Oper Res Int J (2015) 15:95–114
DOI 10.1007/s12351-014-0166-5

ORIGINAL PAPER

Minimizing total tardiness in the permutation flowshop scheduling problem with minimal and maximal time lags

Imen Hamdi · Taicir Loukil

Received: 1 April 2014 / Revised: 10 November 2014 / Accepted: 22 November 2014 /
Published online: 9 December 2014

Abstract In this research, we study permutation flowshop scheduling problem with minimal and maximal time lags while minimizing the total tardiness. The time lags are defined between couples of successive operations of jobs. Each time lag is greater than or equal to a prescribed value called minimal time lag and smaller than or equal to a prescribed value called maximal time lag. A new mathematical formulation is proposed. Upper bounds are provided by applying heuristic procedures based on known and new rules. Then, new lower bounds are derived by applying different methods where the main one is the Lagrangian relaxation. In order to make the last technique a viable approach to the considered problem, an auxiliary formulation is adopted and the Lagrangian multipliers are updated using the subgradient algorithm. Then, results of computational experiments are reported.

Keywords Scheduling · Permutation flowshop · Time lags · Tardiness · Upper bounds · Lower bounds

1 Introduction

The problem that we address in this paper is the permutation flowshop scheduling problem with time lags. Traditionally, flowshop scheduling models assume that for a given job, if its operation k precedes its operation l, then once the processing of the operation k is completed, the operation l may begin to process at any time. However, in several practical situations, a time lag between a couple of successive

I. Hamdi (✉) · T. Loukil
Research Unit LOGIQ, High Institute of Industrial Management, University of Sfax, 3018 Sfax, Tunisia
e-mail: imen.hamdi2007@yahoo.fr

T. Loukil
e-mail: taicir.loukil@fsegs.rnu.tn

operations of jobs is required. Hence, the considered problem can be formulated as follows: there are n jobs to be scheduled for processing on m machines. Each machine can process at most one operation at a time and preemption is not allowed. Each job i is processed on the m machines during $p_{i,1}, p_{i,2}, \ldots, p_{i,m}$ time units successively. We consider here additional constraints that extend the classical model: for each job, a definite amount of time must elapse between two consecutive operations of the same job which must be greater than or equal to a non negative value called minimal time lag (θ^{min}) and smaller than or equal to a non-negative value called maximal time lag (θ^{max}) where $\theta^{max} \geq \theta^{min}$. It is shown by Koulamas (1994) that the classical permutation flowshop scheduling problem with minimizing the total tardiness is NP-hard in the strong sense, then obviously the considered problem is also strongly NP-hard.

Motivation for the considered problem comes from industrial applications. Many industrial situations involving specific manufacturing processes may be modelled using minimal and maximal time lags. For example in the field of biotechnologies and chemistry, the chemical reactions with variable processing times may be represented by time lags (Chu and Proth 1996). In the agriculture field, such constraints arise during the sequence of harvesting operations (Foulds and Wilson 2005). In the food-producing industry, the maximal time lags may be used to model situations when the delay between operations must not be too long in order to avoid deterioration of the product.

The flowshop scheduling problems with time lags are studied largely with the classical criterion the makespan (see Fondrevelle et al. 2006; Dell'Amico 1996; Yu et al. 2004). However, the criteria related to the due dates are the most encountred in the real situations. Among these criteria, the most applicable performance measure is the total tardiness. Generally, each job has a given due date, if it finishes too late it cannot be delivered on time. To the best of our knowledge, this is the first attempt to consider the permutation flowshop scheduling problem with time lags to minimize the total tardiness.

Some approaches are proposed to solve the problem under consideration in this research. A mathematical formulation is proposed, then by running the commercial software CPLEX we can obtain an optimal solution for small size problems. For large size problems, upper bounds are provided by applying some efficient heuristic procedures. They are based on some rules called "Shortest Processing Time," "Adjustment on the Bottleneck Machine," and "Estimated Completion Time" which will be described later. Also, lower bounds are derived by using different methods. The first one is obtained by relaxing the integrality constraints, the second one by relaxing the constraints which state that the machines $1, \ldots, m-1$ can process one job at a time, and the third one is derived by applying the Lagrangian relaxation (LR). This last technique is frequently used in operational research. It can often calculate lower bounds of the optimal solution. Its idea consists in relaxing some constraints (often those that make difficult problem) and inject them into the objective function as a linear combination of penalties. The coefficients of this combination are called Lagrangian multipliers. The most known method that can be used to update the Lagrangian multipliers is the subgradient.

Generally, LR is effective for separable problems, i.e. problems with additive objective functions and additive coupling constraints (constraints that couple subsystems together). If the original integer or mixed integer optimization problem is NP-hard, the relaxed problem after relaxing those coupling constraints can be decomposed into individual subsystem problems which are not NP-hard, then LR is very powerful. It is a trade off, we want to relax just enough coupling constraints so that subproblems can be efficiently solved while not having too many multipliers for good high level convergence and tight lower bound.

Survey of the flowshop models reveals that there are some interesting researches addressing the LR application. Liu et al. (1997) consider the permutation flowshop scheduling problem while minimizing the additive penalties on product tardiness and on releasing raw materials too early. They develop a novel approach based on LR, the number of multipliers is independent of the planning horizon, and the multipliers are updated by using the recently developed reduced complexity bundle method. Recently, the LR is integrated with other approaches to yield very performant results (see Nishi et al. 2009, 2010; Nishi and Hiranakaa 2013). Bulbul et al. (2004) consider the flowshop problem to minimize the sum of tardiness, earliness, and intermediate inventory holding costs. They develop heuristics to minimize the total cost, then they exploit the duality between Dantzig-Wolfe reformulation and LR to enhance these heuristics and to to develop two different lower bounds on the optimal integer solution. The computational study proove that the proposed algorithms have a significant speed advantage over alternate methods.

The rest of the paper is organized as follows: a mathematical formulation of the considered problem is presented in Sect. 2. In Sect. 3, we present the proposed heuristic procedures. In Sect. 4 we present the developed lower bounds and we detail the steps of the LR application. Then, computational results are reported in Sect. 5. Finally, we discuss concluding remarks in Sect. 6.

2 Mathematical formulation

In this section we present a new mathematical formulation for the considered problem. The parameters of the problem are described as follows.

