# Order Scheduling with Tardiness Objective: Improved Approximate Solutions*

Jose M. Framinan† Paz Perez-Gonzalez

Industrial Management, School of Engineering,
University of Seville. Camino de los Descubrimientos s/n. 41092 Seville, Spain

**Abstract**

The problem addressed in this paper belongs to the topic of order scheduling, in which customer orders –composed of different individual jobs– are scheduled so the objective sought refers to the completion times of the complete orders. Despite the practical and theoretical relevance of this problem, the literature on order scheduling is not very abundant as compared to job scheduling. However, there are several contributions with the objectives of minimising the weighted sum of completion times of the orders, the number of late orders, or the total tardiness of the orders. In this paper, we focus in the last objective, which is known to be NP-hard and for which some constructive heuristics have been proposed. We intend to improve this state-of-the-art regarding approximate solutions by proposing two different methods: Whenever extremely fast (negligible time) solutions are required, we propose a new constructive heuristic that incorporates a look-ahead mechanism to estimate the objective function at the time that the solution is being built. For the scenarios where longer decision intervals are allowed, we propose a novel matheuristic strategy to provide extremely good solutions. The extensive computational experience carried out shows that the two proposals are the most efficient for the indicated scenarios.

**Keywords: scheduling, customer order, tardiness, heuristics, matheuristics**

# 1 Introduction

In classic scheduling literature, jobs to be processed are treated as individual entities possibly belonging to different customers, and hence the objectives sought are related to the completion

*Preprint submitted to European Journal of Operational Research. DOI: https://doi.org/10.1016/j.ejor.2017.10.064

†Corresponding author. Tel. +3495 448 7327; fax: +3495 448 7329. E-mail address: framinan@us.es

times of the individual jobs, or to the differences between the completion times and their due dates or deadlines. However, in many real-life situations, a customer order is composed of different products that have to be processed in the shop, and it may be sensible to pursue objectives related to the completion of the order as a whole rather than to the individual jobs within the order. This is caused by the fact that many customers require to receive the complete order, which cannot be shipped before all jobs in the order are completed. Therefore, from their viewpoint, only the completion time of the full order is relevant (Ahmadi et al., 2005). In view of the frequency of real-life settings where this situation arises, a branch of scheduling labelled *order scheduling* has emerged as a new research area.

In this paper, we consider a case of order scheduling in which we have a facility with $m$ machines in parallel. Each machine can produce one (and only one) particular product type, i.e. they are considered to be dedicated machines. The problem under consideration consists of scheduling $n$ customer orders composed of some/all product types. This problem was first formulated by Ahmadi and Bagchi (1990), and several practical applications have been described, including some operations in the paper industry (Leung et al., 2005b), manufacturing of semi finished lenses (Ahmadi et al., 2005), the pharmaceutical industry (Leung et al., 2005a), or the assembly of operations (Leung et al., 2005b). Different objectives related to the orders can be considered for this problem, such as the minimisation of the sum of the completion times (see e.g. Wagneur and Sriskandarajah, 1993, Leung et al., 2005b, Roemer, 2006, Wang and Cheng, 2007, or Framinan and Perez-Gonzalez, 2017), or weighted sum of the completion time (Leung et al., 2007; Wang and Cheng, 2007). Due date related objectives have been also the subject of research: Leung et al. (2006) analyse the order scheduling problem with the objectives of maximum lateness and the total number of late orders. Finally, Xu et al. (2016) minimise the total tardiness if a position-based learning effect in the processing times can be assumed.

Our research focuses on the minimization of the total tardiness (i.e. the sum of the difference between the completion time of an order and its due date if this difference is positive) in the order scheduling setting, a problem that is known to be NP-hard (Ahmadi et al., 2005). Therefore, most research in this topic has focused on developing approximate methods. More specifically, Lee (2013) develops four fast constructive heuristics for the problem. Among them, it turns out that

the so-called OMDD heuristic is clearly the most efficient one, therefore being the best approximate method for the problem. Despite OMDD's performance and modest computational requirements, we believe that new efficient approximate methods can be proposed for this problem along two directions:

1. Whenever the scheduling decision has to be taken in very short or almost real-time intervals, the idea of developing an index to select the next order to be appended at the end of the schedule under construction seems to be very efficient. This is the idea behind OMDD, and also behind other efficient heuristics for other order scheduling problems (see e.g. Leung et al., 2005b). However, the index developed by these heuristics is greedy in the sense that it does not incorporate a look-ahead mechanism, so the contribution to the objective function of the unscheduled orders is not taken into account. Developing such index can improve the performance of these fast, constructive heuristics without compromising their running times.

2. For the case where a longer decision interval is allowed, more sophisticated procedures can be developed. More specifically, the employ of matheuristics –combination of mathematical procedures and metaheuristics– can use the knowledge of the problem domain to provide extremely high-quality solutions.

These two directions are addressed in this paper: First, we propose a new heuristic based on incorporating a look-ahead mechanism in order to be able to assess, not only the potential contribution of the candidate orders to the total tardiness, but also an estimation of the contribution to the objective function of the non scheduled orders. Second, we propose a novel efficient matheuristic strategy that is able to provide very high-quality solutions. To test the efficiency of these methods, we carry out an extensive computational experience that shows that our constructive heuristic outperforms OMDD –which is the state-of-art constructive heuristic for the problem–, and that the matheuristic strategy also outperforms other existing strategies.

The remainder of the paper is as follows: The problem is formally stated in Section 2, together with its background. The proposed constructive heuristic is described in Section 3, while the matheuristic strategy is presented in Section 4. The computational experience and the subsequent analysis of the results are carried out in Section 5. Finally, in Section 6 we discuss the main

conclusions of the research.

## 2 Problem statement and background

The problem under consideration can be formally stated as follows: There is a facility with $m$ dedicated machines in parallel, and $n$ customer orders, each order $j$ ($j = 1, \ldots, n$) with its corresponding due date $d_j$. Each order is composed of some/all the product types that have to be manufactured on one of the $m$ machines without pre-emption. The total amount of processing required by order $j$ on machine $i$ ($i = 1, \ldots, m$) is denoted by $p_{ij}$, i.e. order $j$ contains a number of units of the product type which is manufactured in machine $i$, thus requiring a total of $p_{ij}$ time units of machine $i$. This is equivalent to state that the number of units of a product type requested is different for each customer, which reflects the usual real-life situation. The objective of the decision problem is to sequence the orders so the total tardiness –defined as the sum across all orders of the difference between the completion times of the orders and their due dates whenever this difference is positive– is minimised.

A sequence or solution of the problem is given by $\Pi := (\pi_1, \pi_2, \ldots, \pi_n)$ a permutation of $n$ components, as it has been shown that the same permutation of orders for all machines is optimal for this problem (Lee, 2013). $C_{i,\pi_j}(\Pi)$, the completion time of product type $i$ in the order scheduled in the $j$ position in sequence $\Pi$, can be computed using the following recursive equation:

$$C_{i,\pi_j}(\Pi) = C_{i,\pi_{j-1}}(\Pi) + p_{i,\pi_j} \quad i = 1, \ldots, m, j = 1, \ldots, n \tag{1}$$

where $C_{i,\pi_0}(\Pi) := 0 \quad \forall i$. Thus $C_{\pi_j}(\Pi)$ the completion time of order scheduled in position $j$-th is then

$$C_{\pi_j}(\Pi) = \max_{1 \leq i \leq m} \{C_{i,\pi_j}(\Pi)\} \tag{2}$$

Analogously, the tardiness of order scheduled in position $j$-th is

$$T_{\pi_j}(\Pi) = \max\{C_{\pi_j}(\Pi) - d_{\pi_j}; 0\} \tag{3}$$

The total tardiness is computed as $T(\Pi) = \sum_{j=1}^{n} T_{\pi_j}(\Pi)$. Similarly, $C(\Pi)$ the sum of the completion times of the orders scheduled according to $\Pi$ can be computed as $C(\Pi) = \sum_{j=1}^{n} C_{\pi_j}(\Pi)$.

