = 0.5$						
	A_1	3:30.76	0.0853	2,027,081	48	0.62
40	A_2	5:59.66	0.0926	2,029,231	48	0.62
	A_3	10:08.70	0.4476	2,051,022	48	0.62
	A_1	18:15.17	0.7275	5,189,188	45	1.37
60	A_2	20:24.29	0.7349	5,189,000	45	1.55
	A_3	38:43.69	13.2302	5,733,770	45	1.37
	A_1	1:14:10.25	1.5020	8,211,107	42	0.79
80	A_2	1:18:09.24	1.4863	8,210,998	42	0.79
	A_3	2:36:10.73	8.3622	8,438,701	42	0.79
	A_1	1:59:06.36	5.3600	9,540,374	41	1.26
100	A_2	1:52:27.48	5.3471	9,540,357	41	1.26
	A_3	4:19:29.52	8.3293	10,143,663	40	1.26
	A_1	3:33:16.72	1.3308	13,019,825	37	0.97
200	A_2	6:15:50.78	1.2841	13,020,073	37	0.97
	A_3	26:17:52.63	4:46.57	13,639,751	37	0.97
	A_1	17:42:05.92	7.0524	14,019,667	36	0.24
500	A_2	15:52:55.39	5.9951	14,019,667	36	0.24
$R = 0.6$						
	A_1	2:59.01	0.0608	1,017,391	49	2.71
40	A_2	2:51.86	0.0594	1,017,379	49	2.31
	A_3	7:38.57	0.5191	1,283,222	49	2.71
	A_1	13:23.16	0.0030	2,001,035	48	0.26
60	A_2	13:19.02	0.0022	2,000,982	48	0.26
	A_3	23:51.37	0.1535	2,001,383	48	0.24
	A_1	16:06.71	1.6158	3,548,901	47	1.30
80	A_2	19:42.08	1.5206	4,496,662	46	1.13
	A_3	42:25.34	5.6657	5,154,014	45	1.47
	A_1	24:59.50	0.1235	4,029,291	46	0.50
100	A_2	24:48.82	0.1193	4,029,291	46	0.50
	A_3	58:36.76	0.4002	5,071,945	45	0.50
	A_1	11:26.63	0.3044	2,082,825	48	0.49
200	A_2	13:44.03	0.0985	3,028,215	47	0.49
	A_3	21:10.28	0.8350	2,139,389	48	0.49
	A_1	3:16:36.69	4.1353	5,066,898	45	0.19
500	A_2	3:14:37.67	7.1349	5,066,898	45	0.19

Table 4: (continued)

n	*alg*	*total CPU* (hrs:min:sec)	*avg CPU* (min:sec)	*total nodes*	*solved*	*max gap* (%)
$R = 0.8$						
	A_1	0.12	0.0024	216	50	-
40	A_2	0.10	0.0020	216	50	-
	A_3	0.13	0.0026	216	50	-
	A_1	0.13	0.0026	284	50	-
60	A_2	0.15	0.0030	284	50	-
	A_3	0.17	0.0034	342	50	-
	A_1	0.19	0.0038	218	50	-
80	A_2	0.20	0.0040	218	50	-
	A_3	0.22	0.0044	232	50	-
	A_1	0.25	0.0050	502	50	-
100	A_2	0.29	0.0058	502	50	-
	A_3	0.33	0.0060	746	50	-
	A_1	4:09.77	0.0086	1,000,613	49	0.22
200	A_2	3:58.20	0.0119	1,000,607	49	0.22
	A_3	6:49.31	0.0138	1,001,398	49	0.22
	A_1	6.34	0.1268	11,683	50	-
500	A_2	4.46	0.0892	6,916	50	-
$R = 1.0$						
	A_1	0.07	0.0014	91	50	-
40	A_2	0.10	0.0020	91	50	-
	A_3	0.07	0.0014	91	50	-
	A_1	0.12	0.0024	102	50	-
60	A_2	0.11	0.0022	102	50	-
	A_3	0.12	0.0024	102	50	-
	A_1	0.17	0.0034	169	50	-
80	A_2	0.17	0.0034	169	50	-
	A_3	0.20	0.0040	189	50	-
	A_1	0.19	0.0038	88	50	-
100	A_2	0.21	0.0042	87	50	-
	A_3	0.22	0.0044	101	50	-
	A_1	0.55	0.0110	82	50	-
200	A_2	0.55	0.0110	82	50	-
	A_3	0.56	0.0112	89	50	-
	A_1	2.35	0.0470	81	50	-
500	A_2	2.32	0.0464	81	50	-

References

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

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

[3] 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.

[4] Hall, L.A. (1994). A polynomial approximation scheme for a constrained flow shop scheduling problem. *Math. Opns. Res.* 19, 68-85.

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

[6] Kovalyov, M.Y., and F. Werner (1997). A polynomial approximation scheme for problem $F2\,/r_j/\,C_{\max}$. *Oper. Res. Lett.* 20, 75-79.

[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] Potts, C.N. (1985). Analysis of heuristics for two-machine flow-shop sequencing subject to release dates. *Math. Opns. Res.* 10, 576-584.

[13] 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.

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

Appendix: Branch and bound algorithm

Calculations for root node

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

procedure $Opt(\sigma_1, S, \sigma_2, BestMakespan)$;
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 $BestMakespan$

if $(LB < BestMakespan)$ **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 order

else if *type* $= 2$ **then** calculate lower bounds for *maximal* and sort in nondecreasing order

else (*type* not set)

begin

must calculate and sort lower bounds for both *minimal* and *maximal* to 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* $\neq \emptyset$) **and** ($BestMakespan >$ 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 $\prec$

$S := S \backslash \{job\}$;

if *type* $= 1$ **then** add *job* to σ_1

else if *type* $= 2$ **then** add *job* to σ_2;

$Opt(\sigma_1, S, \sigma_2, BestMakespan)$;{recursive call}

if *type* $= 1$ **then** remove *job* from σ_1

else if *type* $= 2$ **then** remove *job* from σ_2;

$S := S \cup \{job\}$;

restore job to $\prec$

end;{while}

end;{if $(LB < BestMakespan)$ and $(|S| \neq 1)$}

end;{if $nodes \leq 1000000$}

end;

Faculty of Business
McMaster University

WORKING PAPERS - RECENT RELEASES

410. Jiang Chen and George Steiner, "Approximation Methods for Discrete Lot Streamingin Flow Shops", June, 1995.

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.