European Journal of Operational Research 261 (2017) 515–529

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

The single machine weighted mean squared deviation problem

Jordi Pereira[a,*], Óscar C. Vásquez[b]

[a] Department of Engineering and Sciences, Universidad Adolfo Ibáñez, Av. Padre Hurtado 750, Office C216, Viña del Mar, Chile
[b] University of Santiago of Chile, Department of Industrial Engineering, Santiago de Chile, Chile

ARTICLE INFO

Article history:
Received 24 May 2016
Accepted 1 March 2017
Available online 7 March 2017

Keywords:
Scheduling
Single machine
JIT
Branch-and-cut
Dominance properties

ABSTRACT

This paper studies a single machine problem related to the Just-In-Time (JIT) production objective in which the goal is to minimize the sum of weighted mean squared deviation of the completion times with respect to a common due date. In order to solve the problem, several structural and dominance properties of the optimal solution are investigated. These properties are then integrated within a branch-and-cut approach to solve a time-indexed formulation of the problem. The results of a computational experiment with the proposed algorithm show that the method is able to optimally solve instances with up to 300 jobs within reduced running times, improving other integer programming approaches.

1. Introduction

In this paper we consider a single-machine weighted mean squared deviation problem (WMSDP) in which the production environment is simplified into a single machine that performs operations on a set $\mathcal{J}$ of n jobs, where each job j has a processing time $p_j \in \mathbb{N}$, a priority weight $w_j \in \mathbb{Q}^+$, and a common due date $d \in \mathbb{N}$. The machine does not allow preemptions and cannot perform multiple operations simultaneously. The objective of the WMSDP is to find a schedule σ of a set $\mathcal{J}$ of jobs on a single machine such that the sum of the weighted mean squared deviations of the completion times with respect to a common due date $\sum_{j\in\mathcal{J}} w_j(C_j - d)^2$ is minimized (omitting the normalization factor), and the completion time C_j of each job $j \in \mathcal{J}$ is greater than or equal to $\sum_{\sigma_i \le \sigma_j} p_i$, where σ_j is the order of job $j \in \mathcal{J}$ in the schedule σ. Consequently, a non-negative starting time for each job is required. According to the three-field notation used in scheduling, see (Graham, Lawler, Lenstra, & Kan, 1979), the problem is denoted by $1||\sum_j w_j(C_j - d)^2$.

The WMSDP considers scheduling operations within a single processing unit, the machine. Single machine settings are frequent in several industries (Jin, Gupta, Song, & Wu, 2010; Wagner, Davis, & Kher, 2002) as well as in some computer systems (Waldspurger & Weihl, 1994). Moreover, complex machine environments are often decomposed into single machine problems in many practical settings, (Pinedo, 2012, p. 35). Measuring the quality of a schedule according to its earliness and tardiness with respect to a due date has been thoroughly studied in the literature, see (Baker & Scudder, 1990) for an early review and Janiak, Janiak, Krysiak, and Kwiatkowski (2015) for a recent review on a generalized version of the problem. The objective assumes the Just-in-time (JIT) production ideals in which goods are manufactured only when needed and both inventories and delays are penalized, see Jozefowska (2007) for a review of JIT problems and objectives in scheduling.

Different weighted common due date problems with linear penalties have been covered in Bagchi, Chang, and Sullivan (1987a); Feldmann and Biskup (2003); Hino, Ronconi, and Mendes (2005); Liao and Cheng (2007); Nearchou (2008); Sourd (2009); Ying (2008) among others. The WMSDP differs from these works in quadratically penalizing deviations from the goal. Such a scheme avoids that few jobs collect most of the deviations and enforces that deviations are shared as evenly as possible among all jobs, see Jozefowska (2007); Vilà and Pereira (2013).

When a quadratic deviation from a common goal is considered, the objective also portrays environments in which variance minimization is sought. The applicability of variance minimization in computer information retrieval systems was highlighted in Merten and Muller (1972). In Kanet (1981) the importance of such an objective in service and manufacturing settings was also illustrated. Following the example given in Merten and Muller (1972), in Kanet (1981) the author describes the applicability of the unweighted version of the WMSDP for information storage in which the variation of time to access different records should be minimized. In such an environment if the access frequency is known, it is natural to formulate the problem using a different weight for each

* Corresponding author.
E-mail addresses: jorge.pereira@uai.cl (J. Pereira), oscar.vasquez@usach.cl (Ó.C. Vásquez).

http://dx.doi.org/10.1016/j.ejor.2017.03.001
0377-2217/© 2017 Elsevier B.V. All rights reserved.

job. Similarly, if deviations for different clients or operations are known, a formulation that takes into account these weights would be preferred.

Contributions of this work. This paper introduces the WMSDP and describes several structural and dominance properties of the problem. These properties differ from those reported in the literature, see Bansal, Dürr, Thang, and Vásquez (2016); Vásquez (2015), as (1) the objective of the WMSDP is a non-monotone quadratic function, and (2) the optimal schedule does not necessarily start processing jobs in time 0. On the one hand, the structural properties allow us to: (1) state that the optimal solution has no idle time; (2) identify lower and upper bounds on the optimal starting time of the schedule and (3) obtain a bound on the difference between the cost of the optimal schedule and the cost of the optimal schedule with an integer starting time. On the other hand, the dominance properties allow us to state the sufficient conditions that must be satisfied for a pair of jobs $i, j \in \mathcal{J}$, such that in any optimal schedule σ, job i is always scheduled before job j. The dominance properties are integrated as valid inequalities within a branch-and-cut approach to solve a time-indexed integer programming formulation of the problem, which assumes that the starting times of the jobs are integers. In addition to the valid inequalities, the branch-and-cut procedure is strengthened by the use of (1) a variable prefixing rule; (2) two constructive heuristics to find fast, feasible solutions, and (3) an especially tailored branching scheme that makes use of the special characteristics of the problem.

Related work. The WMSDP is an extension of the mean squared deviation problem (MSDP), in which jobs are given different weights that depict their relative importance.

In the MSDP, the objective function evaluates the difference between the completion time of each job and the common due date. Consequently, the optimal schedule contains no idle time and it is possible to establish a relationship between the due date and the starting time. If the due date is sufficiently small, the starting time of the optimal schedule is 0, but if the due date is sufficiently large, the starting time of the optimal schedule is positive. In such a case, increasing the due date by delta time units causes the schedule to be the same but the starting time is increased by the same amount of time units.

Formally speaking, the literature considers three different cases according to the relationship between the mean completion time of the schedules and the due date of the instance: (1) *tightly restricted* instances in which the mean completion time of any schedule is not greater than the due date, and thus the starting time of the optimal schedule is equal to 0; (2) *unrestricted* instances in which the mean completion time of any schedule is greater than the due date, and thus the starting time of the optimal schedule is positive, and (3) *restricted* instances in which the mean completion time of any schedule may be smaller or greater than the due date, and thus the starting time of the optimal schedule is unknown.

Several dynamic programming (DP) algorithms have been suggested for the different versions of the MSDP Ventura and Weng (1995); Weng and Ventura (1996). In a unified problem context, Mondal (2002) proposes a DP algorithm with complexity $O(n^2(d + \max p_j))$, which was improved to $O(nd)$ by Srirangacharyulu and Srinivasan (2013).

The *unrestricted* MSDP is equivalent to the completion time variance (CTV) problem, in which the weights of the jobs are equal and the common due date is a variable of the problem defined by $d := \sum_{j\in J} C_j/|\mathcal{J}|$, see Bagchi, Sullivan, and Chang (1987b). The equivalence transfers the NP-hardness proof from Kubiak (1993) to the MSDP, and consequently to the WMSDP.

For the weighted completion time variance (WCTV) problem, a problem formulation in which the weight of the jobs is arbitrary and the common due date is a variable of the problem defined by $d := \sum_{j\in J} w_j C_j / \sum_{j\in J} w_j$, the question about whether the problem is strongly NP-hard remains open. For the WCTV problem, Nessah and Chu (2010) propose a heuristic and a lower bound based on job splitting. Approximation algorithms have been proposed for the WCTV problem with “agreeable weights” (AWCTV), in which $p_i < p_j$ implies $w_i > w_j$. Cai (1995) proposed a DP algorithm with complexity $O(n \sum_{j\in J} w_j \sum_{j\in J} p_j)$, as well as two fully polynomial time approximation schemes (FPTAS) in which the weights are bounded by a polynomial in n. Woeginger (1999) improves the above result without imposing any restrictions on the weights of the jobs. Cheng and Kubiak (2005) reformulate the AWCTV problem as a pseudo-Boolean quadratic function called *Half Product*, see Badics and Boros (1998), and propose an FPTAS with complexity $O(\log(\max\{\max_{j\in J} p_j, \max_{j\in J} w_j, n\}) n^4/\epsilon)$.

Dominance properties have been shown to improve the performance of exhaustive search procedures by early pruning ineffective partial solutions in problems with a monotone increasing function, Vásquez (2015). Such a function ensures that the optimal schedule can be shifted to the left in such a way that the starting time is known to be 0, and there is no idle time between any two consecutive jobs in the optimal schedule. For instance, Vásquez (2015) studied some structural and dominance properties of the airplane refueling problem, which can be seen as a special single machine scheduling problem. These dominance properties are incorporated into an A* algorithm in order to find the exact solution of the problem, leading to reductions in the number of nodes of the search tree of up to three orders of magnitude. Bansal et al. (2016) investigated some structural and dominance properties of the weighted completion time problems with non-linear objective functions, problem denoted by $1||\sum_j w_j C_j^{\beta}, \beta > 0$ according to Graham et al. (1979), generalizing the previous known results. The impact of these properties is also evaluated incorporating them into an A* algorithm, with reductions of up to four orders of magnitude in the number of generated nodes.

Mixed integer linear programming (MILP) formulations for single machine scheduling problems have been widely considered in the literature. These formulations can be classified according to the decision variables used in the literature, see (Queyranne & Schulz, 1994) for a description, or the more recent work of Keha, Khowala, and Fowler (2009) for a comparison of results. While the proposed structural and dominance properties could be included in different formulations, e.g. when the variables represent a partial ordering among jobs (Dyer & Wolsey, 1990), in this work we choose the time-indexed formulation. In a time indexed formulation a binary variable is associated to each job and its possible starting time. Although the number of variables is larger than in other formulations, it is preferred because: (1) it is known to be effective for several single machine scheduling problems (Bigras, Gamache, & Savard, 2008); and (2) it pre-calculates the quadratic terms of the objective function, leading to an integer linear programming model, which is more amenable to standard off-the-shelf solution methods than the corresponding non-linear model.

Outline of the paper. The remainder of the paper is structured as follows: Section 2 considers different properties that any optimal solution has to fulfill, Section 3 provides a time-indexed integer programming (IP) formulation of the WMSDP, and Section 4 describes a branch-and-cut approach to solve the time-indexed formulation, in which the properties described in Section 2 are used to derive valid inequalities. The results of a computational experiment with the branch-and-cut approach are analyzed in Section 5. Finally, some conclusions and future work are discussed in Section 6.

2. Structural and dominance properties

In this section, we set forth the main theoretical contributions of this work. Section 2.1 studies several structural properties of the problem. These properties provide insights into the problem and will be used in the proposed solution procedure as well as to assess the inaccuracy introduced by the resolution of a formulation that only considers integer starting times, see Section 3. Section 2.2 analyzes possible precedence relations among jobs, which will then be used to obtain valid inequalities within the solution procedure.

2.1. Structural properties

Property 1. *The optimal schedule σ^* does not contain idle time.*

Proof. The proof is by contradiction. Suppose an optimal schedule σ^* which contains idle time between two consecutive jobs j_1 and j_2. Let s_k be the starting time and t_k be the execution time of a job j_k. By case assumption, there is idle time between two consecutive jobs j_1 and j_2 of σ (i.e. $s_1 + t_1 < s_2$). Note that the objective function is strictly decreasing for $s_k \in [0, d - t_k]$ and strictly increasing for $s_k \in [d, \infty)$. Thus, the starting times of one of the two consecutive jobs can be delayed or advanced to reduce the idle time and to decrease the value of the objective function. This fact contradicts the optimality of σ^*. □

Given that the optimal solution contains no idle time, the best objective value $F(\sigma, s_\sigma)$ of a given schedule σ with starting time s_σ corresponds to Eq. (1).

$$F(\sigma, s_\sigma) = \sum_{j\in\mathcal{J}} w_j \left(s_\sigma + \sum_{\sigma_i \le \sigma_j} p_i - d \right)^2, \tag{1}$$

where σ_j denotes the order of job $j \in \mathcal{J}$ in schedule σ.