Since the order scheduling problem with completion time as objective is known to NP-hard for $m \geq 2$ (Roemer and Ahmadi, 1997), so it is our problem. As a consequence, approximate procedures are the best option to ensure solutions of good quality –although without optimality guarantee– in problems of a realistic size. In this regard, Lee (2013) propose four fast constructive heuristics for the problem. These are:

- TPT-EDD: Orders are sequenced in a non-decreasing order of index $\alpha_j$:

$$\alpha_j := \max\Big\{\sum_{i=1}^{m} p_{ij}; m \cdot d_j\Big\} \tag{4}$$

- MPT-EDD: Orders are sequenced in a non-decreasing order of index $\alpha_j$:

$$\alpha_j := \max\Big\{\max_{1\leq i\leq m}\{p_{ij}\}; m \cdot d_j\Big\} \tag{5}$$

- EDD-MCT. Orders are sequenced separately on each machine to construct an index to sort the orders. More specifically:

  - Step 1: For each machine $i$ $(1 \leq i \leq m)$, sequence the orders in non decreasing order of $\max\Big\{p_{ij}; d_j\Big\}$. Compute $C_{EDD-MCT}(i,j)$ the completion time of order $j$ in each one of the so-obtained $m$ sequences.
  - Step 2: Sequence the orders in non decreasing order of the following index:

$$\alpha_j := \max_{1\leq i\leq m}\Big\{C_{EDD-MCT}(i,j)\Big\} \tag{6}$$

- OMDD. This heuristic has the following steps:

  - Step 0: Set $\Omega := (1, 2, \ldots, n)$ (unscheduled orders), $\Pi := \varnothing$ (scheduled orders) and $t_i = 0$ $(1 \leq i \leq m)$.
  - Step 1: Select order $r$ in $\Omega$ with the minimum value of index $\alpha_j$:

$$\alpha_j = \max\Big\{ \max_{1\leq i\leq m}\{t_i + p_{ij}\} - \max_{1\leq i\leq m}\{t_i\}; d_j - \max_{1\leq i\leq m}\{t_i\}\Big\} \tag{7}$$

- Step 2: Remove order $r$ from $\Omega$, append it at the end of $\Pi$ and update $t_i := t_i + p_{i,r}$
- Step 3: If $\Omega$ is not empty, go to Step 1.

Clearly, TPT-EDD and MPT-EDD are basically sorting algorithms, while EDD-MCT has a multi-pass structure. OMDD uses a similar technique to that in the ECT heuristic by Leung et al. (2005b) for the flowtime order scheduling problem, i.e it constructs a sequence by appending the most suitable among the unscheduled orders. The extensive computational experience carried out by Lee (2013) shows that the OMDD is the most efficient among the four. Therefore, in the subsequent computational experiments presented in Section 5 we only include OMDD, which we have re-implemented.

The heuristics presented above require negligible CPU times, so they are well-suited to these cases where the interval to take the scheduling decision is very short. We already mentioned in Section 1 that, when longer decision intervals are allowed, more sophisticated procedures could be used to obtain solutions of very high quality. In recent years, a very efficient option to develop such procedures is the use of matheuristics. Matheuristics can be broadly defined as algorithms in which metaheuristic techniques are combined with mathematical procedures (Fanjul-Peyro et al., 2017). Within the scheduling field, we are aware of the contributions by Billaut et al. (2015); Ta et al. (2015) for the single-machine scheduling problem with tardiness objective, by Della Croce et al. (2014b) for the same layout with tardiness objective and release dates, by Della Croce et al. (2014a) for the 2-machine flowshop scheduling problem with total completion time, by Quang Chieu et al. (2013) for the 2-machine scheduling problem with total tardiness, by Lin and Ying (2016) for the no-wait flowshop scheduling problem with makespan objective, and by Fanjul-Peyro et al. (2017) for the unrelated parallel machine case with additional resources and makespan as objective. We are not aware of the application of matheuristics to the problem under consideration, or to other customer scheduling problems. However, the references cited before use position-based binary variables in the MILP models in the same manner as in our case –see Section 4.1–, so it is worth analysing the strategies employed to check their suitability for our problem. Among these

strategies, the first and most common is the one that we will denote as *Job-Position Fixing* (JPF), employed by Quang Chieu et al. (2013), Della Croce et al. (2014b), Della Croce et al. (2014a), Billaut et al. (2015), and Ta et al. (2015). This strategy consists in, given an initial solution, fixing a subset of contiguous jobs within this solution, and formulating a MILP model of the problem with an additional number of constraints representing the fixed positions of these selected jobs, keeping the rest unscheduled. Since this strategy has proved to provide excellent results in other scheduling problems, it is described in detail in Section 4.2 as it will be used to compare our proposed matheuristic strategy.

Lin and Ying (2016) use a property of the no-wait flowshop scheduling problem with makespan objective so it can be transformed into an special case of the asymmetric travelling salesman problem, which can be modelled solely using binary integer variables. This model is employed within the matheuristic strategy, so their approach cannot be used in our problem. Finally, Fanjul-Peyro et al. (2017) test several matheuristic strategies for the unrelated parallel machine case with additional resources and makespan as objective, including different JPF strategies, as well as a machine-assignment fixing strategy which clearly does not make sense in our problem with dedicated machines. Among these, the most efficient strategy consists of first solving a MILP model to schedule a subset of $g$ jobs, and, at each subsequent iteration $k$ ($k = 1, \ldots$), the job scheduled in the last position plus new $g-1$ unscheduled jobs are added, so the MILP model is solved assuming that the previously scheduled $k \cdot (g-1)$ jobs are fixed in their positions, and only the recently-added $g$ jobs have to be scheduled. This strategy, although potentially interesting for other problems, cannot be efficiently applied –at least in an straightforward manner– to due-date based objective functions (as in our case), as it is likely that, during the first iterations, the tardiness of the set $g$ is zero regardless the specific order in which the $g$ jobs are processed, so the MILP model would possibly find a solution which is not appropriate for the future iterations. Therefore, we will not include this strategy in the comparison carried out in Section 5.

In summary, although there are constructive heuristics available for the problem under consideration, we believe that there is room for improvement by using estimates of the contribution of the unscheduled jobs in the objective function. A proposal in this direction is presented in Section 3. In addition, it will be interesting to develop approximate procedures that use longer CPU times to

provide results of excellent quality. Our proposal is to employ a novel matheuristic strategy, which is described in Section 4.

# 3 A new Constructive Heuristic

The proposed heuristic constructs a solution $\Pi := (\pi_1, \ldots, \pi_n)$ from a set $\mathcal{U}$, representing the set of unscheduled orders. Initially, no order has been scheduled and consequently, $\mathcal{U}$ contains all orders. Then the heuristic starts an iterative process of $n$ steps: in step $k$, each order in $\mathcal{U}$ is selected as candidate to be appended at the end of $\Pi$. To select the chosen order among the candidates, for each order $\omega_l \in \mathcal{U}$ with due date $d_l$, a (partial) sequence $S_l$ is formed by appending $\omega_l$ at the end of $\Pi$, i.e. $S_l = (\pi_1, \ldots, \pi_{k-1}, \omega_l)$.

Then, the following tardiness indicator $\eta_l$ for the $l$-th candidate (order $w_l$) is computed as follows:

$$\eta_l = \max\{C(S_l) - d_l; 0\} + T^*(S_l) \tag{8}$$

where the first term represents the contribution to the total tardiness if selecting $\omega_l$, as $C(S_l)$ is the completion time of the (partial) sequence $S_l$, and $d_l$ its corresponding due date. $T^*(S_l)$ is an estimate of the contribution to the total tardiness of the remaining unscheduled orders.

Note that the contribution of the scheduled orders to the completion times (and therefore to the total tardiness) depends on how the remaining orders are scheduled. Since the objective is to obtain an estimate of a sort of *average* contribution, we assume that the remaining orders are scheduled following a specific sequence, which is fixed from the beginning of the algorithm to reduce the computation burden. More specifically, we use $\Omega$ a given sequence of all orders before starting the iterative procedure, and, for each iteration, we will compute $T^*(S_l)$ as if the remaining orders are scheduled after the orders in $\Pi$ according to the relative ordering given by $\Omega$. In our case we propose using the sequence given by sorting the orders according to the Earliest Due Date (EDD) rule. In this manner, after selecting one order in $\Omega$ to be scheduled, this order is appended at the end of the partial sequence and removed from $\Omega$. Therefore, in the next iteration the remaining orders in $\Omega$ are already sorted in EDD sequence, so they do not have to be computed again.

It is clear that selecting the order with the lowest value of $\eta$ would have the smallest estimated

contribution to the total tardiness. Consequently, the order with the lowest value of $\eta$ is removed from $\mathcal{U}$ and appended at the end of $\Pi$. The procedure is repeated until $\mathcal{U} = \varnothing$. The pseudocode of the heuristic is given in Figure 1. It can be seen that the complexity of this heuristic is determined by the main iteration loop of complexity $O(n^3m)$.

# 4 Matheuristic

In order to present the matheuristic strategy proposed, we first describe in Section 4.1 the MILP model for the problem under consideration, together with a further simplification in order to reduce the number of constraints of the problem. In Section 4.2 we first describe the JPF approach –which is, as discussed in Section 2, the main matheuristic strategy for scheduling problems– in order to illustrate the novel strategy proposed for our problem in Section 4.3.