1. Decision variables

- $X_{i,j} = 1$ if job i is scheduled in position j, 0 otherwise $\forall i \in \{1, 2, \ldots, n\}$, $\forall j \in \{1, 2, \ldots, n\}$
- $C_{j,k}$: completion time of job in position j on machine k, $\forall j \in \{1, 2, \ldots, n\}$, $\forall k \in \{1, 2, \ldots, m\}$

2. Data

- $p_{i,k}$: processing time of job i on machine k, $\forall i \in \{1, 2, \ldots, n\}$, $\forall k \in \{1, 2, \ldots, m\}$
- d_i: due time of job i, $\forall i \in \{1, 2, \ldots, n\}$

- $\theta_{i,k}^{min}$: minimal time lag between machines k and $k+1$ of job i, $\forall i \in \{1, 2, \ldots, n\}$, $\forall k \in \{1, 2, \ldots, m-1\}$
- $\theta_{i,k}^{max}$: maximal time lag between machines k and $k+1$ of job i, $\forall i \in \{1, 2, \ldots, n\}$, $\forall k \in \{1, 2, \ldots, m-1\}$

$$Min\, Z = \sum_{j=1}^{n} T_j \tag{1}$$

$$s/t \quad \sum_{i=1}^{n} X_{i,j} = 1 \quad \forall j \in \{1, 2, \ldots, n\} \tag{2}$$

$$\sum_{j=1}^{n} X_{i,j} = 1 \quad \forall i \in \{1, 2, \ldots, n\} \tag{3}$$

$$C_{j,k+1} \geq C_{j,k} + \sum_{i=1}^{n} X_{i,j}\left(p_{i,k+1} + \theta_{i,k}^{min}\right) \quad \forall j \in \{1, 2, \ldots, n\} \quad and \quad \forall k \in \{1, 2, \ldots, m-1\} \tag{4}$$

$$C_{j,k+1} \leq C_{j,k} + \sum_{i=1}^{n} X_{i,j}\left(p_{i,k+1} + \theta_{i,k}^{max}\right) \quad \forall j \in \{1, 2, \ldots, n\} \quad and \quad \forall k \in \{1, 2, \ldots, m-1\} \tag{5}$$

$$C_{(j-1),k} + \sum_{i=1}^{n}\left(X_{i,j} p_{i,k}\right) \leq C_{j,k} \quad \forall j \in \{2, \ldots, n\} \quad and \quad \forall k \in \{1, 2, \ldots, m\} \tag{6}$$

$$T_j - C_{j,m} + \sum_{i=1}^{n}(d_i X_{i,j}) \geq 0 \quad \forall j \in \{1, 2, \ldots, n\} \tag{7}$$

$$T_j \geq 0 \quad \forall j \in \{1, 2, \ldots, n\} \tag{8}$$

$$X_{i,j} = \{0,\ 1\} \quad \forall i \in \{1, 2, \ldots, n\} \quad and \quad \forall j \in \{1, 2, \ldots, n\} \tag{9}$$

The objective (1) is to minimize the total tardiness. Constraints (2) and (3) are the classical assignment constraints, with (2) insure that each job is assigned to just one sequence position, while (3) insure that each position is filled by only one job. Constraints of types (4) and (5) find the completion time of every job. They specify the conjunctive precedence constraints for the jobs with respect to the minimal and maximal time lags respectively. Constraints of type (6) state that the completion time of job in position j on machine k is greater than or equal to its processing time on this machine plus the completion time of the job in the previous position on this same machine k. Constraints of type (7) find the tardiness of job in position j. Constraints of type (8) force the tardiness of the job in position j to be non negative. Constraints (9) specify the decision variables as binary variables.

The number of decision variables is $n^2 + nm + n$ and the number of constraints is $3nm + n - m$.

3 Heuristic procedures

Upper bounds of the optimal solution are provided by applying some heuristic procedures. They are based on three different rules: the well known rule "Shortest Processing Time (SPT)", and two other new proposed rules: "Adjustment on the Bottleneck Machine (ABM)" and "Estimated Completion Time (ECT)". Then the obtained sequences by using the proposed rules are scheduled by applying the "*Algorithm (A)*" described later.

3.1 Shortest Processing Time (SPT) rule

The *SPT* rule states to sequence the jobs in a nondecreasing order of processing times. As we are interested in the permutation flowshop problem, we may find m different sequences, one sequence for each machine. Then, the jobs are arranged according to $p_{\pi(1),k} \le p_{\pi(2),k} \le \cdots \le p_{\pi(n),k}$ for each machine $k \in \{1, \ldots, m\}$. We can determine another sequence which is the (m+1)th one by using the total processing times $(TP_{\pi(1)} \le TP_{\pi(2)} \cdots \le TP_{\pi(n)})$ as same in (Ghassemi and Olfat 2010). Now, we determine the total tardiness of all the $m + 1$ sequences and we select the final one with the smallest total tardiness. This rule determines the final schedule through the following algorithmic steps:

Step 1: Let $k = 1$.
Step 2: Sequence jobs in a non-decreasing order of $p_{i,k}(p_{\pi(1),k} \le p_{\pi(2),k} \le \cdots \le p_{\pi(n),k})$ on machine k.
Step 3: Define a permutation schedule on all machines using this sequence and calculate the total tardiness of all the jobs by applying the $Algorithm\,(A)$ described later.
Step 4: If $k = m$, go to step 5, otherwise let $k = k + 1$, and go to step 2.
Step 5: Sequence jobs in a non-decreasing order of $TP_i\left(TP_{\pi(1)} \le TP_{\pi(2)} \le \cdots TP_{\pi(n)}\right)$. Define a permutation schedule using this sequence and then calculate the total tardiness of all the jobs by applying the $Algorithm\,(A)$. described later.
Step 6: Select the final schedule with the smallest total tardiness among the $m + 1$ schedules, and stop.

3.2 Adjustment on the Bottleneck Machine (ABM) rule

In this proposed heuristic, we focus on the bottleneck machine $(b \in \{1, 2, \ldots, m\})$ to adjust the number of tardy jobs. This machine is defined as the machine that requires maximum sum of processing times of all jobs amongst all machines. We determine for each job its ready time to be processed on this machine and we derive its due date. Then, we can define two sets of jobs: a set of tardy jobs and a set of early jobs. The adjustment is done by sequencing the set of tardy jobs in a decreasing order of the tardiness values and the early jobs in an increasing order of the earliness values, then we concatenate the sequence of the early jobs to the set of tardy jobs to form the final sequence to be scheduled using the *Algorithm* (A) described later. We detail the steps of the algorithm as follows:

Step 1.

- Determine the release date of job i on machine b

$$r_{i,b} = \left\{ \sum_{l=1}^{b-1} \left(p_{i,l} + \theta_{i,l}^{min} \right) \right\}$$

- Determine the due date of job i on machine b

$$d_{i,b} = d_i - \left\{ \sum_{l=b+1}^{m-1} \left(p_{i,l} + \theta_{i,l}^{min} \right) + p_{i,m} \right\}$$

Step 2.

