Permutation flowshops with exact time lags to minimize maximum lateness

Julien Fondrevelle, Ali Allahverdi, Ammar Oulamara, Marie-Claude Portmann

Permutation flowshops with exact time lags to minimize maximum lateness

Julien Fondrevelle

MACSI Project LORIA-INRIA Lorraine , Ecole des Mines de Nancy, Parc de Saurupt, 54042 Nancy, France,

E-mail: julien.fondrevelle@loria.fr

Ali Allahverdi

Department of Industrial and Management Systems Engineering, College of Engineering and Petroleum, Kuwait University, P.O. Box 5969, Safat, Kuwait ,

E-mail: allahverdi@kuc01.kuniv.edu.kw

Ammar Oulamara and Marie-Claude Portmann

MACSI Project LORIA-INRIA Lorraine , Ecole des Mines de Nancy, Parc de Saurupt, 54042 Nancy, France,

E-mail: {oulamara, portmann}@loria.fr

Abstract

In this paper, we investigate the m-machine permutation flowshop scheduling problem where exact time lags are defined between consecutive operations of every job. The objective is to minimize the maximum lateness. We introduce different job types, depending on their time lags. We study polynomial special cases and provide a dominance relation. We derive lower and upper bounds that are integrated in a branch-and-bound procedure to solve the problem. We perform a computational analysis to evaluate the efficiency of the proposed method.

Keywords : Flowshop, exact time lags, maximum lateness, dominance relation, branch-and-bound procedure

1 Introduction

We consider the problem of scheduling n jobs in an m-machine permutation flowshop where there exist exact time lags between the operations of every job. Each job j is processed successively on the machines $1, 2, \ldots, m$ for $p_{j,1}, p_{j,2}, \ldots, p_{j,m}$ time units respectively. Each machine can process at most one job at a time. Moreover, the time elapsed between every couple of successive operations of the same job must be equal to a prescribed value (exact time lag). We arbitrarily define the time lag between the completion time of the operation on the upstream machine and the starting time of the subsequent operation, processed on the downstream machine (stop-start time lag). Since the processing times are deterministic and known in advance, it is equivalent to consider start-start or stop-stop time lags. When there exists at least one positive exact time lag, permutation schedules, i.e. schedules where the job sequences are the same on all the machines, are no longer dominant, even with two machines. Nevertheless, we consider here only permutation schedules, which are commonly used in industrial applications. The aim is to find a feasible schedule that minimizes the maximum lateness.

The flowshop problem with exact time lags (or exact delays) is a particular case of the flowshop with minimal and maximal time lags. In this situation, the waiting times between the operations are lower- and upper-bounded. Our problem corresponds to the case where for each couple of consecutive operations, the minimal and maximal time lags are equal. Besides, it must be noted that the exact time lag constraints generalize the classical no-wait constraints, for which the waiting time between successive operations equals 0. The no-wait requirement can be found in industries where products must be processed continuously through the stages in order to prevent degradation. Without loss of generality, we consider here the situation in which the time lag is an integer value (positive or negative).

The case of negative time lags corresponds to job overlapping. This can be used to model a sequence-independent setup time that can be performed while the job is still in process on the preceding machine or a removal time that can be executed while the job is already in process on the succeeding machine. Flowshop problems with no-wait and separate setup times exist in several real situations, for instance in chemical, steel or plastic industries ([Allahverdi and Aldowaisan,01]). Another example arises when lot-sizing is taken into account. The first item or subset of the lot may be available for processing on a machine before the completion of the last items on the preceding machine.

When the exact time lag is positive, the job has to wait for a prescribed amount of time between the machines. This may model a transportation time, a communication delay or an additional processing that does not require any machine. Detailed examples of industrial applications of scheduling problems with time lags can be found in [Deppner,04].

