Engineering Optimization
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/geno20

Two-machine flow-shop scheduling to minimize total late work

B. M. T. Lin [a], F. C. Lin [b] & R. C. T. Lee [c]

[a] Department of Information and Finance Management, Institute of Information Management, National Chiao Tung University, Taiwan, 300, R.O.C

[b] Department of Computer Science and Information Engineering, National Chi Nan University, Taiwan, 545, R.O.C

[c] Department of Information Management, National Chi Nan University, Taiwan, 545, R.O.C

Published online: 25 Jan 2007.

To cite this article: B. M. T. Lin, F. C. Lin & R. C. T. Lee (2006) Two-machine flow-shop scheduling to minimize total late work, Engineering Optimization, 38:04, 501-509, DOI: 10.1080/03052150500420439

To link to this article: http://dx.doi.org/10.1080/03052150500420439

Engineering Optimization
Vol. 38, No. 4, June 2006, 501–509

Taylor & Francis
Taylor & Francis Group

Two-machine flow-shop scheduling to minimize total late work

B. M. T. LIN*†, F. C. LIN‡ and R. C. T. LEE§

†Department of Information and Finance Management, Institute of Information Management, National Chiao Tung University, Taiwan 300, R.O.C.
‡Department of Computer Science and Information Engineering, National Chi Nan University, Taiwan 545, R.O.C.
§Department of Information Management, National Chi Nan University, Taiwan 545, R.O.C.

(Received 31 January 2005; revised 8 April 2005; in final form 28 April 2005)

This article considers a two-machine flow-shop scheduling problem of minimizing total late work. Unlike tardiness, which is based upon the difference between the job completion time and the due date, the late work of a job is defined as the amount of work not completed by its due date. This article first shows that the problem remains non-deterministic polynomial time (NP) hard even if all jobs share a common due date. A lower bound and a dominance property are developed to design branch-and-bound algorithms. Computational experiments are conducted to assess the performance of the proposed algorithms. Numerical results demonstrate that the lower bound and dominance rule can help to reduce the computational efforts required by exploring the enumeration tree. The average deviation between the solution found by tabu search and the proposed lower bound is less than 3%, suggesting that the proposed lower bound is close to the optimal solution.

Keywords: Late work; Flow shop; Complexity; Branch-and-bound algorithm

1. Introduction

Together with the rapid growth of applications of scheduling theory, new manufacturing models as well as measures have been proposed and studied in the open literature (Pinedo and Chao 1999). This article studies a scheduling problem of minimizing a new penalty measure, namely the total penalty for unfinished parts of the jobs. Consider a set of jobs $N = \{1, 2, \ldots, n\}$ available from time zero onwards for processing in a two-machine flow shop, which was first motivated by the study by Johnson (1954). Each job must be first processed on machine one and then transferred to machine two for second-stage processing. Operations on both machines are processed in the same sequence; *i.e.* only permutation schedules are considered. The processing time of job i on machine k ($k = 1$ or 2) is denoted by $p_{i,k}$. A specific due date d_i is assigned to each individual job i to define the time before which it is expected to

*Corresponding author. Email: bmtlin@mail.nctu.edu.tw

Engineering Optimization
ISSN 0305-215X print/ISSN 1029-0273 online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/03052150500420439

be completed. If a job cannot be completely finished by its due date, then it is called *late* and a penalty proportional to the amount of remaining unfinished part will be incurred. Let $C_{i,1}$ and $C_{i,2}$ be the completion times of operation one and operation two of job i on the two machines respectively. The *late work* of job i is defined by $Y_i = Y_{i,1} + Y_{i,2} = \min\{\max\{C_{i,1} - d_i, 0\}, p_{i,1}\} + \min\{\max\{C_{i,2} - d_i, 0\}, p_{i,2}\}$. If the two operations of a job are completed by its due date, then no penalty is incurred; otherwise, it will be penalized in proportion to the amount of unfinished parts of operation one and operation two. The goal of the studied problem seeks to construct a job sequence such that the sum of late work penalties over all jobs is minimum. Following the standard three-field notation (Graham *et al.* 1979), this article uses $F2||\sum Y_i$ to denote the studied problem, where $F2$ dictates the two-machine flow shop and the third field specifies the objective function to minimize.