Determine two sets of jobs: set of early jobs (E) and set of tardy jobs (T) on the bottleneck machine.
$T = \{i | r_{i,b} + p_{i,b} > d_{i,b}\}$ and we define $u_{i \in T} = r_{i,b} + p_{i,b} - d_{i,b}$
$E = \{i | r_{i,b} + p_{i,b} < d_{i,b}\}$ and we define $s_{i \in E} = d_{i,b} - p_{i,b} - r_{i,b}$

Step 3.

Sequence the jobs in an increasing order of $s_i \, \forall i \in E$ to provide the sequence E_1
Sequence the jobs in a decreasing order of $u_i \, \forall i \in T$ to provide the sequence T_1
The final sequence (π) to be scheduled is defined by concatenating the sequence E_1 to the sequence $T_1 \Rightarrow \pi = T_1 + E_1$.

3.3 Estimated Completion Time (ECT) rule

This rule is similar to the *SPT* rule, but instead of using the individual job processing times on each machine, we consider the total processing time of a job on the previous machines plus the minimal time lags between each consecutive couple of operations. For each job $i \in \{1, \ldots, n\}$, the found value will be considered then as an estimated ready time for the processing on a specific machine $k \in \{1, 2, \ldots, m\}$. This ready time will be added to the job processing time on the considered machine to find an ECT. Then, we obtain a sequence of jobs on each machine according to the ascending order of the ECT. After determining a sequence on each machine k, we use this sequence as a permutation schedule for all the machines. Then, by having m different sequences, we calculate the total tardiness of each one by applying the "*Algorithm* (A)" and we select the one with the smallest total tardiness. This rule determines the final sequence through the following algorithmic steps:

Step 1: Calculate the ECT of each job $i \in \{1, \ldots, n\}$ as $ECT_{i,k} = \sum_{l=1}^{k-1} \left(p_{i,l} + \theta_{i,l}^{min}\right) + p_{i,k}$ on each machine $k \in \{1, \ldots, m\}$.

Step 2: Sequence jobs on machine k according to a non-decreasing order of ECT_i $(ECT_{\pi(1)} \leq ECT_{\pi(2)} \leq \cdots \leq ECT_{\pi(n)})$.

Step 3: Define a permutation schedule on all machines using this sequence and calculate the total tardiness of all the jobs by applying the *Algorithm* (A) described later.

Step 4: If $k = m$, go to step 5, otherwise let $k = k + 1$, and go to Step 2.

Step 5: Select the final schedule with the smallest total tardiness among the m schedules, and stop.

3.4 Scheduling algorithm

The sequence obtained by applying each of the previous proposed rules are scheduled by using the following algorithm. Here $\pi(i), k$ denote the job in position i in the sequence π on machine k; $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m$ [ith job means the job in position $\pi(i)$]

Algorithm (A)

```
a) The first job is scheduled as soon as possible
- Schedule the first job on the first machine
$C_{\pi(1),1} = p_{\pi(1),1}$
- Schedule successively the other operations of the first job
For $k = 1$ to $m - 1$, do
$C_{\pi(1),k+1} = C_{\pi(1),k} + \theta^{min}_{\pi(1),k} + p_{\pi(1),k+1}$
b) Schedule the other jobs as soon as possible
For $i = 2$ to $n$, do
- Schedule the first operation of the $i^{th}$ job on machine 1
$C_{\pi(i),1} = C_{\pi(i-1),1} + p_{\pi(i),1}$
- Schedule successively the other operations of the $i^{th}$ job with respect to the precedence constraints and the minimal time lags constraints
For $k = 1$ to $m - 1$, do
$C_{\pi(i),k+1} = max\{C_{\pi(i),k} + \theta^{min}_{\pi(i),k}, C_{\pi(i-1),k+1}\} + p_{\pi(i),k+1}$
-Make sure that the maximal time lags constraints are satisfied
For $k = m - 1$ to $1$, do
If $C_{\pi(i),k+1} - C_{\pi(i),k} > \theta^{max}_{\pi(i),k} + p_{\pi(i),k+1}$
Then $C_{\pi(i),k} = C_{\pi(i),k+1} - p_{\pi(i),k+1} - \theta^{max}_{\pi(i),k}$
c) Determine the total tardiness
If $C_{\pi(i),m} > d_{\pi(i)}$
Then $T = T + (C_{\pi(i),m} - d_{\pi(i)})$, $i = i + 1$
```

In the above algorithm, it's shown that there is no matter to check the respect of the maximal time lags for the job in the first position. As it has not any precedent job, its processing on the next machine begins exactly after the minimal time lag. Then, it is straightforward to check the respect of the minimal time lags for the other jobs. However to check the respect of the maximal time lags, the machines are considered in a reverse order [from $(m-1)$ to 1] in order to minimize the number of shifts. If we consider the order of the operations's passage, we may shift an opeartion more than one time. Indeed, shifting an operation on machine k can lead to the non respect of the maximal time lag between this operation and the preceding one. Then, this operation must be shifted again.

The complexity of this algorithm is $O(nm)$.

4 Lower bounds

In this section, we present the developed lower bounds. The first one is derived by relaxing the integrality constraints, the second one is obtained by relaxing the constraints which state that the machines 1,..., $m-1$ can process one job at a time and by using then the constraint programming, and the third lower bound is developed by applying the LR technique.

4.1 The integrality constraints relaxation

The key insight behind this approach is that the closely related integer programming problem is NP-hard. We can therefore relax it to a linear program by removing the integrality constraints. Then, the binary constraints are allowed to take on any real values in the closed interval [0, 1].

Then an example of the integer linear programming can be modeled as follows:

$$\begin{aligned} &min \sum_{i=1}^{n} x_i \\ &s.t \quad x_i + x_j \geq 1 \quad \forall i, j \\ &x_i \in \{0, 1\} \quad \forall i \end{aligned}$$

Its corresponding linear programming after relaxing the integrality constraints is as follows:

$$\begin{aligned} &min \sum_{i=1}^{n} x_i \\ &s.t \quad x_i + x_j \geq 1 \quad \forall i, j \\ &0 \leq x_i \leq 1 \quad \forall i \end{aligned}$$

4.2 Constraint programming based lower bound

Recently, the Constraint Programming (CP) becomes a leading technique for solving complex decision support problems. It is an other fashion to model a problem rather than the integer linear programming. In general, systems based on *CP* are much more expressive and hence easy to understand. It is increasingly used for solving scheduling problems as its flexibility is well suited for real-life scheduling problems. It works by incorporating some restrictions on the possible solution into a programming environment.

