Lower Bounds for the Two-Machine Flow Shop Problem with Time Delays

Mohamed Amine Mkadem, Aziz Moukrim and Mehdi Serairi

Abstract We consider the flow shop problem with two machines and time delays with respect to the makespan, i.e., the maximum completion time. We recall the lower bounds of the literature and we propose new relaxation schemes. Moreover, we investigate a linear programming-based lower bound that includes the implementation of a new dominance rule and a valid inequality. A computational study that was carried out on a set of 480 instances including new hard ones shows that our new relaxation schemes outperform the state of the art lower bounds.

1 Introduction

This paper is devoted to dealing with the flow shop scheduling problem with two machines and time delays, denoted by $F2|l_j|C_{max}$. Let $I = (J, p_1, l, p_2)$ be an instance of $F2|l_j|C_{max}$, where $J = \{1, 2, \dots, n\}$ is a set of n jobs, p_1 and p_2 are the vectors of processing times on the first and the second machines, and l is the vector of the time delays. Each job j has two operations. The first operation (resp. the second operation) must be executed without preemption during $p_{1,j}$ (resp. $p_{2,j}$) time units on M_1 (resp. M_2). For each job $j \in J$, a time delay of at minimum l_j time units must separate the end of the first operation and the start of the second one. The objective is to find a feasible schedule that minimizes the completion time of the last scheduled job on M_2. A feasible schedule is such that at most one operation is processed at a time on a given machine. In addition, the operations are executed without preemption, where interruption and switching of operations are not allowed.

M.A. Mkadem (✉) · A. Moukrim · M. Serairi
Sorbonne universités, Université de Technologie de Compiégne, CNRS,
UMR 7253 Heudiasyc, CS 60319, 60203 Compiégne Cedex, France
e-mail: mohamed-amine.mkadem@hds.utc.fr

A. Moukrim
e-mail: aziz.moukrim@hds.utc.fr

M. Serairi
e-mail: mehdi.serairi@hds.utc.fr

A. Fink et al. (eds.), *Operations Research Proceedings 2016*,
Operations Research Proceedings, DOI 10.1007/978-3-319-55702-1_70

Mitten [2] proves that the permutation flow shop $F2\pi|l_j|C_{max}$, where a feasible schedule consists in having the same job sequence on both machines, can be solved in polynomial time. However, solving our problem as a permutation flow shop does not necessarily provide an optimal solution. $F2|l_j|C_{max}$ is an NP-hard problem in the strong sense even with unit-time operations [3].

The objective of this paper is to introduce new lower bounds. First, we improve the most promising lower bound of the literature. Second, we investigate a linear programming-based lower bound.

2 Combinatorial Lower Bounds

We present here the lower bounds of the literature and propose new ones. Hereafter, $C^*_{max}(I)$ represents the optimal makespan value of instance I and $C_{max}(S)$ stands for the makespan value of schedule S.

First, we survey four lower bounds of Yu [3]. We start by two $O(n)$ basic lower bounds $LB_1 = \max_{1\le j\le n}(p_{1,j} + l_j + p_{2,j})$ and $LB_2 = \max(\sum_{j=1}^{n} p_{1,j} + \min_{1\le j\le n}(l_j + p_{2,j}),$ $\sum_{j=1}^{n} p_{2,j} + \min_{1\le j\le n}(l_j + p_{1,j}))$. Moreover, Yu [3] interested in the problem where each job $j \in J$ is splitted into $\min(p_{1,j}, p_{2,j})$ unitary sub-jobs. The lower bound $LB_3 = \lceil(\sum_{j=1}^{n} \min(p_{1,j}, p_{2,j}).l^u_j)/\sum_{j=1}^{n} \min(p_{1,j}, p_{2,j})\rceil + 1 + \sum_{j=1}^{n} \min(p_{1,j}, p_{2,j})$ was introduced, where $l^u_j = l_j + \max(p_{1,j}, p_{2,j}) - 1$ is the time delay observed by each sub-job derived from $j \in J$.