## 4.1 MILP formulation

A classical formulation of the problem would be based on positional variables, which are known to be the best one for a range of scheduling problems (see e.g. Della Croce et al., 2014b): Let $x_{kj}$ to be a binary variable equal to 1 if order $k$ is sequenced in the $j$-th position, and $C_j$ and $T_j$ the completion time and the tardiness of the order sequenced in the $j$-th position, respectively. Recall that $p_{ik}$ refers to the processing time on machine $i$ of job $k$, and that $d_k$ is the due date of job $k$.

The resulting model MILP is as follows:

$$\min \sum_{j=1}^{n} T_j \tag{9}$$

subject to

$$\sum_{k=1}^{n} x_{kj} = 1 \quad j = 1, \ldots, n \tag{10}$$

$$\sum_{j=1}^{n} x_{kj} = 1 \quad k = 1, \ldots, n \tag{11}$$

**Procedure** *FP*

$\Pi := \varnothing$;
Obtain sequence $\Omega := (\omega_1, \ldots, \omega_n)$ by sorting orders according to EDD rule;
Set $ct_i = 0$ $(1 \leq i \leq m)$;
// Append orders one by one
**for** $j = 1$ **to** $n$ **do**
  // Compute indicator for each unscheduled order
  **for each** $\omega_l \in \Omega$ **do**
    // Compute $ect_i$ the estimated completion times of $w_l$ for each machine $i$;
    **for** $i = 1$ **to** $m$ **do**
      $ect_i = ct_i + p_{i,w_l}$;
    **end**
    // Compute first term of indicator $\eta_l$, see Equation (8)
    $\eta_l = \max\Big\{ \max_{1 \leq i \leq m}\{ect_i\} - d_{w_l}; 0\Big\}$;
    // Compute second term of indicator $\eta_l$
    **for each** $w_k \in \Omega - \{w_l\}$ **do**
      **for** $i = 1$ **to** $m$ **do**
        $ect_i = ect_i + p_{i,w_k}$;
      **end**
      $\eta_l = \eta_l + \max\Big\{ \max_{1 \leq i \leq m}\{ect_i\} - d_{w_k}; 0\Big\}$;
    **end**
  **end**
  // Select order with minimum value of $\eta_l$
  $r := \text{argmin}_{1 \leq l \leq n-j+1}\, \eta_l$;
  Append $\omega_r$ at the end of $\Pi$, i.e. $\Pi := (\pi_1, \ldots, \pi_{j-1}, \omega_r)$;
  Extract $\omega_r$ from $\Omega$, i.e. $\Omega := (\omega_1, \ldots, \omega_{r-1}, \omega_{r+1}, \ldots, \omega_{n-j+1})$;
  // Update completion times
  **for** $i = 1$ **to** $m$ **do**
    $ct_i = ct_i + p_{i,w_r}$;
  **end**
**end**
**end**

Figure 1: Constructive heuristic proposed

$$\sum_{k=1}^{n}\sum_{r=1}^{j} p_{ik}\cdot x_{kr} \leq C_j \quad i=1,\ldots,m \quad j=1,\ldots,n \tag{12}$$

$$C_j - \sum_{k=1}^{n} d_k \cdot x_{kj} \leq T_j \quad j=1,\ldots,n \tag{13}$$

$$x_{kj} \in \{0,1\} \quad T_j \geq 0 \quad C_j \geq 0 \quad k=1,\ldots,n \quad j=1,\ldots,n \tag{14}$$

Equation (9) expresses the objective function. Constraints (10) state that one (and only one) order is sequenced in the $j$-th position, while constraints (11) ensure that each order $k$ is sequenced in one (and only one) position. Constraints (12) state that the completion time of an order is given by the maximum of the processing times of the components of the order in the machines. The definition of tardiness is ensured by constraints (13), while constraints (14) express the domain of the variables of the model.

The resulting MILP model has $n\cdot(m+3)$ constraints and $n\cdot(n+2)$ variables. A reduction of the model can be achieved by removing variables $C_j$: Constraints (13) can be written as $C_j \leq T_j + \sum_{k=1}^{n} d_k \cdot x_{kj}$, so they can be combined with constraints (12) into:

$$\sum_{k=1}^{n}\left(\sum_{r=1}^{j} p_{ik}\cdot x_{kr} - d_k \cdot x_{kj}\right) \leq T_j \quad i=1,\ldots,m \quad j=1,\ldots,n \tag{15}$$

which replace constraints (12) and (13). The resulting model has $n\cdot(m+2)$ constraints and $n\cdot(n+1)$ variables. Therefore, the model minimising (9) subject to constraints (10), (11), (14) and (15) will be used in the following.

## 4.2 JPF matheuristic strategy

The JPF strategy consists in, given a current solution, a subset of contiguous jobs in the solution sequence is left free while the remaining jobs are kept in the same positions. More specifically, given a solution $\Pi$ (which corresponds to a given set of compatible $\bar{x}_{kj}$ variables), a job index $r$, and a so-called *job window* of size $h$, let us define $\bar{S}(\Pi, r, h)$ the index set of the free jobs, i.e. those within the job window. Therefore, jobs that are not in positions $r, r+1, \ldots, r+h-1$ in sequence $\Pi$ are

**Algorithm** *JPF(h, $t_w$, $t_{limit}$)*

Let $\bar{\mathbf{x}}$ contain the positional binary variables corresponding to a heuristic solution $\Pi$.
**repeat**
  Set *improved* = false;
  **repeat**
    Pick $r \in \{1, \ldots, n-h+1\}$ randomly;
    Compute $\bar{S}(\Pi, r, h)$;
    Solve the minimisation of (9) subject to (10), (11), (14), (15), and (16) during $t_w$ seconds. Let $\hat{\mathbf{x}}$ be the so-obtained solution;
    **if** *f($\bar{\mathbf{x}}$) > f($\hat{\mathbf{x}}$)* **then**
      Set $\bar{\mathbf{x}} = \hat{\mathbf{x}}$;
      Set *improved* = true;
    **end**
  **until** *improved* **or** *all r values have been tried*;
**until not** *improved* **or** $t_{limit}$ *exceeded*;
**end**

Figure 2: Procedure based on the JPF strategy

assumed to be fixed, so the corresponding positional binary variables, $x_{kj}$ where $k \notin \overline{S}(\Pi, r, h)$, are set to those corresponding values in the solution for $\Pi$, i.e. $x_{kj} = \overline{x}_{kj}$. These equalities are added to the set of constraints of the original model so this enhanced MILP model –denoted as *window re-optimisation model*– would find the best positions for the solutions in $\bar{S}(\Pi, r, h)$. More specifically, the following set of $n \cdot (n-h)$ constraints is added (assuming $r < n-h$):

$$x_{kj} = \bar{x}_{kj} \quad \forall k \notin \bar{S}(\Pi, r, h) \quad j = 1, \ldots, n \tag{16}$$

Therefore, for a given parameter $h$ and an index $r$, a window re-optimisation model can be solved using a MILP solver. With these ingredients, an iterative procedure naturally arises by exploring the neighbourhood of an index $r$ ($r = 1, \ldots, n-h+1$), where the domain of the index $r$ is explored in random order. This procedure –used by Quang Chieu et al. (2013), Della Croce et al. (2014b), Della Croce et al. (2014a), Billaut et al. (2015), and Ta et al. (2015)– is presented in Figure 2.

## 4.3 Proposed matheuristic strategy

Our proposed strategy –denoted as Job-Position Oscillation, or JPO in the following– also starts with a given solution $\Pi$, but, in our case, when searching for a solution using a MILP model we

restrict the order in position $j$ in $\Pi$ to be displaced back and forth a maximum of $\delta$ positions.

More specifically, given a sequence $\Pi$, for all orders $k$ ($k = 1, \ldots, n$) we find $w_k(\Pi)$ the position of order $k$ in $\Pi$. Then, a set of *allowed* positions for $k$ are found so order $k$ can only move at maximum $\delta$ positions backwards and $\delta$ positions forward ($\delta$ is thus a parameter of the matheuristic that would be called *oscillation*). The set of allowed positions for order $k$ is $\mathcal{P}_k(\Pi, \delta) = \{\underline{w}_k(\Pi, \delta), \underline{w}_k(\Pi, \delta) + 1, \ldots, \bar{w}_k(\Pi, \delta)\}$, where $\underline{w}_k(\Pi, \delta) = \max\{0; w_k(\Pi) - \delta\}$ and $\bar{w}_k(\Pi, \delta) = \min\{n; w_k(\Pi) + \delta\}$. In this manner, for each order $k$ ($k = 1, \ldots, n$), positional variables $x_{kj}$ are not meaningful for $j \notin \mathcal{P}_k(\Pi, \delta)$, and the size of the resulting MILP model is reduced with respect to the number of variables. Analogously, the set of orders that can occupy position $j$ can be defined as $\mathcal{O}_j(\Pi, \delta) = \{k : j \in \mathcal{P}_k(\Pi, \delta)\}$.

