Computers & Operations Research 66 (2016) 141–152

Contents lists available at ScienceDirect

Computers & Operations Research

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

# The robust (minmax regret) single machine scheduling with interval processing times and total weighted completion time objective

Jordi Pereira [a,b,*]

[a] Faculty of Engineering and Sciences, Universidad Adolfo Ibáñez, Av. Padre Hurtado 750, Office C216, Viña del Mar, Chile
[b] Escola Universitària d'Enginyeria Tècnica Industrial de Barcelona, Universitat Politècnica de Catalunya. C. Comte Urgell, 187 1st. Floor, 08036 Barcelona, Spain

ARTICLE INFO

Available online 5 September 2015

*Keywords:*
Scheduling
Single machine
Uncertainty
Robust optimization
Branch-and-bound

ABSTRACT

Single machine scheduling is a classical optimization problem that depicts multiple real life systems in which a single resource (the machine) represents the whole system or the bottleneck operation of the system. In this paper we consider the problem under a weighted completion time performance metric in which the processing time of the tasks to perform (the jobs) are uncertain, but can only take values from closed intervals. The objective is then to find a solution that minimizes the maximum absolute regret for any possible realization of the processing times. We present an exact branch-and-bound method to solve the problem, and conduct a computational experiment to ascertain the possibilities and limitations of the proposed method. The results show that the algorithm is able to optimally solve instances of moderate size (25–40 jobs depending on the characteristics of the instance).



## 1. Introduction

Single machine scheduling is a classical problem thoroughly studied in the literature (see [9,12,16,17,26,33,34] among others). The objective of the problem is to ascertain the optimal sequence in which the tasks (jobs) that define the problem need to be performed in a machine, in order to optimize some performance measure. In addition to its theoretical interest, and its use on real life situations in which only one resource (machine) exists, the problem also models situations in which the production system is taken as a whole (hence, it is considered to be a unity) or there is a bottleneck operation that dictates the performance of the production system.

While the most studied formulations correspond to deterministic problems [9,33,34,17] that is, formulations in which all of the parameters are known and fixed, formulations with different characterizations of the uncertainty have also been studied [30,35,26,14,2,5].

Among the different alternatives to model uncertainty, we highlight the use of probability distribution functions to describe processing times [35,14,5], the representation of possible processing times through scenarios [10,18], and the use of intervals to define the processing time of the jobs [11,30,2].

In this paper we study an interval data minmax regret (IDMR) (see [11,1,30]) formulation of the single machine weighted sum of completion times problem (WCTP) [29,26]. In this formulation, the uncertainty on the processing time is represented using intervals, and the aim is to find a sequence that minimizes the maximum regret for all of the possible scenarios (that is, the sequence that minimizes the maximum absolute deviation between its solution and the optimal solution for each realization of processing times). Hence, the objective reflects a robust criterion which can be associated to a risk-averse decision maker that tries to hedge against the worst-case performance. This problem is referred as the Robust Weighted Completion Time Problem (RWCTP) throughout the paper.

### 1.1. Contributions of this work

While the robust minmax regret sum of completion times problem has been studied in the literature (see [11,19,37,20,15]); to the best of our knowledge its weighted counterpart has not been previously considered. We conjecture that the cause for this lack of research directed toward its resolution is the inapplicability of the classical results used to evaluate a solution in multiple IDMR problems (see, e.g. [1]), which leads to an absence of a viable method to evaluate the maximum regret of a given sequence.

In this paper we deal with the previous issue and we propose two different methods to evaluate the maximum regret of any given solution: one based on enumeration, and a second based on a dynamic programming (DP) formulation [6]. Both methods build

* Correspondence address: Faculty of Engineering and Sciences, Universidad Adolfo Ibáñez, Av. Padre Hurtado 750, Office C216, Viña del Mar, Chile.
*E-mail addresses:* jorge.pereira@uai.cl, jorge.pereira@upc.edu

http://dx.doi.org/10.1016/j.cor.2015.08.010
0305-0548/

upon the identification of a subset of scenarios containing the worst case scenario. Note that this approach differs from other studied IDMR problems in which the worst case scenario and/or the worst case alternative are obtained by solving some classical optimization problem.

Then, a branch-and-bound procedure using a novel lower bound method and a previously known dominance rule [30] is used to enumerate all of the solutions. The method is initialized using a heuristic which is shown to provide a 2-approximation, and if the lower bound or the dominance rule are not able to discard a solution during enumeration, the maximum regret of the said solution is obtained using one of the methods described above.

A computational experiment using instances from previous works on other weighted completion time problems with interval data [30,2] shows that the proposed branch-and-bound is capable of optimally solving instances with up to 25 jobs if the variability of processing times is high, and instances with up to 40 jobs if the variability is low. Furthermore, the applicability and the approximation ratio of a simple heuristic for IDMR problems, the midpoint scenario heuristic (see for example [19,15]) is also considered.

Note that the size of these instances is smaller than the size of the instances solvable for the unweighted case, in which larger instances can be solved to optimality (see [23,8]). Also note that the RWCTP includes an additional level of complexity, as the objective function depends not only on the absolute but also on the relative position of each pair of jobs. Nevertheless, the proposed branch-and-bound method is able to solve medium sized instances to optimality within limited running times. The results for the midpoint scenario heuristic in optimally solved instances show that it provides very reduced optimality gaps, and thus it appears as a good alternative if the solution of larger instances is required.

### 1.2. Paper outline

The rest of the paper is structured as follows. In Section 2 we describe the problem; we study the previous work on related problems and we put forward some notation used in the following sections. Section 3 is devoted to the contributions of this work and the proposed branch-and-bound algorithm. Section 4 sets forth a computational experiment to evaluate the efficacy of the algorithm proposed in Section 3. Finally, Section 5 puts forward the conclusions of this work. Two appendices are included: Appendix A is devoted to the approximation guarantees of the midpoint scenario heuristic for the RWCTP; and Appendix B, which can be found in the electronic Supplementary Material, provides additional results from the computational experiment reported in Section 4.