The fourth lower bound is presented as follows. Let S^* be an optimal schedule and $p_{k,[\ell]}$ the processing time of the job scheduled at position ℓ on M_k, $k \in \{1, 2\}$. Moreover, let j^k be the position of job j on M_k, $k \in \{1, 2\}$. For each job $j \in J$, it holds that $C_{max}(S^*) \ge \sum_{\ell=1}^{j^1} p_{1,[\ell]} + l_j + \sum_{\ell=j^2}^{n} p_{2,[\ell]}$. By adding together the above equations for all jobs and by considering that the makespan is integral, $LB_4 = \lceil(\sum_{j=1}^{n} l_j + \sum_{m=1}^{n} \rho_{1,m} + \sum_{m=1}^{n} \rho_{2,m})/n\rceil$ is a valid lower bound, where $\rho_{k,m}$ is the sum of the m smallest values in $\{p_{k,1}, p_{k,2}, \dots, p_{k,n}\}$.

The following lower bounds were introduced by Dell'Amico [1]. In the first one, it is assumed that all jobs are executed at time 0 on M_1. The problem is then a single-machine scheduling problem with release dates denoted by $1|r_j|C_{max}$. Let I_r be the instance for $1|r_j|C_{max}$ with $r_j = p_{1,j} + l_j$ and $p_j = p_{2,j}$, $j \in J$. Obviously, $L_1 = C^*_{max}(I_r)$ is a valid lower bound on the $F2|l_j|C_{max}$ original instance, which can be computed in $O(n \log n)$-time by scheduling the jobs in a nondecreasing order of $r_j, j \in J$. By interchanging the role of M_1 and M_2, we yield a symmetric lower bound called L_2. Finally, we define the lower bound $LB_5 = \max(L_1, L_2)$.

Solving our problem as a permutation flow shop does not necessarily provide an optimal solution. However, special cases exist where it is true. Dell'Amico [1] proved that permutation schedules are dominant if $l_j \le \min_{1\le i\le n}(p_{1,i} + l_i)$, $j \in J$ and then he introduced the following lower bound. Let $\bar{I} = (J, p_1, \bar{l}, p_2)$ be a new instance that is derived from instance $I = (J, p_1, l, p_2)$, where $\bar{l}_j = \min(l_j, \min_{1\le i\le n}(l_i + p_{1,i}))$, $j \in J$. Since $\bar{I}$ verifies Dell'Amico's [1] conditions, an optimal solution for $\bar{I}$ can be found in polynomial time using Mitten algorithm [2]. Therefore, $LB_6 = C^*_{max}(\bar{I})$ is a valid lower bound.

Furthermore, we introduce two new lower bounds which can be considered as a generalization of LB_6. In fact, Yu [3] extended Dell'Amico's [1] result after showing that the permutation schedules are dominant if $l_j \leq \min_{1\leq i\leq n}(l_i + \max(p_{1,i}, p_{2,i}))$, $j \in J$. Therefore, from an instance $I = (J, p_1, l, p_2)$ of $F2|l_j|C_{max}$ problem, we derive a new instance $\tilde{I}(J, p_1, \tilde{l}, p_2)$, where $\tilde{l}_j = \min(l_j, \min_{1\leq i\leq n}(l_i + \max(p_{1,i}, p_{2,i}))), j \in J$. As a consequence of Yu's [3] result, $LB_1^N = C^*_{max}(\tilde{I})$ is a valid lower bound on instance I, which is computed in $O(n \log n)$-time using Mitten [2].

Moreover, we consider two instances $I = (J, p_1, l, p_2)$ and $I' = (J', p_1, l, p_2)$ of $F2|l_j|C_{max}$, where $J' \subset J$. Then, any valid lower bound on I' is also a valid lower bound on I. A new lower bound called LB_2^N can be obtained by invoking LB_1^N on different sub-instances of I. Interestingly, we consider n sub-instances. We start by the original instance I, the next sub-instance is built from the one in hand by removing the job that has the minimum value of $l_j + \max(p_{1,j}, p_{2,j}), j \in J$.

3 Linear Programming-Based Lower Bound

We consider a mathematical formulation that is based on determining the precedence relationships between jobs on the two machines where it is supposed that the jobs are continuously processed on M_1 and M_2. Indeed, any valid schedule on an $F2|l_j|C_{max}$ instance can be transformed to a schedule with the same makespan value C where jobs are continuously processed on the two machines from time 0 and from time $C - \sum_{j=1}^{n} p_{2,j}$ on M_1 and M_2, respectively.