We focus on the last machine m and we relax the constraints which state that the machines 1,…, $m-1$ can process one job at a time. Then we can determine for each job $i \in \{1,\ldots,n\}$ a release date for its processing on the last machine by summing its processing times on the $m-1$ first machines plus the minimal time lags between them as $r_i = \sum_{l=1}^{m-1} \left(p_{i,j} + \theta_{i,j}^{min}\right)$. Then by solving the following *CP*, we can obtain a lower bound on the optimal solution.

$$\begin{aligned} &Minimize\, sum\,(i\, in\, Jobs)\, maxl\,(endOf(itvs[i]) - DueDate[i],\, 0)\\ &\quad s.t\, startOf(itvs[i]) > \; = release[i] \quad forall\, i\\ &endOf(itvs[i]) > \; = startOf(itvs[i]) + OpDurations[i] \quad forall\, i \end{aligned}$$

Here, “itvs” means an interval of time used to describe the job i on the last machine, it is bounded by a release date “release” and a due date “DueDate”. “OpDurations” is used as the processing time. The formulation above include an objective function used to define the total tardiness and two types of constraints. The first type states that the starting time has to be greater than or equal to the release date defined above. The second type is used to define the completion time for each job i on the last machine.

Some constraint-based tools are provided for imperative languages in the form of libraries. ILOG is one of the most successful companies to produce such tools. High level modeling languages exist for model constraint problems and specifying search strategies such that the OPL language.

4.3 Lagrangian relaxation

In this section, we detail the steps of obtaining a lower bound by using the LR. The basic idea of LR is to have a separable problem formulation in that the objective function and coupling constraints should be additive in terms of basic decision variables or groups of decision variables. Since the identical sequence constraints of a permutation flowshop require the processing sequences of various production stages to be the same, obtaining a separable problem formulation with manageable numbers of variables and constraints is not a straightforward task (Liu et al. 1997).

Here, we use a time-indexed formulation, i.e. the time is devided into slots. We define a long time horizon H and we introduce a new decision variable $t_{j,k}$ to design the starting time of job in position j on machine k, $\forall j \in \{1, 2, \ldots, n\}$, $\forall k \in \{1, 2, \ldots, m\}$. The machines are assumed available from the beginning and during the planning horizon. The time horizon is assumed to be long enough to finish all the slots $(C_{n,m} \leq H)$. Our objective is to determine an optimal assignment of jobs to slots and then determining of operations beginning times of the slots on the different machines. We refer to the idea adopted by (Liu et al. 1997; Baker 1974;

Pinedo 1995) which is to create a virtual sequence. We try to create a virtual sequence while the first slot of this sequence is defined to be processed first, the second slot is defined to be processed the second, and so on. Let s be the slot index of the virtual sequence with $s = 1, 2, \ldots, n$. So that the constraints are modelled in terms of the assignment variables. They still the same ones presented already in the previous mathematical formulation, just the assignment variable $X_{i,s}$ here is used to mean that job i is assigned to a slot s (instead of $X_{i,j}$ which means that the job i is assigned to position j). The decomposition methodology requires to express the constraints by using the starting time variable. Then, the constraints (4) and (5) can be rewritten as follows:

$$t_{s,k+1} \geq t_{s,k} + \sum_{i=1}^{n} X_{i,s}(p_{i,k} + \theta_{i,k}^{min}) \quad \forall s = 1, \ldots, n; \; \forall k = 1, \ldots, m-1 \tag{10}$$

$$t_{s,k+1} \leq t_{s,k} + \sum_{i=1}^{n} X_{i,s}(p_{i,k} + \theta_{i,k}^{max}) \quad \forall s = 1, \ldots, n; \; \forall k = 1, \ldots, m-1 \tag{11}$$

4.3.1 Objective function and reducing the solution oscillation

The objective function is to minimize the additive jobs tardiness: $Z = \sum_{s=1}^{n} T_s$ where $T_s = max(0, \; C_{s,m} - d_s)$ and $d_s = d_i X_{i,s}$. As it is described previously, the problem consists in determining an optimal assignment of jobs to slots in this sequence and then in determining the operations beginning times of the slots on the different machines. There is a problem that can be created when determining the beginning times $(t_{s,k})$ which is its oscillation between a large value $(t_{j,k} = H - p_{j,k})$ and a small value $(t_{j,k} = 0)$. The solution oscillation can alternatively be explained by the geometry of the dual function in which multiplier trajectories generated by the subgradient method zigzag about a set of nondifferentiable points from iteration to iteration (Czerwinski and Luh 1994). This problem is studied already by Czerwinski and Luh (1994) and Luh and Hoitomt (1993). According to them, to reduce the solution oscillation, quadratic tardiness of operations are included in the objective function which can be then obtained as: $Z = \sum_{s=1}^{n} T_s + \sum_{s=1}^{n} \sum_{k=1}^{m} T_{s,k}^2$. The variable $T_{s,k}$ is defined on a scheduling sequence with $T_{s,k} = max(0, \; C_{s,k} - d_{s,k})$ and $C_{s,k} = t_{s,k} + p_{s,k}$ such that $d_{s,k}$ is defined as the same way in (Liu et al. 1997), $d_{s,m} = \underset{s}{max}\{d_s\}$ and $d_{s,k} = d_{s,k+1} - \underset{s}{min}\{p_{s,k+1}\}$. The quadratic tardiness terms are intended to define a period of time in the horizon time. In the dual space, the quatratic terms modify the geometry of the dual function and this, in turn, reduces oscillation between solutions (Luh and Hoitomt 1993).

The data are assumed given: The number of jobs n, the number of machines m, the processing times $\{p_{i,k}\}$, the minimal time lags $\{\theta_{i,k}^{min}\}$, the maximal time lags $\{\theta_{i,k}^{max}\}$and the due dates $\{d_i\}$. The scheduling problem is to select job beginning times $\{t_{i,k}\}$ so as to minimize the total tardiness. Once the beginning times are selected, the completion times $\{C_{s,k}\}$ and the tardiness $\{T_s\}$ can be easily derived.

Two methods are proposed, one constraints type is relaxed for each one. In the first method, we relax the constraints of type (4) in the mathematical formulation given previously [rewritten as constraints (10)] by using the Lagrangian multipliers $\{\upsilon_{s,k}\}$. The total number of these multipliers is equal to $n(m-1)$. In the second method, we relax the constraints of type (5) [rewritten as constraints (11)] by using the Lagrangian multipliers $\{\beta_{s,k}\}$. Also, the total number of multipliers is equal to $n(m-1)$. Three steps are followed by each of the two methods: Obtaining and solving subproblems, constructing a feasible solution, and updating the Lagrangian multipliers.