## 2. Problem settings

### 2.1. Description of the deterministic problem

The one machine scheduling problem with weighted sum of completion times (WCTP) is a classical problem, see for example Pinedo [26]. According to the notation proposed in Graham et al. [12], the problem corresponds to $1 \parallel \sum w_j C_j$, and it can be formally described as follows: $n$ independent jobs $J = \{J_1, J_2, \ldots, J_n\}$ have to be processed over a single machine that can handle at most one job at a time. Each job $J_i$, $1 \le i \le n$, is available for processing at time zero, requires a processing time $p_i > 0$ and has a positive weight $w_i > 0$. Let $C_i$ be the completion time of job $i$, then the objective is to obtain a processing order, a sequence, for the jobs such that the weighted completion time of the jobs, that is $\sum_{i \in J} w_i C_i$, is minimized.

This problem is easily solvable as the optimal sequence is obtained by sorting jobs according to their weighted shortest processing times (WSPT), see Smith [29].

### 2.2. Description of the robust (minmax regret) problem

The Robust (minmax regret) weighted sum of completion time problem (RWCTP) departs from the WCTP formulation by considering that the processing time $p_i$ of each of the jobs is unknown before scheduling but bounded between a given lower $p_i^-$ and an upper $p_i^+$ value. As the processing times are not known in advance, the objective is to minimize the maximum regret of the chosen sequence.

We proceed to formalize the objective and the concept of regret, as well as other terminology which will be used.

Let $S$ be the set of all of the permutations (sequences). For any given sequence $\sigma$ ($\sigma \in S$), we denote as $\sigma(k)$ the job occupying position $k$ in sequence $\sigma$ and as $\pi_\sigma(i)$ the position occupied by job $J_i$ in the sequence. Each possible realization of processing times is known as a scenario ($p \in P$). Throughout the paper we refer to a scenario in which either $p_i = p_i^-$ or $p_i = p_i^+$ holds for every job, as an extreme scenario. Let us also denote the completion time of job $i$ under scenario $p$ as $C_i(\sigma, p)$. Then, expression (1) provides the total weighted completion time of sequence $\sigma$

$$\sum_{i=1}^{n} w_i \cdot C_i(\sigma, p) \tag{1}$$

Let $\sigma_p^*$ be the sequence that minimizes (1) for a given scenario $p$. Then, the regret of a sequence $\sigma$ under scenario $p$ corresponds to $Z_p(\sigma)$

$$Z_p(\sigma) = \sum_{i=1}^{n} w_i \cdot C_i(\sigma, p) - \sum_{i=1}^{n} w_i \cdot C_i(\sigma_p^*, p) \tag{2}$$

The scenario $p \in P$ that maximizes (2) for any given sequence $\sigma$ is known as the worst-case scenario of $\sigma$; its regret is denoted by $Z(\sigma)$; and its optimal sequence $\sigma_p^*$ as the worst-case alternative of $\sigma$

$$Z(\sigma) = \max_{p \in P} \left( \sum_{i=1}^{n} w_i C_i(\sigma, p) - \sum_{i=1}^{n} w_i C_i(\sigma_p^*, p) \right) \tag{3}$$

The objective of the RWCTP is to obtain the sequence that minimizes the maximum regret for all possible scenarios, that is, the sequence $\sigma$ that minimizes $Z$

$$Z = \min_{\sigma \in S} Z(\sigma). \tag{4}$$

Note that optimally solving (4) does not only imply finding a sequence $\sigma$ that minimizes (4), but also finding its worst case scenario, a realization of processing times $p$, and its worst case alternative $\sigma_p^*$, see (2).

### 2.3. Literature review

Several IDRM scheduling problems have been studied in the literature. The robust total completion time problem, the unweighted counterpart of the RWCTP, was considered in Daniels and Kouvelis [11], Kouvelis and Yu [19], Lebedev and Averbakh [20], Yang and Yu [37], and Kasperski and Zielinski [15].

Daniels and Kouvelis [11] and Kouvelis and Yu [19] show that the absolute regret problem is NP-hard, and propose a branch-and-bound for its resolution. The method uses a surrogate relaxation bound and it is able to solve instances with up to 20 jobs within short running times (less than 10 min). These results

were improved in Montemanni [23] using a mixed integer linear programming approach.

In Yang and Yu [37] the absolute, maximum and relative regret versions are examined and all three of the problems are shown to be NP-hard. The authors also propose a dynamic programming method, a greedy heuristic, and a surrogate heuristic to solve the problem, which is also tested in instances with up to 20 jobs.

In Lebedev and Averbakh [20], the authors consider the complexity of special cases of the problem, and identify some polynomially solvable cases.

More recently, Kasperski and Zielinski [15] offered a 2-approximation heuristic for the problem and some related cases, like the case with additional precedence constraints among jobs.

Other IDMR scheduling problems have also been considered, such as the maximum weighted tardiness problem [3] or the permutation flowshop problem with two jobs [4], which were shown to be polynomially solvable. Other studied problems, like the unrelated parallel machine with total completion time [36,8,28], the single machine with uncertain setup times [21], or the single machine with uncertain setup times and processing times [22], are shown to be NP-hard because of their relationship with other optimization problems.

In this later set of works, the solution methods make use of this relationship to provide a solution for the problem (e.g. [8]) or to derive a method to evaluate the maximum regret of a solution, which is then used in specially tailored methods to obtain a solution for the problem in hand (e.g. [28]). Note that while not strictly related to the unweighted case, Conde [8] documents the successful application of an MIP formulation for instances of the IDMR version of the unrelated unweighted parallel machine problem, which includes the single machine problem as a special case.

We would also like to note that other objectives for the problem with WCTP-derived objective and interval data uncertainty on the processing times have been studied. The literature has studied the problem under mean weighted completion times minimization objective [30,2] and the "stability box" maximization objective [31,32].

In Sotskov et al. [30], dominance relations among jobs for the mean weighted completion time problems are investigated. These relations impose a relative order among jobs, and when they fail to provide a complete ordering, two heuristics are proposed to find such an ordering. These rules are not only useful for the mean weighted completion time objective but also for the IDMR case under study in this paper.