The decision variables are defined for each pair of jobs $i, j \in J$, where $X^k_{i,j}$ takes the value 1 if i precedes j on M_k and 0 otherwise, $k \in \{1, 2\}$. Furthermore, $C_{k,j}$ represents the completion time of job j on M_k and the total idle time on M_2 is denoted by L. Using these definitions, the model can be formulated as follows:

$$\mathbf{min} \quad L \tag{1}$$

$$\text{s.t.} \quad X^k_{i,j} + X^k_{j,i} = 1, \qquad \forall i, j \in J \; i \neq j; k \in \{1, 2\} \tag{2}$$

$$X^k_{i,j} \geq X^k_{i,v} + X^k_{v,j} - 1, \qquad \forall i, j, v \in J; k \in \{1, 2\} \tag{3}$$

$$C_{1,i} = \textstyle\sum_{j=1}^{n} p_{1,j}.X^1_{j,i} + p_{1,i}, \quad \forall i \in J \tag{4}$$

$$C_{2,i} \geq C_{1,i} + l_i + p_{2,i}, \qquad \forall i \in J \tag{5}$$

$$C_{2,i} = L + \textstyle\sum_{j=1}^{n} p_{2,j}.X^2_{j,i} + p_{2,i}, \; \forall i \in J \tag{6}$$

$$L \geq 0, \; C_{k,j} \geq 0, \; X^k_{i,j} \in \{0, 1\} \;\; \forall i, j \in J, \; k \in \{1, 2\} \tag{7}$$

The objective function (1) minimizes the total idle time on M_2. Constraints (2) ensure that for each pair of jobs, one of them has to precede the other on each machine. Constraints (3) guarantee the absence of cyclic precedence relationships between all jobs. Constraints (4) and (6) take into account the job's precedence and enforce them to be processed continuously without idle on M_1 and M_2, respectively. In addition, Constraints (5) ensure that a job after being processed on M_1 has to wait its time delay to be executed on M_2. The nature of decision variables L, $C_{k,j}$ and $X^k_{i,j}$ is displayed by Constraints (7).

In order to strengthen the LP relaxation of the model, we propose a valid inequality, which is based on the additional waiting time that a job has to fulfill after being available for processing on M_2. We remark that given a sequence of jobs on M_1, solutions in which the jobs are scheduled on M_2 according to their arrival times are dominant. Therefore, if a job j is preceded by a job i on M_1, then a lower bound on the minimum additional waiting time observed by job j or job i is $w_{i,j}$, where (i) $w_{i,j} = \max(0, l_i + p_{2,i} - p_{1,j} - l_j)$, if $l_i \le p_{1,j} + l_j$ (ii) $w_{i,j} = \max(0, p_{1,j} + l_j + p_{2,j} - l_i)$, if $l_i > p_{1,j} + l_j$.

A lower bound on the total additional waiting time Δ can be obtained by solving the assignment problem where the assignment costs are $\delta_{i,j}$, $i,j \in \{1, \ldots, n\}$. In the following, we describe how the assignment costs are computed. Note that the first scheduled job (resp. the last scheduled job) on M_1 is assumed to be preceded (resp. followed) by a dummy job (job 0, resp. job $n+1$). Obviously, since job 0 cannot precede job $n+1$, and a job cannot precede itself, then we set $\delta_{0,n+1} = \infty$ and $\delta_{j,j} = \infty$, $\forall j \in \{1, \ldots, n\}$.

Remark 1 Let us consider an instance I of $F2|l_j|C_{max}$ and LB (resp. UB) a lower bound (resp. an upper bound) on the value of the makespan. If a schedule of makespan LB exists, then the jobs can be continuously processed without any idle time, from time 0 on M_1 and from time $(LB - \sum_{j=1}^{n} p_{2,j})$ on M_2. Then, we obtain the following assignment costs:

- $\forall i \in \{1, \ldots, n\}$, $\delta_{0,i} = \max(0, LB - \sum_{j=1}^{n} p_{2,j} - l_i - p_{1,i})$
- $\forall i \in \{1, \ldots, n\}$, if $\sum_{j=1}^{n} p_{1,j} + l_i + p_{2,i} > UB$, i cannot be processed at the last position on M_1, then $\delta_{i,n+1} = \infty$. Otherwise $\delta_{i,n+1} = 0$

In order to set $\delta_{i,j}$, $\forall i,j \in \{1, \ldots, n\}$, $i \neq j$, we introduce in the following lemma a new dominance rule.

