On Some Lower Bounds for the Permutation Flowshop Problem

Sebastian Cáceres Gelvez[1,2], Thu Huong Dang[1] and Adam N. Letchford[1]

[1]Department of Management Science, Lancaster University Management School, Lancaster, United Kingdom. E-mail: {s.caceresgelvez,t.h.dang,a.n.letchford}@lancaster.ac.uk

[2]Grupo de Investigación EUREKA UDES, Facultad de Ingenierías, Universidad de Santander, Cúcuta, Colombia

To appear in *Computers and Operations Research*

Abstract

The permutation flowshop problem with makespan objective is a classic machine scheduling problem, known to be $\mathcal{NP}$-hard in the strong sense. We analyse some of the existing lower bounds for the problem, including the "job-based" and "machine-based" bounds, a bound from linear programming (LP), and a recent bound of Kumar and co-authors. We show that the Kumar *et al.* bound dominates the machine-based bound, but the LP bound is stronger still. On the other hand, the LP bound does not, in general, dominate the job-based bound. Based on this, we devise simple iterative procedures for strengthening the Kumar *et al.* and LP bounds. Computational results are encouraging. In particular, we are able to obtain improved lower bounds for the "hard, small" instances of Vallada, Ruiz and Framinan.

Keywords: flowshop scheduling; permutation flowshops; lower bounds

1 Introduction

Machine scheduling problems have received a great deal of attention from the Operational Research and Optimisation communities, and there is a huge literature on them, including several textbooks (e.g., [1, 2, 3, 21]). Here, we focus on the *permutation flowshop scheduling problem with makespan objective*, or PFM for short.

In the PFM, there are m machines numbered from 1 to m, along with n jobs. Each machine can process only one job at a time, and each job

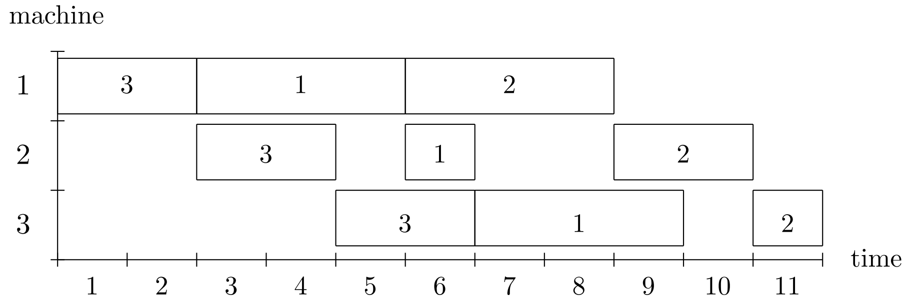

Figure 1: Gantt chart.

must be processed on machine 1, then machine 2, and so on. The amount of time taken to process job j on machine i is known and deterministic, and denoted by p_{ij}. A feasible solution is a permutation of the set of jobs, or *sequence*, and each machine must process the jobs in the order specified by that sequence. The goal is to minimise the time taken to finish processing the last job on the last machine, commonly called the *makespan*.

Let us suppose, for example, that $m = n = 3$, and the processing time matrix is

$$\begin{bmatrix} 3 & 3 & 2 \\ 1 & 2 & 2 \\ 3 & 1 & 2 \end{bmatrix},$$

where rows and columns correspond to machines and jobs, respectively. One can check that an optimal solution, with makespan 11, is obtained by sequencing the jobs in the order 3, 1, 2. The Gantt chart that corresponds to this solution is shown in Figure 1.

Johnson [14] showed that the PFM with $m = 2$ can be solved in polynomial time. For general m, however, the PFM is $\mathcal{NP}$-hard in the strong sense [17]. A wide variety of heuristics have been proposed (see, e.g., the surveys [9, 10, 19, 22]). There are also several exact approaches (e.g., [4, 6, 11, 13, 16, 20, 23, 25, 26]).

Here, however, we focus on *lower-bounding* procedures. In particular, we consider the following four lower bounds:

1. The "machine-based" bound of Ignall and Schrage [12].

2. The "job-based" bound of McMahon and Burton [18].

3. A bound obtained by solving the linear programming (LP) relaxation of a mixed-integer linear program (MILP) due to Stafford *et al.* [23].

4. A bound recently proposed by Kumar *et al.* [15].

We begin by analysing the four bounds from a theoretical point of view. Amongst other things, we prove the following:

- The Kumar *et al.* bound dominates the machine-based bound.
- The LP bound dominates the Kumar *et al.* bound.
- The job-based bound is in general incomparable with the other three bounds.

After that, we propose some simple iterative procedures for strengthening the Kumar *et al.* and LP bounds. These procedures ensure that the resulting bounds are at least as strong as the job-based bound. Finally, we present some computational results on benchmark PFM instances. The results are rather encouraging. In particular, we are able to obtain improved lower bounds for the "hard, small" instances of Vallada *et al.* [27].

The paper has the following structure. Section 2 gives a brief overview of the relevant literature. Section 3 contains our results on the lower bounds. Section 4 describes our strengthening procedures, and Section 5 presents the computational results. Some concluding remarks are made in Section 6.

We assume throughout that the reader is familiar with the basics of integer programming. For detailed treatments of the topic, see, e.g., the books [5, 7]. We also assume without loss of generality that the processing times are non-negative integers. Finally, we let OPT denote the optimal makespan.

2 Literature Review

We now review the relevant literature. Subsection 2.1 recalls the machine-based and job-based bounds. Subsection 2.2 presents the MILP formulation from [23]. Finally, Subsection 2.3 describes the lower-bounding procedure in [15]. For more on flowshop scheduling, see the book [8].

2.1 The machine-based and job-based bounds

We begin by recalling some simple lower bounds on the makespan. The first bound, which we will call L_M, is obtained by computing the load of each machine and picking the largest:

$$L_M = \max_{1 \le i \le m} \left\{ \sum_{j=1}^{n} p_{ij} \right\}.$$

A way to improve L_M was given in [12]. Consider a particular machine i. If $i > 1$, then machine i cannot start processing its first job until that job has been processed on the preceding machines. Also, if $i < m$ then, after machine i has finished processing its last job, that job must be processed on the subsequent machines. Thus, the makespan must be at least:

$$P_i = \sum_{j=1}^{n} p_{ij} + \min_{1 \le j \le n} \left\{ \sum_{r<i} p_{rj} \right\} + \min_{1 \le j \le n} \left\{ \sum_{r>i} p_{rj} \right\}.$$