In Allahverdi et al. [2], several new heuristics are proposed for the mean weighted completion time objective. These heuristic are then compared with those proposed in Sotskov et al. [30] in a large instance set featuring instances with multiple characteristics.

In Sotskov and Lai [31] and Sotskov et al. [32], the stability approach is investigated. The approach is based on the maximization of the volume of a "stability box" for a given sequence, which is defined as the range of values for the processing times of the jobs in which the proposed sequence maintains its optimality conditions. In Sotskov and Lai [31], the authors show that the problem is solvable in $O(n^2)$ time. Furthermore, they investigate the regret between the solution provided by the stability box method and the optimal solution for the scenarios in which the processing times corresponds to the lower, upper and midpoint (the average between the lower and upper bound) values.

### 2.4. Dominance rules

As stated in Section 2.1, the sequence minimizing the WCTP is provided by the weighted shortest processing time first (WSPT) rule, see Smith [29], which is a generalization of the shortest processing time first (SPT) rule to solve the unweighted case.

This rule can be described as a dominance relation among jobs, stating that if condition (5) holds, then job $J_i$ is to be scheduled before job $J_{i'}$ in any optimal sequence (that is, a precedence relation). These precedence relations define a complete ordering of the jobs, and as such provide the optimal sequence

$$\frac{w_i}{p_i} \geq \frac{w_{i'}}{p_{i'}} \tag{5}$$

In Sotskov et al. [30], the authors provide a generalized version of this rule, considering cases in which this dominance relation holds for all of the possible scenarios.

**Theorem 1.** *For the WCTP, job $J_i$ dominates job $J_{i'}$ if and only if (6) holds*

$$\frac{w_i}{p_i^+} \geq \frac{w_{i'}}{p_{i'}^-} \tag{6}$$

**Proof.** See Sotskov et al. [30].□

Let us note that the proof is based on verifying that condition (5) holds for all of the scenarios and their respective realizations of processing times for jobs $J_i$ and $J_{i'}$. Also note that it is possible that two jobs dominate each other. In such a case, we can arbitrarily choose one dominance rule to enforce without modifying optimality.

## 3. The robust (minmax regret) weighted completion time problem with interval processing times

This section puts forward the contributions of this work. As none of the results in the literature can be used to define a method to obtain the worst-case scenario for a given sequence $\sigma$, the topic is discussed in Section 3.1. First we prove that only extreme scenarios need to be considered, and then we provide an alternative to the enumeration of all of the extreme scenarios based on the enumeration of the worst-case alternatives.

The remaining sections deal with the method to obtain a sequence minimizing the maximum regret. Section 3.2 provides a lower bound to detect partial sequences that cannot improve the best-known solution, and Section 3.3 details the branching rule and other features of the proposed branch-and-bound method.

### 3.1. Determination of worst-case scenarios for the RWCTP

In order to evaluate the regret of a given sequence, we need to obtain its worst-case scenario and its worst-case alternative. In this section we describe two possible procedures based on (1) reducing the set of possible worst-case scenarios to the subset of extreme scenarios, or (2) reducing the set of possible worst-case alternatives to a subset of sequences.

In order to present the proposed methods, we need to introduce a preliminary result.

**Theorem 2.** *There is an extreme scenario p that maximizes (7) for any given pair of sequences $\sigma$, $\sigma'$*

$$\sum_{i=1}^{n} w_i \cdot C_i(\sigma, p) - \sum_{i=1}^{n} w_i \cdot C_i(\sigma', p). \tag{7}$$

**Proof.** Suppose that the opposite is true, and let $J_i$ be a job whose processing time realization $p_i$ is not equal to its lower, $p_i^-$, or upper, $p_i^+$, limit.

First, rewrite the terms of (1) as (8). Then, separate the terms in which $p_i$ participates in (7), see (9)

$$\sum_{i=1}^{n} w_i \sum_{\substack{1\le j\le n:\\ \pi_\sigma(j)\le\pi_\sigma(i)}} p_j \tag{8}$$

$$\sum_{j=\pi_\sigma(i)}^{n} w_{\sigma(j)}\cdot p_i - \sum_{j=\pi_{\sigma'}(i)}^{n} w_{\sigma'(j)}\cdot p_i = \left(\sum_{j=\pi_\sigma(i)}^{n} w_{\sigma(j)} - \sum_{j=\pi_{\sigma'}(i)}^{n} w_{\sigma'(j)}\right)\cdot p_i. \tag{9}$$

Note that (9) is linear to $p_i$. Hence, there is always a realization of $p_i$ maximizing (7) that corresponds to $p_i^-$ or $p_i^+$.□

**Remark 1.** Eq. (9) can also be used to identify if $p_i$ should take its maximum (if the slope is positive) or minimum value (if the slope is negative) in the scenario maximizing (7). This remark is later used on the DP method to obtain the worst-case scenario for any given alternative.

**Corollary 1.** *If $\sigma'$ is the worst-case alternative for $\sigma$, then scenario $p$ maximizing (9) is a worst-case scenario for $\sigma$. According to Theorem 2, $p$ is an extreme scenario. Hence, we conclude that only extreme scenarios need to be considered as possible worst-case scenarios for any sequence. Note that this result is a well known result for IDMR problems with linear objective function, which is not the case in the problem under study.*

A straightforward application of Corollary 1 leads to an initial method to evaluate the maximum regret. Let $\mathcal{P}\subset P$ be the subset of extreme scenarios. In order to obtain the regret of any given sequence $\sigma$, we can traverse all of the scenarios; obtain their optimal sequence $\sigma_p^*$; evaluate the objective value of $\sigma$ and $\sigma_p^*$; and keep track of the maximum difference among these two values (the regret of $\sigma$ in the scenario).

Note that the cardinality of $\mathcal{P}$ is $2^n$. Hence, the method described above is only applicable if the number of jobs is small, see the computational experiment reported in Section 4 and the Supplementary online Material for further details.