Property 2. *For any given schedule σ, its optimal starting time s^*_σ is*

$$\max\left\{0, \sum_{j\in\mathcal{J}} w_j \left(d - \sum_{\sigma_i \le \sigma_j} p_i \right) \Big/ \sum_{j\in\mathcal{J}} w_j \right\}. \tag{2}$$

Proof. Eq. (1) is derivable, continuous and convex in s_σ. Consequently, s^*_σ corresponds to equaling the first derivative of (1) to 0, or it is equal to 0 if this value is negative. Convexity also guarantees an optimal value even when only integer values for s_σ are considered. □

Following Property 2, WMSDP instances can be classified as in the MSDP: *tightly restricted* instances in which the weighted mean completion time of any schedule is not greater than the due date, and thus the starting time of the optimal schedule is equal to 0; *unrestricted* instances in which the weighted mean completion time of any schedule is greater than the due date, and thus the starting time of the optimal schedule is positive, and *restricted* instances in which the weighted mean completion time of any schedule may be smaller or greater than the due date, and thus the starting time of the optimal schedule is unknown.

Let $C(\sigma^{WSPT})$ and $C(\sigma^{WLPT})$ be the total cost according to (1) if jobs of a given instance are processed starting from time zero following the weighted shortest processing time WSPT (jobs are sorted by non-decreasing p_j/w_j ratio), and the weighted longest processing time WLPT (jobs sorted by non-decreasing w_j/p_j ratio) respectively. Note that these schedules yield the minimum and maximum weighted mean completion time, i.e $\min_\sigma \sum_{j\in\mathcal{J}} w_j \sum_{\sigma_i \le \sigma_j} p_i = C(\sigma^{WSPT})$ and $\max_\sigma \sum_{j\in\mathcal{J}} w_j \sum_{\sigma_i \le \sigma_j} p_j = C(\sigma^{WLPT})$.

Property 3. *For the* unrestricted *WMSDP, any optimal schedule σ^* will have starting time $s_{\sigma^*} \in [d - C(\sigma^{WLPT})/\sum_{j\in\mathcal{J}} w_j, d - C(\sigma^{WSPT})/\sum_{j\in\mathcal{J}} w_j]$.*

Proof. Proof follows from the $C(\sigma^{WSPT})$ and $C(\sigma^{WLPT})$ values, and Property 2. □

Property 4. *For the* unrestricted *WMSDP, any optimal schedule σ^* will have starting time $s_{\sigma^*} \in [d - \sum_{j\in\mathcal{J}} p_j/2 - \max_{j\in\mathcal{J}} p_j, d - \sum_{j\in\mathcal{J}} p_j/2]$.*

Proof. The proof is by contradiction. Suppose that an optimal schedule σ^* exists with a starting time $s_{\sigma^*} < d - \sum_{j\in\mathcal{J}} p_j/2 - \max_{j\in\mathcal{J}} p_j$. Then sequencing the first job in σ^* after completing all other jobs in the schedule (that is, sequencing the job as the last job while the remaining jobs are unaffected) decreases the objective function. This fact contradicts the optimality of σ^*. For $s_{\sigma^*} > d - \sum_{j\in\mathcal{J}} p_j/2$, the argument is symmetric. □

Properties 3 and 4 allow us to classify instances as *unrestricted*, *restricted* or *tightly restricted*. If $d > C(\sigma^{WLPT})/\sum_{j\in\mathcal{J}} w_j$ and $d > \sum_{j\in\mathcal{J}} p_j/2 - \max_{j\in\mathcal{J}} p_j$, the instance is *unrestricted*. If $d \le \max\{C(\sigma^{WSPT})/\sum_{j\in\mathcal{J}} w_j, \sum_{j\in\mathcal{J}} p_j/2\}$, the instance is *tightly restricted*. In any other case, the instance is *restricted*.

Property 5. *Consider two* unrestricted *instances that differ only in their respective due dates d, d'. Then, the optimal sequence of jobs for both optimal schedules σ^* and σ'^* are identical and their starting times s_{σ^*}, $s'_{\sigma'^*}$ will differ in $d - d'$ time units.*

Proof. For any *unrestricted* instance and optimal schedule σ, a change of $d - d'$ time units in the due date modifies the optimal starting time $d - d'$ time units, see Eq. (1), without altering the sequence nor the value of the objective function. Hence, while the instance remains *unrestricted* to the due date, the sequence of the optimal schedule and its objective function remains unaltered. □

Property 5 combined with Properties 3 and 4 will be later used to reduce the number of variables required to solve *unrestricted* instances using the time-indexed formulation.

We now proceed to establish the relationship between the WCTV problem and the *unrestricted* WMSDP.

Property 6. *The weighted completion time variance (WCTV) problem is equivalent to the* unrestricted *WMSDP.*

Proof. First, we claim that for any given schedule σ and starting time s_σ, the due date d^* that minimizes Eq. (1) is equal to the weighted mean completion time $\sum_{j\in\mathcal{J}} w_j(s_\sigma + \sum_{\sigma_i \le \sigma_j} p_i)/\sum_{j\in\mathcal{J}} w_j$. To prove the claim, consider Eq. (1) with a given schedule and starting time, and a variable due date, d^*. Eq. (1) is derivable, continuous and convex in d. Consequently, d^* corresponds to the weighted mean completion time of the schedule by equaling the first derivative of (1) to 0.

Substitute d in (1) for $\sum_{j\in\mathcal{J}} w_j(s_\sigma + \sum_{\sigma_i \le \sigma_j} p_i)/\sum_{j\in\mathcal{J}} w_j$, which is the weighed mean completion time of σ, to obtain the objective function of the WCTV. Notice that the optimal schedule σ^* for the WCTV problem is also optimal for the *unrestricted* instance with a due date equal to

$$\sum_{j\in\mathcal{J}} w_j \left(s^*_\sigma + \sum_{\sigma^*_i \le \sigma^*_j} p_i \right) \Bigg/ \sum_{j\in\mathcal{J}} w_j,$$

see claim above. Consequently, σ^* is also optimal for an arbitrary *unrestricted* instance, see Property 5, and both problems are equivalent. □

To conclude this Section, we investigate the deviation introduced if only integer-valued starting times for the schedule are considered. Let $\pi^{\mathbb{N}*}$ be the optimal schedule of jobs with an integer starting time $s_{\pi^{\mathbb{N}*}}$. Please note that the sequences of jobs for $\pi^{\mathbb{N}*}$ and σ^* are not necessarily identical.

Property 7.

$$F(\pi^{\mathbb{N}*}, s_{\pi^{\mathbb{N}*}}) - F(\sigma^*, s_{\sigma^*}) < 2\sum_{j\in\mathcal{J}} w_j. \quad (3)$$

Proof. Let Σ be the set of schedule σ with an optimal starting time s^*_σ. We denote by $\sigma^{\mathbb{N}}$ the optimal schedule of jobs with integer starting times and an identical sequence of jobs found in σ; and by $s_{\sigma^{\mathbb{N}}}$ the integer starting time of jobs in $\sigma^{\mathbb{N}}$. Consequently, we have:

$$\begin{aligned}
&F(\pi^{\mathbb{N}*}, s_{\pi^{\mathbb{N}*}}) - F(\sigma^*, s_{\sigma^*})\\
\le& F(\sigma^{*\mathbb{N}}, s_{\sigma^{*\mathbb{N}}}) - F(\sigma^*, s_{\sigma^*}) \quad (4)\\
\le& \max_{\{\sigma\}\in\Sigma} F(\sigma^{\mathbb{N}}, s_{\sigma^{\mathbb{N}}}) - F(\sigma, s_{\sigma^*}) \quad (5)\\
=& \max_{\{\sigma\}\in\Sigma} \sum_{j\in\mathcal{J}} w_j\left((s_{\sigma^{\mathbb{N}}} - s_{\sigma^*})\left(s_{\sigma^{\mathbb{N}}} + s_{\sigma^*} + 2\sum_{\sigma_i\le\sigma_j} p_i - 2d\right)\right)\\
<& \max_{\{\sigma\}\in\Sigma} \sum_{j\in\mathcal{J}} w_j\left(2 + 2\sum_{k\in\mathcal{J}} w_k\left(d - \sum_{\sigma_\ell\le\sigma_k} p_\ell\right)\Big/\sum_{m\in\mathcal{J}} w_m + 2\sum_{\sigma_i\le\sigma_j} p_i - 2d\right)\\
=& \max_{\{\sigma\}\in\Sigma} 2\sum_{j\in\mathcal{J}} w_j + 2\sum_{k\in\mathcal{J}} w_k\left(d - \sum_{\sigma_\ell\le\sigma_k} p_\ell\right)\\
&+ 2\sum_{j\in\mathcal{J}} w_j \sum_{\sigma_i\le\sigma_j} p_i - 2d\sum_{j\in\mathcal{J}} w_j\\
=& 2\sum_j w_j. \quad (6)
\end{aligned}$$

Inequality (4) follows from the optimality of schedule $\pi^{\mathbb{N}*}$. Inequality (5) holds for identical sequences of jobs in $\sigma^{*\mathbb{N}}$ and σ^*. Now, we claim $|s_{\sigma^{\mathbb{N}}} - s_{\sigma^*}| < 1$ by convexity of d in WMSDP and hence we also have $s_{\sigma^*} + s_{\sigma^{\mathbb{N}}} < 2 + 2s_{\sigma^*}$. If we replace the terms, we have (6) and this concludes the proof. □

This bound will be used later in order to evaluate the optimality gap induced by the time-indexed formulation introduced in Section 3.

2.2. Dominance properties

Formally, we differentiate two kinds of dominance properties.

- We say that jobs i, j satisfy a *local dominance property* at time t, denoted by $i{\prec_{\ell(t)}}j$, if whenever in a schedule σ job j starts at time t and is followed immediately by job i, σ is not optimal. We use the notation $i{\prec_{\ell[a,\,b]}}j$ as a shorthand for i $\prec_{\ell(t)}j$ for all $t \in [a, b]$.
- We say that jobs i, j satisfy a *global dominance property* in the time interval $[a, b]$, denoted by i $\prec_{g[a,\,b]}j$, if whenever in a schedule σ we have $a \le C_j - p_j \le C_i - p_i - p_j \le b$, σ is sub-optimal, regardless of i, j being adjacent or not.

Now consider the function $f: t \mapsto (t-d)^2$ and analyze the effect of swapping adjacent jobs i, j when both jobs i and j are completed before or after the due date d. We define the following function on $0 \le t$.

$$\phi_{ij}(t) := \frac{f(t+p_i+p_j) - f(t+p_j)}{f(t+p_i+p_j) - f(t+p_i)}.$$

The above function ϕ_{ij} allows us to algebraically analyze the local dominance property between jobs i, j. Let A, B be two job subsequences, and let t be the sum of processing times of jobs in A plus the starting time s_σ.

For $t \le d - p_i - p_j$, we have

$$\begin{aligned}
&F(AjiB, s_\sigma) > F(AijB, s_\sigma)\\
\Leftrightarrow\ & w_j f(t+p_j) + w_i f(t+p_i+p_j) > w_i f(t+p_i) + w_j f(t+p_i+p_j)\\
\Leftrightarrow\ & w_j(f(t+p_j) - f(t+p_i+p_j)) > w_i(f(t+p_i) - f(t+p_i+p_j))\\
\Leftrightarrow\ & \phi_{ij}(t) > \frac{w_i}{w_j},
\end{aligned}$$

and therefore

$$i \prec_{\ell(t)} j \Leftrightarrow: \phi_{ij}(t) > \frac{w_i}{w_j} \quad t \in [0, d-p_i-p_j]. \quad (7)$$

Symmetrically, for $t \ge d$ we have

$$i \prec_{\ell(t)} j \Leftrightarrow: \phi_{ij}(t) < \frac{w_i}{w_j} \quad t \in [d, \infty). \quad (8)$$

We now proceed to systematically analyze the properties of the function $\phi_{ij}(t)$. Our analysis considers the variable t in two intervals: $t \ge d$ and $d - p_i - p_j \ge t \ge 0$, where function f is known to be increasing or decreasing, respectively. For $t \ge d$, we adopt the dominance properties defined by Bansal et al. (2016) for monotone increasing objective functions. For $t \ge d - p_i - p_j$, we prove new local and global dominance properties for monotone decreasing objective functions. Note that $i{\prec_{g[a,\,b]}}j$ trivially implies $i{\prec_{\ell[a,\,b]}}j$, but the reverse is not always true, see Bansal et al. (2016) for a precise characterization of global dominance properties by local dominance properties in monotone increasing monomial functions, including an example where the implication $i{\prec_{\ell[a,\,b]}}j \Rightarrow i{\prec_{g[a,\,b]}}j$ does not hold.