Shop problems with time lags have been extensively studied in the scheduling literature, but in most cases, only minimal time lags are considered (see for instance [Szwarc,86], [Dell'Amico,96], [Brucker and Knust,99], [Janczewski and Kubale,01]). [Brucker et al.,99] show that various scheduling problems, including flowshop with minimal and maximal time lags, can be reduced to single-machine problems with minimal and maximal time lags between jobs. They propose a branch-and-bound algorithm to minimize the makespan. [Finke et al.,02] propose a general model for the two-machine permutation flowshop with minimal time lags and show that this problem can be polynomially solved using an extension of Johnson's algorithm ([Johnson,54]). [Fondrevelle et al.,05] study the problem of minimizing the makespan in a permutation flowshop with minimal and maximal time lags. Special cases are discussed and a branch-and-bound procedure is developed for the m-machine problem. Concerning the no-wait case, many articles investigate scheduling problems with this constraint. [Hall and Sriskandarajah,96] provide a survey of the research on this topic. From a computational complexity point of view, the two-machine no-wait flowshop problem of minimizing maximum lateness has been shown to be NP-hard ([Roeck,84]). This implies that the problem under study is NP-hard as well. The two-machine no-wait flowshop with separate setup times and maximum lateness as objective function is addressed by [Dileepan,04]. Only a dominance relation and special cases are provided. [Fondrevelle et al.,04] study the same problem where separate removal times are also considered. Special cases are presented and a branch-and-bound algorithm is proposed. To the best of our knowledge, no solution method has been developed for the general problem considered in this paper.

The rest of the paper is organized as follows: section 2 introduces the notations used and defines different types of jobs. In section 3, polynomial cases are presented and a dominance relation is proposed for two-machine problems. In section 4, lower and upper bounds are developed and integrated in a branch-and-bound procedure and some computational results are discussed in section 5.

2 Notations and definitions

In this paper, we use the following notations:

- n: number of jobs
- m: number of machines
- $p_{j,k}$: processing time of job j on machine k
- $\theta_{j,k}$: exact time lag for job j between machine k and machine $k+1$
- $C_{j,k}$: completion time of job j on machine k
- d_j: due date of job j
- $L_j = C_{j,m} - d_j$: lateness of job j

The aim is to determine the job completion times on every machine so that all the constraints are satisfied and the criterion $L_{max} = max\{L_j / 1 \leq j \leq n\}$ is minimized. Since the maximum lateness is a regular criterion, semi-active schedules (i.e. left-shifted schedules) are dominant and we will only consider such schedules.

We state the following definition, which will be useful in the rest of the paper.

Definition. 1 *The lateness of each job can be expressed depending on the completion time on any machine as follows:* $L_j = C_{j,k} - d'_{j,k}$ *where* $d'_{j,k} = d_j - \sum_{k \leq t \leq m-1}(\theta_{j,t} + p_{j,t+1})$ *is the due date for job* j *on machine* k.

Proof. The lateness of job j is defined as $L_j = C_{j,m} - d_j$. Due to the exact time lag constraints, the completion time of job j on any machine i can be computed from the completion time on the succeeding machine: $C_{j,i} = C_{j,i+1} - p_{j,i+1} - \theta_{j,i}$. By induction, we have $C_{j,k} = C_{j,m} - \sum_{k \leq i \leq m-1}(\theta_{j,i} + p_{j,i+1})$, which leads to the stated formula (see Figure 1). □

According to the value of each exact time lag, we will distinguish between the following job types:

- The *covering-shape* jobs, for which there exists a machine k such that the processing period on any other machine is included in the processing period on machine k: $\forall\, 1 \leq i \leq m, C_{j,i} - p_{j,i} \geq$

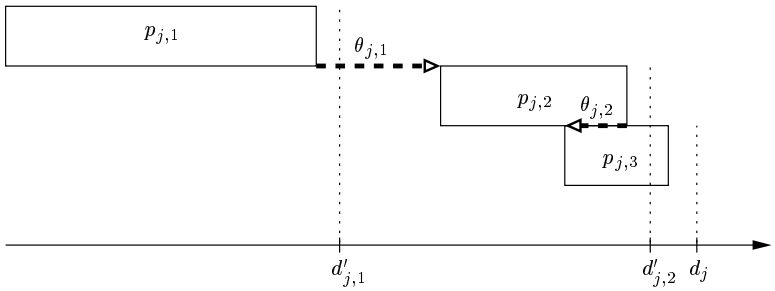

Figure 1: Due date on each machine

$C_{j,k} - p_{j,k}$ and $C_{j,i} \leq C_{j,k}$ (see Figure 2). Depending on the machine index k, such a job will be called k-covering-shape.