An alternative to enumerating scenarios to obtain the worst-case scenario, is to enumerate sequences in order to obtain the worst-case alternative, and then apply Theorem 2 to obtain the worst-case scenario.

We structure the enumeration of sequences using a dynamic program formulation (DP) (see [6,13]). Note that the examination of Eq. (9) indicates that the contribution of a job to the regret between two sequences only depends on the processing time of the job (which needs to be selected in such a way that Eq. (9) is maximized); and the weight of the jobs scheduled after the said job in $\sigma$ and $\sigma'$. As $\sigma$ is known in advance, the maximum regret can be obtained as the sum of the contribution of each job to the total regret, which has the optimal substructure required to solve a problem using a DP formulation.

Furthermore, since the worst-case alternative is the optimal sequence for the worst-case scenario, the search can be limited to sequences that are potentially optimal in some scenario. Hence, the proposed formulation uses the dominance relations indicated in Section 2.4 in order to impose precedence constraints among jobs, and thus trying to avoid the enumeration of schedules that are not optimal in any scenario (see for early applications of DP approaches to sequencing problems with precedence constraints [13,27]).

We proceed to introduce the notation required to describe the recurrence relation of the DP model: let $\mathcal{J}$ be a subset of the jobs, $\mathcal{J}\subseteq J$. We say that $\mathcal{J}$ is a non-dominated subset if, for every job $J_i\in\mathcal{J}$, all the predecessors of job $J_i$ according to the dominance rule described in Section 2.4 are also in $\mathcal{J}$.

Let $c(\mathcal{J},J_i)$ be the contribution to the regret if job $J_i$ is assigned to the $|\mathcal{J}|+1$-th position in the sequence with the jobs in $\mathcal{J}$ sequenced in the $|\mathcal{J}|$-th first positions, see Eq. (10); and let $C(\mathcal{J})$ be the maximum regret of the jobs sequenced in $\mathcal{J}$

$$c(\mathcal{J},J_i)=\max\left\{\begin{array}{l}\left(\sum_{j=\pi_\sigma(i)}^{n} w_{\sigma(j)} - \sum_{j\in J-\mathcal{J}} w_j\right)\cdot p_i^-,\\ \left(\sum_{j=\pi_\sigma(i)}^{n} w_{\sigma(j)} - \sum_{j\in J-\mathcal{J}} w_j\right)\cdot p_i^+\end{array}\right\}. \tag{10}$$

Then, the recurrence relations correspond to Eqs. (11) and (12); $C(J)$ provides the maximum regret of sequence $\sigma$. Furthermore, the worst-case alternative $\sigma'$ is obtained by reconstructing the decisions taken to reach a final state, subset $\mathcal{J}=J$, from an initial state, subset $\mathcal{J}=\emptyset$, and the worst-case scenario $p$ is to be obtained by considering the processing time (be it $p_i^-$ or $p_i^+$) used in Eq. (10)

$$C(\emptyset)=0, \tag{11}$$

$$C(\mathcal{J})=\max_{\forall J_i\in\mathcal{J}}\{C(\mathcal{J}\backslash J_i)+c(\mathcal{J}\backslash J_i,J_i)\}. \tag{12}$$

A bound on the number of subsets $\mathcal{J}$ that need to be considered is $\sum_{i=1}^{n}\binom{n}{i}$; but the number of feasible subsets is reduced by the introduction of dominance relations. Hence, the applicability of the DP formulation is limited by the number of precedence constraints introduced by the dominance relation. The computational experiment (see Section 4) shows that the dominance rule enables the evaluation of larger instances than the extreme scenario method.

### 3.2. Lower bounds for the RWCTP

In order to provide a method to avoid the enumeration of all of the possible sequences and their evaluation by means of the procedures proposed in Section 3.1, we propose to use a lower bound on the maximum regret for any given subsequence. The method is based on considering a scenario, its optimal sequence and the optimal continuation of the subsequence under evaluation.

Let $\sigma^m$ be a subsequence with $m$ sequenced jobs. Using the arguments provided in Smith [29], it is straightforward to prove that for any given scenario $p$ ( $p\in P$) the continuation of subsequence $\sigma^m$ that minimizes the total weighted completion time under scenario $p$ corresponds to sequencing all of the jobs not in $\sigma^m$ in its corresponding WSPT order. Let $C(\sigma)$ be the total weighted completion time of the subsequence and its optimal continuation, and let $C(\sigma_p^*)$ be total weighted completion time of the optimal sequence for scenario $p$. Then $C(\sigma)-C(\sigma_p^*)$ is a lower bound on $Z(\sigma)$ for any possible continuation of subsequence $\sigma^m$.

Let us remark that the previous approach incurs in two issues, which need to be discussed.

First, the lower bound provided by the previous approach when no jobs have been scheduled is 0. The issue was already faced by a branch-and-bound method for the unweighted robust version of the problem (see [11]). In the aforementioned work, the authors propose a lower bound on the regret which uses a surrogate relaxation of multiple scenarios. This relaxation provides a non-trivial lower bound when no jobs have been scheduled. Unfortunately, this method requires the identification of worst-case scenarios for the solution provided by each iteration of the surrogate relaxation. These scenarios are then included in the surrogate relaxation until no further improvement is possible. As the identification of worst-case scenarios has been deemed as a difficult task (see Section 3.1), we ruled out a similar approach. Consequently, we are to obtain worse lower bounds in the initial stages of the search, in exchange for a faster calculation of these bounds.

Second, the problem of finding the scenario that maximizes the lower bound for any given $\sigma^m$ is a difficult problem by itself. Consequently, we propose a heuristic followed by a local search in order to reach a scenario that maximizes the regret.

The heuristic to obtain a regret maximizing scenario is based on the observation that, for any given sequence $\sigma$, the realization of processing times for a given job $J_i$ in the worst-case scenario tends to take its upper value, $p_i^+$, if the job is sequenced in the early positions of $\sigma$, and its lower value, $p_i^-$, if the job is sequenced in the later positions of $\sigma$.