The late-work concept was first introduced by Blazewicz (1984) in a parallel-machine production environment. The total late-work measure has practical significance in such real-world applications as data collection (Blazewicz 1984, Blazewicz and Finke 1987) and corn harvesting (Potts and Van Wassenhove 1992a, 1992b). The study was later extended to single-machine (Hariri *et al.* 1995, Kethley and Alidaee 2002, Potts and Van Wassenhove 1992a, 1992b), flow-shop (Blazewicz *et al.* 2005) and open-shop environments (Blazewicz *et al.* 2004). The known results concerning total late-work criteria are summarized in table 1. The research studies that are most relevant to this study are by Blazewicz *et al.* (2005), Hariri *et al.* (1995), Lin and Hsu (2005) and Potts and Van Wassenhove (1992a). Hariri *et al.* (1995) considered the single-machine environment with the objective of minimizing total weighted late work. They presented a proof to show that $1||\sum w_i Y_i$ is non-deterministic polynomial time (NP) hard and developed a pseudo-polynomial time dynamic programming algorithm. Lin and Hsu (2005) considered $1|r_i|\sum Y_i$, in which release dates are introduced. They showed that $1|r_i, d_i = d|\sum Y_i$ and $1|r_i, pmtn|\sum Y_i$ are polynomially solvable and developed a branch-and-bound algorithm for the general case. Study on late-work criteria in flow shops first appeared in an article by Blazewicz *et al.* (2005) so as to minimize weighted late work. A reduction from PARTITION was given to establish the NP-hardness of $F2|d_i = d|\sum w_i Y_i$. They also developed a pseudo-polynomial time dynamic program. This article shows that the unweighted case $F2|d_i = d|\sum Y_i$ remains NP-hard and uses the results obtained by Hariri *et al.* (1995) and Lin and Hsu (2005) to develop a branch-and-bound algorithm for $F2||\sum Y_i$.

Table 1. Known results about the total late-work criterion.

Problem	Complexity	Source
$1\|\|\Sigma Y_i$	NP-hard	Potts and Wassenhove (1992a)
$1\|pmtn\|\Sigma Y_i$	$O(n \log n)$	Potts and Wassenhove (1992a)
$1\|d_i = d\|\Sigma Y_i$	$O(n)$	Potts and Wassenhove (1992a)
$1\|p_i = p\|\Sigma Y_i$	$O(n \log n)$	Potts and Wassenhove (1992a)
$1\|pmtn\|\Sigma w_i Y_i$	$O(n \log n)$	Hariri *et al.* (1995)
$1\|d_i = d\|\Sigma w_i Y_i$	$O(n)$	Hariri *et al.* (1995)
$1\|p_i = p\|\Sigma w_i Y_i$	$O(n^3)$	Hariri *et al.* (1995)
$1\|r_i, d_i = d\|\Sigma Y_i$	$O(n \log n)$	Lin and Hsu (2005)
$1\|r_i, pmtn\|\Sigma Y_i$	$O(n \log n)$	Lin and Hsu (2005)
$P\|r_i\|\Sigma w_i Y_i$	NP-hard	Blazewicz (1984)
$P\|r_i, pmtn\|\Sigma w_i Y_i$	$O(n^7 \log n)$	Blazewicz and Finke (1987)
$Pk\|r_i, pmtn\|\Sigma w_i Y_i$	$O(k^3 n^7 \log kn)$	Blazewicz and Finke (1987)
$F2\|d_i = d\|\Sigma w_i Y_i$	NP-hard	Blazewicz *et al.* (2005)
$O\|r_i, pmtn\|\Sigma w_i Y_i$	$O(n^3)$	Blazewicz *et al.* (2004)
$O2\|d_i = d\|\Sigma Y_i$	$O(n)$	Blazewicz *et al.* (2004)
$O2\|d_i = d\|\Sigma w_i Y_i$	NP-hard	Blazewicz *et al.* (2004)

The rest of this article is organized as follows. Section 2 is dedicated to the NP-hardness of a special case where a common due date is assumed. Section 3 introduces a lower bound and a dominance property to develop branch-and-bound algorithms. In section 4, computational experiments are conducted to assess the effectiveness of the proposed properties. Finally, some concluding remarks are given in section 5.