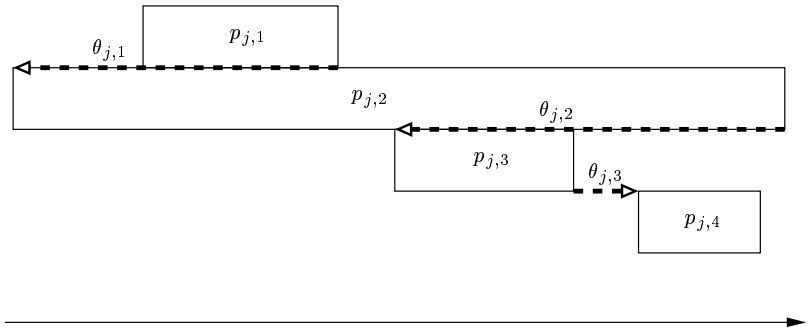

Figure 2: Covering-shape job

- The *no-covering-shape* jobs, for which the processing periods on the machines are all disjoint. This corresponds to the case where the exact time lags are non-negative: $\forall\, 1 \leq i \leq m-1, \theta_{j,i} \geq 0$ (see Figure 3).

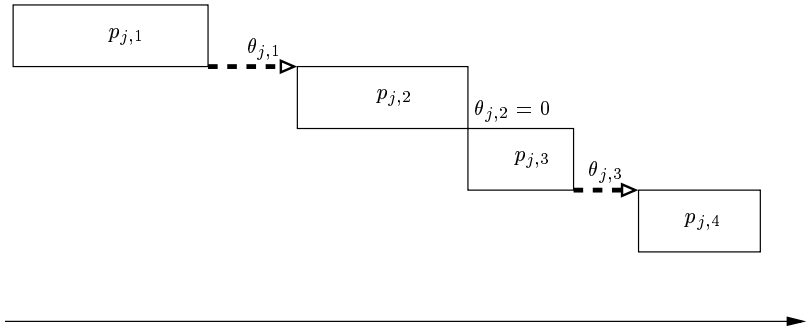

Figure 3: No-covering-shape job

- The *mix-covering-shape* jobs, which do not belong to the previous job classes (see Figure 4).

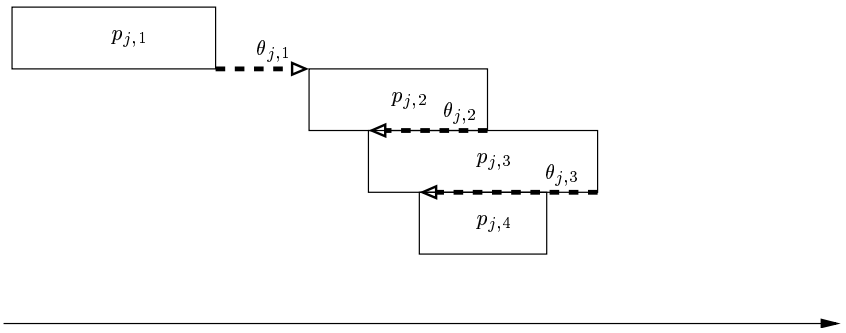

Figure 4: Mix-covering-shape job

3 Special cases

3.1 Polynomial cases

Theorem. 1 *If, for a given machine k, all the jobs are k-covering-shape, then an optimal schedule is obtained by using the Earliest Due Date (EDD) rule on the due dates $d'_{j,k}$ on machine k.*

Proof. Suppose that all the jobs are k-covering-shape. Consider an arbitrary schedule where the job sequence on machine k is $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$. Due to the definition of k-covering-shape jobs, the earliest starting time for the first job scheduled $\pi(1)$ will be on machine k and the latest completion time for that job will be on machine k as well. Thus, without loss of generality, $\pi(1)$ will start on machine k at time 0 and will finish on this machine at time $C_{\pi(1),k} = p_{\pi(1),k}$. The processing on the other machines will be imposed by the exact time lags. Similarly, the second job $\pi(2)$ will be processed on machine k between $C_{\pi(1),k}$ and $C_{\pi(1),k} + p_{\pi(2),k}$, while the processing periods on the other machines are included in this time interval. More generally, the i-th job $\pi(i)$ will be scheduled on machine k between $\sum_{1 \leq h \leq i-1} p_{\pi(h),k}$ and $\sum_{1 \leq h \leq i} p_{\pi(h),k}$. Therefore, the problem is equivalent to a single-machine problem with processing times $p_{j,k}$ and due dates $d'_{j,k}$ on this machine. It is a well known result that EDD provides an optimal schedule for this problem. □