The Lagrangian multiplier $\upsilon_{s,k}$ can be interpreted as the cost of violating the respect of the minimal time lags between each consecutive operations of each job. The Lagrangian multiplier $\beta_{s,k}$ can be interpreted as the cost of violating the respect of the maximal time lags between each consecutive operations of each job. The optimal values of the Lagrangian multipliers are obtained by solving the Lagrangian dual problem by applying a subgradient algorithm as it will be presented later.

A specified stopping criterion is used here as non-improvements in the objective value over a number of consecutive iterations are met.

4.3.2 Decomposition framework and solving subproblems

As it is described previously, we aim to determine the beginning times of the slots on the different machines. So that, the constraints can be expressed in terms of the beginning times. In this first step, we detail the decomposition framework and the solving subproblems for each method.

4.3.3 Method 1: relaxation of constraints (4)

The resulting first relaxed problem is:

$$\underset{\{X_{i,s},t_{s,k}\}}{Min} \quad L$$

with $L = \sum_s T_s + \sum_s \sum_k T_{s,k}^2 - \sum_s \sum_k \upsilon_{s,k}\left(t_{s,k+1} - t_{s,k} - \sum_i \left(p_{i,k} + \theta_{i,k}^{min}\right) X_{i,s}\right)$

The formulation obtained already for the objective function can be divided into two subproblems types that will be individually solved.

- *Operation beginning time for operations* $k = 1, 2, \ldots, m-1$

$$\underset{t_{s,k}}{Min}\, L_{s,k}$$

$L_{s,k} \equiv T_{s,k}^2 - (\upsilon_{s,k-1} - \upsilon_{s,k})t_{s,k} \quad \forall s \in \{1, 2, \ldots, n\}$

The subproblem is a piecewise quadratic cost function with a convex envelop. The solution can be obtained by differentiating its convex envelop.

we suppose $\varepsilon = \upsilon_{s,k-1} - \upsilon_{s,k}$

The subproblem is rewritten as: $Min\ L_{s,k}\ \ with\ L_{s,k} = T_{s,k}^2 - \varepsilon t_{s,k}$

the solution is $t_{s,k} = d_{s,k} - p_{s,k} - \frac{\varepsilon}{2}$

As it's mentioned previously, the operation's quadratic tardiness is introduced in the objective function to reduce the solution oscillation. To confirm this idea, we suppose that this tardiness is not included in the objective function. Then, the beginning time $t_{s,k}$ depends only on the sign of ε. When this term is negative, $t_{s,k}$ has to be as large as possible $(t_{s,k} = H - p_{s,k})$. On the other hand, if this term is positive, $t_{s,k}$ has to be as small as possible $t_{s,k} = 0$. So, this solution oscillation between small and large beginning time for iteration to iteration make difficult the convergence to meaningful multipliers. Then, the introduction of the operation tardiness in the objective function to obtain intermediate beginning time solution. This subproblem can be solved in $O(mn)$.

- *Slot assignment and operation beginning time for operation $k = m$.*

The tardiness is defined as $T_s = max\left(0, C_{s,m} - \sum_i d_i X_{i,s}\right)$, the beginning time of the last operation $t_{s,m}$ is coupled with the job assignment variable $X_{i,s}$ in order to obtain T_s.

$$\min_{X_{i,s}} L_s$$

with $L_s \equiv \sum_{i=1}^{n} \left[\sum_{k=1}^{m-1} \left[\upsilon_{s,k}\left(p_{i,k} + \theta_{i,k}^{min}\right)\right] X_{i,s}\right] + \min_{t_{s,m}} L_{s,m}$

where $L_{s,m}$ is defined as: $L_{s,m} = T_s + T_{s,m}^2 - (\upsilon_{s,m}) t_{s,m}$

The beginning time $t_{s,m}$ is solved to minimize $L_{s,m}$ which has a quadratic cost function with convex envelop. The solution is $t_{s,m} = 1 + d_{s,m} - p_{s,m} - \frac{\varepsilon}{2}$. This subproblem can be solved in $O(n)$.

The variable $X_{i,s}$ is then determined by assigning the job associated with the lowest subproblem value to slot s. Then, we can obtain an optimal value for each subproblem given by $L_{s,k}^{*}$ *and* L_s^{*}. These values are then used to solve the dual problem which is formed as: $\max_{\upsilon_{s,k}} L$ with $L \equiv \sum_{s=1}^{n} \sum_{k=1}^{m-1} L_{s,k}^{*} + \sum_{s=1}^{n} L_s^{*}$.

4.3.4 Method 2: relaxation of constraints (5)

The resulting second relaxed problem is:

$$\underset{\{X_{i,s}, t_{s,k}\}}{Min} \; L$$

with $L = \sum_s T_s + \sum_s \sum_k T_{s,k}^2 - \sum_s \sum_k \beta_{s,k}\left(t_{s,k} + \sum_i X_{i,s}\left(p_{s,k} + \theta_{s,k}^{max}\right) - t_{s,k+1}\right)$

The formulation obtained already for the objective function can be divided into two subproblems types that will be individually solved as the same way in the method 1.

- *Operation beginning time for operations $k = 1, 2, \ldots, m - 1$*

$$\underset{t_{s,k}}{Min}\, L_{s,k}$$

$L_{s,k} \equiv T^2_{s,k} + \left(\beta_{s,k-1} - \beta_{s,k}\right)t_{s,k} \quad \forall s \in \{1, 2, \ldots, n\}$

The subproblem has a piecewise quadratic cost function with a convex envelop. The solution can be obtained by differentiating its convex envelop.

we suppose $\varepsilon' = \beta_{s,k-1} - \beta_{s,k}$

The subproblem is rewritten as: $Min\ L_{s,k}\ \ with\ L_{s,k} = T^2_{s,k} + \varepsilon' t_{s,k}$

the solution is $t_{s,k} = d_{s,k} - p_{s,k} - \frac{\varepsilon'}{2}$

- *Slot assignment and operation beginning time for operation $k = m$*

$$\underset{X_{i,s}}{Min}\, L_s$$

with $L_s \equiv \sum_{i=1}^{n} \left[\sum_{k=1}^{m-1} \left[\beta_{s,k}\left(-p_{i,k} - \theta^{max}_{i,k}\right)X_{i,s}\right] + \underset{t_{s,k}}{min} L_{s,m}\right]$ where $L_{s,m}$ is defined as: $L_{s,m} = T_s + T^2_{s,m} + (\beta_{s,m})t_{s,m}$.