This enables us to increase L_M to:

$$L_M^+ = \max_{1 \le i \le m} \{P_i\}.$$

Another bound, which we will call L_J, is obtained by computing the total processing time of each job and picking the largest:

$$L_J = \max_{j=1}^{n} \left\{ \sum_{i=1}^{m} p_{ij} \right\}.$$

A way to improve L_J was given in [18]. Consider a particular job j. Every other job either comes before or after j. Thus, the makespan must be at least:

$$Q_j = \sum_{i=1}^{m} p_{ij} + \sum_{s \neq j} \min \{p_{1s}, p_{ms}\}.$$

This enables us to increase L_J to:

$$L_J^+ = \max_{j=1}^{n} \{Q_j\}.$$

2.2 The Stafford *et al.* formulation

Stafford *et al.* [23] formulated PFM as an MILP, by adapting the formulation of the job-shop scheduling problem in [28]. We have a binary variable x_{jk} for $j, k = 1, \ldots, n$, taking the value 1 if and only if job j is assigned to the k-th position in the sequence. We also have non-negative continuous variables f_{ik}, representing the time at which machine i finishes processing the k-th job in the sequence. The formulation is:

$$\begin{aligned}
\min \quad & f_{mn} && && (1)\\
\text{s.t.} \quad & \textstyle\sum_{k=1}^{n} x_{jk} = 1 && (j = 1, \ldots, n) && (2)\\
& \textstyle\sum_{j=1}^{n} x_{jk} = 1 && (k = 1, \ldots, n) && (3)\\
& f_{11} = \textstyle\sum_{j=1}^{n} p_{1j} x_{j1} && && (4)\\
& f_{i,k+1} \ge f_{ik} + \textstyle\sum_{j=1}^{n} p_{ij} x_{j,k+1} && (i = 1, \ldots, m;\ k = 1, \ldots, n-1) && (5)\\
& f_{i+1,k} \ge f_{ik} + \textstyle\sum_{j=1}^{n} p_{i+1,j} x_{jk} && (i = 1, \ldots, m-1;\ k = 1, \ldots, n) && (6)\\
& x_{ik} \in \{0, 1\} && (i = 1, \ldots, m;\ k = 1, \ldots, n) && (7)\\
& f_{ik} \ge 0 && (i = 1, \ldots, m;\ k = 1, \ldots, n). && (8)
\end{aligned}$$

The objective function (1) is self-explanatory. The constraints (2) and (3) are standard assignment constraints. The constraint (4) states that the finishing time of the first job on the first machine is equal to the processing time of that job. The constraints (5) state that, on any given machine, the finishing time of a job is at least the finishing time of the previous job plus the time taken to process the given job. The constraints (6) state that, for any given job, the finishing time of that job on a machine is at least the finishing time of that job on the previous machine plus the time taken to process the given job. The constraints (7) and (8) are trivial.

2.3 The Kumar *et al.* bound

Very recently, Kumar *et al.* [15] presented a lower-bounding procedure for the permutation flowshop problem with the objective of minimising the sum of the completion times. We describe it here, because it also yields a lower bound for PFM.

For each machine i, we sort the p_{ij} values in non-decreasing order. The sorted values are then denoted by $\tau_{i1}, \ldots, \tau_{in}$. We also let σ_{ik} denote $\sum_{k'=1}^{k} \tau_{i,k'}$. That is, σ_{ik} is the minimum time needed for machine i to process k jobs.

Next, for $i = 1, \ldots, m$ and $k = 1, \ldots, n$, we compute a lower bound γ_{ik} on the time at which machine i finishes processing the k-th job in the sequence. This is done as follows. For all k, γ_{1k} is set to σ_{1k}. For all i, γ_{i1} is set to

$$\min_j \left\{ \sum_{i'=1}^{i} p_{i',j} \right\}.$$

For $i = 2, \ldots, m$ and $k = 2, \ldots, n$, γ_{ik} is set to the larger of the following four values:

$$\begin{aligned}
\beta_{ik}^1 &= \sigma_{ik} + \gamma_{i-1,1} \\
\beta_{ik}^2 &= \sigma_{i,k-1} + \gamma_{i1} \\
\beta_{ik}^3 &= \max_{1 \le i' \le i} \left\{ \gamma_{i',k-1} + \min_j \left\{ \sum_{i''=i'}^{i} p_{i'',j} \right\} \right\} \\
\beta_{ik}^4 &= \max_{1 \le i' < i} \left\{ \gamma_{i',k} + \min_j \left\{ \sum_{i''=i'+1}^{i} p_{i'',j} \right\} \right\}.
\end{aligned}$$

At the end of the procedure, γ_{mn} is a lower bound for the PFM.

3 Analysis of Existing Bounds

In this section, we analyse some of the existing lower bounds. Subsection 3.1 concerns the machine-based and job-based bounds. Subsection 3.2 concerns the Kumar *et al.* bound. Finally, Subsection 3.3 concerns the bound obtained by solving the LP relaxation of the Stafford *et al.* MILP.

3.1 On the machine-based and job-based bounds

First, we prove some simple results about the "machine-based" bounds (L_M and L_M^+) and the "job-based" bounds (L_J and L_J^+).

Lemma 1 $L_M \geq OPT/m$.

Proof. We can obtain a feasible PFM solution by processing all of the jobs on the first machine, then processing all of the jobs on the second machine, and so on. The makespan of this solution is $\sum_{i=1}^m \sum_{j=1}^n p_{ij}$, which is no more than $m\, L_M$ by definition. Thus, $OPT \leq m\, L_M$ or, equivalently, $L_M \geq OPT/m$. □

Lemma 2 $L_J \geq OPT/n$.

Proof. Similar to the previous lemma. □

Lemma 3 *For any $m, n \geq 1$, and any small $\epsilon > 0$, there exists a PFM instance such that $L_M^+ < L_J/(m - \epsilon)$.*

Proof. Let m and n be given. Let b be a large positive integer. Suppose that $p_{i1} = b$ for all i, but all other processing times are equal to 1. One can check that $L_J = mb$ and $L_M^+ = b + m + n - 2$. As b tends to infinity, L_J/L_M^+ tends to m from below. □

Lemma 4 *For any $m, n \geq 1$, and any small $\epsilon > 0$, there exists a PFM instance such that $L_J^+ < L_M/(n - \epsilon)$.*

Proof. Similar to the previous lemma, except that we set $p_{1j} = b$ for all j, and all other processing times to 1. We then have $L_M = nb$ and $L_J^+ = b + m + n - 2$. □

Of course, these last two lemmas imply that L_M^+ and L_J^+ can be arbitrarily close to OPT/m and OPT/n, respectively.

3.2 On the Kumar *et al.* bound

Next, we consider the Kumar *et al.* bound.

Lemma 5 $\gamma_{mn} \geq L_M^+$.