This result generalizes the special cases presented in [Fondrevelle et al.,04] for only two machines. In case of no-wait flowshop with separate setup and removal times, a job j with processing, setup and removal times on machine k respectively denoted by $t_{j,k}$, $s_{j,k}$ and $r_{j,k}$ and due date e_j, can be replaced in our model by a job j with processing time $p_{j,k} = s_{j,k} + t_{j,k} + r_{j,k}$ on machine k, exact time lag $\theta_{j,k} = -r_{j,k} - s_{j,k+1}$ between machines k and $k+1$ and due date $d_j = e_j + r_{j,m}$.

3.2 Dominance relations for two-machine problems

We extend the dominance relations presented by [Dileepan,04] for the two-machine no-wait flowshop with separate setup times to the two-machine flowshop with exact time lags. Consider a sequence $\alpha = (S_1, i, j, S_2)$ where job i precedes immediately job j, and a sequence $\beta = (S_1, j, i, S_2)$ which is identical to α, except that j precedes immediately i (where S_1, S_2 denote partial sequences). The objective is to find conditions under which α dominates β.

As mentioned earlier, the no-wait flowshop with setup times is a particular case of our problem. Following our notations, the conditions proposed in [Dileepan,04] can be expressed as follows:

Property. 1 *[Dileepan,04]*

- *Case A: If*
  - $p_{i,1} + \theta_{i,1} \leq min_{1 \leq u \leq n}\{p_{u,2} + \theta_{u,1}\}$,
  - $p_{j,1} + \theta_{j,1} \leq min_{1 \leq u \leq n}\{p_{u,2} + \theta_{u,1}\}$,
  - $p_{i,2} + \theta_{i,1} \leq p_{j,2} + \theta_{j,1}$
  - *and* $d_i \leq d_j$.

  Then solution α dominates solution β.

- *Case B: If*
  - $p_{i,1} + \theta_{i,1} \geq max_{1 \leq u \leq n}\{p_{u,2} + \theta_{u,1}\}$,
  - $p_{j,1} + \theta_{j,1} \geq max_{1 \leq u \leq n}\{p_{u,2} + \theta_{u,1}\}$,
  - $p_{j,2} + \theta_{j,1} \leq p_{i,2} + \theta_{i,1}$
  - *and* $d'_{i,1} \leq d'_{j,1}$.

  Then solution α dominates solution β.

The general ideas used to establish this property are the following:

- Case A: If

- in solution α there is no idle time on machine 2 between the end of S_1 and the completion of j,
- in solution β there is no idle time on machine 2 between the end of S_1 and the completion of i,
- machine 1 becomes available sooner after j in solution α than after i in solution β,
- and i has a smaller due date than j on machine 2,

then solution α dominates solution β.

- Case B: If
  - in solution α there is no idle time on machine 1 between the end of S_1 and the completion of j,
  - in solution β there is no idle time on machine 1 between the end of S_1 and the completion of i,
  - machine 2 becomes available sooner after j in solution α than after i in solution β,
  - and i has a smaller due date than j on machine 1,

  then solution α dominates solution β.

In each case of Property 1, the first two conditions are sufficient to avoid idle time as mentioned previously. However, it is possible to state other conditions that are less restrictive and for which the result still holds. These conditions apply to more instances than the previous ones. Let x denote the last job of the partial schedule S_1 (if S_1 is empty, let $p_{x,1} = p_{x,2} = \theta_{x,1} = 0$). The new conditions can be expressed as follows:

Property. 2
- *Case A': If*
  - $p_{i,1} + \theta_{i,1} \leq min(p_{x,2} + \theta_{x,1}, p_{j,2} + \theta_{j,1})$,
  - $p_{j,1} + \theta_{j,1} \leq min(p_{x,2} + \theta_{x,1}, p_{i,2} + \theta_{i,1})$,
  - $p_{i,2} + \theta_{i,1} \leq p_{j,2} + \theta_{j,1}$
  - *and* $d_i \leq d_j$.

then solution α dominates solution β.