For $t \ge d$, we have the following properties:

Lemma 1 (Lemma 5 and Lemma 6 in Bansal et al. (2016))**.** *If $p_i \neq p_j$ and $t \ge d$, then $\phi_{ij}(t)$ is strictly monotone. In particular:*

- *If $p_i > p_j$, then $\phi_{ij}(t)$ is strictly increasing, concave and $\phi_{ij}(t) \in [\phi_{ij}(0), \frac{p_i}{p_j})$.*
- *If $p_i < p_j$, then $\phi_{ij}(t)$ is strictly decreasing, convex and $\phi_{ij}(t) \in (\frac{p_i}{p_j}, \phi_{ij}(0)]$*

See Fig. 1 for an illustration of the properties claimed for $t \ge d$.

Property 8 (from Theorem 1 in Bansal et al. (2016))**.** *Let a be a time value greater than or equal to d. The implication $i{\prec_{\ell[a,\,\infty)}}j \Rightarrow i{\prec_{g[a,\,\infty)}}j$ holds when $p_i \le p_j$.*

Property 9 (from Theorem 2 in Bansal et al. (2016))**.** *The implication $i{\prec_{\ell[d,\,\infty)}}j \Rightarrow i{\prec_{g[d,\,\infty)}}j$ holds.*

Fig. 2 shows the global dominance properties for $t \ge d$.

We obtain similar properties for $t \le d - p_i - p_j$. In this case, the function $f: t \mapsto (t-d)^2$ is decreasing.

Lemma 2. *If $p_i \neq p_j$ and $t \le d - p_i - p_j$, then $\phi_{ij}(t)$ is strictly monotone and bounded, in particular:*

- *If $p_i > p_j$, then $\phi_{ij}(t)$ is strictly increasing, convex and $\phi_{ij}(t) \in [\phi_{ij}(0), (\frac{p_i}{p_j})^2]$.*
- *If $p_i < p_j$, then $\phi_{ij}(t)$ is strictly decreasing, concave and $\phi_{ij}(t) \in [(\frac{p_i}{p_j})^2, \phi_{ij}(0)]$.*

Proof. To prove the monotonicity of $\phi_{ij}(t)$, we use its first and second derivatives

$$\phi'_{ij}(t) = 2\frac{p_i}{p_j}\frac{p_i - p_j}{(p_j + 2(t + p_i - d))^2}$$

$$\phi''_{ij}(t) = -8\frac{p_i}{p_j}\frac{p_i - p_j}{(p_j + 2(t + p_i - d))^3}.$$

Clearly, the function $\phi_{ij}(t)$ is strictly increasing and convex when $p_i > p_j$, and strictly decreasing and concave when $p_i < p_j$.

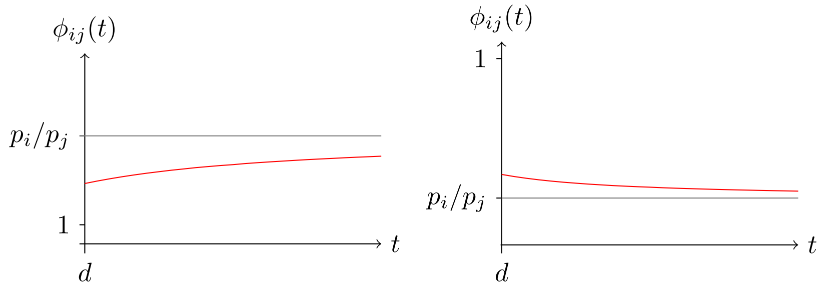
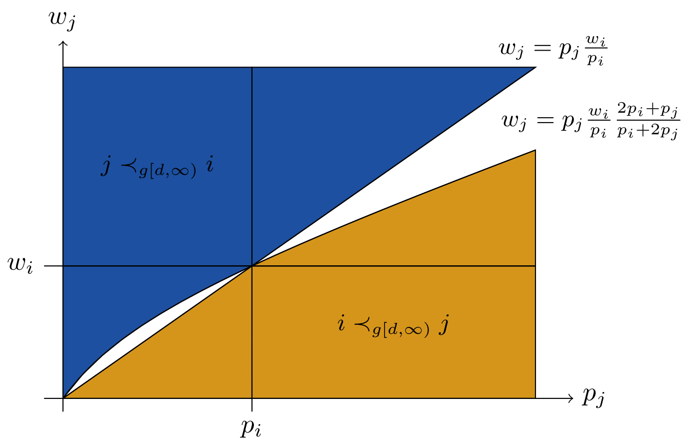


Fig. 1. Function $\phi_{ij}(t)$ for $t \geq d$ in case $p_i > p_j$ (left) and case $p_i < p_j$ (right).



w_j
$w_j = p_j \frac{w_i}{p_i}$
$w_j = p_j \frac{w_i}{p_i} \frac{2p_i+p_j}{p_i+2p_j}$
$j \prec_{g[d,\infty)} i$
w_i
$i \prec_{g[d,\infty)} j$
p_j
p_i

Fig. 2. Global dominance properties for $t \geq d$. In this comparability map, a job is represented by a point in $\mathbb{R}$ where the abscissa corresponds to its processing time and the ordinate to its weight.

To bound the function $\phi_{ij}(t)$, we analyze its codomain. Since $\phi_{ij}(t)$ is strictly monotone, it suffices to evaluate its extreme points. At $t = d - p_i - p_j$, we have $\phi_{ij}(d - p_i - p_j) = (\frac{p_i}{p_j})^2$, and at $t = 0$, we have $\phi_{ij}(0) = \frac{p_i}{p_j}(1 - \frac{p_i - p_j}{p_j + 2(p_i - d)})$. □

Fig. 3 provides a description of the properties for $t \leq d - p_i - p_j$.

Property 10. *Let a be a time value smaller than or equal to $d - p_i - p_j$. The implication $i \prec_{\ell[0,\,a]} j \Rightarrow i \prec_{g[0,\,a]} j$ holds when $p_i \leq p_j$.*

Proof. This proof holds for any decreasing penalty function f and follows the idea of the proof from Bansal et al. (2016). For a given starting time $s_\sigma \geq 0$, we consider that the sum of processing times of jobs in A, B plus the starting time s_σ is smaller than or equal to $b \leq d - p_i - p_j$. First, if $F(AjBi, s_\sigma) \geq F(AjiB, s_\sigma)$, then by $i \prec_{\ell[0,\,b]} j$ we have $F(AjBi, s_\sigma) > F(AijB, s_\sigma)$. For the remaining case we assume that $F(AjBi, s_\sigma) < F(AjiB, s_\sigma)$ and show that $F(AjBi, s_\sigma) > F(AijB, s_\sigma)$. For $i \prec_{\ell[0,\,b]} j$, it suffices to show an even stronger inequality

$$F(AijB, s_\sigma) - F(AjiB, s_\sigma) > F(AiBj, s_\sigma) - F(AjBi, s_\sigma)$$

or equivalently

$$F(AjBi, s_\sigma) - F(AjiB, s_\sigma) > F(AiBj, s_\sigma) - F(AijB, s_\sigma).$$

The right-hand side is negative by assumption, so it is sufficient to show

$$F(AjBi, s_\sigma) - F(AjiB, s_\sigma) > \min_{t \in [0,b]} \phi_{ij}(t)(F(AiBj, s_\sigma) - F(AijB, s_\sigma)) \tag{9}$$

since $\phi_{ij}(t) \leq 1$ by $p_i \leq p_j$.

We introduce the following notation. Denote the jobs in B by $1, \ldots, k$ for some integer k. For some job $1 \leq h \leq k$, let l_h denote the total processing time of jobs i, j and all jobs $h \in B$ plus the starting time s_σ. By definition of ϕ_{ij}, we have

$$f(p_i + p_j + l_h) - f(p_j + l_h) = \phi_{ij}(l_h)(f(p_i + p_j + l_h) - f(p_i + l_h)),$$

which implies

$$w_h(f(p_j + l_h) - f(p_i + p_j + l_h)) \\ \geq \min_t \phi_{ij}(l_h) w_h(f(p_i + l_h) - f(p_i + p_j + l_h)) \tag{10}$$

by considering that f is a decreasing function. To analyze the contribution of jobs i, j, we observe that according to $i \prec_{\ell[0,\,b]} j$ we have

$$\phi_{ij}(t) w_j \geq w_i \qquad \forall t \in [0, b],$$

which implies

$$w_i(f(a + p_i + p_j + l_h) - f(a + p_j + p_i)) \\ \geq w_j \min_{t \in [0,b]} \phi_{ij}(t)(f(a + p_i + p_j + l_h) - f(a + p_i + p_j)) \tag{11}$$

by considering that f is a decreasing function.

Summing up (10) for every $1 \leq h \leq k$ and (11) yields (9) as required, which completes the proof. □

Property 11. *The implication $i \prec_{\ell[0,d-p_i-p_j]} j \Leftrightarrow i \prec_{g[0,d-p_i-p_j]} j$ holds when $p_i \geq p_j$ and $w_j > w_i$.*

Proof. The proof holds for any decreasing penalty function f. For a given starting time $s_\sigma \geq 0$, we consider that the sum of processing times of jobs in A, B plus the starting time s_σ is smaller than or equal to $d - p_i - p_j$. For convenience, we define the jobs in B as 1, ..., k for some integer k. For some job $1 \leq h \leq k$ we denote by l_h the total processing time of all jobs $h \in B$ plus the starting time s_σ, $\Delta(y) = \sum_{h \in B} w_h f(l_h + y)$, and $\sum_{k \in B} p_k = K$.

We show

$$F(AiBj, s_\sigma) < F(AjBi, s_\sigma) \\ \equiv w_i f(t + p_i) + \Delta(p_i) + w_j f(t + p_i + p_j + K) < w_j f(t + p_j) \\ + \Delta(p_j) + w_i f(t + p_i + p_j + K) \\ \equiv w_i[f(t + p_i) - f(t + p_i + p_j + K)] - w_j[f(t + p_j) \\ - f(t + p_i + p_j + K)] < \Delta(p_j) - \Delta(p_i).$$

The right side is positive since f is decreasing and $p_i \geq p_i$. Hence, it suffices to show

$$w_i[f(t + p_i) - f(t + p_i + p_j + K)] - w_j[f(t + p_j) \\ - f(t + p_i + p_j + K)] < 0,$$

which holds for any $t \leq d - p_i - p_j - K$ by the decreasing penalty function $w_j > w_i$ and $p_i \geq p_j$. □

Fig. 4 shows the global dominance properties for $t \leq d - p_i - p_j$.

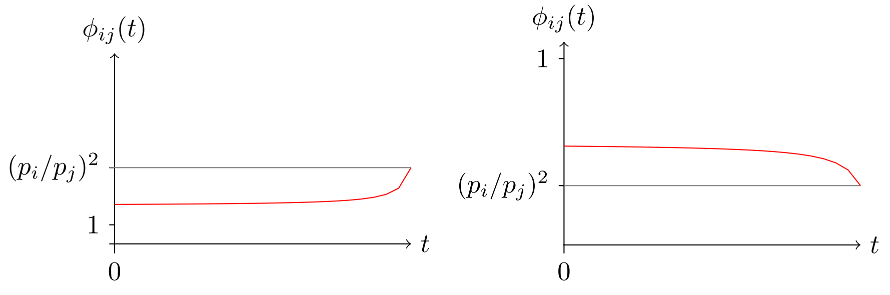
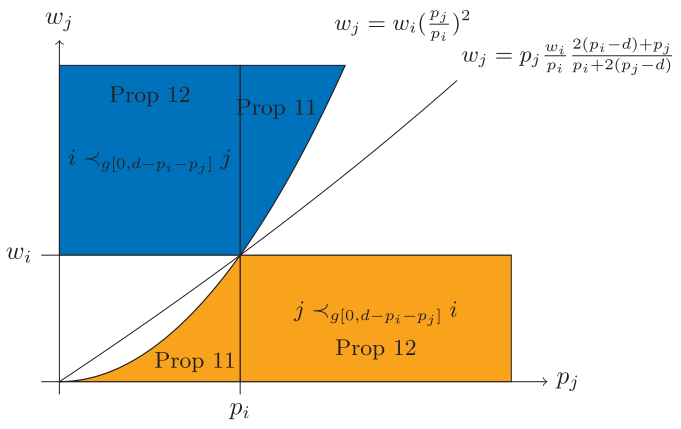


Fig. 3. Function $\phi_{ij}(t)$ for $t \le d - p_i - p_j$ in case $p_i > p_j$ (left) and case $p_i < p_j$ (right).



w_j
$w_j = w_i(\frac{p_j}{p_i})^2$
$w_j = p_j \frac{w_i}{p_i} \frac{2(p_i-d)+p_j}{p_i+2(p_j-d)}$
Prop 12
Prop 11
$i \prec_{g[0,d-p_i-p_j]} j$
w_i
$j \prec_{g[0,d-p_i-p_j]} i$
Prop 12
Prop 11
p_j
p_i

Fig. 4. Global dominance properties for $t \le d - p_i - p_j$. In this comparability map, a job is represented by a point in $\mathbb{R}$ where the abscissa corresponds to its processing time and the ordinate to its weight.