The reduced MILP model can be expressed as follows:

$$\min \sum_{j=1}^{n} T_j \tag{17}$$

subject to

$$\sum_{k \in \mathcal{O}_j(\Pi, \delta)} x_{kj} = 1 \quad j = 1, \ldots, n \tag{18}$$

$$\sum_{j \in \mathcal{P}_k(\Pi, \delta)} x_{kj} = 1 \quad k = 1, \ldots, n \tag{19}$$

$$\sum_{k \in \mathcal{O}_j(\Pi, \delta)} \left( \sum_{r=\underline{w}_k(\Pi, \delta)}^{\min\{j; \bar{w}_k(\Pi, \delta)\}} p_{ik} \cdot x_{kr} - d_k \cdot x_{kj} \right) \leq T_j \quad i = 1, \ldots, m \quad j = 1, \ldots, n \tag{20}$$

$$\begin{aligned} x_{kj} \in \{0, 1\} \quad k = 1 \ldots, n \;\; j \in \mathcal{P}_k(\Pi, j) \\ T_j \geq 0 \quad j = 1, \ldots, n \end{aligned} \tag{21}$$

Equation (17) expresses the objective function. Constraints (18) state that one (and only one) order, among the set of orders allowed in this position, is sequenced in the $j$-th position, while constraints (19) ensure that each order $k$ is sequenced in one (and only one) of its allowed posi-

tions. Constraints (20) state that the completion time of an order is given by the maximum of the processing times of the components of the order in the machines. Finally, constraints (21) express the domain of the variables in the reduced model.

The following property establishes the reduction in the MILP model:

**Property 1.** *For $\delta$ an oscillation parameter with $\delta < n$, the number of variables $x_{kj}$ in the reduced MILP is $n(2\delta + 1) - \delta(\delta + 1)$.*

*Proof.* First, note that the number of variables depends on the number of elements in set $\mathcal{P}_k(\Pi, \delta)$ for each $k$. In turn, the cardinality of $\mathcal{P}_k(\Pi, \delta)$, denoted $|\mathcal{P}_k(\Pi, \delta)|$, depends on the position $j$ occupied by order $k$. Therefore, $|\mathcal{P}_k(\Pi, \delta)|$ depends on $j$ as follows:

- If $j \leq \delta \Rightarrow |\mathcal{P}_k(\Pi, \delta)| = j + \delta$
- If $\delta + 1 \leq j \leq n - \delta \Rightarrow |\mathcal{P}_k(\Pi, \delta)| = 2\delta + 1$
- If $j > n - \delta \Rightarrow |\mathcal{P}_k(\Pi, \delta)| = \delta + n - j + 1$

Since, in the reduced model, we compute $\mathcal{P}_k(\Pi, \delta) \ \ \forall \ \ k = 1, \ldots, n$, the total number of variables is equal to

$$\sum_{j=1}^{n} |\mathcal{P}_k(\Pi, \delta)| = \sum_{j=1}^{\delta}(j + \delta) + \sum_{j=\delta+1}^{n-\delta} (2\delta + 1) + \sum_{j=n-\delta+1}^{n} (\delta + n - j + 1) =$$

$$= \sum_{j=1}^{\delta} j + \delta^2 + (n - \delta - \delta)(2\delta + 1) + (n - (n - \delta))(\delta + n + 1) - \sum_{j=n-\delta+1}^{n} j =$$

$$= \delta^2 + (n - 2\delta)(2\delta + 1) + \delta(\delta + n + 1) - \delta(n - \delta) = n(2\delta + 1) - \delta(\delta + 1)$$

Note that the resulting value is $n$ times the number of positions as if it would be possible to move $\delta$ positions back and forth, minus the sum of the $\delta$ first and last indices.

□

The next corollary establishes the conditions for the resulting MILP model to be smaller than the original one:

**Corollary 1.** *The number of variables $x_{kj}$ in the resulting MILP is lower than in the original MILP if and only if $\delta < n - 1$.*

*Proof.* According to the property above, the number of variables $x_{kj}$ in the resulting MILP is $n(2\delta + 1) - \delta(\delta + 1)$, being $n^2$ in the original MILP. It follows that

$$n(2\delta + 1) - \delta(\delta + 1) < n^2 \Leftrightarrow n^2 - n(2\delta + 1) + \delta(\delta + 1) > 0 \Leftrightarrow$$

$$\Leftrightarrow (n - (\delta + 1))(n - \delta) > 0 \Leftrightarrow n < \delta \text{ or } n > \delta + 1$$

Since $\delta < n$ by hypothesis, then the number of variables $x_{kj}$ in the resulting MILP is lower than $n^2$ if and only if $n > \delta + 1$. □

The last corollary determines the number of variables in the resulting MILP:

**Corollary 2.** *The number of variables in the resulting MILP is* $(2n - \delta)(\delta + 1)$.

*Proof.* The proof is obvious taking into account Property 1 and the fact that the reduced MILP model also includes $n$ continuous variables $T_j$. □

Clearly, the proposed strategy can be embedded in an iterative procedure, which is shown in Figure 3. Since the size of $\delta$ may play a crucial role in the strategy, we include a mechanism for avoiding the procedure not to be trapped in a solution, i.e. if the current solution does not improve the best-so-far solution, then $\delta$ is increased so a less-restricted MILP model is solved. Note that, since the time allotted to this bigger problem is also $t_w$, probably the solution found by the solver is not of a very good quality, but at least this mechanism allows the procedure to escape. In some preliminary tests, we found that 10 is a good value to increase $\delta$, so it was used for the experiments.

Finally, note that the property presented in this Section and its corollaries can be employed to properly set the parameters of the strategy: In a first step, preliminary tests can be conducted to determine the problem size $n$ that can be optimally solved within $t_w$ seconds. Then, using the second corollary, $\delta$ could be determined for each instance size accordingly. However, in order to conduct a fair comparison with the JPF strategy, in this paper we will not make use of these results and the values of $t_w$ and $t_{limit}$ would be the same for all strategies.

**Algorithm** *JPO($\delta$, $t_w$, $t_{limit}$)*

Let $\bar{\mathbf{x}}$ contain the positional binary variables corresponding to a heuristic solution $\Pi$.
$\Pi_{best} := \Pi$
$\delta_{curr} := \delta$
**repeat**
  **for** *k=1,...,n* **do**
    Compute $\mathcal{P}_k(\Pi, \delta_{curr})$
  **end**
  Solve the minimisation of (17) subject to (18), (19), (20), (21) using $\mathcal{P}_k(\Pi, \delta_{curr})$ $(k = 1, \ldots, n)$ during $t_w$ seconds. Store in $\bar{\mathbf{x}}$ the so-obtained solution;
  Store in $\Pi$ the sequence corresponding to solution $\bar{\mathbf{x}}$
  **if** *f($\mathbf{x_{best}}$) > f($\bar{\mathbf{x}}$)* **then**
    $\Pi_{best} := \Pi$
    $\mathbf{x_{best}} := \bar{\mathbf{x}}$
    $\delta_{curr} := \delta$
  **else**
    $\delta_{curr} := \delta_{curr} + 10$
  **end**
**until** $t_{limit}$ *is not exceeded*;
**end**

Figure 3: Algorithm proposed

# 5 Computational experiments

In this section we describe the experiments carried out to assess the performance of the constructive heuristic and the matheuristic proposed in Sections 3 and 4 respectively. In Section 5.1 we present the test instances employed to conduct the experiments. The algorithms under comparison and the criteria employed to evaluate their performance are presented in Section 5.2. Finally, in Sections 5.3 and 5.4 we describe the results of the experiments.

## 5.1 Testbed design

The testbed design is aimed at two goals:

1. To assess the efficiency of the constructive heuristic proposed in Section 3 as compared to that in the existing literature, most notably the OMDD heuristic by Lee (2013).

2. To assess the efficiency of the matheuristic strategy proposed in Section 4.

Regarding the first goal, in order to conduct a fair comparison, we develop a testbed –labelled in

the following $SMALL$ – that replicates the instance sizes and parameters presented in Lee (2013). More specifically, $n \in \{10, 20, 30, 40, 50\}$ and $m \in \{2, 5, 8\}$. The processing times for job $k$ in order $j$ are generated following a uniform distribution $U[1, 100]$. The due date $d_j$ for each order is generated randomly from a uniform distribution $U[P(1 - TF - RDD/2), P(1 - TF + RDD/2)]$, where:

- $P$ is the sum of processing times divided by the number of machines, i.e. $P = \sum_{k=1}^{m} \sum_{j=1}^{n} p_{jk}/m$ and thus can be computed for each problem instance.
- $RDD$ is the range of the due dates: $RDD \in \{0.2, 0.5, 0.8\}$.
- $TF$ is the tardiness factor of the due dates: $TF \in \{0.2, 0.5, 0.8\}$.