- *Case B': If*
  - $p_{i,1} + \theta_{i,1} \geq max(p_{x,2} + \theta_{x,1}, p_{j,2} + \theta_{j,1})$,
  - $p_{j,1} + \theta_{j,1} \geq max(p_{x,2} + \theta_{x,1}, p_{i,2} + \theta_{i,1})$,
  - $p_{j,2} + \theta_{j,1} \leq p_{i,2} + \theta_{i,1}$
  - *and* $d'_{i,1} \leq d'_{j,1}$.

  then solution α dominates solution β.

A similar proof to that presented in [Dileepan,04] can be used to demonstrate that in case A' or in case B', solution α dominates solution β.

It would be possible to generalize this to a problem with an arbitrary number m of machines, but as m increases, the conditions become more and more complex and restrictive.

4 A branch-and-bound method

In this section, we propose a branch-and-bound algorithm to solve the problem of minimizing the maximum lateness in an m-machine permutation flowshop with exact time lags. As mentioned earlier, we can restrict the search for an optimal solution to semi-active schedules. For a given job sequence π, the optimal placement of the jobs with respect to π on all the machines can be determined polynomially, by scheduling the jobs $\pi(1), \pi(2), \ldots, \pi(n)$ successively as soon as possible. This result is similar to that presented in [Fondrevelle et al.,05] and leads us to use a classical scheme based on Ignall and Schrage's method ([Ignall and Schrage,65]). Nodes at depth k of the search tree are associated with initial partial sequences of k jobs. At each separation, a job is added at the end of the current partial sequence. A depth-first search rule is adopted in the branching procedure. An initial upper bound is provided by the heuristics presented in Section 4.2. The value of the upper bound is then updated each time a new solution with lower objective value is found.

4.1 Lower bounds

Suppose that the initial partial sequence is $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(h))$, in which the first h jobs have been scheduled. The completion times and the lateness of these jobs are exactly determined.

We first define m lower bounds $LB_1, LB_2, \ldots, LB_m$ where $LB_k (k = 1, \ldots, m)$ proceeds as follows: for the jobs that have not been scheduled yet, we only take into account the processing on machine k (by relaxing the capacity constraints on all the machines except k and possibly accepting that the operations on these machines might start before time 0). As defined in Section 2, the lateness L_j of each job j can be computed from the completion time on machine k and the due date on this machine, i.e. $L_j = C_{j,k} - d'_{j,k}$. Using a similar argument as in the proof of Theorem 1, we could show that the relaxed problem is equivalent to a single-machine problem for the remaining jobs, with processing times $p_{j,k}$ and due dates $d'_{j,k}$ and where the machine becomes available at time $C_{\sigma(h),k}$ (the last job scheduled in the current partial sequence is denoted by $\sigma(h)$). An optimal solution to this problem is provided by EDD applied on $d'_{j,k}$. Let L_k^{EDD} be the corresponding maximum lateness value. Then lower bound LB_k is given by $LB_k = max(L_\sigma, L_k^{EDD})$ where $L_\sigma = max\{L_{\sigma(i)} / 1 \leq i \leq h\}$ denotes the maximum lateness of the current partial schedule. The global lower bound LB is defined by $LB = max\{LB_k / 1 \leq k \leq m\}$.

4.2 Upper bounds

For $1 \leq k \leq m$ we define the heuristic H_k as follows: apply EDD on the due dates $d'_{j,k}$ on machine k and construct the corresponding schedule. The best criterion value obtained can be used as an initial upper bound and will be denoted by H_{EDD}.

We also propose to improve this value through an iterative procedure based on NEH method ([Nawaz et al.,83]). The principle of this frequently used scheme is as follows: starting from an initial job list, the schedule is constructed step by step by successively inserting the jobs of the list at the best position in the partial sequence, so as to minimize the objective function. We choose to apply NEH iteratively a prescribed number of times: at each iteration, the final sequence obtained at the previous step is used as initial job list. The sequence with the best value throughout the iterations is kept as solution. Depending on the criterion used to construct the initial job list, we define three heuristics $NEH(TT)$, $NEH(JL)$ and $NEH(H_{EDD})$:

- In $NEH(TT)$, the initial list is sorted in decreasing order of total processing time $\sum_{1 \leq k \leq m} p_{j,k}$ of jobs.
- In $NEH(JL)$, the initial list is sorted in decreasing order of total length $\sum_{1 \leq k \leq m-1} (p_{j,k} + \theta_{j,k}) +$

$p_{j,m}$ of jobs.