Lemma 3. *Fix two jobs i and j. If there exists $t^* \in \mathbb{R}^+$ such that*

$$\frac{w_i}{w_j} = \phi_{ij}(t^*),$$

then t^* is unique and equal to

$$t^* = \frac{w_j p_i(p_i + 2(p_j - d)) - w_i p_j(p_j + 2(p_i - d))}{2(w_i p_j - w_j p_i)}. \tag{12}$$

Proof. We assume that there exists $t^* \in \mathbb{R}^+$ such that $w_i/w_j = \phi_{ij}(t^*)$. The strict monotonicity of $\phi_{ij}(t)$ and the interval of the function, Lemmas 1 and 2, imply uniqueness of t^*. We have

$$\frac{w_i}{w_j} = \phi_{ij}(t^*) = \frac{(t^* + p_i + p_j - d)^2 - (t^* + p_j - d)^2}{(t^* + p_i + p_j - d)^2 - (t^* + p_i - d)^2}$$

and arranging the above expression, we obtain

$$t^* = \frac{w_j p_i(p_i + 2(p_j - d)) - w_i p_j(p_j + 2(p_i - d))}{2(w_i p_j - w_j p_i)}.$$

□

Based on the previous analysis, the dominance properties of the WMSDP can be summarized as follows.

Corollary 1 (the dominance properties)**.** *Let i, j be two jobs, with $p_i > p_j$.*
If $t \ge d$

if $p_i/p_j < w_i/w_j$	then $i \prec_{g[d,\infty)} j$	by Lemma 1 and Property 9
if $w_i/w_j < \phi_{ij}(d)$	then $j \prec_{g[d,\infty)} i$	by Lemma 1 and Property 9
else $\exists t^*$:	$i \prec_{\ell[d,t^*)} j$	
	and $j \prec_{g[t^*,\infty)} i$	by Lemma 1 and Property 8

if $t \le d - p_i - p_j$

if $(p_i/p_j)^2 \le w_i/w_j$	then $j \prec_{g[0,d-p_i-p_j]} i$	by Lemma 2 and Property 10
if $w_i/w_j \le 1$	then $i \prec_{g[0,d-p_i-p_j]} j$	by Lemma 2 and Property 11
if $w_i/w_j \le \phi_{ij}(0)$	then $i \prec_{\ell[0,d-p_i-p_j]} j$	by Lemma 2
else $\exists t^*$:	$j \prec_{g[0,t^*]} i$	
	and $i \prec_{\ell(t^*,d-p_i-p_j]} j$	by Lemma 2 and Property 10.

We remark that for $t \in (d - p_i - p_j, d)$ the local-global dominance properties are not characterized, as the completion times of both jobs can be in both the decreasing and increasing intervals of the function f. Also note that the dominance properties in the increasing interval allow us to compare a larger number of job pairs i, j than in the decreasing interval. Consequently, the number of job sequences satisfying all dominance properties in the increasing interval is smaller than in the decreasing interval.

3. Mathematical formulation

Time-indexed formulations discretize the planning horizon T into a number of periods where period t starts at time t and ends at time $t + 1$. The formulation uses binary variables x_{jt} for each job and period that take value equal to 1 if job $j \in \mathcal{J}$ starts being processed at time $t = 0, \ldots, T - 1$, and 0 otherwise.

In order to ascertain T, a bound on the maximum completion time of all of the jobs $\mathcal{J}$ is required. For the WMSDP the bound is obtained from the properties derived in Section 2.1 according to the instance class (be it tightly restricted, restricted or unrestricted).

For *tightly restricted* instances, the starting time of the optimal schedule is 0, and thus its horizon is $T = \sum_{j \in \mathcal{J}} p_j$. For *unrestricted* instances, the due date is shifted using Properties 3 and 5, (i.e., the equivalent due date d is the minimum value such that $d - C(\sigma^{WLPT}) / \sum_{j \in \mathcal{J}} w_j > 0$. Then, the optimal starting time is no later than $d - C(\sigma^{WSPT}) / \sum_{j \in \mathcal{J}} w_j$ and, therefore, $T = \sum_{j \in \mathcal{J}} p_j + \lceil d - C(\sigma^{WSPT}) / \sum_{j \in \mathcal{J}} w_j \rceil$ w.l.o.g..

For *restricted* instances, note that the maximum starting time from the *unrestricted* instances holds (the minimum starting time is 0 rather than a positive value). Consequently, the horizon can be set to $T = \sum_{j \in \mathcal{J}} p_j + \lceil d - C(\sigma^{WSPT}) / \sum_{j \in \mathcal{J}} w_j \rceil$ w.l.o.g..

For any pair of integers i, j: $i \le j$, denote the set $\{i, i+1, \ldots, j\}$ by $[i{:}\ j]$. Let x_{jt}, $j \in \mathcal{J}$, $t \in [0 : T - p_j]$ be a binary variable that takes value 1 if the machine starts processing job j in time slot t and 0 otherwise, and let c_{jt} be the contribution to the weighted sum of squared deviations of job j when the machine finishes processing the said job in time instant t, see Eq. (13).

$$c_{jt} = w_j(t - d)^2 \qquad \forall j \in \mathcal{J},\ \forall t \in [p_j : T] \tag{13}$$

The contribution to the objective function of job j when its finishing time is t can be precalculated using (13), and thus (14)–(17) constitutes a valid integer linear programming formulation for the WMSDP.

$$\text{minimize} \sum_{j \in \mathcal{J}} \sum_{t \in [0:T-p_j]} c_{j,t+p_j} x_{jt} \tag{14}$$

$$\sum_{t \in [0:T-p_j]} x_{jt} = 1 \qquad \forall j \in \mathcal{J} \tag{15}$$

$$\sum_{j \in \mathcal{J}} \sum_{t'=[\max\{0,t-p_j\}:t]} x_{j,t'} \le 1 \qquad \forall t \in [0 : T] \tag{16}$$

$$x_{jt} \in \{0,1\} \qquad \forall j \in \mathcal{J},\ t \in [0 : T - p_j] \tag{17}$$

Objective (14) minimizes the weighted sum of squared deviations for all of the jobs. Constraint set (15) forces each job to be performed, constraint set (16) imposes that a maximum of a single job is to be performed in the machine in each time slot, and constraint set (17) defines the domain of the variables. As the contributions of each variable to the objective function can be calculated beforehand, model (14)–(17) constitutes an Integer Linear Programming (ILP) formulation in which the variables report the starting times of the jobs. Moreover, if the starting time of the schedule is set to be equal to an integer value, and the processing times of the jobs are integer numbers, then the above mentioned formulation provides the optimal solution to the WMSDP.

Note that the set of variables of formulation (14)–(17) could also be associated to the completion time of the jobs in the machine without modifying the structure of the problem. In both cases, the ILP is usually known as a time-indexed formulation in the machine scheduling literature (Sousa & Wolsey, 1992).

3.1. Properties of the optimal solution to the linear relaxation

The linear relaxation of model (14)–(17) provides insights into the optimal solution of the problem. These insights are used to develop a primal heuristic for the problem, which will be discussed in Section 4.4.

The heuristic is based on a reinterpretation of the optimal solution of the linear relaxation as the solution to an m-machine parallel version of the problem. Consider the variables x_{jt} that take a positive value and represent the value as a fraction. Let m be equal to the least common multiple of the denominators of these fractions.

Divide each job $j \in \mathcal{J}$ into m separate jobs, each with the same processing time as the original job and a weight equal to the original weight divided by m. It is straightforward to show that the optimal solution of the linear programming relaxation is equivalent to the optimal solution of the above parallel m-machine WMSDP in which the right-hand side of constraint set (16) is equal to m and the constraints impose machine availability. Moreover, the following property holds:

Property 12. *The schedule of each machine satisfies the local and global dominance properties among the derived jobs.*

Proof. Note that each machine constitutes a separate instance of the WMSDP. Consequently, the optimality conditions must hold for each separate machine. □

Property 12 leads to a natural method to provide candidate solutions for the problem in hand. Indeed, if the schedule of any of the machines corresponds to $|\mathcal{J}|$ jobs, each derived from a different, original job, the schedule may constitute an optimal solution for the problem under study, as it fulfills all of the dominance properties described in Section 2.2. Note that while the local-global properties hold for any machine, it does not ensure that the schedule of any machine contains a job derived from each original job.

4. Resolution within a branch-and-cut framework

Time-indexed formulations for single machine scheduling problems are known to be effective for several single machine scheduling problems (Bigras et al., 2008) as the formulation provides tight lower bounds on the objective value. Consequently, this Section describes the use of the formulation described in Section 3 in which the local-global properties are included as valid inequalities. Therefore, the proposed method falls within the branch-and-cut framework.

These inequalities embody the core of Section 4.1. The branch-and-cut method is further enhanced with primal heuristics, a different branching scheme that tries to take advantage of the special characteristics of the problem, and a variable prefixing procedure that also uses the dominance properties. These enhancements are discussed in Sections 4.2 to 4.4.

4.1. Valid inequalities

4.1.1. Formulation of the global-local properties as linear constraints

In addition to the constraint sets (15)–(17), local-global dominance properties, see Section 2, represent additional conditions (valid inequalities) that optimal solutions satisfy.

Each of the four types of local-global properties described in Section 2 may be represented as linear inequalities in different forms. As some of these properties can be represented using a large number of constraints, it is natural to introduce them in the formulation as needed within a branch-and-cut framework.

Local property $i \prec_{\ell(t^*, d-p_i-p_j]} j$. This local property states that if jobs i and j are to be consecutively scheduled, and the starting time of job i is to belong to time interval $(t^*, d-p_i-p_j]$, then job i must precede job j. Consequently, constraint set (18), which avoids job j to precede job i, must hold for each pair of jobs i, j such that $i \prec_{\ell(t^*, d-p_i-p_j]} j$:

$$x_{j,t-p_j} + x_{i,t} \le 1 \qquad \forall t \in [\max\{\lfloor t^*+1\rfloor, p_j\} : d - p_i - p_j] \tag{18}$$

It is possible to modify (18) so as to consider that jobs i and j cannot be performed simultaneously. Thus, if job i is to be scheduled in time slot t, then job j cannot be scheduled within time slots $[t-p_j : t]$. Therefore, (18) can be rewritten as (19), which is equivalent to (18) for integer solutions but not for their linear relaxation:

$$\sum_{t' \in [t-p_j:t]} x_{j,t'} + x_{i,t} \le 1 \qquad \forall t \in [\max\{\lfloor t^*+1\rfloor, p_j\} : d - p_i - p_j] \tag{19}$$

Local property $i \prec_{\ell[d,t^*)} j$. As in the previous case, the local property states a local order between jobs i and j if the machine consecutively processes both jobs starting within $[d, t^*)$. Therefore, constraint set (20) must be satisfied for each pair of jobs i, j such that $i \prec_{\ell[d,t^*)} j$ applies. Moreover, following the logic described above, (21) surrogates (20).

$$x_{j,t-p_j} + x_{i,t} \le 1 \qquad \forall t \in [d + p_j : \lceil t^* - 1 \rceil] \tag{20}$$

$$\sum_{t' \in [t-p_j:t]} x_{j,t'} + x_{i,t} \le 1 \qquad \forall t \in [d + p_j : \lceil t^* - 1 \rceil] \tag{21}$$

Global property $i \prec_{g[0,t^*]} j$, and its special case $i \prec_{g[0,d-p_i-p_j]} j$ with $t^* = d - p_i - p_j$. The global property forces job i to be scheduled before job j if both jobs start being processed no later than t^*.

Eq. (22) provides a compact representation of the property. The summation on the left-hand side equals the starting time of job i if the machine starts processing job i within the time slots $[0: \lfloor t^* \rfloor]$ and 0 otherwise. As the right-hand side can only take non-negative values, if job i does not start its processing time within the said time slots, constraint (22) is always satisfied. Conversely, the machine starts processing job i within the said time slots and job j must be processed after job i, whether job j starts processing in the time slots $[0: \lfloor t^* \rfloor]$ or in later time slots.

$$\sum_{t \in [0:\lfloor t^* \rfloor]} t \cdot x_{it} \le \sum_{t \in [0:T-p_j]} t \cdot x_{jt} \tag{22}$$

An alternative linear representation of the same property corresponds to constraint set (23), which can be combined with the logic described in (19) and (21) to improve its ability to eliminate partial solutions; and it is more suitable for branch-and-cut approaches. Therefore, specific inequalities from (23) can be added to the linear formulation whenever a relaxed solution does not satisfy the inequality.

$$\sum_{t' \in [t:\lfloor t^* \rfloor]} x_{it'} + \sum_{t' \in [0:t]} x_{jt'} \leq 1 \qquad \forall t \in [0 : \lfloor t^* \rfloor] \tag{23}$$

Constraint set (23) ensures that if the machine starts processing job i in a time slot $t \in [0\colon \lfloor t^* \rfloor]$, then job j cannot be processed at earlier intervals, that is, in time slots $t' \in [0 : t - p_j]$, or simultaneously to job i, that is, within the time slots $t' \in [t - p_j : t]$.