2. Non-deterministic polynomial time hardness result

This section mainly includes the proof that the studied problem remains NP-hard even if there is only one due date common to all jobs. Denote this special case by $F2|d_i = d|\sum Y_i$. As mentioned above, the weighted case $F2|d_i = d|\sum w_i Y_i$ problem was shown to be NP-hard by Blazewicz *et al.* (2005). The reduction in this article is also based upon PARTITION, which is NP-complete in the ordinary sense (Garey and Johnson 1979).

PARTITION: Given an integer B and a set of A of s positive integers $\{x_1, x_2, \ldots, x_s\}$ such that $\sum_{i=1}^{s} x_i = 2B$, does there exist a partition A_1 and A_2 of set A such that $\sum_{x_i \in A_1} x_i = \sum_{x_i \in A_2} x_i = B$?

THEOREM 1 *The decision version of the $F2|d_i = d|\sum Y_i$ problem is NP-complete.*

Proof It is clear that the decision counterpart of $F2|d_i = d|\sum Y_i$ belongs to NP. Given an instance of PARTITION, an instance of the $F2|d_i = d|\sum Y_i$ problem consisting of $n = s + 1$ jobs is constructed as follows:

ordinary jobs: $p_{i,1} = x_i\omega$, $p_{i,2} = x_i$, $1 \leqslant i \leqslant s$, ω is an integer greater than B, say $\omega = B + 1$;
enforcer job: $p_{s+1,1} = 0$, $p_{s+1,2} = \omega B$;
common due date $d = \omega B + B$.

The following will establish the claim that there is a specified partition for PARTITION if and only if there is a schedule for $F2|d_i = d|\sum Y_i$ whose total late work is exactly ωB. ■

$\Rightarrow$) Let subsets A_1 and A_2 be a partition as specified for set A. Schedule enforcer job $s + 1$ as the first job in the flow shop. The jobs corresponding to the elements of A_1 then follow in arbitrary order. The remaining jobs are scheduled arbitrarily at the rear part of the schedule. See figure 1 for an illustration. Note that the late machine-two operations can be postponed and processed consecutively at the rear part of the schedule without sacrificing the solution quality. In the figure, each late machine-two operation is processed immediately on completion of its corresponding machine-one operation so as to emphasize the sequencing order. It is not difficult to verify that the enforcer job and the jobs corresponding to the elements of A_1 are all early. Moreover, the first of the remaining jobs has an early part of length B

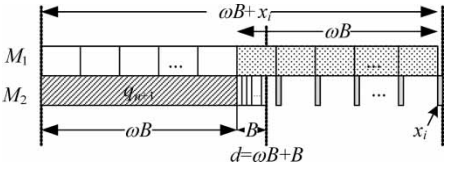

Figure 1. Gantt chart of the schedule corresponding to the specified partition.

on machine-one. Therefore, the total late work of the remaining jobs on both machines is $\sum_{x_i \in A_2} x_i\omega - B + \sum_{x_i \in A_2} x_i = \omega B - B + B = \omega B$.