Based on the premise that already sequenced jobs were sequenced too early, and non-sequenced jobs are to be sequenced too late, the scenario is constructed as follows: if job $J_i$ belongs to subsequence $\sigma^m$, then realization $p_i$ is $p_i^+$, and $p_i^-$ otherwise. The complexity of obtaining the processing times, the worst-case alternative, the continuation of $\sigma^m$, and the evaluation of the regret is dominated by the ordering operations required to schedule jobs in WSPT order. Hence, the total time complexity of the heuristic is $O(n \log n)$.

The local search defines the neighborhood of a scenario as the scenarios obtained by changing the realization $p_i$ of a job $J_i$ from its lower (upper) limit to its upper (lower) limit. The exploration of the neighborhood is organized in a first descent fashion (that is, whenever an improving exchange is found, it is automatically implemented).

The ordering in which the jobs are considered affects the behavior of the local search. Therefore, and in order to reduce the effect of a single order, jobs are considered in the ordering given by the worst-case alternative. The time complexity of each evaluation is $O(n)$. To reach this complexity, we have to take into account that the move only modifies the order of job $J_i$ in both sequences, and that the new positions of job $J_i$ can be obtained in $O(n)$ time. As the evaluation is also performed in $O(n)$ time, the total complexity is $O(n)$.

The neighbor operator is repeated until no further improvement is possible (that is, we have a locally optimal scenario with respect to its neighborhood), or the regret provided by any scenario is greater than the maximum regret of the best-known solution (that is, the lower bound provided by the scenario allows us to fathom the subsequence).

### 3.3. A branch-and-bound method for the minmax regret weighted completion time problem

In order to obtain a solution method for the RWCTP, the lower bound, see Section 3.2, and the dominance rule proposed in Sotskov et al. [30], see Section 2.4, are integrated into a branch-and-bound procedure.

The proposed branch-and-bound constructs a sequence by consecutively appending jobs to a subsequence. A branch is defined by the selection of the next job to perform. In order to fulfil the dominance rules, these dominance rules are considered as precedence relations, and only those jobs in which all of its predecessors already belong the subsequence under construction are considered as candidates for branching.

Branching is performed using a depth-first strategy and the lower bound on the maximum regret is calculated as in Section 3.2. Whenever a complete solution is found, one of the evaluation methods proposed in Section 3.1 is applied, and the best known solution is changed if it improves upon the best-known solution.

The branch-and-bound is initialized with a heuristic upper bound. Based on the results of previous methods for other IDMR problems, the midpoint scenario heuristic is chosen see for multiple examples of use of the midpoint scenario heuristic for robust minmax regret optimization problems [11,15,7,8,24,25,28].

The midpoint scenario corresponds to the optimal solution for a scenario in which the processing time of each job is equal to $p_i = (p_i^- + p_i^+)/2$ for every job. The sequence for the midpoint scenario is then evaluated and its maximum regret is used as the initial upper bound. Note that this heuristic also provides a 2-approximation guarantee, see Appendix A.

Finally, during preliminary tests with the implemented method, we verified that the time required by the branch-and-bound was dependent on the ordering of the jobs in the instance. Therefore, jobs are reordered according to their WSPT order for the midpoint scenario. Note that this reordering affects which branching decisions are explored first. The proposed ordering gives priority to solutions that do not depart from the midpoint scenario, as its deviation from the optimal solution in many other IDMR is known to be small [24,25,8].

## 4. Computational experiment

The computational experiment tests (a) the applicability of the different proposed methods to evaluate the maximum regret of a solution; (b) the efficiency of the proposed branch-and-bound method under varying characteristics of the instances; (c) the quality of the proposed lower bound; (d) the performance of the midpoint scenario heuristic; and (e) the effect of the characteristics of the instances on the maximum regret. The following sections give some implementation details, see Section 4.1, describe the instances used for comparison purposes, see Section 4.2, and describe the results of the computational experiment, see Section 4.3. Additional results are provided in the electronic Supplementary Material Appendix B.

### 4.1. Implementation details

The algorithm was coded in C++ and compiled using the GNU GCC compiler 4.9.0. All of the computational experiments were run on an Intel Xeon machine with an 8-core 2.66 GHZ CPU and 32-GB RAM running the Linux operating system. While the computer has eight cores, the code does not make use of any form of parallelism, and consequently only one core is used in the computational experiments. Furthermore, two run time limits (600 and 3600 s) per instance were considered. The 600 s time limit was chosen in accordance to the running times reported in [11], in which the time required to solve the largest considered instances (20 jobs) is approximately 600 s. The 3600 s time limit considers the benefits of increasing the running time, and to verify if the 600 s time limit is too restrictive for the performance of the algorithm.

### 4.2. Description of the instances

The literature reports two different approaches to generate instances with interval data and total weighted completion time objective [30,2]. These instances were generated in order to evaluate different versions of WCTP with interval data. We test our method with instances created using both generators, and hence two different instance sets were constructed. Below, we describe the characteristics of the instances of each set.

*Sotskov instances*: These instances are generated using the procedure proposed in Sotskov et al. [30]. For each job, we generate the weight from a uniform distribution, $w_i = U[1, 50]$. In order to generate the lower and upper bounds on the processing times, we first generate a central value from a uniform distribution $C_i = U[1, 200]$, and then the lower ($p_i^-$) and upper bounds ($p_i^+$) are generated as: $p_i^- = C_i \cdot (1 - \delta/100)$ and $p_i^+ = C_i \cdot (1 + \delta/100)$. $\delta$ is a

parameter that controls the variability of processing times. The parameter $\delta$ is set to 1, 2, 5, 10, 15, 25, 50, 75 and 100.

*Allahverdi instances*: These instances are generated using the procedure proposed in Allahverdi et al. [2]. For each job, we generate the weight from a uniform distribution, $w_i = U[1, 50]$. The upper bound is generated from a uniform distribution, $p_i^+ = U[1, 100]$ and the lower bound is generated from a uniform distribution $p_i^- = U[p_i^+ - D, p_i^+]$, in which $D$ is a parameter that controls the variability of processing times. Note that $p_i^-$ may be a negative value, therefore if $p_i^- \leq 1$ we set $p_i^- = 1$. As in [2], the parameter $D$ is set to 10, 20, 30, 40, 50 and 60.