Global property $i \prec_{g[t^*,\infty)} j$ and its special case $i \prec_{g[d,\,\infty)} j$ with $t^* = d$. As in the previous case, two alternative formulations are considered. Eq. (24) provides a compact formulation of the global property, while (25) provides a set of inequalities suitable for branch-and-cut approaches.

$$\sum_{t \in [\lceil t^* \rceil : T - p_i]} t\, x_{it} \leq M \sum_{t \in [0 : \lceil t^* \rceil - 1]} x_{jt} + \sum_{t \in [\lceil t^* \rceil : T - p_j]} t\, x_{jt} \tag{24}$$

The left-hand side of constraint (24) is 0 whenever job i is schedule before t^*, hence the inequality holds (the right-hand side cannot take negative values). Otherwise, the left-hand side provides the starting time of job i in the machine, and the right-hand side should be greater than or equal to the left-hand side whenever (1) job j is processed before interval $[t^*, \infty)$; or (2) job j is processed within the interval and the schedule satisfies the property.

The first summation of the right-hand side is equal to 1 when the first case applies. After multiplication for a sufficiently large constant M, (24) holds. Note that M can be set equal to T. The second summation of the right-hand side calculates the starting time of job j; consequently, the starting time of job j and the constraint ensure that the starting time of job i is no larger than the starting time of job j, thus ensuring the global property.

The condition can be written as constraint set (25), which is sparser (each constraint has fewer non-zero coefficients) and does not have big-M coefficients. Each constraint forces the global property when job i is scheduled in the machine at time t.

$$\sum_{t' \in [t:T-p_i]} x_{it'} + \sum_{t' \in [\lceil t^* \rceil : t]} x_{jt'} \leq 1 \qquad \forall t \in [\lceil t^* \rceil : T - p_i] \tag{25}$$

4.1.2. Introduction of valid inequalities

Whenever the optimal solution to the linear relaxation is obtained, we try to add valid inequalities described in Section 4.1 to tighten the linear relaxation. In addition to the cuts provided by the ILP solver, local constraints (19) and (21), and global constraints (23) and (25) that are not fulfilled by the optimal solution of the relaxation are added to the formulation.

The procedure used to detect violated inequalities is of enumerative nature; that is, all of the inequalities are tested and added if the current relaxed solution is found to be infeasible. In order to avoid unnecessary computations and to reduce the time required to test for feasibility, a preprocessing step is performed to identify local and global precedence rules between each pair of jobs. These rules are obtained according to the conditions stated in Section 2, and if a breakpoint value, t^*, is required, its value is calculated and stored.

Then, whenever a solution, $\hat{x}$, to the linear relaxation is obtained, the accumulated value for each job j of its corresponding variables before and after the due date, d, is computed. Denote these values by $a_j^- = \sum_{t \in [0:d-p_j]} \hat{x}_{jt}$ and $a_j^+ = \sum_{t \in [d:T-p_j]} \hat{x}_{jt}$. Then, the following steps are performed to test for valid inequalities:

- For each local property $i \prec_{\ell[t^*, d-p_i-p_j)} j$ and time slot $t \in [\lceil t^* \rceil : d - p_i - p_j]$ such that $(a_i^- + a_j^-) > 1$ and $\hat{x}_{it} > 0$, its corresponding inequality in (19) is tested.
- For each local property $i \prec_{\ell[d,\,t^*)} j$ and time slot $t \in [d\colon \lfloor t^* \rfloor]$ such that $(a_i^+ + a_j^+) > 1$ and $\hat{x}_{it} > 0$, the corresponding inequality in (21) is considered.
- For each global property $i \prec_{g[0,t^*)} j$ and time slot $t \in [0\colon \lfloor t^* \rfloor]$ such that $(a_i^- + a_j^-) > 1$ and $\hat{x}_{it} > 0$, the corresponding inequality in (23) is tested.
- For each global property $i \prec_{g[t^*,\,\infty)} j$ and time slot $t \in [\lceil t^* \rceil : T - 1]$ such that $(a_i^+ + a_j^+) > 1$ and $\hat{x}_{it} > 0$, the corresponding inequality in (25) is considered.

4.2. Branching scheme

The basic method included in the MILP solver branches by setting a variable either to its upper bound or to its lower bound. This branching scheme ignores the special characteristics of the problem and may lead to unbalanced search trees. Therefore, a more sophisticated method is proposed. The method also tries to create subproblems in which the linear formulation leads to the addition of additional cutting planes (that is, the identification of precedence rules).

The proposed scheme branches by including a constraint that splits the set of possible starting times for a job into two parts. Essentially, the branching rule introduces a constraint in each subproblem as follows: let $\bar{t}$ be the dividing period, then one branch includes constraint (26), while the second branch includes constraint (27).

$$\sum_{t \in [0:\bar{t}-1]} x_{it} = 1 \tag{26}$$

$$\sum_{t \in [\bar{t}:T-p_i]} x_{it} = 1 \tag{27}$$

While it would be possible to implement the branching scheme using the above constraints, the implementation sets $x_{it} = 0,\ \forall t \in [\bar{t} : T - p_i]$ in one branch and $x_{it} = 0,\ \forall t \in [0 : \bar{t} - 1]$ in the other branch.

In order to determine the job and the time period to branch, the following algorithm, see Algorithm 1, is used. For each job

Algorithm 1 Branching rule.

```
Input: A solution to the linear programming relaxation $\hat{x}$.
Output: A branching decision (job $\hat{\imath}$ and time period $\hat{t}$) or identification of integrality of the current partial solution.
1: for $i \in \mathcal{J}$ do
2:     $t_i^- := \min_{t \in [0:T-p_i] \wedge \hat{x}_{it} > 0} \{t\}$,
3:     $t_i^+ := \max_{t \in [0:T-p_i] \wedge \hat{x}_{it} > 0} \{t\}$ ;
4: $\hat{\imath} := \arg\max_{i \in \mathcal{J}} \{t_i^+ - t_i^-\}$;
5: if $t_{\hat{\imath}}^+ = t_{\hat{\imath}}^-$ then
6:     The current solution is integral. No branching required
7: else
8:     if $t_{\hat{\imath}}^- < d < t_{\hat{\imath}}^+$ then
9:         $\hat{t} := d$
10:    else
11:        $\hat{t} := \lceil (t_{\hat{\imath}}^+ + t_{\hat{\imath}}^-)/2 \rceil$
12: return $\hat{\imath}$, $\hat{t}$
```

i, the algorithm calculates the earliest and latest time periods, t, in which the corresponding variable $\hat{x}_{it}$ has a positive value, and branches on the job for which the interval between both values is the largest.

If the interval contains the time slots preceding and succeeding the due date d, the division is made in such a way that the partition forces the job to be scheduled before or after the due date, $\hat{t} = d$. Otherwise, $\bar{t}$ is set to the average between the minimum and the maximum t in which some x_{it} take a positive value.

The proposed branching rule is based on the three following observations. First, the optimal solution to the linear relaxation contains few variables with fractional values. Some of these variables tend to correspond to one job and several completion times with equal (or similar) objective coefficients; hence, the completion times are equally (or nearly equally) spaced from the common due date. In this case, the linear relaxation fails to set the job completion before or after the due date, but when the job is forced to start before or after it, the cost coefficients impose a natural ordering among jobs. Second, the contribution of a job to the objective is related to its completion time. According to the first observation, the jobs with a larger interval are scheduled in time periods that are further from the due date, hence the rule prioritizes branching in jobs with larger contributions to the objective function. Third, forcing a job to be scheduled before or after the due date enables the method to introduce additional valid inequalities according to the local and global properties.

4.3. Variable prefixing

One of the main drawbacks of time-indexed formulations is the large number of variables required by the model. Consequently, the use of techniques to reduce their number by prefixing the value of some of these variables to their value before solving a linear programming relaxation is likely to reduce the total running time of the algorithm.

The proposed branch-and-cut procedure uses the dominance properties and the branching decisions to fix the value of some variables in the root node and in each branching decision.

We first consider two variable prefixing techniques at the root node. These methods require the resolution of special subset-sum knapsack problems, each considering the subset of jobs that share a global precedence property with a job i:

- Let $\mathcal{J}^i \subset \mathcal{J}$ be the subset of jobs j such that $i \prec_{g[d,\infty)} j$. If $\sum_{j\in\mathcal{J}^i} p_j - \max_{j\in\mathcal{J}^i} p_j > d$, then some of the jobs in $\mathcal{J}^i$ start after d and, according to the global dominance properties, job i is to start before the processing times of this subset of jobs. More formally, for each job $i \in \mathcal{J}$ and its subset of jobs $\mathcal{J}^i$, we aim at finding the minimum processing time among jobs that cannot be scheduled before the due date d (the minimum processing time of jobs that are to start after the job i). This problem is equivalent to selecting the jobs that maximize their sum of processing times while satisfying that the sum of all but one job in the subset is smaller than the due date d, see model (28)–(30), in which variables $\mathbf{x} \in \{0,1\}^{|\mathcal{J}^i|}$ take value equal to 1 if the job belongs to the subset and 0 otherwise.

$$z = \max \sum_{j\in\mathcal{J}^i} p_j x_j \tag{28}$$

$$\sum_{j\in\mathcal{J}^i} p_j x_j - \max_{j\in\mathcal{J}^i} \{p_j x_j\} < d \tag{29}$$

$$x_j \in \{0,1\} \qquad \forall j \in \mathcal{J}^i \tag{30}$$

Notice that there exists an optimal solution to (28)–(30) that contains the job with the largest processing time among the jobs in $\mathcal{J}^i$ (the proof is by a contradiction, as it would be possible to substitute the job with the largest processing time in the subset for a job with a larger processing time not in the subset, improving the objective function while not modifying the feasibility of the solution).

Based on the previous observation and the integrality of the processing times, model (28)–(30) can be solved as a subset-sum knapsack problem, and job i cannot start later than $\hat{s} = T - p_i - \left(\sum_{j\in\mathcal{J}^i} p_j - z\right)$ in any optimal schedule, therefore $x_{it} = 0,\ \forall t \in [\hat{s} : T - p_i]$.
- Let $\mathcal{J}^i \subset \mathcal{J}$ be the subset of jobs j such that $j \prec_{g[0,d-p_i-p_j]} i$. If the sum of processing times among jobs in $\mathcal{J}^i$ exceeds $h - d$, then a subset of the jobs in $\mathcal{J}^i$ is to be scheduled before the due date and, according to the global precedence property, job i is to start being processed after the processing times of this subset of jobs. If the subset $\mathcal{J}^* \subset \mathcal{J}^i$ maximizes the processing time performed after the due date, then the sum of the processing times of the jobs belonging to $\mathcal{J}^i \setminus \mathcal{J}^*$ constitutes a lower bound on the total processing time that must be processed before job i is scheduled in any optimal schedule and thus we can safely set to 0 all variables fulfilling: $x_{it} = 0\ \forall t \in [0 : \sum_{j\in\mathcal{J}^i\setminus\mathcal{J}^*} p_j]$.

The previous prefixing methods consider that it is unavoidable for some subsets of jobs to be scheduled within specific time intervals. Note that whenever a branching decision is made, it is possible to apply additional prefixing rules based on the fact that in order to satisfy global dominance properties some jobs are to be performed within specific time intervals. While some of these rules were implemented and tested, they were deemed inefficient and their application is not reported in the results provided in Section 5.

4.4. Primal heuristics

In order to find fast primal solutions as well as to verify optimality within reduced running times, two constructive heuristics are proposed. An initial heuristic, H1, uses the possibly fractional solution as the priority index of a dispatching procedure, while a second heuristic, H2, is based on the property defined in Section 3 to test for candidate solutions. In addition to the constructive heuristics, a neighborhood-based local search is used in order to improve the solutions provided by the heuristics. Both heuristics make use of the optimal solution to the linear relaxation of the mathematical formulation and are calculated whenever the linear programming solver looks for a primal solution.

While the implementation calculates different heuristics only the best found solution is returned to the linear programming solver. This approach prioritizes the use of the primal heuristic as a method to report the optimal or near-optimal objective value early in the search and neglects possible advantages that the linear programming solver may obtain from the additional solutions, like solution polishing subroutines.

4.4.1. Priority based heuristic (heuristic H1)

This heuristic iteratively constructs a schedule σ by using a fractional solution $\hat{x}$ to decide the order of the jobs. Once the schedule is obtained, the best starting time is evaluated. The heuristic works as follows. Initially, the solution is empty and the time t^c is set to the minimum value in which a variable takes positive value, $\hat{x}_{it^c} > 0$. In each iteration, the heuristic selects a job to include in the schedule σ according to the accumulated value of the variables $\hat{x}$ up to the time period t^c; and updates t^c by summing the processing time of the selected job. Algorithm 2 depicts the proposed method.