$T_s = max\left(0,\ C_{s,m} - \sum_i X_{i,s} d_i\right)$. The beginning time $t_{s,m}$ is solved to minimize $L_{s,m}$ which has a quadratic cost function with convex envelop. The solution is $t_{s,m} = 1 + d_{s,m} - p_{s,m} - \frac{\varepsilon'}{2}$.

The variable $X_{i,s}$ is then determined by assigning the job associated with the lowest subproblem value to slot s. Then, we can obtain an optimal value for each subproblem given by $L^*_{s,k}\ and\ L^*_s$. These values are then used to solve the dual problem which is formed as: $\underset{\beta_{s,k}}{max} L$ with $L \equiv \sum_{s=1}^{n} \sum_{k=1}^{m-1} L^*_{s,k} + \sum_{s=1}^{n} L^*_{s,k}$.

4.3.5 Constructing a feasible solution

Generally the found solution may be associated with an infeasible schedule as an unscheduled sequence can be obtained when a job i is assigned to several slots. A way to avoid this problem is to apply the following heuristc: when a job is assigned to several slots, only the one with the smallest index is kept. All the jobs assigned to succeeding positions are advanced in the scheduling sequence to fill up the gap. Then, the complement of the sequence is added to the one already obtained. This algorithm requires $O(n)$ time.

Example There are five jobs to be scheduled and which have to be assigned to five slots in the scheduling sequence. Assume that the dual solution obtained is [1 1 5 3 3], job 1 is assigned to the first and second slots; also job 3 is assigned to the fourth and fifth slots which is an unscheduled sequence. By Applying the proposed heuristic, the sequence become [1 5 3 - -]. Then, we complete the sequence by adding the missing jobs in the ascending order to obtain the sequence [1 5 3 2 4].

4.3.6 Updating the Lagrangian multipliers

To update the Lagrangian multipliers, we adopt the subgradient method. It consists in updating iteratively the variables of the dual problem (Lagrangian multipliers). Each iteration z consists in two steps:

1. For $\upsilon_{s,k}^{z}$, *and* $\beta_{s,k}^{z}$ (Lagrangian multipliers in the zth iteration), calculate $L(\upsilon_{s,k}^{z})$, *and* $L(\beta_{s,k}^{z})$ to obtain the subgradient of L in the point $(\upsilon_{s,k}^{z})$ and $(\beta_{s,k}^{z})$ successively.
2.
   - Calculate $\upsilon_{s,k}^{z+1} = \upsilon_{s,k}^{z} + S_{\upsilon}^{z}\left(\left(t_{s,k} - t_{s,k+1}\right) + \sum_{i=1}^{n} X_{i,s}\left(p_{i,k} + \theta_{i,k}^{min}\right)\right) \quad \forall k = 1, \ldots, m-1$ and $\upsilon_{s,k}^{z+1} = \upsilon_{s,k}^{z} + S_{\upsilon}^{z}\left(t_{s,k} + \sum_{i=1}^{n} X_{i,s}p_{i,k}\right) \quad \forall k = m$
   - Calculate $\beta_{s,k}^{z+1} = \beta_{s,k}^{z} + S_{\beta}^{z}\left(\left(t_{s,k+1} - t_{s,k}\right) - \sum_{i=1}^{n} X_{i,s}\left(p_{i,k} + \theta_{i,k}^{max}\right)\right) \forall k = 1, \ldots, m-1$ and $\beta_{s,k}^{z+1} = \beta_{s,k}^{z} + S_{\beta}^{z}\left(t_{s,k} - \sum_{i=1}^{n} X_{i,s}p_{i,k}\right) \forall k = m$

There are several types of step size rules. We use the constant step size as the sub gradient algorithm is guaranteed to converge to within some range of the optimal value. The step size is given by:

$$S_{\upsilon}^{z} = \gamma \frac{L^{U} - L^{z}}{\sum_{s=1}^{n} \sum_{k=1}^{m-1} \left\{\left(t_{s,k} - t_{s,k+1}\right) + \sum_{i=1}^{n} X_{i,s}\left(p_{i,k} + \theta_{i,k}^{min}\right)\right\}^{2}}$$

$$S_{\beta}^{z} = \gamma \frac{L^{U} - L^{z}}{\sum_{s=1}^{n} \sum_{k=1}^{m-1} \left\{\left(t_{s,k+1} - t_{s,k}\right) + \sum_{i=1}^{n} X_{i,s}\left(p_{i,k} + \theta_{i,k}^{max}\right)\right\}^{2}}$$

with $0 \leq \gamma \leq 2$

Here L^{z} is the value of L at the zth iteration and L^{U} is an estimate of the optimal solution.

4.3.7 Example

To explain better the steps of the proposed LR technique, we use the following example of method 1. We consider 3 jobs, 3 slots, and 3 machines and we define the minimal time lags between them (Table 1).

Then, we detail the steps of the proposed algorithm:

We begin by an iteration counter = 0, we just can determine the starting times as follows (Table 2).

The first obtained $L = 3$, and we consider the sequence 1-2-3 as we can't determine the L_s value in this iteration (Table 3). Then we can determine the Lagrangian multipliers values for the next iteration such that $S_{\upsilon}^{1} = 0.02$.

Then, we can determine the objective value as it was described previously and the sequence still 1-2-3 (Table 4).

Table 1 The data

	$p_{i,1}$	$p_{i,2}$	$p_{i,3}$	$\theta^{min}_{i,1}$	$\theta^{min}_{i,2}$	$\theta^{max}_{i,1}$	$\theta^{max}_{i,2}$	$d_{i,1}$	$d_{i,2}$	$d_{i,3}$
Job 1	3	4	2	3	3	4	4	5	8	10
Job 2	4	3	3	1	2	3	3	5	8	10
Job 3	5	2	3	2	4	3	6	5	8	10

Table 2 Lagrangian multipliers

$t_{s,k}$	1	2	3
1	2	4	9
2	1	5	8
3	0	6	8

Table 3 The obtained Lagrangian multipliers $\upsilon_{s,k}$ and ε values

$\upsilon_{s,k}$	1	2	3	ε	1	2	3
1	0.08	0.04	0.22	1	−0.08	−0.04	−0.18
2	0.02	0.10	0.22	2	−0.02	−0.08	−0.18
3	0.02	0.08	0.22	3	−0.02	−0.06	−0.18

Table 4 The $L_{s,k}$ and L_s values

$L_{s,k}$	1	2	3
1	0.06	0.16	1.08
2	0.02	0.04	2.04
3	0	0.06	2.06

Then, $L = 6$ for this iteration such that the last column represent the L_s values.