$\Leftarrow$) Suppose that there is a schedule for $F2|d_i = d|\sum Y_i$ whose total late work is no more than B. Note that the total processing length of all jobs on the two machines is $2\omega B + \omega B + 2B$. Because $d = \omega B + B$, to ensure that the total late work is no greater than B, on both machines the time span before the due date must be fully filled; *i.e.* no idle time is allowed on either machine before the due date. This observation leads to the fact that the enforcer job must be scheduled first. Let N_1 be the set of ordinary jobs that are completely early on machine-one. First the situation when $\sum_{i \in N_1} x_i$ is strictly smaller than B is considered. In this case, the completion time on machine-two of the last job in N_1 is strictly earlier than $\omega B + B$. Because on machine-one the processing of the immediate successor following N_1 is not completed by or at the due date $d = \omega B + B$, a non-zero idle time will be introduced into its machine-two processing before the due date. As a sequel, $\sum_{i \in N_1} x_i \geqslant B$ must hold. On the other hand, if $\sum_{i \in N_1} x_i > B$, then $\sum_{i \in N_1} x_i \geqslant B + 1$. Therefore, on machine-one the largest completion time of the jobs in N_1 is $\sum_{i \in N_1} p_{i,1} = \sum_{i \in N_1} \omega x_i \geqslant \omega B + \omega$, which is greater than $d = \omega B + B$. It is a contraction to the assumption that all jobs of N_1 are completely early on machine-one. Therefore, $\sum_{i \in N_1} x_i$ must equal B exactly, and thus a desired partition is attained. From the above reduction, the proof is readily complete.

The result implies that it is very unlikely to design polynomial time algorithms for producing optimal solutions to $F2|d_i = d|\sum Y_i$. Because Blacewicz *et al.* (2005) have proposed a pseudo-polynomial algorithm for the weighted version $F2|d_i = d|\sum w_i Y_i$, the $F2|d_i = d|\sum Y_i$ problem is ordinary NP-hard. In the next section, some useful properties will be developed and deployed to reduce the solution-finding time.

3. Branch-and-bound algorithm

This section is dedicated to the development of a branch-and-bound algorithm, which is one of the most commonly adopted approaches to tackling hard combinatorial optimization problems. The efficiency of a branch-and-bound algorithm mainly depends on such structural properties as lower bounds and dominance rules, which can help to trim off unnecessary branches in the enumeration tree that corresponds to the solution space. In most of the objective functions for flow-shop scheduling problems, the objective value is dependent on the completion of machine-two operations. For the total late-work criterion, it is, however, necessary to consider the performance measure on both machines. Therefore, it is viable to find a lower bound for the sequence of operations on each machine and then to combine two lower bounds into an aggregate lower bound for the original problem. In addition to lower bounds, a dominance rule will also be used to reduce the time required by exploring the solution space.

To develop a lower bound, the machine-two operations are ignored and only the operations on the first machine are considered. The problem then becomes the single-machine problem $1||\sum Y_i$, using $\{p_{i,1}, p_{i,2}, \ldots, p_{i,n}\}$. Although the single-machine $1||\sum Y_i$ problem is still NP-hard, Potts and Wassenhove (1992a) proposed a pseudo-polynomial dynamic programming algorithm to solve it optimally. Therefore, given a partial schedule σ for the original problem, the machine-one operations of the unscheduled jobs can be collected and the single-machine case is solved by appealing to this dynamic program.

The discussion now proceeds to when the variant involves only the machine-two operations. Given a partial schedule σ of the original $F2||\sum Y_i$ problem, let $t_1(\sigma)$ and $t_2(\sigma)$ be the completion times of the last job scheduled in σ on machine-one and machine-two respectively. Note that any machine-two operation of job $i \in N \backslash \{\sigma\}$ cannot start its processing earlier than

$\max\{t_1(\sigma) + p_{i,1}, t_2(\sigma)\}$. Following this line of reasoning, $\max\{t_1(\sigma) + p_{i,1}, t_2(\sigma)\}$ may be treated as the release time for machine-two operation of unscheduled job i. As a result, this becomes the single-machine schedule problem of the machine-two operations of the remaining unscheduled jobs under release-date constraints, $1|r_i|\sum Y_i$. Similarly, this problem remains NP-hard. Lin and Hsu (2005) presented an $O(n \log n)$ algorithm for the case $1|pmtn, r_i|\sum Y_i$, where preemption is allowed. The optimal solution value of $1|pmtn, r_i|\sum Y_i$ is a lower bound on the optimal solution value $1|r_i|\sum Y_i$; *i.e.* there is a lower bound for the problem involving only the machine-two operations. In an optimal solution to the original $F2||\sum Y_i$ problem, the total late work contributed by the machine-one operations cannot be smaller than the solution derived using the dynamic program, and similarly the total late work contributed by the machine-two operations cannot be smaller than the solution derived by using the preemption relaxation. Therefore, by combining these two values, an aggregate lower bound on the optimal solution value of $F2||\sum Y_i$ can be obtained.