Proof. Consider a fixed machine $i < m$. We have:

$$\begin{aligned}
\gamma_{mn} &\geq \beta^4_{mn} \\
&\geq \gamma_{in} + \min_j \left\{ \sum_{i''=i+1}^{n} p_{i'',j} \right\} \\
&\geq \beta^1_{in} + \min_j \left\{ \sum_{i''=i+1}^{n} p_{i'',j} \right\} \\
&= \sigma_{in} + \gamma_{i-1,1} + \min_j \left\{ \sum_{i''=i+1}^{n} p_{i'',j} \right\} \\
&= \sum_{j=1}^{n} p_{ij} + \min_j \left\{ \sum_{i''=1}^{i-1} p_{i'',j} \right\} + \min_j \left\{ \sum_{i''=i+1}^{n} p_{i'',j} \right\}.
\end{aligned}$$

This last expression is equal to P_i for the given i. Moreover, we have $\gamma_{mn} \geq \beta^1_{mn} = P_m$. Thus, $\gamma_{mn} \geq P_i$ for all i. □

Together with Lemma 1, this implies that $\gamma_{mn} \geq OPT/m$.

A natural question at this point is whether γ_{mn} can ever be larger than L^+_M. The following example shows that, in fact, it can be larger than both L^+_M and L^+_J simultaneously:

Example 1: Suppose that $m = n = 4$, and let the matrix of processing times be

$$\begin{bmatrix} 1 & 4 & 4 & 4 \\ 6 & 4 & 4 & 4 \\ 1 & 2 & 2 & 2 \\ 2 & 4 & 4 & 4 \end{bmatrix}.$$

One can check that $OPT = 25$, $L_M = 18$, $L^+_M = 22$, $L_J = 14$ and $L^+_J = 23$. One can also check that $\gamma_{11} = 1$, $\gamma_{12} = 5$, $\gamma_{21} = 7$, $\gamma_{22} = 11$, $\gamma_{13} = 9$, $\gamma_{31} = 8$, $\gamma_{32} = 13$, $\gamma_{23} = 15$, $\gamma_{33} = 17$, $\gamma_{14} = 13$, $\gamma_{24} = 19$, $\gamma_{34} = 21$, $\gamma_{41} = 10$, $\gamma_{42} = 16$, $\gamma_{43} = 20$, and $\gamma_{44} = 24$. Thus, the Kumar *et al.* bound is 24 for this instance, which is greater than both L^+_M and L^+_J. □

On the other hand, we have the following negative result, concerning the relationship between γ_{mn} and the "job-based" lower bound L_J.

Lemma 6 *γ_{mn} can be smaller than L_J.*

Proof. Consider again the PFM instance described in the proof of Lemma 3. One can check that $L_J = mb$ for this instance. One can also check that (i) $\gamma_{ik} = i+k-1$ for $i = 1, \dots, m$ and $k = 1, \dots, n-1$, and (ii) $\gamma_{in} = b+n+i-2$ for $i = 1, \dots, n$. So $\gamma_{mn} = b + m + n - 2$ for this instance. Setting m to 2 yields the result. □

3.3 On the LP bound

The *continuous relaxation* of the Stafford *et al.* formulation is obtained by replacing the binary conditions (7) with the weaker conditions $x_{jk} \in [0,1]$. This relaxation is an LP, which can be solved efficiently. For brevity, we will just call it "the LP". We will also call the resulting lower bound the *LP bound*, and we denote it by L_c.

We will show that the LP bound is stronger than the Kumar *et al.* bound. To do this, we will need a series of lemmas.

Lemma 7 *If a vector $x \in [0,1]^{n^2}$ satisfies equations (2) and (3), then*

$$\sum_{k'=1}^{k} \left(\sum_{j=1}^{n} p_{ij} x_{j,k'} \right) \geq \sigma_{ik} \tag{9}$$

holds for $i = 1, \ldots, m$ and $k = 1, \ldots, n$.

Proof. We set up a minimum-cost flow problem to find a vector x that minimises the left-hand side of (9). For $j = 1, \ldots, n$, we have a source node with a supply of 1. For $k' = 1, \ldots, k$, we have a sink node with a demand of 1. The flow in the arc from source j to sink k' represents the value of $x_{jk'}$, and the cost of the arc is set to p_{ij}.

Since all supplies and demands are integral, the minimum-cost flow problem has an integral optimal solution. Moreover, since $k \leq n$, such a solution uses exactly k arcs. Now, observe that the cost of an arc does not depend on the sink. Thus, an optimal solution is obtained by sending one unit of flow from the source with the smallest p_{ij} value to the first sink, then one unit of flow from the source with the second smallest p_{ij} value to the second sink, and so on. This flow has a cost of σ_{ik}. □

Lemma 8 *If (x^*, f^*) is a feasible solution to the LP, then $f^*_{1k} \geq \gamma_{1k}$ for $k = 1, \ldots, n$.*

Proof. The LP contains the equation (4), along with the following constraints of type (5):

$$f_{1,k'+1} \geq f_{1,k'} + \sum_{j=1}^{n} p_{1j} x_{j,k'+1} \quad (k' = 1, \cdots, k-1).$$

Adding all of these and simplifying yields:

$$f_{1k} \geq \sum_{k'=1}^{k} \left(\sum_{j=1}^{n} p_{1j} x_{j,k'} \right) \geq \sigma_{1k} = \gamma_{1k},$$

where the second inequality follows from Lemma 7. □

Lemma 9 *If (x^*, f^*) is a feasible solution to the LP, then $f^*_{i1} \geq \gamma_{i1}$ for $i = 1, \dots, m$.*

Proof. The LP contains the equation (4), along with the following constraints of type (6):

$$f_{i'+1,1} \geq f_{i',1} + \sum_{j=1}^{n} p_{i'+1,j} x_{j1} \quad (i' = 1, \cdots, i-1).$$

Adding all of these and simplifying yields:

$$f_{i1} \geq \sum_{j=1}^{n} \left(\sum_{i'=1}^{i} p_{i',j} \right) x_{j1} \geq \sum_{j=1}^{n} \gamma_{i1} x_{j1} = \gamma_{i1} \sum_{j=1}^{n} x_{j1} = \gamma_{i1},$$

where the second inequality follows from the definition of γ_{i1} and the last equation follows from (3). □

Lemma 10 *If (x^*, f^*) is a feasible solution to the LP, then*

$$f_{ik} \geq \beta^1_{ik}$$

holds for $i = 2, \dots, m$ and $k = 2, \dots, n$.

Proof. Note that the LP contains the following constraint of type (6):

$$f_{i1} \geq f_{i-1,1} + \sum_{j=1}^{n} p_{ij} x_{j1},$$

along with the following constraints of type (5):

$$f_{i,k'} \geq f_{i,k'-1} + \sum_{j=1}^{n} p_{ij} x_{j,k'} \qquad (k' = 2, \cdots, k).$$

Summing these together and simplifying yields

$$f_{ik} \geq f_{i-1,1} + \sum_{k'=1}^{k} \sum_{j=1}^{n} p_{ij} x_{j,k'}.$$

Together with Lemmas 7 and 9, this gives:

$$f_{ik} \geq \sigma_{ik} + \min_{j} \left\{ \sum_{i'=1}^{i-1} p_{i',j} \right\},$$

which proves the result. □

Lemma 11 *If (x^*, f^*) is a feasible solution to the LP, then*

$$f_{ik} \geq \beta_{ik}^2$$

holds for $i = 2, \ldots, m$ and $k = 2, \ldots, n$.