Note that the combination of $TF = 0.8$ and $RDD = 0.5|0.8$ may generate negative due dates, a case that it is avoided by drawing a new due date until it is non-negative. For each combination of parameters, 20 instances have been generated, resulting in a total of 2,700 instances.

Regarding the second goal –to assess the efficiency of the matheuristic strategy–, the $SMALL$ testbed is not sufficient and therefore we build a second testbed –denoted $BIG$– containing instances of larger sizes. Furthermore, the performance of the constructive heuristics can be also tested in $BIG$, so a more complete assessment is obtained. This larger testbed contains instance sizes with $n \in \{100, 150, 200, 300\}$, and $m \in \{5, 10\}$. The same procedure for processing times and due date generation as in $SMALL$ has been adopted, and also 20 instances of the same size and combination of parameters have been generated. Therefore, this testbed is composed of 1,440 instances.

## 5.2 Algorithms under comparison

Regarding constructive heuristics –tested on $SMALL$ and $BIG$ testbeds–, the following algorithms have been compared:

- The $EDD$ (Earliest Due Date) heuristic for base comparison.
- The $OMDD$ heuristic by Lee (2013).
- The proposal presented in Section 3, denoted as $FP$.

Regarding matheuristic strategies, we use the $BIG$ testbed to compare $JPF(h, t_w, t_{limit})$ –embedded in the procedure in Figure 2–, and $JPO(\delta, t_w, t_{limit})$ –embedded in the procedure in Figure 3–. Clearly, in order to conduct a fair comparison, the total amount of time $t_{limit}$ and the time to solve the MILP model $t_w$ should be the same for each strategy. In our experiments, we use the values $t_{limit} = 600$ seconds and $t_w = 60$ seconds, as these are the values most commonly employed for the JPF strategy, see e.g. Della Croce et al. (2014b). Finally, different values of $h$ and $\delta$ are tested to cover different possibilities. After some preliminary experiments, the most promising values are $h \in \{30, 40\}$ and $\delta \in \{10, 20\}$. In addition, an optimal solution for all problems in both $SMALL$ and $BIG$ has been tried by solving the MILP model in Section 4.1 using the Gurobi 7 solver (Gurobi Optimization Inc., 2017) with the same time limit $t_{limit} = 600$ seconds.

In summary, we have tallied the results obtained by the following procedures:

- $JPF(30, 60, 600)$, denoted as JPF30 in the results.
- $JPF(40, 60, 600)$, denoted as JPF40 in the results.
- The $JPO(10, 60, 600)$, denoted as JPO-10 in the results.
- The $JPO(20, 60, 600)$, denoted as JPO-20 in the results.
- The MILP model running during 600 seconds, denoted as MILP in the results.

The results obtained by the constructive heuristics and the matheuristics on each instance have been recorded using the following three indicators:

- $RDI$ (Relative Deviation Index). When testing a set of heuristics $H$, the $RDI$ obtained by heuristic $s \in H$ when applied to instance $t$ is defined as follows:

$$RDI_{st} = \begin{cases} 0 & \text{if } \min_{h \in H} T_{ht} = \max_{h \in H} T_{ht} \\ \frac{T_{st} - \min_{h \in H} T_{ht}}{\max_{h \in H} T_{ht} - \min_{h \in H} T_{ht}} \cdot 100 & \text{otherwise} \end{cases} \tag{22}$$

  being $T_{st}$ the tardiness value obtained by heuristic $s$ in instance $t$. In our case $\min_{h \in H} T_{ht}$ is the optimal value if the MILP model is able to find the optimal solution within the assigned computation time, or the best solution found among the heuristics and matheuristics

otherwise. Note that $RDI$ is an standard indicator for measuring the quality of approximate solutions in scheduling problems where due dates are involved (see e.g. Kim, 1993, Fernandez-Viagas and Framinan, 2015, or Karabulut, 2016).

- Optimality, $Opt \in \{0, 1\}$, being 1 if the solution obtained is equal to that provided by the MILP (if the MILP has reached the optimum); and 0 otherwise.
- CPU time in seconds.

The heuristics and matheuristics have been coded in C# using Microsoft Visual Studio 2013 and Gurobi 7 .NET API in the case of the matheuristics. The algorithms have been run in an Intel Core i7-3770 with 3.4 GHz and 16 GB RAM computer. Special care has been taken to use the same data structures and methods.

## 5.3 Results for the $SMALL$ testbed

Table 1 shows the results for the different values of $m$ and $n$, including the average and standard deviation of the $RDI$ values –except for the MILP column–, the percentages of times that a method has found the optimal value, and the average computational time required by each method. Average and standard deviation $RDI$ are zero for MILP as the total tardiness found for all instances is the minimum across all heuristic under comparison. Table 2 shows the same results aggregated with respect to $TF$ and $RDD$. From both tables, it can be seen that FP obtains the best results regardless the problem size in terms of $RDI$, and the highest percentage of optima. Additionally, it is the method with lowest standard deviation values, which speaks for its relative robustness. Tukey 95% confidence intervals show that there are statistical differences with respect to the $RDI$ values obtained by FP with respect to EDD and OMDD for different values of $TF$ (see Figure 4a), and for different values of $RDD$ (see Figure 4b). Finally, regarding the CPU times, all tested heuristics provide their results almost instantaneously.

## 5.4 Results for the $BIG$ testbed

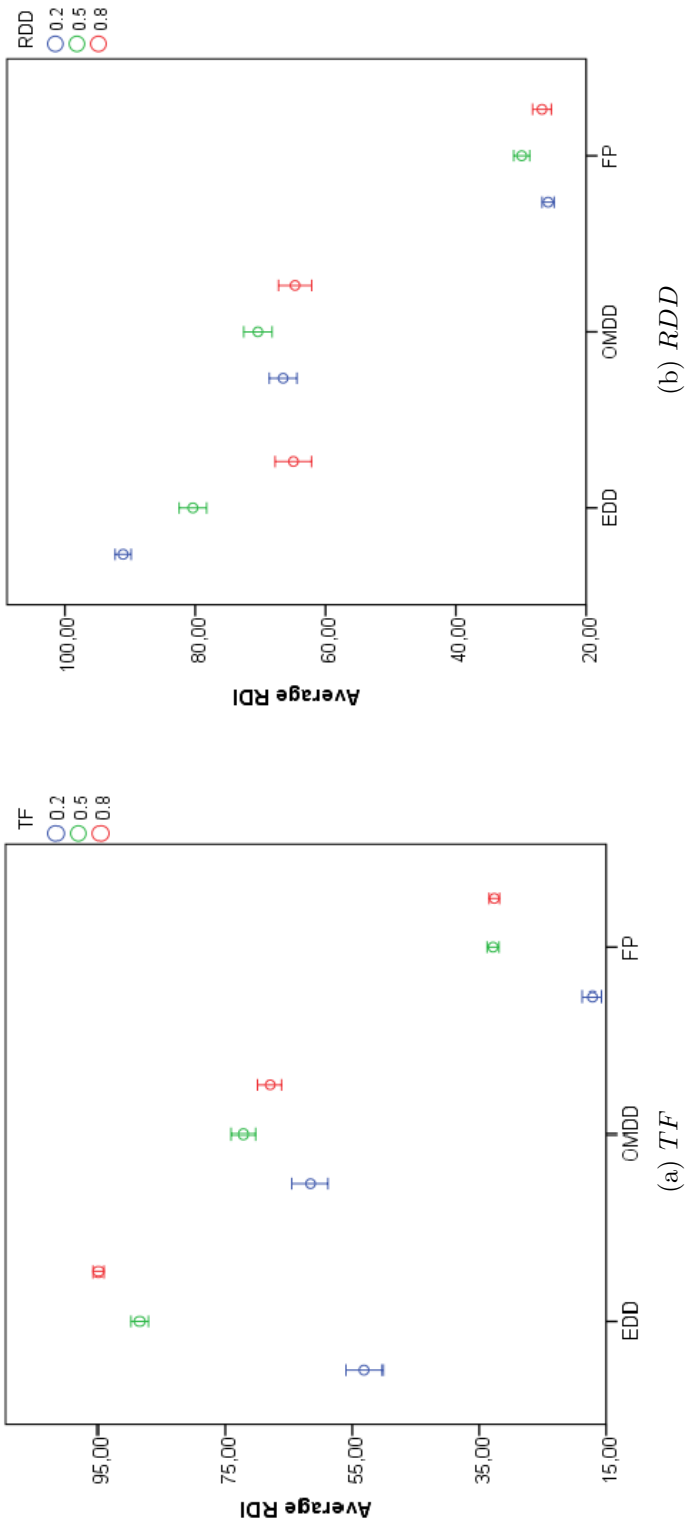
Table 3 shows the results for the different constructive heuristics and matheuristics grouped by problem size. CPU times are not shown as all methods (except all constructive heuristic, which