- In $NEH(H_{EDD})$, the initial job list corresponds to the best sequence provided by the heuristic H_{EDD}.

5 Computational results

We conducted a computational analysis to evaluate the performance of the proposed solution procedure. In the case of two machines, we also compared this new general approach with the one that we used in [Fondrevelle et al.,04] for a more specific problem. Our algorithms were coded in C, and the computational experiments were run on a PC Pentium, 1.2 GHz.

We first used the same instances as in [Fondrevelle et al.,04] as benchmarks. These 10-instance classes correspond to two-machine no-wait flowshop problems with separate setup and removal times, which were shown to be particular cases of our problem (mix-covering-shape jobs only, with partial covering between every couple of successive operations). Details about the generation of these classes are given in the Appendix A.

We also generated new benchmark classes for 5-machine problems, according to the classification given in Section 2. Each class contains 10 instances and the number of jobs is set to 16, except for class 12 in which $n = 14$.

- Class 10 corresponds to no-covering-shape jobs only, the processing times of which are randomly drawn between 20 and 50. The time lags are in the interval $[0; 100]$.

- Class 11 corresponds to mix-covering-shape jobs only, the processing times of which are randomly drawn between 20 and 50. $\theta_{j,k}$ is generated between $-p_{j,k+1}$ and 0 so as to have partial covering between every couple of successive operations of the jobs.

- Classes 12 and 13 correspond to covering-shape jobs only. For each job j, a random integer is drawn between 1 and 5 to determine the machine k_j such that j is k_j-covering-shape. The processing times on all the machines except k_j are generated between 20 and 50, and the time lags that are not related to k_j are in the interval $[-20; 20]$. p_{j,k_j}, θ_{j,k_j-1} and θ_{j,k_j} are computed such that j is k_j-covering-shape. Although such problems do not correspond to real situations,

Table 1: Performance of the heuristics

Class	H_{EDD}	$NEH(H_{EDD})$	$NEH(TT)$	$NEH(JL)$
1	22.1	11.9	8.6	8.9
2	24.3	9.1	6.3	7.3
3	17.7	4.1	2.4	2.9
4	12.9	4.9	5.1	4.2
5	9.8	4.8	3.3	4.8
6	32.9	12.5	10.9	10.9
7	25.9	10.3	12.9	8.6
8	29.1	13.7	8.0	8.4
9	14.8	4.1	3.3	5.3
10	38.9	5.6	5.8	5.3
11	33.5	11.5	7.4	9.4
12	26.2	7.1	5.5	6.0
13	32.5	9.1	6.4	8.0
Average	24.7	8.4	6.6	6.9

it could be interesting to apply our solution method to them in order to evaluate its efficiency in these cases.

Following the method proposed in [Potts and Van Wassenhove,82], we generated the due dates in a range $[Px, Py]$, where P is a lower bound on the makespan and $x = 1 - T - R/2$, $y = 1 - T + R/2$. T is the tardiness factor, which was set to 0.6 and R is the due date range set to 0.75.

We first compared the performance of the heuristics presented in 4.2. For each instance, the relative error (in %) between the solution found by the heuristic considered and the optimal solution, obtained without time limit, was computed. The average values for each class are given in Table 1. Since the CPU times for the heuristics are very small (less than 0.1 second), we do not report them here.
As can be seen from Table 1, H_{EDD} is outperformed by the iterative NEH-based methods. This result holds for every instance. $NEH(H_{EDD})$ is slightly outperformed by $NEH(TT)$ and $NEH(JL)$, the average relative errors of which are in the same range and do not increase with the number of machines. Moreover, for each of these heuristics, there exists at least one instance in each class on which the heuristic dominates the two others.