Proof. Note that the LP contains the following constraints of type (5):

$$f_{i,k'+1} \geq f_{i,k'} + \sum_{j=1}^{n} p_{ij} x_{j,k'} \qquad (k' = 1, \cdots, k-1).$$

Summing these together and simplifying yields:

$$f_{ik} \geq f_{i1} + \sum_{k'=2}^{k} \sum_{j=1}^{n} p_{ij} x_{j,k'}. \tag{10}$$

Now Lemma 9, together with the definition of γ_{i1}, shows that

$$f_{i1} \geq \min_j \left\{ \sum_{i'=1}^{i} p_{i',j} \right\}.$$

Moreover, the argument used in Lemma 7 shows that the second term on the right-hand side of (10) is at least $\sigma_{i,k-1}$. Thus, we have

$$f_{ik} \geq \sigma_{i,k-1} + \min_j \left\{ \sum_{i'=1}^{i} p_{i',j} \right\}$$

which proves the result. □

Lemma 12 *If (x^*, f^*) is a feasible solution to the LP, then*

$$f_{ik} \geq \max_{1 \leq i' < i} \left\{ f_{i',k} + \min_j \left\{ \sum_{i''=i'+1}^{i} p_{i'',j} \right\} \right\} \tag{11}$$

holds for $i = 2, \ldots, m$ and $k = 2, \ldots, n$.

Proof. The LP contains the following constraints of type (6):

$$f_{i''+1,k} \geq f_{i'',k} + \sum_{j=1}^{n} p_{i''+1,j} x_{jk} \quad (i'' = i', \cdots, i-1)$$

Adding all of these and simplifying yields:

$$\begin{aligned} f_{ik} &\geq f_{i',k} + \sum_{j=1}^{n} \Big(\sum_{i''=i'+1}^{i} p_{i'',j} \Big) x_{j,k} \qquad (12) \\ &\geq f_{i',k} + \min_j \left\{ \sum_{i''=i'+1}^{i} p_{i'',j} \right\} \Big(\sum_{j=1}^{n} x_{j,k} \Big) \\ &\geq f_{i',k} + \min_j \left\{ \sum_{i''=i'+1}^{i} p_{i'',j} \right\}. \end{aligned}$$

This inequality applies for every $1 \leq i' < i$. Therefore, inequality (11) is obtained. $\square$

Lemma 13 *If (x^*, f^*) is a feasible solution to the LP, then*

$$f_{ik} \geq \max_{1 \leq i' \leq i} \left\{ f_{i',k-1} + \min_j \left\{ \sum_{i''=i'}^{i} p_{i'',j} \right\} \right\}. \tag{13}$$

holds for $i = 2, \ldots, m$ and $k = 2, \ldots, n$.

Proof. The LP contains the following constraint of type (5):

$$f_{i',k} \geq f_{i',k-1} + \sum_{j=1}^{n} p_{i',j} x_{j,k}$$

Adding this to inequality (12) and simplifying yields:

$$\begin{aligned} f_{ik} &\geq f_{i',k-1} + \sum_{j=1}^{n} \Big(\sum_{i''=i'}^{i} p_{i'',j} \Big) x_{j,k} \\ &\geq f_{i',k-1} + \min_j \left\{ \sum_{i''=i'}^{i} p_{i'',j} \right\} \Big(\sum_{j=1}^{n} x_{j,k} \Big) \\ &\geq f_{i',k-1} + \min_j \left\{ \sum_{i''=i'}^{i} p_{i'',j} \right\} \end{aligned}$$

This inequality applies for every $1 \leq i' \leq i$. Therefore, inequality (13) is obtained. $\square$

Lemma 14 *If (x^*, f^*) is a feasible solution to the LP, then $f^*_{22} \geq \gamma_{22}$.*

Proof. Lemma 10 tells us that $f_{22} \geq \beta^1_{22}$, and Lemma 11 tells us that $f_{22} \geq \beta^2_{22}$. Moreover, Lemma 12 tells us that

$$f_{22} \geq f_{12} + \min_j \{p_{2,j}\} \geq \gamma_{12} + \min_j \{p_{2,j}\} = \beta^4_{22}.$$

Finally, Lemma 13 tells us that

$$\begin{aligned} f_{22} &\geq \max_{1 \leq i' \leq 2} \left\{ f_{i',1} + \min_j \left\{ \sum_{i''=i'}^{2} p_{i'',j} \right\} \right\} \\ &\geq \max_{1 \leq i' \leq 2} \left\{ \gamma_{i',1} + \min_j \left\{ \sum_{i''=i'}^{2} p_{i'',j} \right\} \right\} = \beta^3_{22}. \end{aligned}$$

$\square$

Armed with Lemmas 8 to 14, we can now present the main result of this subsection:

Theorem 1 *If* (x^*, f^*) *is a feasible solution to the LP, then* $f^*_{ik} \geq \gamma_{ik}$ *for all* i *and* k*.*

Proof. Lemmas 8 and 9 show that the result holds for $i = 1$ and $k = 1$. Together with Lemma 14, this implies that the result holds also for $i+k \leq 4$. To complete the proof, we will use induction on $i + k$. That is, we will show that, if (a) N is an integer between 4 and $m + n - 1$, and (b) $f_{ik} \geq \gamma_{ik}$ for $i + k \leq N$, then $f_{ik} \geq \gamma_{ik}$ also holds when $i + k = N + 1$.

So, let N be given, and let i, k be integers such that $2 \leq i \leq m$, $2 \leq k \leq n$ and $i + k = N + 1$. From Lemmas 10 and 11, we already know that

$$f_{ik} \geq \max\left\{\beta^1_{ik},\, \beta^2_{ik}\right\}. \tag{14}$$

Now, Lemma 12 tells us that

$$f_{ik} \quad \geq \quad \max_{1 \leq i' < i} \left\{ f_{i',k} + \min_j \left\{ \sum_{i''=i'+1}^{i} p_{i'',j} \right\} \right\}.$$

By the induction hypothesis, the term $f_{i',k}$ on the right-hand side is at least as large as $\gamma_{i',k}$, since $i' + k \leq N$. From this, we conclude that

$$f_{ik} \geq \beta^3_{ik}. \tag{15}$$

Similarly, Lemma 13 tells us that

$$f_{ik} \quad \geq \quad \max_{1 \leq i' \leq i} \left\{ f_{i',k-1} + \min_j \left\{ \sum_{i''=i'}^{i} p_{i'',j} \right\} \right\}$$

By the induction hypothesis, the term $f_{i',k-1}$ on the right-hand side is at least as large as $\gamma_{i',k-1}$, since $i' + k - 1 \leq N$. From this, we conclude that

$$f_{ik} \geq \beta^4_{ik}. \tag{16}$$

The result then follows from (14), (15) and (16). □

Corollary 1 $L_c \geq \gamma_{mn}$*, i.e., the LP bound dominates the Kumar* et al. *bound.*