Once a schedule σ is obtained, the optimal, non-necessarily integral, starting time, s_σ, of the first job can be obtained by minimizing Eq. (1), see Property 2 in Section 2, which is a derivable, continuous and convex function. As the solution method only considers integer starting times, $\lfloor s_\sigma \rfloor$, $\lceil s_\sigma \rceil$ and 0 are considered as candidate starting times.

Algorithm 2 Heuristic H1.

```
Input: A solution to the linear programming relaxation $\hat{x}$.
Output: A heuristic solution $\sigma$.
1: $t^c := \min t : \exists x_{it} > 0$; $S := \emptyset$
2: while $S \neq \mathcal{J}$ do
3:   for $i \in \mathcal{J}$ do
4:     $h_i := \sum_{t\in[0:t^c]} x_{it}$
5:   let $i^*$ be $arg\,max_{i\in\mathcal{J}\wedge i\notin S}\{h_i\}$
6:   $S = S \cup \{i^*\}$; $t^c := t^c + p_{i^*}$; $\sigma_{|S|} := i^*$
```

4.4.2. Enumerative heuristic (heuristic H2)

The heuristic is based on the properties of the optimal solution to the linear relaxation described in Section 3.1. According to these properties, the optimal solution of the linear relaxation, $\hat{x}$, can be seen as the optimal solution to an m-machine problem. Let l_t be the fraction of the machine used in a time period t, $l_t = \sum_{j\in\mathcal{J}} \sum_{t'=[\max\{0,t-p_i\}:t]} x_{j,t'}$, that is the left-hand side of constraint set (16) for time period t, and let $l_{-1} = 0$. Denote the set of starting times of the schedule of each of the m machines by T^-. Following the optimality of each schedule, Property 12, and the non-idleness of any optimal schedule, Property 1, the starting time of any schedule corresponds to the set of time periods in which machine usage increases, $T^- = \{t : t \in [0 : T] \wedge l_t > l_{t-1}\}$.

Obviously, it is possible to construct at least one schedule of jobs starting from each $t \in T^-$ in which whenever a job finishes processing another job starts processing. This schedule constitutes the optimal schedule of one of the m machines, see Section 3.1, and whenever such a schedule contains one job derived from each job of the original problem, it constitutes a valid solution for the WMSDP.

Algorithm 3 Heuristic H2.

```
Input: A solution to the linear programming relaxation $\hat{x}$.
Output: Possibly, a heuristic solution $\sigma$.
1: Obtain $T^-$, and let $\sigma := null$; $S := \emptyset$
2: for $t \in T^-$ do
3:   for $j \in \mathcal{J} : \hat{x}_{jt} > 0$ do
4:     $\sigma_{(1)} := j$
5:     recursive($\hat{x}, S \cup \{j\}, t + p_{jt}, \sigma$)
```

Heuristic H2, see Algorithm 3, enumerates all possible candidate schedules for the m machines using the recursive function described in Algorithm 4. Whenever the recursive function reports a feasible solution, its best starting time is evaluated as in heuristic H1, and the best solution among those evaluated is considered the output of the heuristic. Note that the identification of the m machines is not required. The description only makes use of the said interpretation as it is illustrative of the rationale behind the heuristic.

Algorithm 4 Function **recursive**($\hat{x}$, S, t, σ).

```
Input: A solution to the linear programming relaxation $\hat{x}$; the set
  of already assigned jobs $S$; the current time $t$; the solution un-
  der construction $\sigma$; the cost of the current partial solution.
1: for $j \in \mathcal{J} : \hat{x}_{jt} > 0$ do
2:   $\sigma_{(|S|+1)} := j$
3:   if $S \cup \{j\} = \mathcal{J}$ then
4:     report $\sigma$ as a feasible solution
5:   else
6:     call recursive($\hat{x}, S \cup \{j\}, t + p_{jt}, \sigma$)
```

While heuristic H2 is of enumerative nature, the implementation requires reduced running times (fractions of a second) for all of the considered cases. Moreover, heuristic H2 may fail to report a solution, or may report multiple solutions.

4.4.3. Local search

Whenever heuristic H1 or heuristic H2 provide a new solution, a local search method is applied to the corresponding schedule. The method tries to improve the current solution by applying small changes to it. The neighborhood used in this work corresponds to the 2-exchange in which the neighbors of a schedule correspond to other schedules in which the position of only two jobs differs between them.

During the neighbor exploration, whenever a neighbor improves the incumbent, it becomes the new incumbent. These steps are repeated until no further change can yield any other improved solutions, that is, until a local optimal solution has been found. Notice that in order to evaluate each exchange, we need to evaluate the best starting time for the schedule.

5. Computational experiment

5.1. Implementation details and description of the instances

The proposed branch-and-cut method was implemented in C++ using the Concert Technology framework provided by IBM ILOG CPLEX library version 12.6 and compiled using the GNU GCC compiler 4.9.0. The experiments were run on an Intel Xeon multicore machine with 128 gigabytes RAM running the CentOS release 6.7 Linux operating system, and 32 processors each running at 2.00 gigahertz. Note that CPLEX was set to use a single processor and multiple processes were run simultaneously on the machine allotting 16 gigabytes RAM per process (with the exception of the instances with 300 jobs and maximum processing time equal to 100, in which the total 128 gigabytes was allotted to each process). Default CPLEX parameters were used with the exception of the relative MILP tolerance gap of CPLEX, which was set to 0 instead of its default value (1×10^{-4}), and the default linear programming solver for the root node, which was set to the barrier algorithm. The first change increases the running time but avoids discarding a possible optimal solution due to relatively low differences among feasible solutions, whereas the second provides improvements on the time required to solve the root node.

The processing time and weights of each of the jobs of an instance were generated following the proposal from Höhn and Jacobs (2012). The generator uses two parameters, the number of jobs of the instance, $n = |\mathcal{J}|$, and a parameter σ that controls the relative complexity of the instance based on the relationship to the Smith ratio between tasks. Moreover, the common due date is ascertained according to different types of instances, *unrestricted*, *restricted* and *tightly restricted*.

The method described in Höhn and Jacobs (2012) randomly generates the processing time, p_j of each job $j \in \mathcal{J}$, from a discrete uniform distribution $\mathcal{U}\{1, 100\}$, thus the maximum processing time of a job, t^{max}, is 100. The weight, w_j, of each task is calculated as: $w_j = p_j \cdot 2^{\mathcal{N}(0,\sigma^2)}$, where $\mathcal{N}(0, \sigma^2)$ indicates that the exponent is drawn from a normal distribution with mean 0 and standard deviation σ. According to Höhn and Jacobs (2012), smaller values of σ generate instances with fewer global-local dominance properties.

The common due date is randomly drawn from a uniform distribution which varies according to the instance data and the type of desired instance. For *tightly restricted instances*, the due date is randomly drawn from

$$\mathcal{U}\{0, C(\sigma^{WSPT})/\sum_{j\in\mathcal{J}} w_j\}.$$

For *restricted* instances, the due date is randomly drawn from

$$\mathcal{U}\left\{C(\sigma^{WSPT})\Big/\sum_{j\in\mathcal{J}} w_j, C(\sigma^{WLPT})\Big/\sum_{j\in\mathcal{J}} w_j\right\}.$$

Finally, for *unrestricted* instances, the due date d is modified following Property 5, see Section 2.1, and the due date is set to $d = \lceil C(\sigma^{WLPT})/\sum_{j\in\mathcal{J}} w_j\rceil$.

Two alternative maximum processing times have been considered, $t^{max} = 25$ and $t^{max} = 50$. For these instances, the processing times are drawn from $\mathcal{U}\{1, 25\}$ and $\mathcal{U}\{1, 50\}$. The decision to test different ranges for the processing times was motivated by the high dependency on the results that time-indexed formulations show on different scheduling horizons and processing time ranges.

The final benchmark instance set is composed of 2475 instances. Five instances were generated per combination of number of jobs $|\mathcal{J}| = \{25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 300\}$, level of complexity $\sigma = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, type of due date {*tightly restricted*, *restricted*, *unrestricted*}, and maximum processing time $t^{max} = \{25, 50, 100\}$.

5.2. Computational results

In order to validate the usefulness of the proposed methods, different experiments were performed with the following goals: (1) verify whether the proposed algorithm improves upon the use of the off-the-shelf solver; (2) determine the effect of the different components of the algorithm on its overall performance; (3) check the effect of the different parameters of the instances on the behavior of the algorithm; (4) check the limitations of the procedure; and (5) verify the effect of limiting the starting times to integer values, see Property 7.

A computational test was run to solve all of the 2475 instances using the default CPLEX configuration (with the exception of the tolerance gap and the default LP solver for the root node, as already mentioned) as well as the different versions of the algorithm according to the application or not of the main components of the algorithm; namely the branching rule, additional inequalities and the primal heuristic (and thus a total of 8 alternative procedures were tested). A time limit of 14400 seconds (4 hours) was imposed in order to verify the ability of the algorithms to reach optimality when computation time was not considered as a critical issue.

Table 1 reports the results obtained by (1) the default configuration of CPLEX and the two most promising tested procedures, (2) a version that only includes the heuristic and the branching rule (but not the inequalities), and (3) a version that includes all three components. The results of Table 1 group instances according to the number of jobs and the maximum processing time of the jobs, since these two features of the instances control the number of variables required by the time-indexed formulation, and thus they constitute the main reference on the complexity of an instance for the off-the-shelf CPLEX implementation.

For each group of instances, composed of 75 instances, three metrics are reported: average running time (whether the algorithm reached the time limit or finished due to verifying optimality), the number of optimal solutions found (out of the 75 instances), and the average gap between the best solution and the best bound reported by the algorithm. The gap is calculated as $100 \cdot (UB - LB)/UB$, in which UB and LB stand for the upper and the lower bounds reported by the method after stopping their computation.

The results reported in Table 1 highlight the limits of the default CPLEX configuration to solve the problem. While instances with large ($t^{max} = 100$) processing times and up to 75 jobs can be solved to optimality, the performance of the algorithm deteriorates significantly for larger instances and the problem cannot be solved to optimality even when large computation times are allowed.

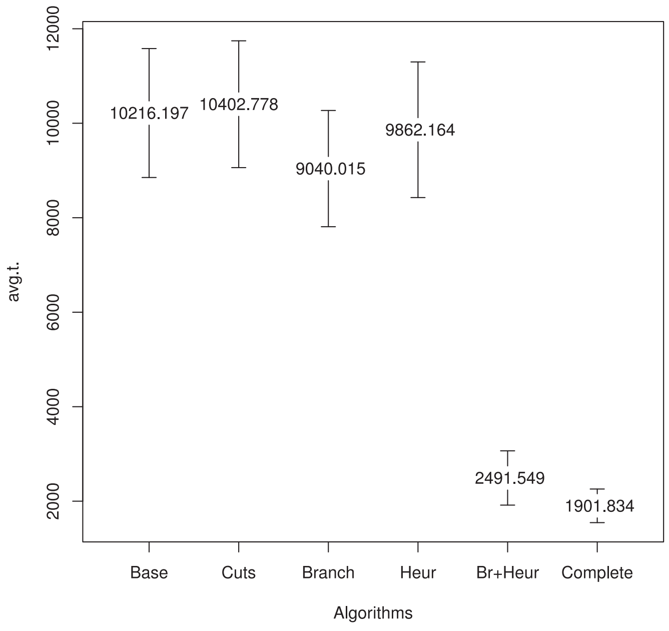


Fig. 5. Results of different versions of the algorithm for instances with $|\mathcal{J}| = 150$ jobs and maximum processing times $t^{max} = 100$. The average running time (and its 95% confidence interval) for each version of the algorithm is reported.

Notice that the gaps remain relatively small with the exception of the largest tested instances, thus showing the quality of the linear relaxation as well as the capacity of the primal heuristic to find good feasible solutions. Nonetheless, when the size of the instance increases, the ability of the default CPLEX configuration to deal with such instances decreases, since it has to rely on enumeration, which becomes more computationally inefficient as the number of variables increases.

The results provided in Table 1 for the proposed algorithms show that both methods clearly outperform the default CPLEX configuration. Moreover, the difference becomes more increasingly apparent as the size of the instance grows, leading us to conclude that the proposed method is to be preferred over the default CPLEX configuration.