In addition to the variability parameter and the instance generator, we consider instances with a different number of jobs, (10, 15, 20, 25, 30, 35 and 40). For each combination of instance generator, variability parameter (be it $\delta$ or $D$) and number of jobs, 50 different instances are constructed for a total of 5250 instances.

### 4.3. Results

#### 4.3.1. Results for the maximum regret evaluation methods

As previously noted, the methods to evaluate the maximum regret of a sequence are of enumerative nature, and it is to be expected that their applicability is dependent on the instance size. Furthermore, the characteristics of the instance may also modify its behavior. Hence, Table 1 explores the limits of applicability of the proposed evaluation methods.

Table 1 reports the time required to evaluate the solution provided by the midpoint scenario for different instance sizes and variability of processing times. For each instance size and variability parameter, the table reports the average (among all of the instances) and the maximum time (reported in parenthesis) required for the evaluation of the regret of the midpoint scenario using the dynamic programming approach.

An additional column reporting the time required to evaluate the maximum regret through the enumeration of scenarios is also provided. Note that if the set of scenarios is enumerated, the running time does not depend on the variability parameter, and thus the required time is only affected by the instance size. Furthermore, no maximum times are reported as the differences between the required computing time among different instances with the same number of jobs is minimal.

Table 1 only reports results for instances with 15 or more jobs and for variability level 25 or larger, as instances with a smaller number of jobs or lower variability can be easily evaluated within a fraction of a second. Moreover, if a time limit of 600 s is reached without obtaining the maximum regret of the solution, the evaluation of the regret is deemed as unpractical, the procedure is stopped and no result is provided (these results are marked as >600).

The results show that the enumeration of the worst-case alternatives using the DP approach is to be the preferred option, unless the level of variability among processing times is extremely high ( $\delta = 100$). The difference between the average and the maximum required time shows a high variability among different instances with similar characteristics. We investigated individual results and detected a strong correlation between the total time required and the number of dominance relations among the jobs in each instance (that is, for any given number of jobs $n$, and variability level $\delta$, there is a strong negative correlation between the number of dominance relations in any given instance and the time required to evaluate the instance). A similar table to Table 1 with the results for the *Allahverdi* instances is provided in the Supplementary Material. For the *Allahverdi* instances all of the tested variability levels and instance sizes were solved within the imposed time limit, with smaller running times than those reported for the *Sotskov* instances.

Nonetheless, the DP approach clearly outperforms the enumeration of scenarios, and it is to be the preferred method to evaluate the regret. Therefore, we limit the remaining analysis to the use of the DP regret evaluation method, and we conclude that instances with up to $n=40$ jobs and moderate levels of variability ($\delta = 50$, $D = 60$) can be evaluated within the imposed time limits.

As previously noted, the ability of the DP method to improve upon the enumeration of scenarios depends on the variability of the processing times, which directly affects the number of dominance relations detected by the dominance rule, see Section 2.4. Table 2 provides the average ratio of dominance relations between tasks with respect to the total possible number of dominance relations. This percentage is calculated using (13), where $A$ is the number of dominance relations that the procedure detects, and the denominator indicates the maximum number of dominance rules that the instance may contain

$$100 \cdot \frac{A}{n \cdot (n-1)/2} \tag{13}$$

The results from Table 2 confirm that the performance of the DP is correlated to the number of dominance relations. While the

**Table 1**
Average and maximum (in parenthesis) running times required to evaluate the maximum regret of the midpoint scenario heuristic solution for the *Sotskov* instance set. For each instance size (row) and variability value $\delta$ (column), we report the time (in seconds) required by the enumeration of alternatives. A final column (column scenarios) reports the time required by the enumeration of scenarios. If the required running time exceeded the 600 s, the problem is deemed as unsolved (>600 is reported).

| $n \setminus \delta$ | 25 | 50 | 75 | 100 | Scenarios |
|---|---|---|---|---|---|
| 15 | 0 | 0 | 0.00 (0.04) | 0.04 (0.08) | 0.002 |
| 20 | 0 | 0.00 (0.04) | 0.08 (0.73) | 2.36 (2.94) | 0.05 |
| 25 | 0 | 0.05 (0.85) | 1.30 (8.26) | 155.79 (165.2) | 1.82 |
| 30 | 0 | 0.27 (2.01) | 24.25 (33.18) | >600 | 63.15 |
| 35 | 0.01 (0.01) | 3.10 (38.02) | >600 | >600 | >600 |
| 40 | 0.07 (0.74) | 22.03 (326.46) | >600 | >600 | >600 |

**Table 2**
Average percentage of dominance relations detected by the method proposed in Section 2.4 for the *Sotskov* and *Allahverdi* instance set. For each instance size (row) and variability value $\delta$ or $D$ (column), the percentage, calculated using (13), is reported.

| $n$ | Sotskov | | | | | | | | | Allahverdi | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 5 | 10 | 15 | 25 | 50 | 75 | 100 | 10 | 20 | 30 | 40 | 50 | 60 |
| 10 | 99.2 | 98.0 | 95.5 | 89.7 | 84.6 | 75.5 | 52.4 | 25.3 | 0.0 | 90.7 | 84.8 | 79.8 | 72.0 | 65.2 | 57.6 |
| 15 | 98.9 | 97.7 | 95.2 | 90.0 | 85.4 | 73.7 | 48.1 | 23.8 | 0.0 | 91.9 | 84.7 | 79.2 | 69.6 | 68.7 | 59.6 |
| 20 | 98.9 | 97.9 | 94.9 | 89.3 | 84.1 | 75.9 | 48.9 | 25.3 | 0.0 | 92.7 | 85.6 | 80.5 | 69.9 | 64.9 | 61.2 |
| 25 | 99.0 | 98.0 | 94.5 | 89.9 | 84.7 | 74.4 | 51.4 | 25.6 | 0.0 | 92.8 | 84.5 | 79.7 | 71.8 | 66.1 | 59.3 |
| 30 | 99.0 | 98.0 | 94.3 | 89.4 | 83.4 | 74.1 | 50.4 | 26.9 | 0.0 | 92.1 | 86.4 | 78.2 | 73.4 | 68.8 | 60.0 |
| 35 | 98.9 | 98.0 | 95.0 | 89.7 | 84.4 | 74.6 | 48.7 | 26.2 | 0.0 | 92.8 | 84.8 | 77.5 | 72.2 | 65.5 | 58.0 |
| 40 | 98.9 | 98.1 | 94.7 | 89.5 | 84.1 | 74.2 | 48.5 | 25.8 | 0.0 | 92.7 | 85.6 | 79.0 | 71.3 | 66.4 | 59.9 |