Together with the previous results, this gives the following chain of inequalities:

$$OPT \geq L_c \geq \gamma_{mn} \geq L^+_M \geq L_M \geq OPT/m.$$

A natural question at this point is whether L_c can ever be larger than γ_{mn}. It turns out that L_c can be larger than both γ_{mn} and L_J^+ simultaneously:

Example 1 (cont.): Recall that, for this instance, we have $OPT = 25$, $L_J^+ = 23$ and $\gamma_{mn} = 24$. One can check that $L_c = 25$ for this instance. $\square$

On the other hand, we have the following negative result:

Proposition 1 *For any $m, n \geq 2$, and any small $\epsilon > 0$, there exists a PFM instance such that $L_c/L_J < \epsilon + (m+n-1)/mn$.*

Proof. Consider once more the PFM instance described in the proofs of Lemmas 3 and 6, and recall that $L_J = mb$. We can obtain a feasible LP solution by setting every x variable to $1/n$ and setting f_{ik} to $(i+k-1)(b+n-1)/n$ for all i and k. Thus, $L_c \leq (m+n-1)(b+n-1)/n$. As b tends to infinity, the ratio L_c/L_J tends to $(m+n-1)/mn$ from above. $\square$

We remark that, by setting m to a large value, the ratio $(m+n-1)/mn$ can be made to approach $1/n$. Similarly, by setting n to a large value, the ratio can be made to approach $1/m$. Note also that $L_c \geq L_M^+ \geq L_J/m$. We suspect that $L_c \geq L_J/n$ as well. In fact, we make the following conjecture.

Conjecture 1 *$L_c \geq OPT/n$.*

To close this section, we remark that L_c does not always take integer values. This is shown in the following example.

Example 2: Suppose that $m = n = 3$, and let the matrix of processing times be

$$\begin{bmatrix} 2 & 1 & 3 \\ 3 & 4 & 5 \\ 4 & 3 & 5 \end{bmatrix}.$$

One can check that $L_c = 17.5$ for this example. $\square$

4 Improved Bounds

In this section, we present some improved bounding procedures. First, in Subsection 4.1, we show how to implement the Kumar *et al.* procedure so that it runs in $O(m^2n + mn\log n)$ time. Then, in Subsection 4.2, we present a bounding procedure that is slightly slower than the one of Kumar *et al.*, but yields a stronger lower bound in some cases. Finally, in Subsection 4.3, we use the output from the procedure in Subsection 4.2 to strengthen the LP bound.

4.1 Efficient implementation

Observe that, in the definition of both β^3_{ik} and β^4_{ik}, we take a maximum over i' and a minimum over j. Thus, if the Kumar *et al.* procedure is implemented in a naive way, it takes $O\big(m^2n^2\big)$ time. A more efficient implementation is given in Algorithms 1 and 2. Algorithm 1 computes the σ_{ik} values. It also computes values called α_{ij}, where

$$\alpha_{ij} = \sum_{i'=1}^{i} p_{i',j} \qquad \big(i = 1, \ldots, m;\ j = 1, \ldots, n\big).$$

Algorithm 2 then uses those values to compute the β and γ values.

One can check that Algorithm 1 runs in in $O(mn \log n)$ time and $O(mn)$ space, whereas Algorithm 2 runs in $O\big(m^2n\big)$ and $O(mn)$ space. We remark that, in practice, m tends to be smaller than n.

Algorithm 1: Computing the σ and α values

input : number of machines m, number of jobs n, processing times p_{ij}
for $i = 1, \ldots, m$ **do**
  Sort the p_{ij} values in $O(n \log n)$ time;
  Let τ_{i1} to τ_{in} be the sorted values;
  Set σ_{i1} to τ_{i1};
  for $k = 2, \ldots, n$ **do**
    Set σ_{ik} to $\sigma_{i,k-1} + \tau_{ik}$;
  end
end
for $j = 1, \ldots, n$ **do**
  Set α_{1j} to p_{1j};
end
for $i = 2, \ldots, m$ **do**
  for $j = 1, \ldots, n$ **do**
    Set α_{ij} to $\alpha_{i-1,j} + p_{ij}$;
  end
end
output: Arrays containing the σ_{ik} and α_{ij} values

4.2 Strengthened procedure

Now, recall from Subsection 3.2 that γ_{mn} is not guaranteed to be as strong as L_J. In this subsection, we will improve the Kumar *et al.* procedure in such a way that the resulting bound is guaranteed to be at least as large as L_J^+.

We will need a little additional notation. For $i = 1, \ldots, m$ and $j = 1, \ldots, n$, we define

$$\lambda(i,j) = \min\big\{p_{1j}, p_{ij}\big\}.$$

Algorithm 2: Computing the β and γ values

input : number of machines m, number of jobs n,
arrays containing the σ_{ik} and α_{ik} values

for $k = 1, \ldots, n$ **do**
    Set γ_{1k} to σ_{1k};
end
for $i = 1, \ldots, m$ **do**
    Set γ_{i1} to $\min_j \left\{\alpha_{ij}\right\}$;
end
for $i = 2, \ldots, m$ **do**
    for $k = 2, \ldots, n$ **do**
        Let $\beta^1_{ik} = \sigma_{ik} + \gamma_{i-1,1}$;
        Let $\beta^2_{ik} = \sigma_{i,k-1} + \gamma_{i1}$;
        Set γ_{ik} to the larger of β^1_{ik} and β^2_{ik};
    end
end
for $i = 2, \ldots, m$ **do**
    for $i' = 1, \ldots, i$ **do**
        for $j = 1, \ldots, n$ **do**
            Let $\delta_j = \alpha_{ij} - \alpha_{i'-1,j}$;
        end
        Let $\Delta = \min_j\{\delta_j\}$;
        for $k = 2, \ldots, n$ **do**
            if $\gamma_{i',k-1} + \Delta > \gamma_{ik}$ **then**
                Increase γ_{ik} to $\gamma_{i',k-1} + \Delta$;
            end
        end
    end
    for $i' = 1, \ldots, i-1$ **do**
        for $j = 1, \ldots, n$ **do**
            Let $\delta_j = \alpha_{ij} - \alpha_{i',j}$;
        end
        Let $\Delta = \min_j \left\{\delta_j\right\}$;
        for $k = 2, \ldots, n$ **do**
            if $\gamma_{i',k} + \Delta > \gamma_{ik}$ **then**
                Increase γ_{ik} to $\gamma_{i',k} + \Delta$;
            end
        end
    end
end

output: Array containing the γ_{ik} values

We then have the following lemma.