To cut off unnecessary branches further, the relationship between any two jobs that are arranged consecutively is examined. For partial schedule σ and two unscheduled jobs i and j, we consider two (partial) schedules $\sigma' = \sigma ij$ and $\sigma'' = \sigma ji$. If

(a) $C_{j,2}(\sigma') \leqslant C_{i,2}(\sigma'')$ and
(b) $Y_i(\sigma') + Y_j(\sigma') \leqslant Y_j(\sigma'') + Y_i(\sigma'')$,

then

(i) σ' and σ'' have the same completion time on machine-one,
(ii) on machine-two, the completion time of σ' is no later than that of σ'' and
(iii) the total late work of σ' is no greater than that of σ''.

The three consequence parts jointly imply that the best solution value fulfilled from σ'' cannot be smaller than that fulfilled from σ'; *i.e.* the branch corresponding to partial schedule σ'' is fathomed. If conditions (a) and (b) are both satisfied with equalities, then in implementation it is possible that σ'' and σ' will trim off each other. Therefore, another condition

(c) $i < j$

is added.

The discussion is summarized in the following property.

Dominance property: Let $\sigma' = \sigma ij$ and $\sigma'' = \sigma ji$ be defined from the partial schedule σ and two unscheduled jobs i and j. If

(a) $C_{j,2}(\sigma') \leqslant C_{i,2}(\sigma'')$,
(b) $Y_i(\sigma') + Y_j(\sigma') \leqslant Y_j(\sigma'') + Y_i(\sigma'')$ and
(c) $i < j$

are satisfied, then the subtree rooted at σ'' can be trimmed off without further exploration.

4. Computational evaluation

To test the performance of the properties when they are incorporated into the design of branch-and-bound algorithms, a series of computational experiments was performed. The experiments consist of two parts. For small-scale problems ($n \leqslant 30$), the experiments are aimed at studying the actual improvement that the lower bound and dominance property can make. For medium-scale problems ($n \leqslant 100$), the focus is mainly on the gap between the lower bound and

the solution produced by a tabu search algorithm. The gap is defined as $[(\mathrm{TS} - \mathrm{LB})/\mathrm{LB}] \times 100\%$, where TS denotes the solution value produced by tabu search and LB is the lower bound.

The algorithms were implemented in C++ and a personal computer with an Intel Pentium 4, 1.7 GHz central processing unit and 256 MB random-access memory was used as the platform. Following the convention adopted in the generation of test data for flow shops, the processing times $p_{i,k}$ were randomly generated from a discrete uniform distribution over [1, 10]. Moreover, to allow potential discrepancy among jobs, another set of test cases were generated with processing times over a wider interval [1, 100]. To generate due dates with different levels of tightness, first the total processing length $p_{i,1} + p_{i,2}$ of each job i is computed, and then the aggregate processing lengths are sorted in non-decreasing order. Denote the ith-largest aggregate length by p'_i. The due date of the job corresponding to p'_i was randomly generated from the interval $(p'_i, p'_i + \sum_{k=1}^{i} p'_{n-k+1}/\beta]$, where β is a parameter used to control the tightness of the due dates. When the value of β is large, the due dates are tight and most of the jobs will be late. On the other hand, if the value of β is small, most of the jobs will be early. To avoid these two extreme cases, preliminary tests were conducted and the value of β set to 3, 5 or 7. Moreover, a limit of 1800 s was used as a threshold for the execution of branch-and-bound algorithms; *i.e.*, if the algorithm cannot finish exploring the solution space, it aborts with a failure. The limit was chosen based on the fact that the maximum value of the system variable used to store the clock ticks in Linux is equivalent to 1800 s. To avoid problems caused by resetting the ticks counter, this limit was adopted. For each problem size n and degree of tightness, β, ten sets of jobs were generated.