While the use of the special branching scheme and the primal heuristics suffices to solve most instances (including almost all of the instances with $|\mathcal{J}| \leq 150$, or $t^{max} \leq 50$), the solution of larger instances benefits from the valid inequalities derived from the dominance relations, specially regarding the ability of the algorithm to obtain small optimality gaps in the largest tested instances. We conjecture that the bottleneck of the proposed method in the largest instances corresponds to the resolution of the linear programming relaxations. In such a situation, the dominance properties provide stronger valid inequalities than default cutting planes, reducing the number of times that the linear programming relaxation is re-optimized within each branching node. Note that for some instances with $|\mathcal{J}| \geq 300$ and $p^{max} = 100$, the configurations that did not make use of the proposed valid inequalities could not solve the root node (hence the large average optimality gap reported in Table 1).

To further illustrate the relative importance of each algorithmic component, the results of the different versions of the algorithm are analyzed in Fig. 5. Fig. 5 provides the average running times as well as the confidence intervals for six different versions of the algorithm for the subset of instances with 150 jobs and maximum processing time of 100 time units. These instances were chosen because, although they are difficult to solve, they can be solved within reduced optimality gaps by the default CPLEX configuration.

Table 1
Comparison among the default CPLEX configuration (columns CPLEX default), the inclusion of the proposed branching and heuristic scheme (columns *Branching + heuristic*) and the proposed algorithm including the cuts derived from the dominance properties (columns *Complete*). Instances are grouped according to their number of jobs (column $|\mathcal{J}|$) and maximum processing time (column p.times). For each algorithm and grouping of instances, three values are reported: (1) the average running time in seconds (column t(second)); (2) the number of reported optimal solutions (column #Opt); and (3) the average optimality gap (column Gap).

$\lvert\mathcal{J}\rvert$	t^{max}	CPLEX default			*Branching + heuristic*			*Complete*		
		t(second)	#Opt	Gap	t(second)	#Opt	Gap	t(second)	#Opt	Gap
25	25	0.9	75	0	0.8	75	0	0.8	75	0
	50	3.3	75	0	2.6	75	0	2.3	75	0
	100	13	75	0	11.3	75	0	10	75	0
50	25	5.8	75	0	3.3	75	0	3.1	75	0
	50	39.9	75	0	13.1	75	0	11.4	75	0
	100	209.6	75	0	64.5	75	0	54.5	75	0
75	25	28.8	75	0	8.8	75	0	8	75	0
	50	569.7	75	0	42.7	75	0	36.8	75	0
	100	3386.3	66	0.0	253.4	75	0	204	75	0
100	25	188.8	75	0	23.3	75	0	21.2	75	0
	50	1540	73	0.0	94.8	75	0	74.6	75	0
	100	7993.6	42	0.0	596	75	0	540.8	75	0
125	25	279.5	75	0	43.6	75	0	39.6	75	0
	50	3476.5	67	0.0	180.9	75	0	149.3	75	0
	100	8801.6	33	0.1	1168	75	0	1075	75	0
150	25	1019.3	75	0	82.3	75	0	80.9	75	0
	50	6260.7	51	0.0	357.7	75	0	326.5	75	0
	100	10216.2	26	5.8	2491.5	73	2.7	1901.8	75	0
175	25	2307.9	71	0.0	136.6	75	0	141	75	0
	50	7440.4	48	0.0	648.3	75	0	517.5	75	0
	100	12090	17	20.3	4779	70	6.7	3443.1	74	1.3
200	25	4010.8	65	0.0	195.1	75	0	207.1	75	0
	50	9576.9	32	0.0	992.1	75	0	927.7	75	0
	100	12357.6	15	30.4	7463.8	53	16	5877.4	61	6.7
225	25	5125.7	59	0.0	321.9	75	0	332.3	75	0
	50	9322.7	33	0.8	1669.1	74	0.0	1361.4	75	0
	100	12887	12	36.4	9279.7	46	20	7539.3	58	5.3
250	25	6380.9	54	0.0	802.3	74	0.0	888.4	74	0.0
	50	11707.5	18	4.6	4287.8	68	2.7	3281.4	71	0.0
	100	12677.5	14	37.4	10450.6	42	21.3	9352.1	50	5.3
300	25	9276.1	37	0.0	1191.8	75	0	1525.7	73	0.0
	50	11213.7	21	5.2	5525.5	60	1.3	5757.7	60	0.0
	100	14223.8	2	55.2	13720.1	8	42.7	13061.1	17	18.7

The six methods reported are: *Base*, which corresponds to the default CPLEX configuration; methods *Cuts*, *Branch* and *Heur*, which correspond to versions of the algorithm in which only one element has been introduced, be it the inequalities, the branching rule, or the primal heuristic respectively; method *Br+Heur*, which reports the results obtained when the branching rule and primal heuristic are included; and *Complete*, which provides the results of the method that uses all three of the algorithmic components.

A comparison of these intervals shows that minimal improvements over the default CPLEX method are achieved by each component separately. The introduction of the cuts is even harmful to the performance of the algorithm (in terms of average running times). According to additional metrics, the source of the degradation is an increase in the running time required by the default heuristic. Consequently, we conclude that the built-in primal heuristic is misguided by the additional inequalities, leading to a worsening of results in terms of computational times.

The use of both the heuristic and the branching rules, method *Br+Heur*, provides a clear improvement of results, since it reduces the average time devoted to obtaining primal feasible solutions. Additionally, the specific branching rule significantly reduces the enumeration tree. We do not report results of the other combinations of two elements (namely branching rule or heuristic with the introduction of inequalities) because they are not competitive with the combination of the branching rule and the primal heuristic. Further gains can be obtained by introducing the valid inequalities provided by the dominance properties, as shown by the smaller average value associated to the *Complete* algorithm.

As the intervals of algorithms *Br+Heur* and *Complete* intersect and the differences between both methods reported in Table 1 are small for most instance sizes, we further investigate whether and when the differences between both methods are significant. According to the generation procedure, see (Höhn & Jacobs, 2012), the variability of weights, controlled by parameter σ, controls the number of global and local properties among tasks. Table 2 provides a comparison of results between algorithms *Br+Heur* and *Complete* on the larger instances, $|\mathcal{J}| > 150$ and $t^{max} = 100$. Table 2 provides results for instances grouped according to the number of jobs, $|\mathcal{J}|$ and the weight variability, σ. For each group of instances, composed of 15 instances, the average running time (whether the algorithm reached the time limit or verified optimality) and the number of optimal solutions are reported. Moreover, the p-value of a Wilcoxon signed-rank paired test on the required average running time for each group of instances is also reported. The Wilcoxon signed-rank paired test was chosen as it assesses the performance of the algorithm when the effect of each specific instance is blocked.

The results of Table 2 highlight that the differences are statistically significant, specifically for instances with larger values of weight variability (larger σ), and instance sizes solvable within the imposed time limit. For the largest instances, the differences are not significant given that both algorithms fail to obtain the optimal solution, and thus their running times are equal (tied). For instances with smaller levels of variability (hence, with fewer dominance properties among jobs), both algorithms are not statistically different. Also note that while the introduction of the dominance

Table 2
Comparison of the performance of algorithms *Br+Heur* and *Complete*. For each group of 15 instances grouped according to the number of jobs, row $\mathcal{J}$, and their weight variability, column σ, the following two values for each algorithm, be it *Br+Heur* and *Complete*, are reported: the average running time, column t(second), and number of optimal solutions found, column #Opt, . Finally, the p-value of the Wilcoxon signed-rank paired test is also provided (* significant at $p< 0.1$; ** significant at $p< 0.05$; *** significant at $p< 0.01$; **** significant at $p< 0.001$).

$\|\mathcal{J}\|$	σ	*Br+Heur*		*Complete*		
		t(second)	#Opt	t(second)	#Opt	p-value
175	0.1	2772.4	15	2057.6	15	0.3591
	0.3	3753.8	14	3617.2	14	0.4144
	0.5	4817.7	14	3464.6	15	0.1070
	0.7	7720.3	12	4176.0	15	0.0008 ****
	0.9	4830.9	15	3899.9	15	0.0181 **
200	0.1	5750.3	11	4055.0	13	0.2635
	0.3	4847.7	14	4124.8	14	0.0069 ***
	0.5	10150.4	7	7995.4	10	0.0059 ***
	0.7	6783.6	12	5472.3	13	0.0546 *
	0.9	9787.2	9	7739.3	11	0.0191 **
225	0.1	8162.6	10	7351.6	12	0.5049
	0.3	8885.1	10	6338.9	14	0.0253 **
	0.5	8022.9	11	6669.9	13	0.0302 **
	0.7	9690.1	9	8112.3	11	0.0831 **
	0.9	11637.9	6	9223.8	8	0.0142 **
250	0.1	8305.4	11	7032.5	13	0.6750
	0.3	10465.3	9	9037.6	11	0.2240
	0.5	11813.0	7	11631.5	8	0.7598
	0.7	11484.1	7	9508.2	10	0.0059 ***
	0.9	10185.1	8	9550.5	8	0.0243 **
300	0.1	13365.2	3	13307.8	2	1
	0.3	13812.6	1	12836.8	3	0.3710
	0.5	13912.4	2	12999.8	4	0.4227
	0.7	13505.0	2	12155.3	7	0.0224 **
	0.9	14000	0	14000	0	1

properties enables the algorithm to solve instances in shorter running times, the effect of the dominance properties seems to be smaller than the effects of the branching rule and the heuristic method.

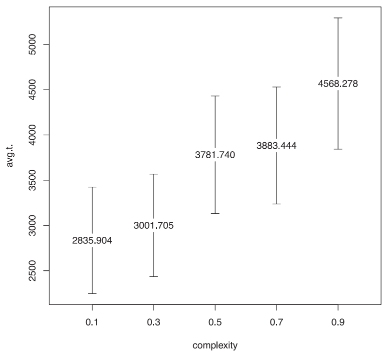
Notice that Table 2 shows that the instances with larger weights variability are more difficult to solve than instances with smaller weight variability. In order to provide a comparison of the relative importance of the different characteristics of the instances in the behavior of the proposed algorithm, Fig. 6 provides plots of the average running time of the *Complete* algorithm when solving instances of varying characteristics. Fig. 6(a) groups all of the instances according to the degree of difficulty introduced by the correlation among the processing times and the weights (parameter σ), while Fig. 6(b) groups all of the instances according to their type of due dates (be it *tightly restricted*, *restricted* or *unrestricted*). For each grouping of instances, the average running time among all of the instances together with their confidence intervals are reported.

Fig. 6(a) shows similar results to those provided in Table 2. The average computational time increases as the level of complexity increases. Note that previous works, see (Höhn & Jacobs, 2012), deem instances with larger amounts of dominance relations easier to solve. We believe that the origin of such differences stems from the objective function (in the WMSDP dominance properties only apply to a subset of the scheduling horizon, while in Höhn and Jacobs (2012) the dominance property applies to the whole horizon) and from the solution method (in a enumeration based method a dominance property implies an immediate reduction of the search tree, while in the branch-and-cut approach it provides possibly tighter lower bounds).

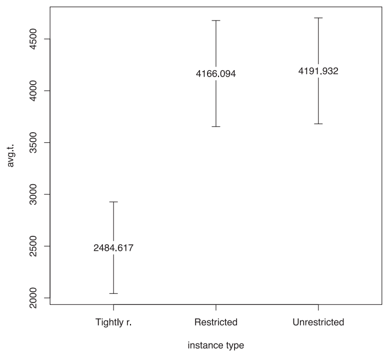
Fig. 6(b) also shows an increase in the average running time required by the *restricted* and the *unrestricted* instances when compared to *tightly restricted* instances. This result is a consequence of the differences in the planning horizon, which is smaller for *tightly restricted* instances than for *restricted* or *unrestricted* ones, see Section 3. As the planning horizon is smaller, the formulation requires fewer variables, leading to easier-to-solve problems. This difference highlights the additional complexity posed by *restricted* and *unrestricted* instances, in which the optimal solution defines the starting time of the sequence, when compared to *tightly restricted* instances, in which the starting time is known to be 0. Also note that the similarity between *restricted* and *unrestricted* instances is a consequence of Property 5, which imposes a bound on the starting time and thus on the planning horizon.

Table 3 provides detailed results of the *Complete* algorithm for all of the instances. The table provides the average number of variables (in thousands), the average and maximum number of nodes in the branch-and-cut enumeration tree, the average and maximum running times (if the algorithm was stopped due to the maximum run time limit, the allotted running time of 14400 seconds is considered), and the average number of order-based inequalities used by the solver. Table 3 does not report optimality gaps or number of optimal solutions found, as these metrics are reported in Table 1.

The results support the efficiency of the proposed algorithm. Instances with more than a million binary variables (like $|\mathcal{J}| = 150$ and $t^{max} = 100$) can be solved within reduced running times (the average running time for large-size instances is around 30 minutes) while exploring very small search trees. For larger instances, the average running time of the complete algorithm is still smaller than the allotted time. Note that for the largest tested instances



(a) Average results according to the level of complexity (σ)



(b) Average results according to the type of due date (tightly restricted, restricted or unrestricted)

Fig. 6. Graphical representation of the average running time required by the proposed algorithm with instances of varying characteristics. Fig. 6(a) groups instances according to their σ, while Fig. 6(b) groups instances according to their due date type.