Lemma 15 *Consider a fixed triple (i,j,k). If job j is one of the first k jobs in the sequence, then the time at which machine i finishes processing the k-th job in the sequence must be at least*

$$\mu(i,j,k) = \alpha_{ij} + \min\left\{ \sum_{j' \in S} \lambda(i,j') : S \subseteq \{1,\ldots,n\} \setminus \{j\},\ |S| = k-1 \right\}.$$

Proof. By definition, the total amount of time needed to process job j on the first i machines is α_{ij}. Now, consider any job $j' \neq j$ that is also one of the first k jobs in the sequence. This job must come either before or after job j. If it comes before j, then machine 1 must process it before it starts processing job j. If it comes after job j, then machine i must process it before it finishes the k-th job in the sequence. In either case, job j' contributes at least $\lambda(i,j')$ to the time at which machine i finishes the k-th job. The result then follows from the fact that there are $k-1$ candidates for j'. □

This means that, for a given i and k, the time at which machine i finishes the k-th job must be at least the k-th smallest value of $\mu(i,j,k)$. Let us call this value β^5_{ik}. We can then improve the Kumar *et al.* procedure as follows. In Algorithm 2, instead of setting γ_{ik} to the larger of β^1_{ik} and β^2_{ik}, we set it to the larger of β^1_{ik}, β^2_{ik} and β^5_{ik}.

One can check that

$$\beta^5_{mn} = \max_j \left\{ \alpha_{mj} + \sum_{j' \neq j} \lambda(i,j') \right\} = L^+_J.$$

Thus, at the end of the improvement procedure, we can be sure that γ_{mn} will be no smaller than L^+_J.

To compute the β^5 coefficients efficiently, we use Algorithm 3. One can check that the algorithm runs in $O\big(mn^2 \log n\big)$ time and $O(mn)$ space.

We will call the strengthened lower bound γ^+_{mn}. From the above discussion, it follows that $\gamma^+_{mn} \geq \max\left\{\gamma_{mn}, L^+_J\right\}$. Interestingly, the inequality can be strict. This is shown in the following example.

Example 3: Suppose that $m = 4$ and $n = 3$, and let the matrix of processing times be

$$\begin{bmatrix} 99 & 73 & 84 \\ 91 & 10 & 72 \\ 94 & 94 & 96 \\ 29 & 3 & 33 \end{bmatrix}.$$

One can check that $\gamma_{mn} = 370$, $L^+_J = 349$ and $\gamma^+_{mn} = 444$. □

Algorithm 3: Computing the β^5_{ik} values

```
input : number of machines m, number of jobs n, processing times p_ij
for i = 1, ..., m do
    for j = 1, ..., n do
        Let λ(i, j) = min {p_1j, p_ij};
    end
    Sort the λ(i, j) values in non-decreasing order;
    for k = 1, ..., n do
        Let λ'(i, k) be the k-th value in the sorted list;
    end
end
Create a 1-dimensional array MU of size n;
for i = 2, ..., m do
    Set SUM to λ'(i, 1);
    for k = 2, ..., n do
        for j = 1, ..., n do
            Set MU[j] to α_ij + max {SUM, SUM + λ'(i, k) − λ(i, j)};
        end
        Sort the array MU in non-decreasing order;
        Set β^5_ik to the k-th element in the array MU;
        Increase SUM by λ'(i, k);
    end
end
output: Array containing the β^5_ik values
```

4.3 Strengthening the LP relaxation

Observe that the procedure in Subsection 4.2 attempts to increase not only γ_{mn}, but also γ_{ik} for all i and k. Let us call the strengthened values γ^+_{ik}. By definition, in any feasible solution to the Stafford *et al.* MILP, f_{ik} must be at least γ^+_{ik} for all i and k. Accordingly, we can strengthen the LP relaxation by adding the trivial constraint $f_{ik} \geq \gamma^+_{ik}$ for $i = 1, \ldots, m$ and $k = 1, \ldots, n$. We will call the resulting lower bound L^+_c.

From the previous results, we have $L^+_c \geq L_c \geq \gamma_{mn} \geq L^+_M$ and $L^+_c \geq \gamma^+_{mn} \geq L^+_J$. It turns out that L^+_c can be strictly larger than both L_c and γ^+_{mn} simultaneously.

Example 3 (cont): Recall that $\gamma^+_{mn} = 444$. One can check that $L_c \approx 434.11$ and $L^+_c \approx 448.98$. □

This example also shows that L^+_c can be fractional.

5 Computational Results

In order to gain further insight into the relative strengths and weaknesses of the various lower bounds, we conducted some computational experiments on benchmark PFM instances. Subsection 5.1 gives the results for the classical instances of Taillard [24], and Subsection 5.2 gives the results for the "small, hard" instances presented in Vallada *et al.* [27].

All algorithms were coded in C$^\#$, compiled with Visual Studio 2022, and run on a 2.4GHz Intel i5-1135G7 processor with 16GB of RAM under Windows 10. To solve the LPs, we used the simplex solver of `CPLEX` v. 12.10, with default settings.

5.1 Taillard instances

The Taillard instances have $n \in \{20, 50, 100, 200, 500\}$ and $m \in \{5, 10, 20\}$. There are ten instances for each combination of n and m with $n \in \{20, 50, 100\}$ and $m \in \{5, 10, 20\}$. There are also ten instances for each of the following combinations of n and m: $(200, 10)$, $(200, 20)$ and $(500, 10)$. This makes 120 instances in total. The optimal values for the instances with $m \in \{5, 10\}$ can be found in [24]. At the time of writing, the best known lower and upper bounds for the other instances were available on Taillard's personal web site.[1]

For each instance, we computed the following six lower bounds: L^+_J, L^+_M, γ_{mn}, γ^+_{mn}, L_c and L^+_c. Then, for each instance and each bound, we computed the gap between the lower bound and the best known upper bound, expressed as a percentage of the upper bound. Table 1 shows the average

[1] `http://mistic.heig-vd.ch/taillard/` (accessed 16/12/22)

Table 1: Average percentage gaps for Taillard instances

m	n	L_J^+	L_M^+	γ_{mn}	γ_{mn}^+	L_c	L_c^+
5	20	13.88	2.35	2.35	2.35	1.70	1.68
	50	26.39	0.80	0.80	0.80	0.59	0.59
	100	28.82	1.06	1.06	1.06	0.64	0.64
10	20	16.07	8.48	8.48	8.13	6.48	6.31
	50	21.84	2.10	2.10	2.10	1.68	1.65
	100	26.88	0.80	0.80	0.80	0.56	0.55
	200	28.49	0.66	0.66	0.66	0.45	0.45
20	20	14.73	17.00	17.00	14.25	13.30	12.50
	50	22.17	8.17	8.17	8.17	6.99	6.99
	100	25.79	3.74	3.74	3.74	3.00	3.00
	200	28.60	1.54	1.54	1.54	1.18	1.18
	500	31.12	0.54	0.54	0.54	0.44	0.44

percentage gap for each set of ten instances and each bound. (The detailed results will be made available at the Lancaster University Data Repository, under the heading "Permutation Flowshop Problem".)[2]

We see that L_J^+ was extremely weak compared to the other bounds, with the single exception of the case $m = n = 20$, where it was slightly stronger than L_M^+ and γ_{mn}. Remarkably, the bounds L_M^+ and γ_{mn} were identical for all 120 instances. Moreover, γ_{mn}^+ was better than γ_{mn} only for some of the instances with $m \in \{10, 20\}$ and $n = 20$. (In fact, γ_{mn}^+ was better on only 9 out of 120 instances.) As for the LP-based bounds, we see that L_c was always stronger than γ_{mn}, which is consistent with Corollary 1. We also see that L_c^+ was a little stronger than L_c in some cases. (In fact, it was better on only 8 out of 120 instances.) We remark that both L_c and L_c^+ reached the optimal value for the `tai-20-5-7` instance.