| | | MILP | | EDD | | | | OMDD | | | | FP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | Opt | Time | Average | St. Dv | %Opt | Time | Average | St. Dv | %Opt | Time | Average | St. Dv | %Opt | Time |
| 2 | 10 | 180 | 0.0436 | 75.9297 | 40.0837 | 17.78 | 0.0000 | 43.5491 | 39.6692 | 20.00 | 0.0000 | 22.3157 | 22.4437 | 23.89 | 0.0000 |
| | 20 | 180 | 0.0985 | 77.8515 | 34.7471 | 10.00 | 0.0000 | 60.6777 | 36.9657 | 8.33 | 0.0000 | 24.4407 | 19.8386 | 14.44 | 0.0000 |
| | 30 | 180 | 0.1520 | 82.4306 | 30.6172 | 6.11 | 0.0000 | 68.3169 | 33.0699 | 5.00 | 0.0000 | 26.1245 | 19.6421 | 12.22 | 0.0000 |
| | 40 | 180 | 0.1938 | 83.5197 | 28.4011 | 3.33 | 0.0000 | 69.2476 | 33.3303 | 3.33 | 0.0000 | 29.0727 | 20.8933 | 9.44 | 0.0000 |
| | 50 | 180 | 2.9693 | 72.7579 | 40.4282 | 16.67 | 0.0001 | 53.1584 | 38.5828 | 13.33 | 0.0000 | 23.1598 | 18.9296 | 18.89 | 0.0000 |
| 5 | 10 | 180 | 19.6285 | 81.5004 | 33.0318 | 10.00 | 0.0000 | 65.3690 | 34.1127 | 6.67 | 0.0000 | 29.6373 | 18.8066 | 12.78 | 0.0002 |
| | 20 | 167 | 88.3402 | 77.8111 | 32.9170 | 10.18 | 0.0000 | 77.2824 | 28.1561 | 3.59 | 0.0001 | 27.6686 | 17.4563 | 14.37 | 0.0000 |
| | 30 | 161 | 104.7956 | 78.2377 | 32.2667 | 7.45 | 0.0000 | 79.8393 | 27.1188 | 3.11 | 0.0000 | 28.9105 | 17.4305 | 11.80 | 0.0001 |
| | 40 | 157 | 106.7195 | 78.3025 | 37.7940 | 19.11 | 0.0000 | 56.3522 | 36.2819 | 13.38 | 0.0000 | 25.9515 | 17.3728 | 20.38 | 0.0004 |
| | 50 | 112 | 252.6086 | 82.6439 | 31.2004 | 14.29 | 0.0000 | 71.3124 | 31.2947 | 12.50 | 0.0001 | 30.2097 | 18.6205 | 16.96 | 0.0002 |
| 8 | 10 | 95 | 303.8654 | 81.1507 | 31.7504 | 15.79 | 0.0000 | 76.0789 | 29.7459 | 10.53 | 0.0001 | 31.1384 | 16.3511 | 18.95 | 0.0004 |
| | 20 | 85 | 337.9329 | 78.5165 | 32.3313 | 15.29 | 0.0000 | 83.5724 | 25.4359 | 5.88 | 0.0000 | 29.4887 | 16.7947 | 24.71 | 0.0002 |
| | 30 | 117 | 227.3777 | 77.0892 | 39.2341 | 29.06 | 0.0000 | 53.8274 | 36.2292 | 23.08 | 0.0001 | 26.1182 | 17.5978 | 29.06 | 0.0002 |
| | 40 | 93 | 315.3696 | 75.7516 | 36.3348 | 23.66 | 0.0000 | 72.0354 | 34.6164 | 16.13 | 0.0000 | 29.2427 | 18.1286 | 29.03 | 0.0008 |
| | 50 | 71 | 375.3700 | 78.2734 | 33.2023 | 25.35 | 0.0000 | 77.1922 | 30.9672 | 18.31 | 0.0000 | 29.0536 | 16.4601 | 25.35 | 0.0007 |
| Total | | 2138 | 142.3643 | 78.7844 | 34.5021 | 13.66 | 0.0000 | 67.1874 | 35.0141 | 10.20 | 0.0000 | 27.5022 | 18.6567 | 17.63 | 0.0002 |

Table 1: $SMALL$ Testbed: Constructive Heuristics for $m$ and $n$.

| | | MILP | | EDD | | | | OMDD | | | | FP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $TF$ | $RDD$ | Opt | Time | Average | St. Dv | %Opt | Time | Average | St. Dv | %Opt | Time | Average | St. Dv | %Opt | Time |
| 0.2 | 0.2 | 300 | 1.2919 | 85.3032 | 25.0632 | 1.33 | 0.0000 | 66.3451 | 35.9498 | 3.00 | 0.0000 | 25.4339 | 19.6942 | 6.33 | 0.0003 |
| | 0.5 | 300 | 2.2589 | 56.9273 | 42.7857 | 23.67 | 0.0000 | 71.7665 | 40.0724 | 14.33 | 0.0001 | 19.7393 | 23.2666 | 35.67 | 0.0001 |
| | 0.8 | 300 | 3.0981 | 17.0550 | 35.3555 | 72.33 | 0.0000 | 46.6315 | 49.0762 | 50.67 | 0.0000 | 6.4722 | 18.7154 | 81.67 | 0.0002 |
| 0.5 | 0.2 | 268 | 79.2077 | 92.3365 | 16.4831 | 0.00 | 0.0000 | 65.9069 | 31.8415 | 1.87 | 0.0000 | 26.4286 | 10.0053 | 0.75 | 0.0003 |
| | 0.5 | 255 | 121.9268 | 89.2307 | 18.6282 | 0.00 | 0.0000 | 72.2798 | 30.7637 | 1.57 | 0.0000 | 36.4116 | 15.0155 | 0.00 | 0.0004 |
| | 0.8 | 184 | 248.5713 | 83.6032 | 23.4848 | 0.00 | 0.0000 | 77.9867 | 28.5685 | 1.09 | 0.0001 | 35.4010 | 16.9840 | 2.17 | 0.0002 |
| 0.8 | 0.2 | 188 | 253.0023 | 95.5541 | 11.1448 | 0.00 | 0.0000 | 67.2780 | 28.8994 | 1.06 | 0.0001 | 25.6560 | 10.5417 | 0.00 | 0.0002 |
| | 0.5 | 176 | 275.5275 | 94.9089 | 11.5306 | 0.00 | 0.0000 | 67.0934 | 27.9306 | 0.00 | 0.0000 | 33.5191 | 11.4920 | 0.00 | 0.0001 |
| | 0.8 | 167 | 296.3949 | 94.1409 | 12.9322 | 0.00 | 0.0000 | 69.3989 | 27.8018 | 0.60 | 0.0001 | 38.4578 | 13.8997 | 0.00 | 0.0001 |
| Total | | 2138 | 142.3643 | 78.7844 | 34.5021 | 13.66 | 0.0000 | 67.1874 | 35.0141 | 10.20 | 0.0000 | 27.5022 | 18.6567 | 17.63 | 0.0002 |

Table 2: $SMALL$ Testbed: Constructive Heuristics for $TF$ and $RDD$.




(a) $TF$ (b) $RDD$

Figure 4: $SMALL$ Tesbed: Tukey 95% confidence intervals for $RDI$ depending on $TF$ and $RDD$

obtained their solutions in less than 1 second) have consumed the time limit. It can be seen that the lowest average $RDI$ is obtained by JPO-10 and JPO-20 (around 3%), whereas the JPF strategies lag behind with a $RDI$ roughly four times higher. Furthermore, JPO-10 and JPO-20 also yield the lowest standard deviation. For the small subset of instances (up to $n = 200$ and $m = 5$) MILP yields the best results, but for bigger instances the solver collapses and is not able to find even good solutions within the time limit. Overall, the worst $RDI$ values are obtained by EDD and OMDD, in line with the results in Section 5.3.

The results for the different values of $TF$ and $RDD$ are shown in Table 4. It can be seen that all methods –except MILP and OMDD– obtain their best results for the case $TF = 0.2$ and $RDD = 0.8$, so it may be considered the easiest case. Figure 5 shows the 95% Tukey confidence intervals for $RDI$ for the different methods (EDD and OMDD results are removed due to their poor performance). The best results regarding the average and standard deviation of $RDI$ are given by JPO-10 and JPO-20 without statistical differences between them.

Table 5 shows the number of instances with their optimal value obtained by the MILP in 600 seconds, and the percentage of these instances that is found by each approach. It can be seen that JPO-10 and JPO-20 are able to find almost all optimal values also found by the MILP (93.7% and 94.5%, respectively).