Table 2 shows the numerical results for the branch-and-bound algorithm. Columns LB correspond to the algorithm equipped with the lower bound only. LB_Dom indicates the incorporation of both the lower bound and the dominance rule. Three types of information were kept as follows: Avg_Node, average number of nodes explored for the instances that have been successfully solved; Avg_Time, average execution time required by the successfully solved instances; #Opt, number of instances that have been successfully solved. Exploring a search tree of size $n!$ using crude enumeration usually can solve instances with 12 or 13 jobs. With the lower bound and the dominance rule, the algorithm can solve most instances with up to 30 jobs. The statistics first show that incorporating two properties simultaneously can provide better performances with respect to the number of visited nodes and the elapsed execution times. It is also interesting to see that the application of the dominance property does not incur too much computational load at each node. For example, consider the scenarios with $n = 25$ and $\beta = 5$ for LB and LB_Dom. The ratio of the number of nodes explored by LB_Dom to that explored by LB is $3\,611\,820/11\,763\,800 = 0.307$. Similarly, the ratio of their execution times is $92.086/288.442 = 0.319$. The statistics also suggest that the performance of the algorithm is better when the job-processing times are generated from the shorter interval [1, 10]. A second observation is made on the effects of due date tightness. When the due dates are tighter, *i.e.* β is larger, the algorithm will visit more nodes and thus take a longer execution time.

Table 3 summarizes the numerical results of the second part of the experiments. It appears that different interval lengths from which processing times were generated do not have significant impact on the gaps between the approximate solutions provided by tabu search and the lower bound values. However, the tightness of due dates still plays a significant role. Gaps become larger when the given due dates are relatively larger. As a general observation, it may be said that the average gaps are less than 3% between the 240 test cases. This observation in the mean time demonstrates the tightness of our proposed lower bound.

Table 2. Numerical results for the branch-and-bound algorithm.

		$p_i \in [1, 10]$						$p_i \in [1, 100]$					
		LB			LB_Dom			LB			LB_Dom		
n	β	Avg_Node	Avg_time	#Opt	Avg_Node	Avg_time	#Opt	Avg_Node	Avg_time	#Opt	Avg_Node	Avg_time	#Opt
10	3	477	0.006	10	381	0.000	10	452	0.008	10	355	0.006	10
	5	1 222	0.008	10	798	0.006	10	1 722	0.033	10	1 069	0.019	10
	7	2 454	0.019	10	1 340	0.010	10	2 484	0.051	10	1 446	0.031	10
12	3	903	0.010	10	734	0.005	10	551	0.060	10	1 653	0.042	10
	5	2 128	0.039	10	1 331	0.014	10	5 651	0.134	10	3 096	0.081	10
	7	8 503	0.098	10	2 888	0.041	10	4 482	0.169	10	2 804	0.111	10
14	3	3 713	0.044	10	2 451	0.031	10	11 409	0.336	10	5 532	0.173	10
	5	6 237	0.077	10	4 080	0.052	10	7 360	0.282	10	4 377	0.166	10
	7	13 273	0.184	10	5 838	0.086	10	33 149	1.358	10	15 596	0.700	10
16	3	6 451	0.086	10	3 514	0.053	10	29 511	1.309	10	14 012	0.648	10
	5	27 344	0.366	10	11 112	0.158	10	30 544	1.269	10	15 268	0.670	10
	7	68 624	0.952	10	25 181	0.389	10	773 582	31.578	10	147 228	6.981	10
18	3	20 912	0.342	10	10 755	0.172	10	121 519	5.326	10	57 280	2.622	10
	5	136 351	2.552	10	49 093	0.925	10	68 255	3.197	10	26 674	1.338	10
	7	1 910 370	32.000	10	309 733	5.586	10	1 162 890	61.764	10	304 443	17.438	10
20	3	114 870	2.231	10	45 899	0.878	10	81 759	3.938	10	35 743	1.878	10
	5	160 031	3.152	10	69 007	1.411	10	918 015	47.996	10	205 962	11.614	10
	7	1 972 320	41.322	10	407 924	9.053	10	4 680 728	315.445	10	870 196	60.777	10
25	3	3 699 560	101.491	10	673 347	18.405	10	5 744 130	220.252	10	1 349 170	51.864	10
	5	11 763 800	288.442	9	3 611 820	92.086	10	6 013 670	246.296	7	3 281 320	131.233	10
	7	7 506 770	208.495	10	1 016 960	32.263	10	1 312 040	105.806	3	5 980 120	403.266	9
30	3	4 256 164	132.816	9	3 744 051	115.826	10	7 585 302	724.504	4	6 618 136	594.965	9
	5	18 090 878	583.491	5	13 894 589	432.300	9	11 451 931	1034.640	1	2 753 549	264.187	1
	7	12 852 938	563.106	4	10 292 675	385.578	4	–	–	–	–	–	–