It is also apparent in the table that all bounds apart from L_J^+ tend to get weaker as m increases. Interestingly, however, all bounds apart from L_J^+ tend to get *stronger* as n increases. We do not have a convincing explanation for this phenomenon.

Table 2 shows the average running times in seconds. As before, each figure is the average over ten instances of the given size. A first observation is that the running times for the first three bounds are negligible. Computing γ_{mn}^+ takes slightly longer, but still takes less than one second even for the larger instances.

As for the LP-based bounds, the running time seems to grow only linearly as m increases, but grows rapidly as n increases. This is probably because the number of variables in the LP is $n(n + m)$. We remark that, in the

[2] `http://www.research.lancs.ac.uk/portal/en/datasets/search.html`

Table 2: Average running times (seconds) for Taillard instances

m	n	L_J^+	L_M^+	γ_{mn}	γ_{mn}^+	L_c	L_c^+
5	20	0.002	0.006	0.016	0.018	0.132	0.134
	50	0.002	0.007	0.016	0.020	0.430	0.434
	100	0.004	0.007	0.016	0.022	0.857	0.920
10	20	0.002	0.007	0.018	0.021	0.260	0.264
	50	0.003	0.007	0.016	0.021	0.505	0.584
	100	0.004	0.007	0.016	0.026	1.769	1.872
	200	0.011	0.009	0.018	0.048	11.216	11.659
20	20	0.002	0.007	0.016	0.021	0.352	0.429
	50	0.002	0.008	0.022	0.032	1.108	1.206
	100	0.005	0.010	0.024	0.047	5.829	6.202
	200	0.014	0.013	0.031	0.101	62.795	71.432
	500	0.075	0.024	0.056	0.463	2396.341	2407.510

majority of practical applications, n is likely to be larger than m.

5.2 Vallada *et al.* instances

Vallada *et al.* [27] gave some evidence that the Taillard instances are relatively easy for their size. They created some smaller instances that were designed to be hard for the exact techniques that existed at the time. These instances have $n \in \{10, 20, 30, 40, 50, 60\}$ and $m \in \{5, 10, 15, 20\}$. There are ten instances for each combination of n and m, making 240 instances in total. At the time of writing, these instances were also available on the web.[3]

As well as creating the instances and making them available on the web, Vallada *et al.* [27] computed lower bounds for them. We will call their lower bound L_V.

Table 3 shows the average percentage gap for each set of ten instances and each of seven bounds. The table has an identical format to Table 1, except for the third column showing the percentage gaps for L_V. We see that, as before, L_M^+ and γ_{mn} are identical for all instances. Moreover, L_V is only slightly stronger.

As one might expect, L_J^+ was useful only when the number of jobs is small relative to the number of machines. Interestingly, for these instances, L_J^+ was better than L_M^+ if and only if $n < 2m$ (80 instances out of 240). Similarly, γ_{mn}^+ tended to be better than γ_{mn} when $n \leq 2m$ (81 instances); and L_c^+ tended to be better than L_c under the same condition (59 instances). We remark that optimal or best-known upper bounds were reached for the

[3] `http://soa.iti.es/problem-instances` (accessed 16/12/22)

Table 3: Average percentage gaps for Vallada *et al.* instances

m	n	L_V	L_J^+	L_M^+	γ_{mn}	γ_{mn}^+	L_c	L_c^+
	10	18.72	6.16	19.33	19.33	5.64	14.09	5.51
	20	7.06	16.14	7.16	7.16	7.08	4.85	4.85
5	30	3.72	24.29	3.78	3.78	3.78	2.73	2.73
	40	3.08	20.47	3.32	3.32	3.32	2.06	2.06
	50	2.17	26.12	2.18	2.18	2.18	1.34	1.34
	60	1.78	26.30	1.78	1.78	1.78	1.18	1.18
	10	25.41	7.26	27.23	27.23	7.18	18.44	7.12
	20	14.68	14.32	15.32	15.32	12.15	10.54	9.79
10	30	10.50	17.56	10.65	10.65	10.65	7.46	7.46
	40	6.70	19.39	6.76	6.76	6.76	4.99	4.99
	50	5.44	21.75	5.48	5.48	5.48	3.66	3.66
	60	4.39	20.20	4.58	4.58	4.58	2.88	2.88
	10	27.76	9.14	29.62	29.58	7.43	18.48	7.37
	20	19.31	15.58	20.24	20.24	13.60	13.60	11.60
15	30	14.97	17.04	15.53	15.53	14.11	11.65	11.38
	40	11.41	20.51	11.63	11.63	11.63	8.84	8.84
	50	9.04	21.16	9.20	9.20	9.20	6.98	6.98
	60	7.63	23.93	7.77	7.77	7.77	5.91	5.91
	10	26.18	11.07	27.61	27.61	9.80	17.20	9.79
	20	22.00	13.27	23.08	23.08	12.27	16.20	12.08
20	30	17.86	17.38	17.92	17.92	15.68	13.65	12.99
	40	15.69	19.55	15.97	15.97	15.21	12.40	12.40
	50	13.08	22.19	13.40	13.40	13.40	10.61	10.61
	60	10.76	22.31	10.87	10.87	10.87	8.76	8.76

`VFR-10-5-1` and `VRF-10-5-10` instances when using L_J^+, γ_{mn}^+ and L_c^+.

As before, all bounds apart from L_J^+ tend to get weaker as m increases, but stronger as n increases. We remark that, in the majority of practical applications, n is likely to be larger than m.

Finally, Table 4 shows the average running times in seconds for the Vallada *et al.* instances. Here, the running times for the first four bounds are negligible. Computing the LP-based bounds takes slightly longer, but it still only takes a few seconds, even for the largest instances.