5 Computational experiments

Some computational experiments are done to test the effectiveness of the proposed methods. The mathematical formulation is tested by running CPLEX 11., and the algorithms are implemented with MATLAB 7.6. These computational experiments are run on a DELL PC/2.20 GHz with 4.00 Go RAM.

The processing times are generated randomly from a uniform distribution between 20 and 50, the minimal and maximal time lags are generated from a uniform distribution in the intervals [0, 7] and [0, 14] successively as the same way in (Fondrevelle et al. 2006). The due dates are generated as the same way in (Nishi et al. 2009) by the random numbers on uniform distribution between

$[P(1-\alpha-\gamma/2), P(1-\alpha+\gamma/2)]$. P is a lower bound of the makespan which is given as follows:

$$P = max\left\{\left(\max_{1\le k\le m}\sum_{i=1}^{n}p_{i,k} + \min_{i}\sum_{l=1}^{k-1}\left(p_{i,l}+\theta_{i,l}^{min}\right) + \min_{i}\sum_{l=k+1}^{m}\left(p_{i,l}+\theta_{i,l}^{min}\right)\right), \max_{i}\sum_{k=1}^{m}\left(p_{i,k}+\theta_{i,k}^{min}\right)\right\}$$

α is the tardiness factor of jobs to determine the length of due dates, and γ is the dispersion range of due dates. A larger α generate tight due dates, and smaller γ generates a small range of due dates (Nishi et al. 2009). α and γ are fixed to 0.4 and 0.8 respectively.

5.1 Evaluation of the heuristic algorithms and effect of the time lags on the results

First experiments are done to distuinguish the influence of the minimal and maximal lags on the results, on the CPU time and the number of nodes. They allow us to define four different classes of problems: problems with minimal and maximal time lags ($\theta^{min}=7$, $\theta^{max}=14$); problems without time lags constraints ($\theta^{min}=0$, $\theta^{max}=+\infty$); problems with minimal time lags only ($\theta^{min}=7$, $\theta^{max}=+\infty$); and problems with maximal time lags only ($\theta^{min}=0$, $\theta^{max}=14$). On the other hand, these experiments aim to test the performance of the proposed heuristic algorithms by determining the deviation percentage from the optimal solution. It is calculated as follows $\frac{UB-Op}{Op}\times 100$ (noted % in the table) where the UB is the solution found by

Table 5 The effect of the time lags on the results and performance of the heuristics

(n, m)	$(\theta^{min}, \theta^{max})$	*CPLEX*			*SPT* (%)	*ABM* (%)	*ECT* (%)
		T	*CPU(s)*	*Nodes*			
(5, 5)	(7, 14)	52	0.14	15	4.31	3.17	1.12
	(0, $+\infty$)	47	0.03	6	1.81	2.44	0.81
	(7, $+\infty$)	50	0.05	15	1.66	2.13	2.50
	(0,14)	48	0.37	41	4.45	1.82	1.37
(10, 5)	(7, 14)	250	0.45	549	2.75	1.15	0.01
	(0, $+\infty$)	130	0.19	107	2.16	2.68	1.15
	(7, $+\infty$)	210	0.13	163	4.13	3.58	2.86
	(0, 14)	123	0.33	756	3.36	2.34	1.81
(15, 5)	(7, 14)	369	42.82	14,469	4.13	1.95	0.91
	(0, $+\infty$)	260	10.78	3,322	2.67	3.07	1.93
	(7, $+\infty$)	320	15.41	5,992	3.25	2.68	0.42
	(0, 14)	277	34.31	16,669	2.01	4.11	2.07
(20, 4)	(7, 14)	613	52.14	27,995	1.13	3.45	0.12
	(0, $+\infty$)	555	23.24	10,889	2.77	1.05	0.09
	(7, $+\infty$)	598	18.56	15,423	3.13	3.67	3.57
	(0, 14)	568	55.76	44,239	3.62	2.63	1.45

the heuristic procedure and *Op* is the optimal solution obtained by running CPLEX to the proposed mathematical formulation. The results are drown in Table 5. We set four different configurations for number of jobs $n \in \{5, 10, 15, 20\}$, and the number of machines is set to 4 and 5. For each problem, ten instances are generated; then the average value of the total tardiness is determined. A total of 160 runs are executed (16 problems $\times 10$ tested instances for each one). Note that a limit of 20,000 s is set.

From Table 5, it's obvious that the minimal and maximal time lags have a significant effect on the objective value, the CPU time, and the number of nodes required to solve each problem. The tardiness values (T) are greater for the problems with only minimal time lags and both minimal and maximal time lags. it's not surprising as the makespan is greater for these problems which results in higher tardiness values. However, the problems with only maximal time lags and with both minimal and maximal time lags are hardest to be solved than the classical permutation flowshop problems and with only minimal time lags problems. They require more CPU time to be solved and the greatest number of nodes for almost all the problems sizes. The heuristic procedures are solved in less than 1 second for all the tested problems.

The proposed heuristic procedures perform very will. The deviation percentage from the optimal solution range between 1.13 and 4.45 for the *SPT* heuristic, between 1.05 and 4.11 for the *ABM* heuristic, and between 0.01 and 3.57 for the *ECT* heuristic. The last one performs better among the three proposed procedures. It provides the best solution 15 times among 16 found solutions. On the other hand, we can confirm that the deviation percentage depends on the tardiness value, it decreases with the increasing tardiness values. So that, the *ECT* heuristic is able to find the optimal solution for problems with potentially high total tardiness.

5.2 Evaluation of the lower bounds

Second experiments are done to test the effectiveness of the proposed lower bounds. Many problems with different sizes are generated. They are classified into small size problems and large size problems. For the first type, we set four different configurations for the number of jobs $n \in \{5, 10, 15, 20\}$ and three configurations for the number of machines $m \in \{4, 5, 10\}$. To evaluate the performance of the developed Lower Bounds (LB), we determine for each one its deviation percentage from the optimal solution as: $\frac{Op-LB}{LB} \times 100$.