Table 3. Average gap between the tabu search solution and the lower bound.

n	β	$p_i \in [1, 10](\%)$	$p_i \in [1, 100](\%)$
40	3	2.069	2.162
	5	2.540	2.356
	7	3.036	3.496
50	3	1.374	1.492
	5	2.292	2.598
	7	3.600	3.106
70	3	0.857	1.065
	5	1.240	1.654
	7	2.729	3.092
100	3	0.978	0.881
	5	0.994	1.300
	7	1.702	1.707

5. Concluding remarks

This article has considered the minimization of total late work in a two-machine flow shop. A lower bound and a dominance property have been proposed. Branch-and-bound algorithms incorporating these two properties were tested through a computational study. Statistics have showed the significance of the lower bound and the dominance rule in reducing the computational efforts required by exploring the enumeration tree. Comparisons between lower bound values and approximate solution values also suggest that the proposed lower bound is fairly close to the optimal solution.

As mentioned above, although the late-work criterion has practical implications as well as theoretical challenges, it has not been extensively studied in the literature. There could be many interesting problems to study. For example, it is of research interest to consider such relationship as precedence constraints or release dates.

Acknowledgement

The authors are partially supported by the National Science Council of the R.O.C. under contracts NSC-93-2416-H-009-026 and NSC-93-2213-E-260-009.

References

Blazewicz, J., Scheduling preemptible tasks on parallel processors with information loss. *Technique Sci. Inf.*, 1984, **3**(6), 415–420.

Blazewicz, J. and Finke, G., Minimizing mean weighted execution time loss on identical and uniform processors. *Inf. Processing Lett.*, 1987, **24**, 259–263.

Blazewicz, J., Pesch, E., Sterna, M. and Werner, F., Open shop scheduling problems with late work criteria. *Discrete Appl. Math.*, 2004, **134**, 1–24.

Blazewicz, J., Pesch, E., Sterna, M. and Werner, F., The two-machine flow-shop problem with weighted late work criterion and common due date. *Eur. J. Operational Res.*, 2005, **165**(2), 408–415.

Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-completeness*, 1979 (Freeman: San Francisco, CA).

Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnoy Kan, A.H.G., Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.*, 1979, **5**, 287–326.

Hariri, A.M.A., Potts, C.N. and Wassenhove, L.N., Single machine scheduling to minimize total weighted late work. *ORSA J. Comput.*, 1995, **7**, 232–242.

Johnson, S.M., Optimal two- and three-stage production schedules with setup times included. *Nav. Res. Logistics Q.*, 1954, **1**(1), 61–68.

Kethley, R.B. and Alidaee, B., Single machine scheduling to minimize total weighted late work: a comparison of scheduling rules and search algorithms. *Comput. Ind. Engng.*, 2002, **43**, 509–528.

Lin, B.M.T. and Hsu, S.W., Minimizing total late work on a single machine with release and due dates, in *2005 SIAM Conference on Computational Science and Engineering*, Orlando, Florida, USA, February 2005.

Pinedo, M. and Chao, X., *Operations Scheduling with Applications in Manufacturing and Services*, 1999 (McGraw-Hill: Singapore).

Potts, C.N. and Van Wassenhove, L.N., Single machine scheduling to minimize total late work. *Operations Res.*, 1992a, **40**, 586–595.

Potts, C.N. and Van Wassenhove, L.N., Approximation algorithms for scheduling a single machine to minimize total late work. *Operations Res. Lett.*, 1992b, **11**, 261–266.