ratio of dominance relations remains relatively constant for different number of jobs, the DP also depends on the number of jobs, and thus it is to be expected that the applicability of the branch-and-bound method will depend on both the variability and the instance size.

Since there are some instances for which the evaluation of the maximum regret for the midpoint scenario was deemed infeasible (instances for which the imposed time limit of 600 s was insufficient to reach the maximum regret), their corresponding results for the branch-and-bound or the heuristic procedures are left blank (indicating that no result is available).

#### 4.3.2. Results for the RWCTP

Tables 3–8 provide the results of the branch-and-bound method described in Section 3.3 to obtain the sequence that minimizes the maximum regret with a running time limit of 600 s. In order to evaluate the proposed branch-and-bound method, we provide the following metrics: (1) the number of optimal solutions found within the imposed time limit (Table 3); (2) the average as well as the standard deviation of the time required to solve these instances to optimality (Tables 4 and 5); (3) the average number of explored nodes in the branch-and-bound before verifying optimality (Tables 6 and 7); and (4) the average number of nodes pruned by the lower bound proposed in Section 3.2 (Table 8).

We do not report optimality gaps (difference between the best upper and lower bounds) because the proposed lower bound does not provide a bound unless part of the solution sequence has already been constructed. Since the exploration of the branch-and-bound tree follows a depth-first rule, there are several cases in which some of the descendants of the root node are never explored and the trivial lower bound (the bound equals 0 for the root node) is not improved. Therefore, the number of nodes fathomed by the lower bound constitutes the only available measure of the performance of the proposed lower bound. Note that the same set of results, under a 3600 s time limit, are reported in the Supplementary Material, and the differences between the results reported with both time limits are discussed in this Section.

Table 3 reports the number of verified optimal solutions (out of 50 instances) that the branch-and-bound algorithm is able to solve to optimality for the *Allahverdi* and *Sotskov* instance set. The results show that, for small instance sizes (up to 20 jobs), the method is capable of reaching and verifying optimality within reduced running times for any level of variability on the processing time intervals. For medium sized instances (25 and 30 jobs), the algorithm is capable of solving all of the instances with low variability intervals, but it fails to verify optimality in most of the instances with high variability. For larger instances (35 and 40 jobs), the capability of optimally solving the problem is limited to those instances with low variability. Note that these instances already pose a problem for the scenario evaluation procedure, which is not capable of evaluating some of these instances.

If we compare the results with a 600 s time limit with the results with a 3600 s time limit, a similar trend is observed. The branch-and-bound method is able to reach and verify optimality for 226 additional instances, including most of the instances with 25 jobs (with the exception of the *Sotskov* instances with high variability, $\delta \geq 75$); the majority of the additional optimal solutions correspond to instances for which different instances with the same characteristic were already solved within the 600 s time limit. Consequently, we consider that the resolution of instances with larger number of jobs or larger variability is not to be expected by increasing the running time, and algorithmic improvements would be required in order to optimally tackle larger instances.

Tables 4 and 5 report the average and the standard deviation of the running time required to reach optimality. Consequently, if no solution is found within the allotted time (600 s), no value is reported, and if only one instance is solved within the time limit, the standard deviation does not apply. Both tables, as well as their corresponding tables for a 3600 s time limit provided in the Supplementary Material, indicate a high running time variability among instances from the same group.

The aforementioned behavior is frequent in branch-and-bound algorithms, as the branching order, the ability of the lower bound to prune early areas of the search space, and the specific characteristics of some instances (like the dominance relations) have a great influence on the branch-and-bound. Nonetheless, for most of the combinations of instance characteristics (instance generator, number of jobs and variability of processing times) the expected running time is below the imposed time limit and only for a small subset of instances the algorithm fails to reach optimality (or takes much longer than expected), which leads to large standard deviations.

The results also show that for small instance sizes (up to 20 jobs), with the exception of the *Sotskov* instances with $\delta = 100$, and any level of variability, the algorithm is capable of reaching

**Table 3**
Results of the branch-and-bound procedure for the *Sotskov* and the *Allahverdi* instance sets. For each instance size (row) and variability value, $\delta$ or $D$, (column), the number of verified optimal solutions (out of 50 instances) in which the optimal solution was found within the running time limit (600 s) is reported.

| $n$ | Sotskov | | | | | | | | | Allahverdi | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 5 | 10 | 15 | 25 | 50 | 75 | 100 | 10 | 20 | 30 | 40 | 50 | 60 |
| 10 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| 15 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| 20 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 47 | 29 | 50 | 50 | 50 | 50 | 50 | 50 |
| 25 | 50 | 50 | 50 | 50 | 50 | 48 | 41 | 15 | 0 | 50 | 50 | 50 | 49 | 45 | 49 |
| 30 | 50 | 50 | 50 | 50 | 48 | 35 | 3 | 0 | | 50 | 49 | 44 | 40 | 23 | 16 |
| 35 | 50 | 50 | 50 | 48 | 27 | 3 | 0 | | | 50 | 38 | 21 | 7 | 1 | 0 |
| 40 | 50 | 50 | 50 | 24 | 3 | 0 | 0 | | | 50 | 21 | 2 | 1 | 0 | 0 |