For the large size problems, we set the number of jobs to $n \in \{30, 50, 100\}$ and the number of machines to 5 or 10. As the *CPLEX* is unable to solve problems with size larger than $(n = 20, m = 4)$, we use the *ECT* based heuristic (UB) instead of the optimal solution to determine the deviation percentage. It is then calculated as $\frac{UB-LB}{LB} \times 100$. 10 instances are tested for each problem size, then the average value is determined. The minimal and maximal time lags are generated from a uniform distribution in the intervals [0, 7] and [0, 14] respectively. It is worthy noted that we aim also to distuinguish the effect of the minimal and maximal time lags constraints on the the quality of the developed lower bounds. Here $LB1$ is the lower bound obtained by relaxing the integrality constraints.

For small size problems shown in Table 6, the CP based-lower bound is shown to be the best one and the average deviation percentage from the optimal solution doesn't exceed 2.64. This lower bound and both methods 1 and 2 perform better than the $LB1$. Method 2 which consists in relaxing the maximal time lags constraints is better. However, relaxing the minimal time lags constraints presented in method 1 can't provide a great enhancement. This result is not surprising, as it is known that problems with only maximal time lags are hardest to be solved than the problems with only minimal time lags according to the previous experiments. Then by relaxing the maximal time lags, the problems become easier to be solved and the result is better. The computation time of the proposed both methods 1 and 2 is less than 1 second, and then consume less time than $LB1$. Also, it is shown that the $LB1$ and the CP based lower bound suffer from much consuming time to solve problems and they prove their inability to solve large size problems.

In Table 7, we evaluate the performance of the LR for large size problems.

We extend the experiments to large size problems. We can confirm that the LR performs good until the size ($n = 100$, $m = 10$) with manageable CPU time for the two methods while confirming that method 2 consumes less CPU time. Also, in view of the difference between the two methods, we can confirm the results obtained for the small size problems.

Table 6 Performance of the lower bounds for small size problems

(n, m)	$CPLEX\ (Op)$	$LB1$		$Method\ 1$ (%)		$Method\ 2$ (%)		CP	
		%	$CPU(s)$	%	$CPU\ (s)$	%	$CPU\ (s)$	%	$CPU\ (s)$
(5, 5)	52	15.14	0.04	10.13	0.02	8.00	0.03	2.65	0.03
(10, 5)	250	20.61	0.32	13.23	0.08	10.39	0.07	3.06	0.05
(10, 10)	476	19.01	0.55	15.50	0.11	12.00	0.10	2.27	1.28
(15, 5)	369	15.13	13.45	12.30	0.08	10.00	0.06	1.75	8.34
(15, 10)	628	17.34	41.23	13.30	0.15	10.13	0.10	2.45	20.76
(20, 4)	613	18.19	38.43	14.00	0.19	12.13	0.22	3.67	33.13
Average		17.57		13.07		10.42		2.64	

Table 7 Performance of LR for large size problems

(n, m)	$ECT\ (UB)$	$Method\ 1$ (%)		$Method\ 2$ (%)	
		%	$CPU\ (s)$	%	$CPU\ (s)$
(30, 5)	936	20.45	1.12	15.89	1.33
(30, 10)	2,820	21.34	2.28	13.56	2.06
(50, 5)	2,744	18.77	3.61	15.55	0.71
(50, 10)	5,433	18.45	3.51	14.77	2.16
(100, 10)	7,748	17.55	9.35	13.34	8.50

6 Conclusion

The permutation flowshop scheduling problem with minimal and maximal time lags to minimize the total tardiness was studied in this paper. We proposed its corresponding mathematical formulation. Computationally efficient heuristic algorithms based on different rules are proposed to provide upper bounds for large size problems. Then, lower bounds based on different methods are derived where the LR is the main one. A solution methodology was presented and the Lagrangian multipliers were updated using the subgradient algorithm. Two methods of the LR were proposed which were distuinguished by the type of the relaxed constraints. Experimental results show a great performance of the proposed heuristic procedures and especially the one based on the *ECT* rule. On the other hand, both small and large problem sizes proved the good quality of the developed lower bounds in a few *CPU* time for the LR.

Several directions for further work can be of major interest: the proposed LR technique can be hybridized with other approaches such as the column generation and then we can assess the enhancement that can be occured to the results. It could be interesting also to study similar problems with non classical performance criteria such as the total tardiness and earliness.

References

Baker KR (1974) Introduction to sequencing and scheduling. Wiley, New York

Bulbul K, Kaminsky P, Yano C (2004) Flow shop scheduling with earliness, tardiness, and intermediate inventory holding costs. Nav Res Logist 51:407–445

Chu C, Proth JM (1996) Single machine scheduling with chain structured precedence constraints and separation time windows. IEEE Trans Robot Autom 12:835–844

Czerwinski CS, Luh PB (1994) Scheduling products with bills of materials using an improved Lagrangian relaxation technique. IEEE Trans Robot Autom 10:99–111

Dell'Amico M (1996) Shop problems with two machines and time lags. J Oper Res 44:777–787

Fondrevelle J, Oulamara A, Portmann MC (2006) Permutation flowshop scheduling problems with maximal and minimal time lags. Comput Oper Res 33:1540–1556

Foulds LR, Wilson JM (2005) Scheduling operations for the harvesting of renewable resources. J Food Eng 70:281–292

Ghassemi TF, Olfat LA (2010) A set of algorithms for solving the generalized tardiness flowshop problems. J Ind Syst Eng 4:156–166

Graham R, Lawler E, Lenstra JK, Rinnooy Kan AHG (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discret Math 5:287–326

Koulamas C (1994) The total tardiness problem: review and extensions. Oper Res 42:1025–1041

Liu G, Luh PB, Resch R (1997) Scheduling permutation flowshops using the lagrangian relaxation technique. Ann Oper Res 70:171–189

Luh PB, Hoitomt DJ (1993) Scheduling of manufacturing systems using the Lagrangian relaxation technique. IEEE Xplore 38:1066–1080

Nishi T, Isoya Y, Inuiguchi M (2009) An integrated column generation and lagrangian relaxation for flowshop scheduling problems: systems, man and cybernetics. SMC. IEEE international conference. pp 11–14

Nishi T, Hiranakaa Y, Inuiguchi M (2010) Lagrangian relaxation with cut generation for hybrid flowshop scheduling problems to minimize the total weighted tardiness. Comput Oper Res 37:189–198

Nishi T, Hiranakaa Y (2013) Lagrangian relaxation and cut generation for sequence-dependent setup time flowshop scheduling problems to minimise the total weighted tardiness. Int J Product Res 51:4778–4796

Pinedo M (1995) Scheduling theory, algorithms, and systems. Prentice Hall, Upper Saddle River, New Jersey

Yu W, Hoogeveen H, Lenstra JK (2004) Minimising makespan in a two machine flowshop with delays and unit time operations is NP-hard. J Sched 7:333–348