Lemma 1 *Let $I = (J, p_1, l, p_2)$ be an instance of $F2|l_j|C_{max}$ and two jobs $i,j \in J$ such that $p_{1,j} + l_j \le p_{1,i} + l_i \le p_{2,j} + l_j$. For any schedule S of I, if j and i are adjacent on M_1 then j should precede i on M_1.*

Proof Let us suppose that j is executed before i on M_1. First, thanks to the relation $p_{1,i} + l_i \le p_{2,j} + l_j$, i is ready for processing on M_2 while the processing of job j has not yet ended. Then these two jobs are executed continuously without idle on M_2. Second, since $p_{1,j} + l_j \le p_{1,i} + l_i$, the operations $O_{2,j}$ and $O_{2,i}$ would have started earlier than if i had preceded j on M_1.

Corollary 1 *Let $I = (J, p_1, l, p_2)$ be an instance of $F2|l_j|C_{max}$ and two jobs $i,j \in J$. If $p_{1,j} + l_j \le p_{1,i} + l_i \le p_{2,j} + l_j$, then $\delta_{i,j} = \infty$. Otherwise $\delta_{i,j} = w_{i,j}$.*

Similarly, by interchanging the role of M_1 and M_2, we obtain Δ' another lower bound on the total additional waiting time. Therefore, the following valid inequality holds: $\sum_{j=1}^{n} C_{2,j} \geq \sum_{j=1}^{n} C_{1,j} + \sum_{j=1}^{n}(p_{2,j} + l_j) + \max(\Delta, \Delta')$.

4 Computational Results

We present in this section the computational results of the new lower bounds and we compare their performance. We test them on a set of six classes A–F that was proposed by Dell'Amico [1]. Furthermore, preliminary computational results conducted on the literature classes show that previous lower bounds give bad performance when time delays are very large compared to processing times. To that aim, we introduce two new classes of instances where the processing times on M_1 and M_2 and the time delays are randomly generated between $[1 \dots \alpha]$, $[1 \dots \beta]$ and $[1 \dots \gamma]$, respectively, where $\alpha = \beta = 20$ and $\gamma = \frac{n}{2}10$ (resp. $\alpha = \beta = 100$ and $\gamma = \frac{n}{2}100$) for class 1 (resp. class 2). For each class, the number of jobs is $n = 10, 30, 50, 100, 150, 200$. For each combination of class and number of jobs, 10 instances were randomly generated. All algorithms were coded in C++ and compiled under CentOS 6.6. Moreover, we used CPLEX 12.6 to implement the linear programming-based lower bound. The experiments were conducted on an Intel(R) Xeon(R) @ 2.67 GHz processor. For pages limitation, we interest only to the most competitive lower bounds.

In the following, we denote by LB_3^N the optimal objective value that is obtained after solving the LP relaxation of the mathematical model (1)–(7) including the valid inequality and by LB_4^N a version of LB_3^N without Constraints (3). We conducted preliminary computational results on LB_3^N and LB_4^N. Clearly, LB_3^N dominates LB_4^N. However, LB_4^N offers a good trade-off between effectiveness and efficiency. Indeed, for all the considered instances where $n < 100$, LB_4^N achieves the same lower bound values as LB_3^N within a very short time. The average computational time of LB_4^N on these instances is 0.77 s while LB_3^N needs 61.54 s. Furthermore, LB_3^N fails to solve all large scale instances (i.e. $n \geq 100$) within 1800 s, while LB_4^N solves them in an average time of 1.47 s.