To evaluate the quality of the branch-and-bound procedure, we performed it on each instance with a computational time limit of 1200 seconds. For each class, we report in Table 2 the number N of problems for which the algorithm achieved the time limit, the average computational time t (in

Table 2: Performance of the branch-and-bound algorithm

Class	N	t	t_{dom}	N'	t'	t'_{dom}
1	0	13.8	9.2	1	107.1	/
2	0	13.2	10.7	1	171.6	/
3	0	22.4	19.4	2	121.6	/
4	0	13.6	8.5	4	251.7	/
5	0	35.5	14.2	1	59.7	/
6	0	0.4	0.3	0	99.7	/
7	0	3.7	2.9	1	88.2	/
8	0	8.9	6.5	0	166.4	153.6
9	1	10.2	5.7	3	182.5	139.7
10	0	155.3	/	/	/	/
11	0	115.7	/	/	/	/
12	0	71.2	/	/	/	/
13	3	209.4	/	/	/	/

seconds) for the problems optimally solved before the time limit, and the average computational time t_{dom} (in seconds) when the dominance relation is taken into account (only for 2-machine problems). The corresponding values obtained with the method proposed in [Fondrevelle et al.,04] are indicated with a prime symbol. Note that the dominance test used in [Fondrevelle et al.,04] is more restrictive than the one we use here and concerns only classes 8 and 9.

If we compare the performance of our new branch-and-bound procedure and that of the one proposed by [Fondrevelle et al.,04], we can note a significant improvement in computational time: all the two-machine problems except 1 are optimally solved by the new algorithm and the average computational time is divided by a factor between 5 and 250 except for class 5. It could be surprising that a method developped for a more general problem outperforms a solution approach dedicated to a particular case. Such a gain is partly due to the improvement of the lower bound. Moreover, the dominance relation, which is more frequently used than the previous one, appears to perform quite well since it results in saving 25% of the computational time in average. As far as the 5-machine problems are concerned, it seems that problems with covering-shape jobs only are more difficult to solve than problems with no-covering-shape jobs only or problems with mix-covering-shape jobs only.

We also conducted another series of experiments to evaluate the influence of the number of machines m on the performance of the branch-and-bound. Three new classes denoted by 11A, 11B and 11C were generated similarly as class 11, with m equal to 2, 10 and 15 respectively. Table 3 presents the

Table 3: Influence of m on the computational time

Class	m	N	τ	Q	$\rho = \tau/Q$	$\lambda = \rho/m$
11A	2	0	5.6	5.086×10^6	1.10×10^{-6}	0.55×10^{-6}
11	5	0	115.7	47.50×10^6	2.44×10^{-6}	0.48×10^{-6}
11B	10	1	500.9	111.85×10^6	4.48×10^{-6}	0.45×10^{-6}
11C	15	5	1449.9	223.35×10^6	6.49×10^{-6}	0.43×10^{-6}

number N of problems (out of 10) for which the algorithm achieved the time limit (1200 seconds) and the average computational time τ (in seconds) when no time limit is imposed. Additionnally, we report the average number Q of nodes evaluated (i.e. how many times the lower bound is computed), $\rho = \tau/Q$ and $\lambda = \rho/m$. Therefore ρ corresponds to the average time to evaluate one node (in seconds). Without time limitation, the maximum CPU time was 3720 seconds for an instance with 15 machines.

We can consider the value of λ as constant since it varies from 0.43×10^{-6} to 0.55×10^{-6}. This does not only hold on average, but also for every instance. By definition, this means that the average time for the branch-and-bound algorithm to evaluate one node increases linearly with the number of machine, which is in agreement with the computational complexity of the lower bound ($O(m)$). Besides, the number of nodes Q seems also to be roughly linear in m. This empirical result needs to be confirmed or contradicted by further experiments. The increase in the number of visited nodes is partially explained by the fact that the lower bound becomes less tight as the number of machines grows.

6 Conclusion

We study the problem of minimizing maximum lateness in m-machine permutation flowshops with exact time lags between consecutive operations of the jobs. The exact time lags generalize the classical no-wait constraint and may be used to model no-wait problems with separate setup and removal times. A branch-and-bound method is proposed to solve optimally this NP-hard problem. The computational results show that it outperforms previous algorithms and that it may be improved significantly in case of two machines using a dominance relation.