Figure 6a and Figure 6b show the $RDI$ results for the different values of $TF$ and $RDD$ respectively. Regarding both $TF$ and $RDD$, JPO-10 and JPO-20 are the best methods for all the cases, with statistically significant differences. Note that, for all levels of $TF$ and $RDD$, the matheuristics provide better results than the MILP, except for $RDD = 0.2$, for which JPF30 and JPF40 yield the worst results.

# 6 Conclusions

In this paper we have addressed the order scheduling problem with tardiness objective. This relevant problem is known to be NP-hard and some approximate procedures have been proposed in the literature. However, the best of these approximate procedures –the OMDD heuristic– constructs the solution using a sort of greedy approach that does not take into account the influence of the un-

| | | MILP | | EDD | | OMDD | | FP | | JPF30 | | JPF40 | | JPO-10 | | JPO-20 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $m$ | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. |
| 100 | 5 | 0.4249 | 1.5842 | 75.8836 | 36.2000 | 76.3879 | 32.5870 | 28.1583 | 16.2231 | 9.1539 | 13.0665 | 8.0341 | 12.5516 | 2.5108 | 3.5405 | 1.5980 | 2.2723 |
| | 10 | 0.3663 | 1.2037 | 75.6993 | 34.0286 | 81.1083 | 31.2168 | 28.3408 | 14.4362 | 9.3379 | 10.8317 | 7.1504 | 9.3101 | 2.7241 | 3.0117 | 1.9414 | 2.4127 |
| 150 | 5 | 0.8695 | 2.5765 | 75.2493 | 38.1812 | 73.9006 | 34.6715 | 28.3333 | 17.1303 | 12.9779 | 13.2058 | 10.1073 | 11.7700 | 2.6471 | 3.3405 | 2.4923 | 2.7585 |
| | 10 | 1.8391 | 8.1341 | 73.2243 | 36.2489 | 80.3663 | 32.0992 | 27.1169 | 15.4473 | 15.6852 | 14.1440 | 11.9285 | 13.3002 | 3.4201 | 3.8830 | 4.0158 | 5.0691 |
| 200 | 5 | 3.1597 | 14.7614 | 73.9637 | 37.5205 | 75.6839 | 35.2777 | 26.8274 | 15.2615 | 19.3853 | 18.6541 | 15.9389 | 16.2789 | 4.5527 | 5.9757 | 3.8734 | 5.6741 |
| | 10 | 24.1207 | 41.4226 | 66.9661 | 37.1182 | 72.7739 | 33.7597 | 23.4698 | 13.8574 | 19.3034 | 19.0851 | 14.6996 | 14.9472 | 3.6024 | 6.2140 | 5.5247 | 13.3185 |
| 300 | 5 | 68.8998 | 45.8978 | 49.6267 | 37.6335 | 48.0088 | 34.8463 | 16.7224 | 12.6413 | 18.9055 | 21.2045 | 17.1532 | 20.3951 | 3.8253 | 6.6391 | 2.4195 | 2.9848 |
| | 10 | 99.4073 | 7.4645 | 34.2460 | 28.1289 | 34.6116 | 26.3312 | 10.0189 | 8.9507 | 12.7131 | 13.9664 | 15.8150 | 15.9425 | 1.7083 | 4.3433 | 2.9588 | 4.8990 |
| Total | | 24.8859 | 42.4712 | 65.6074 | 38.4929 | 67.8552 | 36.3086 | 23.6235 | 15.7354 | 14.6828 | 16.3474 | 12.6034 | 15.0571 | 3.1238 | 4.8712 | 3.1030 | 6.0946 |

Table 3: *BIG* Testbed: Average and standard deviation of *RDI* for $m$ and $n$.

| | | MILP | | EDD | | OMDD | | FP | | JPF30 | | JPF40 | | JPO-10 | | JPO-20 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $TF$ | $RDD$ | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. | Average | St. Dv. |
| 0.2 | 0.2 | 12.6406 | 33.1246 | 82.6237 | 27.6662 | 67.4529 | 32.8243 | 22.9393 | 9.7291 | 10.3496 | 11.2853 | 10.1640 | 11.6993 | 1.5049 | 2.4013 | 0.6684 | 1.3895 |
| | 0.5 | 23.3530 | 39.6864 | 26.9079 | 36.0637 | 68.5929 | 44.9437 | 10.5292 | 16.5212 | 6.7103 | 15.0086 | 6.6393 | 15.8835 | 0.2562 | 3.1623 | 0.4369 | 3.2758 |
| | 0.8 | 30.0000 | 45.9696 | 0.0000 | 0.0000 | 2.5000 | 15.6615 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| 0.5 | 0.2 | 15.6313 | 36.4205 | 88.6131 | 18.5731 | 80.2830 | 20.9890 | 27.4939 | 7.9622 | 24.8517 | 14.3148 | 21.0305 | 13.3403 | 7.6264 | 3.6875 | 4.5944 | 3.4811 |
| | 0.5 | 25.9979 | 43.5824 | 72.5766 | 30.7256 | 72.3054 | 30.4154 | 30.7112 | 14.3087 | 6.5545 | 4.7493 | 6.0599 | 5.1843 | 1.6486 | 2.0983 | 2.3301 | 1.9439 |
| | 0.8 | 29.1370 | 42.5805 | 58.4392 | 32.3659 | 73.9778 | 34.7141 | 23.0107 | 16.1770 | 21.3494 | 20.0195 | 19.9492 | 19.0541 | 3.8002 | 7.7332 | 9.3029 | 14.7276 |
| 0.8 | 0.2 | 22.3518 | 41.6165 | 93.4975 | 10.1962 | 84.9061 | 13.0521 | 27.6684 | 6.9662 | 39.4596 | 14.6855 | 31.6327 | 16.4291 | 9.0421 | 5.1675 | 2.7949 | 2.4815 |
| | 0.5 | 30.7497 | 46.1569 | 87.9885 | 14.3257 | 83.1761 | 15.6152 | 35.0778 | 8.2690 | 12.7606 | 5.4162 | 9.8002 | 5.5179 | 1.8998 | 2.2792 | 3.2062 | 2.4591 |
| | 0.8 | 34.1119 | 47.1832 | 79.8197 | 21.0955 | 77.5024 | 21.6615 | 35.1808 | 11.9604 | 10.1093 | 5.9280 | 8.1544 | 5.6345 | 2.3366 | 2.8500 | 4.5930 | 3.4407 |
| Total | | 24.8859 | 42.4712 | 65.6074 | 38.4929 | 67.8552 | 36.3086 | 23.6235 | 15.7354 | 14.6828 | 16.3474 | 12.6034 | 15.0571 | 3.1238 | 4.8712 | 3.1030 | 6.0946 |

Table 4: *BIG* Testbed: Average and standard deviation of *RDI* for $TF$ and $RDD$.

| $m$ | $n$ | MILP | EDD | OMDD | FP | JPF30 | JPF40 | JPO-10 | JPO-20 |
|---|---|---|---|---|---|---|---|---|---|
| 5 | 100 | 57 | 45.61 | 35.09 | 49.12 | 57.89 | 57.89 | 98.25 | 98.25 |
| | 150 | 49 | 65.31 | 48.98 | 65.31 | 67.35 | 67.35 | 85.71 | 89.80 |
| | 200 | 29 | 93.10 | 79.31 | 93.10 | 93.10 | 93.10 | 96.55 | 96.55 |
| | 300 | 10 | 100.00 | 80.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 |
| 10 | 100 | 51 | 39.22 | 35.29 | 43.14 | 49.02 | 49.02 | 90.20 | 90.20 |
| | 150 | 26 | 84.62 | 73.08 | 84.62 | 88.46 | 88.46 | 96.15 | 96.15 |
| | 200 | 16 | 81.25 | 68.75 | 87.50 | 93.75 | 93.75 | 100.00 | 100.00 |
| | 300 | 0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Total | | 238 | 63.03 | 51.68 | 65.13 | 69.75 | 69.75 | 93.70 | 94.54 |

Table 5: *BIG* Testbed: Number of optimum for $MILP$ and % of optimal solutions for the rest of methods for $m$ and $n$.