In order to present a detailed image of the performance of lower bounds LB_3, LB_4, LB_5, LB_6, LB_2^N and LB_4^N, a pairwise comparison between them is given in Table 1. In this table, we illustrate for each pair of lower bounds LB_{row} and LB_{col}, which are displayed in some given row and column, respectively, the percentage of times where $LB_{col} > LB_{row}$. We observe on classes A–F that LB_2^N outperforms LB_5 in 10.83% of instances and LB_6 in 26.38% of instances. However, on the new classes 1–2, we notice that LB_4^N provides a much better performance than the rest, since it outperforms LB_5 and LB_2^N in 77.5% and 75% of instances, respectively.

To get a better picture of the lower bounds performance, we provide in Table 2 the average percentage deviation (over the instances of each class) with respect to the maximal lower bound value, that is delivered by the considered lower bounds. Note that (-) means that the average CPU time is less than 10^{-2} s. From Table 2, we

Table 1 Pairwise comparison between lower bounds

	Classes A–F						Classes 1–2					
	LB_3	LB_4	LB_5	LB_6	LB_2^N	LB_4^N	LB_3	LB_4	LB_5	LB_6	LB_2^N	LB_4^N
LB_3	–	63.33	99.44	98.05	99.44	100	–	46.66	60.83	52.5	61.66	100
LB_4	36.66	–	99.72	98.33	99.72	100	50	–	61.66	54.16	61.66	100
LB_5	0.55	0.27	–	3.33	10.83	2.5	39.16	38.33	–	7.5	52.5	77.5
LB_6	1.94	1.66	23.33	–	26.38	2.77	47.5	45.83	67.5	–	72.5	79.16
LB_2^N	0.55	0.27	0	0	–	1.94	38.33	38.33	0	0	–	75
LB_4^N	0	0	97.5	97.22	98.05	–	0	0	20.83	19.16	23.33	–

Table 2 Relaxation performance by class

Class	LB_3		LB_4		LB_5		LB_6		LB_2^N		LB_4^N	
	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
A	29.9	–	30.26	–	0.04	–	0.3	–	**0**	–	19.81	0.82
B	27.8	–	28.26	–	0.06	–	0.33	–	**0.01**	–	17.82	0.81
C	23.96	–	24.16	–	0.89	–	2.56	–	**0.69**	–	13.73	0.79
D	32.24	–	32.29	–	0.04	–	0.03	–	**0**	–	21.46	0.84
E	53.96	–	46.26	–	0.003	–	0.02	–	**0**	–	33.09	0.82
F	53.32	–	45.85	–	0.02	–	0.22	–	**0**	–	32.93	0.85
1	10.35	–	11	–	1.78	–	2.19	–	1.56	–	**0.92**	0.58
2	10.42	–	10.42	–	13.89	–	19.26	–	13.66	–	**0.11**	0.69
Avg	30.24	–	28.56	–	2.09	–	3.11	–	**1.99**	–	17.48	0.77

observe that the average gaps significantly depend on the classes. On one hand, LB_2^N exhibits an average gap of 1.99% on all classes. However, for the instances of class 2, its average gap jumps to 13.66%. On the other hand, LB_4^N presents a much better performance on the new classes. Indeed, the average gap of this bound is equal to 0.92% and 0.11% on class 1 and class 2, respectively.

5 Conclusion

This paper addressed the two-machine flow shop problem with time delays. We recalled the lower bounds of the literature and proposed new ones. In particular, the linear relaxation of a mathematical formulation with the consideration of a valid inequality and a dominance rule provides the best performance on a set of 120 new instances. Future research needs to be focused on investigating new valid inequalities and dominance rules in order to improve the resolution of the considered model.

Acknowledgements This work is carried out in the framework of the Labex MS2T (Reference ANR-11-IDEX-0004-02). It is also partially supported by the ATHENA project (ANR-13-BS02-0006-02).

References

1. Dell'Amico, M.: Shop problems with two machines and time lags. Oper. Res. **44**(5), 777–787 (1996)
2. Mitten, L.G.: Sequencing n jobs on two machines with arbitrary time lags. Manag. Sci. **5**(3), 293–298 (1959)
3. Yu, W.: The two-machine flow shop problem and the one-machine total tardiness problem. Ph.D. thesis, Eindhoven University of Technology, The Netherlands (1996)