6 Concluding Remarks

The permutation flowshop problem with makespan objective is a classic problem in machine scheduling. We analysed some of the existing lower bounds and proved several dominance relations. We also showed how to strengthen two of the lower bounds: the one obtained by solving the LP

Table 4: Average running times (seconds) for Vallada *et al.* instances

m	n	L_J^+	L_M^+	γ_{mn}	γ_{mn}^+	L_c	L_c^+
5	10	0.002	0.006	0.010	0.011	0.053	0.054
	20	0.002	0.006	0.010	0.011	0.067	0.069
	30	0.002	0.006	0.011	0.011	0.124	0.125
	40	0.002	0.007	0.011	0.012	0.204	0.208
	50	0.002	0.006	0.011	0.012	0.237	0.240
	60	0.002	0.006	0.010	0.012	0.234	0.243
10	10	0.003	0.008	0.014	0.012	0.083	0.091
	20	0.002	0.007	0.011	0.012	0.104	0.112
	30	0.002	0.006	0.010	0.012	0.458	0.461
	40	0.002	0.006	0.011	0.013	1.177	1.181
	50	0.002	0.006	0.010	0.013	0.315	0.342
	60	0.002	0.006	0.010	0.014	0.424	0.458
15	10	0.001	0.006	0.010	0.012	0.057	0.065
	20	0.002	0.006	0.010	0.013	0.133	0.165
	30	0.002	0.006	0.010	0.013	0.538	0.586
	40	0.002	0.006	0.011	0.014	3.195	3.203
	50	0.003	0.007	0.012	0.015	0.545	0.588
	60	0.003	0.007	0.011	0.017	0.831	0.889
20	10	0.002	0.006	0.010	0.012	0.061	0.077
	20	0.002	0.006	0.010	0.013	0.178	0.430
	30	0.002	0.006	0.011	0.014	0.759	0.983
	40	0.002	0.007	0.012	0.016	3.040	3.051
	50	0.002	0.007	0.013	0.017	0.559	0.622
	60	0.003	0.007	0.013	0.019	1.112	1.190

relaxation of the Stafford *et al.* [23] integer programming model, and the one of Kumar *et al.* [15]. The computational results, on the instances of Taillard [24] and Vallada *et al.* [27], show that our strengthened procedures lead to improved bounds when the number of jobs is relatively small compared to the number of machines. Moreover, our improvements incur negligible additional computing time.

An interesting topic for future research is the development of strong *cutting planes* for the Stafford *et al.* formulation. (See [5, 7] for details on cutting-plane approaches to integer programming.) We hope to work on this topic in a future paper.

Acknowledgement: The first author gratefully acknowledges funding from the Engineering and Physical Sciences Research Council (EPSRC) under grant reference EP/V520214/1, and from the Ministerio de Ciencia, Tecnología e Innovación of Colombia (MINCIENCIAS) through the call "885 de 2020—Doctorados en el Exterior".

References

[1] K.R. Baker & D. Trietsch (2019) *Principles of Sequencing and Scheduling* (2nd edn). Hoboken, NJ: Wiley.

[2] J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt & J. Weglarz (2019) *Handbook on Scheduling: From Theory to Practice*. Cham: Springer.

[3] P. Brucker (2007) *Scheduling Algorithms* (5th edn). Berlin: Springer.

[4] J. Carlier, & I. Rebaï (1996) Two branch and bound algorithms for the permutation flow shop problem. *Eur. J. Oper. Res.*, 90, 238–251.

[5] D.-S. Chen, R.G. Batson & Y. Dang (2010) *Applied Integer Programming*. Hoboken, NJ: Wiley.

[6] R. Companys & M. Mateo (2007) Different behaviour of a double branch-and-bound algorithm on Fm—prmu— C_{max} and Fm—block—C_{max} problems. *Comput. Oper. Res.*, 34, 938–953.

[7] M. Conforti, G. Cornuéjols & G. Zambelli (2014) *Integer Programming*. Cham, Switzerland: Springer.

[8] H. Emmons & G. Vairaktarakis (2013) *Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications*. New York: Springer.

[9] V. Fernandez-Viagas, R. Ruiz & J.M. Framinan (2017) A new vision of approximate methods for the permutation flowshop to minimise makespan: state-of-the-art and computational evaluation. *Eur. J. Oper. Res.*, 257, 707–721.

[10] J.M. Framinan, J.N. Gupta & R. Leisten (2004) A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *J. Oper. Res. Soc.*, 55, 1243–1255.

[11] J. Gmys, M. Mezmaz, N. Melab & D. Tuyttens (2020) A computationally efficient branch-and-bound algorithm for the permutation flow-shop scheduling problem. *Eur. J. Oper. Res.*, 284, 814–833.

[12] E. Ignall & L. Schrage (1965) Application of the branch and bound technique to some flow-shop scheduling problems. *Oper. Res.*, 13, 400–412.

[13] T.A. Jessin, S. Madankumar & C. Rajendran (2020) Permutation flow-shop scheduling to obtain the optimal solution/a lower bound with the makespan objective. *Sādhanā*, 45, 1–19.

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

[15] S.S. Kumar, C. Rajendran & R. Leisten (2021) Bounding strategies for obtaining a lower bound for N-job and M-machine flowshop scheduling problem with objective of minimising the total flowtime of jobs. *Int. J. Oper. Res.*, 41, 244–269.

[16] T. Ladhari & M. Haouari (2005) A computational study of the permutation flow shop problem based on a tight lower bound. *Comput. Oper. Res.*, 32, 1831–1847.

[17] J.K. Lenstra, A.H.G. Rinnooy Kan & P. Brucker (1977) Complexity of machine scheduling problems. *Ann. Discr. Math.*, 1, 343–362.

[18] G.B. McMahon & P.G. Burton (1967) Flow-shop scheduling with the branch-and-bound method. *Oper. Res.*, 15, 473–481.

[19] T.E. Morton & D.W. Pentico (1993) *Heuristic Scheduling Systems.* Chichester: Wiley.

[20] C.H. Pan (1997) A study of integer programming formulations for scheduling problems. *Int. J. Syst. Sci.*, 28, 33–41.

[21] M.L. Pinedo (2016) *Scheduling: Theory, Algorithms and Systems* (5th edn). New York: Springer.

[22] R. Ruiz & C. Maroto (2005) A comprehensive review and evaluation of permutation flowshop heuristics. *Eur. J. Oper. Res.*, 165, 479–494.

[23] E.F. Stafford, F.T. Tseng & J.N. Gupta (2005) Comparative evaluation of MILP flowshop models. *J. Oper. Res. Soc.*, 56, 88–101.

[24] E. Taillard (1993) Benchmarks for basic scheduling problems. *Eur. J. Oper. Res.*, 64, 278–285.

[25] C.P. Tomazella & M.S. Nagano (2020) A comprehensive review of branch-and-bound algorithms: guidelines and directions for further research on the flowshop scheduling problem. *Exp. Syst. Appl.*, 158, article 113556.

[26] F.T. Tseng, E.F. Stafford & J.N. Gupta (2004) An empirical analysis of integer programming formulations for the permutation flowshop. *Omega*, 32, 285–293.

[27] E. Vallada, R. Ruiz & J.M. Framinan (2015) New hard benchmark for flowshop scheduling problems minimising makespan. *Eur. J. Oper. Res.*, 240, 666–677.

[28] H.M. Wagner (1959) An integer linear-programming model for machine scheduling. *Nav. Res. Logist. Q.*, 6, 131–140.