Table 3
Results of the proposed algorithm. For each number of jobs, column $\mathcal{J}$, and processing time range, column p.times, six values are reported: (1) average number of variables of the linear model, column variables (av.), reported in thousands of variables; (2) the average number of nodes of the branch-and-cut tree, column nodes (av.); (3) the maximum number of nodes of the branch-and-cut tree, column nodes (max.); (4) the average running time in seconds, column t(second) (av.); (5) the maximum running time in seconds, column t(second) (max); and (6) the average number of inequalities introduced by the precedence rules, column cuts.

$\|\mathcal{J}\|$	t^{max}	Variables	Nodes		t(second)		Cuts
		(av.)	(av.)	(max.)	(av.)	(max.)	(av.)
25	25	8.1	0.6	7	0.8	1.5	4.3
	50	16.0	0.7	9	2.3	6.2	7.7
	100	30.9	0.7	7	10.0	19.9	5.7
50	25	32.4	3.7	29	3.1	7.3	16.3
	50	64.4	2.7	19	11.4	31.1	19.4
	100	125.0	2.6	29	54.5	140.4	19.6
75	25	72.3	7.3	47	8.0	21.0	27.5
	50	142.4	8.3	113	36.8	206.9	36.0
	100	286.2	7.6	79	204.0	1273.2	44.9
100	25	131.8	15.9	307	21.2	110.4	37.4
	50	252.4	8.8	43	74.6	238.6	41.8
	100	502.8	12.3	105	540.8	1815.2	72.0
125	25	202.3	17.6	111	39.6	109.9	44.1
	50	398.1	13.8	97	149.3	397.3	59.1
	100	793.2	19.9	149	1075	4668.9	74.7
150	25	294.6	28.7	260	80.9	416.2	57.8
	50	578.3	24.3	281	326.5	2018.9	76.7
	100	1124.6	20.5	137	1901.8	9277.8	101.9
175	25	400.6	38.8	347	141.0	1209.4	62.5
	50	781.0	30.7	150	517.5	1891.6	81.9
	100	1543.5	28.5	191	3443.1	14400	125.5
200	25	518.3	45.8	304	207.1	1145.3	68.3
	50	1016.0	41.2	343	927.7	3725.6	98.0
	100	2027.3	31.0	146	5877.4	14400	122.6
225	25	654.5	54.4	324	332.3	1694.9	83.1
	50	1283.7	48.8	577	1361.4	14372.8	94.5
	100	2553.6	27.8	130	7539.3	14400	129.4
250	25	818.2	99.9	1218	888.4	14380.2	84.6
	50	1579.3	92.7	490	3281.4	14400	127.9
	100	3147.3	20.3	77	9352.1	14400	142.5
300	25	1165.8	105.2	808	1525.7	14400	102.8
	50	2299.4	79.1	314	5757.7	14400	126.5
	100	4546.4	9.1	39	13061.1	14400	154.1

$|\mathcal{J}| = 300$ and $t^{max} = 100$, the average running time shows the inability of the algorithm to obtain the optimal solution within the four-hour time limit, attesting the computational limits of the method.

Finally, we also verified the effect of limiting the search to the subset of integer starting times for each job. Using Property 7, it is possible to bound the maximum difference between the best solution that the time-indexed formulation may find and the optimal solution of the instance if other starting times were allowed. The gap introduced by the limitation is $100 \cdot 2 \sum_j w_j / UB$, in which UB corresponds to the optimal solution found by the time indexed formulation.

The maximum gap was found in an instance with 25 jobs, and its value is smaller than 0.03%. For the largest instances, 300 jobs, the average value is below 0.0004%. Therefore, we can conclude that the deviation introduced by considering only integer starting times is negligible (it is smaller than the default gap allowed by CPLEX).

6. Conclusions

In this paper we introduce and study a weighted version of the mean squared deviation problem (WMSDP), a classical scheduling problem motivated by the Just-in-Time (JIT) philosophy and objectives. In order to solve the problem we investigate several of its properties, including dominance properties, which define the necessary conditions that must be satisfied for a pair of jobs in any optimal schedule. In particular, we adapted properties from the previous work related to monotone increasing objective function and showed new dominance properties for a monotone decreasing function.

These relations are then used to define valid inequalities that are integrated within a branch-and-cut procedure. The resulting procedure improves upon the use of the standard solver, as it is shown by the results of an extensive computational experiment. The resulting algorithm is capable of solving instances with up to 300 jobs within reasonable running times, outperforming the ability of the off-the-shelf solver.

Please note that the use of a branch-and-cut method has some inherent disadvantages over other alternative methods, like column generation based methods, see (Desaulniers, Desrosiers, & Solomon, 2005). In such methods, a master problem with a subset of the variables is considered, and variables are added and removed from the master problem according to their reduced costs. While column generation methods contain an additional level of complexity (associated to the pricing operations and to the dynamic management of the variables), the resolution of smaller problems may lead to more efficient solution methods.

While the alternative method was considered and an initial implementation was developed, it was deemed inferior to the proposed procedure in terms of its ability to solve the instances used in the experimental results, and thus these results are not provided in this paper. The result may be caused, among other reasons by: (1) the ability of CPLEX to manage problems with the considered size (even when problems with over one million variables are solved to optimality); and (2) the high level of degeneracy in the master problem, which needs to be addressed with special methods (a job that was not conducted in our preliminary implementation). Nonetheless, for larger instance sizes, the use of a column generation method is to be considered as an option, since the bottleneck of the current implementation is the resolution of the root node of the IP, and the solver would face additional issues if larger linear relaxations were to be solved.

Furthermore, the development of a column generation framework would lead the paper astray from its main contributions, which are the identification of dominance properties among jobs for the problem under study, and the efficient use of these rules within a linear programming based framework. Along the same lines, we do not report the applicability of these rules within other enumeration based methods, like branch-and-bound and A*, see (Bansal et al., 2016; Höhn & Jacobs, 2012; Vásquez, 2015), or in alternative formulations, see (Höhn & Jacobs, 2012). We believe that the influence of such rules within enumerative procedures has been widely reported, yet these methods would fail to solve instances of the size reported in the present study, which are up to one order of magnitude larger than the instances considered in other studies, see (Höhn & Jacobs, 2012).

Finally, we would like to stress the generality of the developed dominance properties, and their applicability to other problems with monotone decreasing objective functions or, as in the case of the WMSDP, with both decreasing and increasing intervals of the objective function.

Acknowledgments

Óscar C. Vásquez has been partially supported by FONDECYT grant 11140566. This research was partially supported by the supercomputing infrastructure of the NLHPC (ECM-02) in which some preliminary computational tests were conducted. The authors would like to express their gratitude to the anonymous reviewers of this paper whose suggestions have greatly improved the quality of this work.

References

Badics, T., & Boros, E. (1998). Minimization of half-products. *Mathematics of Operations Research, 23*(3), 649–660.

Bagchi, U., Chang, Y., & Sullivan, R. (1987a). Minimizing absolute and squared deviations of completion times with different earliness and tardiness penalties and a common due date. *Naval Research Logistics (NRL), 34*(5), 739–751.

Bagchi, U., Sullivan, R., & Chang, Y. (1987b). Minimizing mean squared deviation of completion times about a common due date. *Management Science, 33*(7), 894–906.

Baker, K., & Scudder, G. (1990). Sequencing with earliness and tardiness penalties. a review. *Operations Research, 38*, 22–36.

Bansal, N., Dürr, C., Thang, N., & Vásquez, Ó. C. (2016). The local–global conjecture for scheduling with non-linear cost. *Journal of Scheduling*, 1–16.

Bigras, L., Gamache, M., & Savard, G. (2008). Time-indexed formulations and the total weighted tardiness problem. *INFORMS Journal on Computing, 20*, 133–142.

Cai, X. (1995). Minimization of agreeably weighted variance in single machine systems. *European Journal of Operational Research, 85*(3), 576–592.

Cheng, J., & Kubiak, W. (2005). A half-product based approximation scheme for agreeably weighted completion time variance. *European Journal of Operational Research, 162*(1), 45–54.

Desaulniers, G., Desrosiers, J., & Solomon, M. (2005). *Column generation*. Springer.

Dyer, M., & Wolsey, L. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics, 26*, 255–270.

Feldmann, M., & Biskup, D. (2003). Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. *Computers & Industrial Engineering, 44*(2), 307–323. doi:10.1016/S0360-8352(02)00181-X.

Graham, R., Lawler, E., Lenstra, J., & Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics, 5*(2), 287–326.

Hino, C. M., Ronconi, D. P., & Mendes, A. B. (2005). Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. *European Journal of Operational Research, 160*(1), 190–201. http://dx.doi.org/10.1016/j.ejor.2004.03.006.

Höhn, W., & Jacobs, T. (2012). An experimental and analytical study of order constraints for single machine scheduling with quadratic cost. In *Proceedings of the 14th workshop on algorithm engineering and experiments (alenex)* (pp. 103–117). SIAM.

Janiak, A., Janiak, W. A., Krysiak, T., & Kwiatkowski, T. (2015). A survey on scheduling problems with due windows. *European Journal of Operational Research, 242*(2), 347–357. http://dx.doi.org/10.1016/j.ejor.2014.09.043.

Jin, F., Gupta, J., Song, S., & Wu, C. (2010). Single machine scheduling with sequence-dependent family setups to minimize maximum lateness. *Journal of the Operational Research Society*, 1181–1189.

Jozefowska, J. (2007). *Just-in-time scheduling* (1st). Springer.

Kanet, J. J. (1981). Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly, 28*(4), 643–651.

Keha, A., Khowala, K., & Fowler, J. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering, 56*, 357–367.

Kubiak, W. (1993). Completion time variance minimization on a single machine is difficult. *Operations Research Letters, 14*(1), 49–59.

Liao, C. J., & Cheng, C. C. (2007). A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. *Computers & Industrial Engineering, 52*(4), 404–413. http://dx.doi.org/10.1016/j.cie.2007.01.004.

Merten, A., & Muller, M. (1972). Variance minimization in single machine scheduling. *Management Science, 18*, 518–528.

Mondal, S. (2002). Minimization of squared deviation of completion times about a common due date. *Computers & Operations Research, 29*(14), 2073–2085.

Nearchou, A. (2008). A differential evolution approach for the common due date early/tardy job scheduling problem. *Computers & Operations Research, 35*(4), 1329–1343. doi:10.1016/j.cor.2006.08.013.

Nessah, R., & Chu, C. (2010). A lower bound for weighted completion time variance. *European Journal of Operational Research, 207*(3), 1221–1226.

Pinedo, M. L. (2012). *Scheduling: Theory, algorithms and systems* (4th). Springer.

Queyranne, M., & Schulz, A. (1994). *Polyhedral approaches to machine scheduling* Preprint 408-1994, TUBerlin.

Sourd, F. (2009). New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS Journal on Computing, 21*(1), 167–175. doi:10.1287/ijoc.1080.0287.

Sousa, J., & Wolsey, L. (1992). A time indexed formulation of non-preemtive single machine scheduling problems. *Mathematical Programming, 54*, 353–367.

Srirangacharyulu, B., & Srinivasan, G. (2013). An exact algorithm to minimize mean squared deviation of job completion times about a common due date. *European Journal of Operational Research, 231*(3), 547–556.

Vásquez, Ó. C. (2015). For the airplane refueling problem local precedence implies global precedence. *Optimization Letters, 9*(4), 663–675.

Ventura, J., & Weng, M. (1995). Minimizing single-machine completion time variance. *Management Science, 41*(9), 1448–1455.

Vilà, M., & Pereira, J. (2013). Exact and heuristic procedures for single machine scheduling with quadratic earliness and tardiness penalties. *Computers & Operations Research, 40*, 1819–1828.

Wagner, B. J., Davis, D. J., & Kher, H. V. (2002). The production of several items in a single facility with linearly changing demand rates. *Decision Sciences, 33*(3), 317–346.

Waldspurger, C. A., & Weihl, W. E. (1994). Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of the 1st usenix conference on operating systems design and implementation* (p. 1). USENIX Association.

Weng, X., & Ventura, J. (1996). Scheduling about a given common due date to minimize mean squared deviation of completion times. *European Journal of Operational Research, 88*(2), 328–335.

Woeginger, G. (1999). An approximation scheme for minimizing agreeably weighted variance on a single machine. *INFORMS Journal on Computing, 11*(2), 211–216.

Ying, K. C. (2008). Minimizing earliness–tardiness penalties for common due date single-machine scheduling problems by a recovering beam search algorithm. *Computers & Industrial Engineering, 55*(2), 494–502. http://dx.doi.org/10.1016/j.cie.2008.01.008.