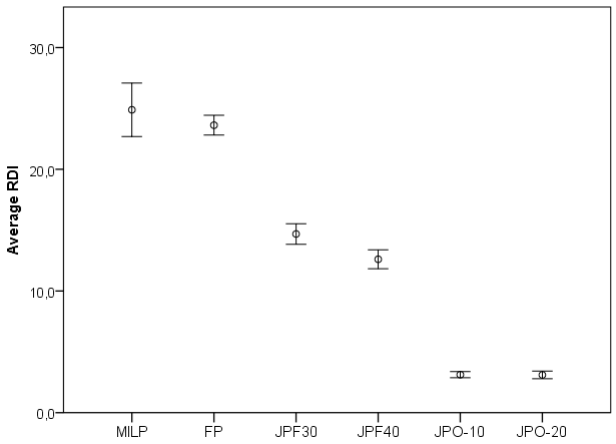
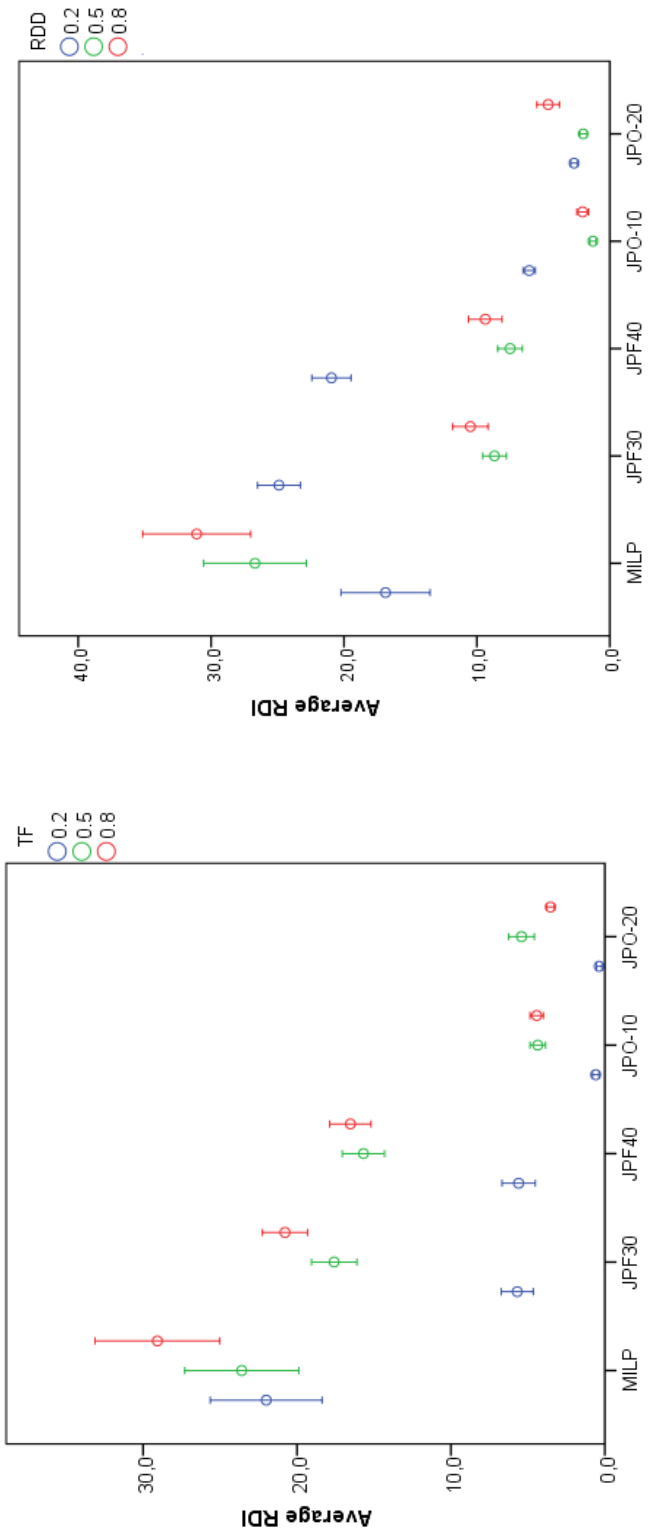



Figure 5: *BIG* Tesbed: Tukey confidence intervals for *RDI*




(a) $TF$ (b) $RDD$

Figure 6: $BIG$ Tesbed: Tukey confidence intervals for $RDI$ depending on $TF$ and $RDD$

scheduled jobs in the objective function. In this paper we propose a new constructive heuristic with a look-ahead mechanism that attempts to estimate this contribution, thus making a more balanced decision. The computational experience carried out shows that our proposal clearly outperforms OMDD and is able to provide good results in negligible CPU time.

For the scenario where bigger decision intervals are allowed, more sophisticated procedures could be used to provide excellent (close-to-optimal) solutions using the available CPU time. Along this line, we propose a matheuristic strategy that is able to find around 94% of the optimal solutions that a MILP solver can find within the same time limit. For the larger instances where the MILP solver cannot obtain good solutions within the given time limit, the proposed strategy provides high-quality solutions, substantially better than those found by other matheuristic strategies using the same CPU time, or by the constructive method presented above. Furthermore, the matheuristic strategy proposed is very robust with respect to its main design parameter.

# Acknowledgements

The authors wish to thank the referees for their comments on the earlier versions of the manuscript. This research has been funded by the Spanish Ministry of Science and Innovation, under grant "PROMISE" with reference DPI2016-80750-P.

# References

Ahmadi, R. and Bagchi, U. (1990). Scheduling of mult-jobs customer orders in multi-machine enviroments. In *ORSA/TIMS, Philadelphia.*

Ahmadi, R., Bagchi, U., and Roemer, T. A. (2005). Coordinated scheduling of customer orders for quick response. *Naval Research Logistics*, 52(6):493–512.

Billaut, J.-C., Della Croce, F., and Grosso, A. (2015). A single machine scheduling problem with two-dimensional vector packing constraints. *European Journal of Operational Research*, 243(1):75–81.

Della Croce, F., Grosso, A., and Salassa, F. (2014a). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, 213(1):67–78.

Della Croce, F., Salassa, F., and T'Kindt, V. (2014b). A hybrid heuristic approach for single machine scheduling with release times. *Computers and Operations Research*, 45:7–11.

Fanjul-Peyro, L., Perea, F., and Ruiz, R. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *European Journal of Operational Research*, 260(2):482–493.

Fernandez-Viagas, V. and Framinan, J. (2015). NEH-based heuristics for the permutation flowshop scheduling problem to minimise total tardiness. *Computers and Operations Research*, 60:27–36.

Framinan, J. and Perez-Gonzalez, P. (2017). New approximate algorithms for the customer order scheduling problem with total completion time objective. *Computers and Operations Research*, 78:181–192.

Gurobi Optimization Inc. (2017). Gurobi optimizer version 7.0.

Karabulut, K. (2016). A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops. *Computers & Industrial Engineering*, 98:300 – 307.

Kim, Y.-D. (1993). Heuristics for flowshop scheduling problems minimizing mean tardiness. *Journal of the Operational Research Society*, 44(1):19–28.

Lee, I. S. (2013). Minimizing total tardiness for the order scheduling problem. *International Journal of Production Economics*, 144(1):128–134.

Leung, J.-T., Li, H., and Pinedo, M. (2005a). *Multidisciplinary Scheduling: Theory and Applications*. Springer.

Leung, J. Y. T., Li, H., and Pinedo, M. (2005b). Order Scheduling in an Environment with Dedicated Resources in Parallel. *Journal of Scheduling*, 8(5):355–386.

Leung, J. Y.-T., Li, H., Pinedo, M., and Zhang, J. (2007). Minimizing total weighted completion time when scheduling orders in a flexible environment with uniform machines. *Information Processing Letters*, 103(3):119–129.

Leung, J. Y.-T., Li, H., and Pinedo, M. A. I. (2006). Scheduling orders for multiple product types with due date related objectives. *European Journal of Operational Research*, 168(2):370–389.

Lin, S.-W. and Ying, K.-C. (2016). Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. *Omega (United Kingdom)*, 64:115–125.

Quang Chieu, T., Gen, W., Billaut, J.-C., and Bouquard, J.-L. (2013). Resolution of the $F2||\sum t_j$ scheduling problem by genetic algorithm and matheuristic. In *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management, IEEE - IESM 2013*.

Roemer, T. and Ahmadi, R. (1997). The complexity of scheduling customer orders. In *INFORMS Conference 1997, Dallas*.

Roemer, T. A. (2006). A note on the complexity of the concurrent open shop problem. *Journal of Scheduling*, 9(4):389–396.

Ta, Q., Billaut, J.-C., and Bouquard, J.-L. (2015). Matheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem. *Journal of Intelligent Manufacturing*.

Wagneur, E. and Sriskandarajah, C. (1993). Openshops with jobs overlap. *European Journal of Operational Research*, 71(3):366–378.

Wang, G. and Cheng, T. (2007). Customer order scheduling to minimize total weighted completion time. *Omega (United Kingdom)*, 35(5):623–626.

Xu, J., Wu, C.-C., Yin, Y., Zhao, C., Chiou, Y.-T., and Lin, W.-C. (2016). An order scheduling problem with position-based learning effect. *Computers and Operations Research*, 74:175–186.