References

[Allahverdi and Aldowaisan,01] Allahverdi A, Aldowaisan T (2001) Minimizing total completion time in a no-wait flowshop with sequence-dependent additive changeover times. *Journal of the Operational Research Society* 52: 449-462

[Brucker et al.,99] Brucker P, Hilbig T, Hurink J (1999) A branch and bound algorithm for a single-machine scheduling problem with positive and negative time-lags. *Discrete Applied Mathematics* 94: 77-99

[Brucker and Knust,99] Brucker P, Knust S (1999) Complexity results for single-machine problems with positive finish-start time-lags. *Computing* 63: 299-316

[Dell'Amico,96] Dell'Amico M (1996) Shop problems with two machines and time lags. *Operations Research* 44: 777-787

[Deppner,04] Deppner F (2004) Ordonnancement d'atelier avec contraintes temporelles entre opérations (*in French*). Ph.D. thesis, Institut National Polytechnique de Lorraine

[Dileepan,04] Dileepan P (2004) A note on minimizing maximum lateness in a two-machine no-wait flowshop. *Computers and Operations Research* 31: 2111-2115

[Finke et al.,02] Finke G, Espinouse M-L, Jiang H (2002) General flowshop models : job dependent capacities, job overlapping and deterioration. *International Transactions in Operational Research* 9: 399-414

[Fondrevelle et al.,04] Fondrevelle J, Allahverdi A, Oulamara A (2004) Two-machine no-wait flowshop scheduling problem to minimize maximum lateness with separate setup and removal times. Technical Report A04-R-406 LORIA-INRIA Lorraine

[Fondrevelle et al.,05] Fondrevelle J, Oulamara A, Portmann M-C (2005) Permutation flowshop scheduling problems with maximal and minimal time lags. *Computers and Operations Research* (to appear)

[Hall and Sriskandarajah,96] Hall N, Sriskandarajah C (1996) A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research* 44: 510-525

[Ignall and Schrage,65] Ignall EJ, Schrage LE (1965) Application of the branch and bound technique to some flow-shop scheduling problems. *Operations Research* 13: 400-412

[Janczewski and Kubale,01] Janczewski R, Kubale M (2001) Scheduling unit execution time tasks with symmetric time-lags. *Journal of Applied Computer Science* 9: 45-51

[Johnson,54] Johnson SM (1954) Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly* 1: 61-68

[Nawaz et al.,83] Nawaz M, Enscore Jr EE, Ham I (1983) A heuristic algorithm for the m-machine, n-job flow shop sequencing problem. *Omega* 11: 91-95

[Potts and Van Wassenhove,82] Potts CN, Van Wassenhove LN (1982) A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters* 1: 177-181

[Roeck,84] Roeck H (1984) Some new results in flowshop scheduling. *Zeitschrift fur Operations Research* 28: 1-16

[Szwarc,86] Szwarc W (1986) The flow shop problem with time lags and separated setup times. *Zeitschrift fur Operations Research* 30: B15-B22

A Appendix - Generation of the benchmarks in [Fondrevelle et al.,04]

This section describes how the benchmarks used in [Fondrevelle et al.,04] were generated. We recall that this model corresponds to a two-machine no-wait flowshop with separate setup and removal times. Job j ($j \in \{1, \ldots, n\}$) has a processing time $t_{j,k}$, a setup time $s_{j,k}$ and a removal time $r_{j,k}$ on machine k ($k \in \{1, 2\}$) and a due date e_j that applies to the completion time on machine 2 (the removal time is not taken into account to compute the lateness of the job).

9 classes were generated according to the following parameters:

- n: the number of jobs
- g: the percentage of jobs belonging to the first group. Jobs are divided into two groups. Processing times of jobs in the first (resp. second) group are in a range $[1, 100]$ (resp. $[40, 60]$)
- k_s: the ratio of maximum setup time. Setup times are generated in a range $[0, k_s \times t_{max}]$ where t_{max} denotes the maximum processing time (100 or 60 depending on the group)
- k_r: the ratio of maximum removal time, which is defined similarly as k_s
- R: the due date range
- T: the tardiness factor.

The due dates are generated as in the new benchmarks, following the method of [Potts and Van Wassenhove,82]. All data were drawn from discrete uniform distributions and each class contains 10 instances. Table 4 reports the values of the parameters for each class.

Table 4: Classes of instances used in [Fondrevelle et al.,04]

Class	n	g	k_s	k_r	T	R
1	16	50%	0.5	0.5	0.6	0.75
2	16	50%	0.5	0.5	0.6	0.6
3	16	50%	2	2	0.6	0.75
4	16	50%	2	0.5	0.6	0.75
5	16	0%	0.5	0.5	0.6	0.75
6	16	100%	0.5	0.5	0.6	0.75
7	18	50%	0.5	0.5	0.6	0.75
8	16	50%	0.5	0	0.6	0.75
9	16	50%	2	0	0.6	0.75