**Table 4**
Results of the branch-and-bound procedure for the *Sotskov* instance set. For each instance size (row) and variability value $\delta$ (column), two values are reported: the average time (av.) and the standard deviation (sd.) among the different instances for which the algorithm obtains the optimal solution within the imposed (600 s) time limit. We do not report results for instances with $\delta = 1$, $\delta = 2$, as all of the instances are solved within a fraction of a second.

| $\delta$ | 5 | | 10 | | 15 | | 25 | | 50 | | 75 | | 100 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | av. | sd. | av. | sd. | av. | sd. | av. | sd. | av. | sd. | av. | sd. | av. | sd. |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0.1 | 0.2 | 0.4 | 0.5 | 12.5 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0.1 | 0.2 | 0.3 | 0.5 | 6.9 | 15.8 | 28.7 | 48.8 | 155 | 153.2 |
| 25 | 0 | 0 | 0.6 | 1.7 | 2.6 | 7.3 | 20.9 | 54.9 | 121.8 | 153.7 | 261.8 | 162.8 | | |
| 30 | 1.5 | 4.5 | 8.2 | 16.6 | 60.1 | 114.8 | 171.8 | 170 | 281.7 | 58 | | | | |
| 35 | 2.3 | 5.8 | 69.4 | 95.1 | 181.3 | 187.3 | 294.9 | 35.6 | | | | | | |
| 40 | 11.4 | 16 | 217 | 204.5 | 151.7 | 47.4 | | | | | | | | |

**Table 5**
Results of the branch-and-bound procedure for the *Allahverdi* instance set. For each instance size (row) and variability value $D$ (column), two values are reported: the average time (av.) and the standard deviation (sd.) among the instances for which the algorithm obtains the optimal solution within the imposed (600 s) time limit.

| $D$ | 10 | | 20 | | 30 | | 40 | | 50 | | 60 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | av. | sd. | av. | sd. | av. | sd. | av. | sd. | av. | sd. | av. | sd. |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0.1 | 0 | 0 | 0 | 0.1 |
| 20 | 0 | 0 | 0 | 0 | 0.1 | 0.3 | 1.6 | 6 | 1.9 | 4.8 | 4.3 | 18.9 |
| 25 | 0 | 0.1 | 1.5 | 6.2 | 9.9 | 40.2 | 29.6 | 67.8 | 42.7 | 82.1 | 83.8 | 129 |
| 30 | 0.7 | 1.3 | 8.1 | 15.8 | 63.3 | 111.2 | 139.4 | 168.5 | 174.6 | 174.7 | 245.5 | 195.4 |
| 35 | 4.1 | 7.9 | 113.6 | 136.7 | 201.3 | 156.9 | 221.5 | 114.5 | 441.2 | | | |
| 40 | 50.6 | 82.5 | 230.6 | 188 | 339.7 | 318.9 | 177.5 | | | | | |

**Table 6**
Results of the branch-and-bound procedure for the *Sotskov* instance set. For each instance size (row) and variability value $\delta$ (column), two values are reported: the average number of nodes (in thousands), column av., and the maximum number of nodes (in thousands), column max., explored by the branch-and-bound algorithm before verifying optimality within the imposed (600 s) time limit.

| $\delta$ | 1 | | 2 | | 5 | | 10 | | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | av. | max | av. | max | av. | max | av. | max | av. | max |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 | 0.6 |
| 15 | 0 | 0 | 0 | 0.1 | 0 | 0.2 | 0.1 | 0.5 | 0.4 | 2.6 |
| 20 | 0 | 0.1 | 0 | 0.3 | 0.4 | 3.7 | 1.6 | 10.7 | 5.8 | 68.4 |
| 25 | 0 | 0.1 | 0.1 | 1.3 | 1.5 | 12.8 | 30.2 | 611.4 | 183.2 | 2879.5 |
| 30 | 0.1 | 0.3 | 0.5 | 2.6 | 58.8 | 964.8 | 427.1 | 4770 | 3772.5 | 32,443.2 |
| 35 | 0.3 | 2.8 | 1.5 | 25.1 | 62 | 1017 | 3067.2 | 20,979.1 | 8400.7 | 30,860.9 |
| 40 | 0.5 | 3.4 | 6.3 | 60.7 | 289.1 | 1968.3 | 7084.7 | 21,121.9 | 6074.8 | 7493.1 |
| $\delta$ | 25 | | 50 | | 75 | | 100 | | | |
| $n$ | av. | max | av. | max | av. | max | av. | max | | |
| 10 | 0.1 | 1.1 | 0.3 | 2.5 | 0.5 | 3.2 | 1.1 | 3.3 | | |
| 15 | 2.8 | 34.7 | 12.3 | 121.2 | 43.5 | 446 | 242.6 | 1492.7 | | |
| 20 | 27.7 | 312.9 | 595.6 | 6412.7 | 2202.2 | 13,746.9 | 11,342.4 | 63,735.2 | | |
| 25 | 1760.2 | 23,787.2 | 9187.8 | 48,666.2 | 19,044.8 | 52,894 | | | | |
| 30 | 11,818.6 | 38,851.2 | 18,806.4 | 20,814.1 | | | | | | |
| 35 | 15,171.1 | 17,314.6 | | | | | | | | |
| 40 | | | | | | | | | | |

**Table 7**
Results of the branch-and-bound procedure for the *Allahverdi* instance set. For each instance size (row) and variability value $D$ (column), two values are reported: the average number (in thousands) of nodes, av., and the maximum number (in thousands) of nodes, max., explored by the branch-and-bound algorithm before verifying optimality within the imposed (600 s.) time limit.

| $D$ | 10 | | 20 | | 30 | | 40 | | 50 | | 60 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | av. | max | av. | max | av. | max | av. | max | av. | max | av. | max |
| 10 | 0 | 0.1 | 0 | 0.2 | 0 | 0.4 | 0.1 | 0.2 | 0.1 | 1 | 0.1 | 0.6 |
| 15 | 0.1 | 0.3 | 0.7 | 24 | 0.7 | 10.1 | 2 | 25.1 | 2.9 | 27.1 | 3.3 | 26.1 |
| 20 | 0.3 | 2.4 | 1.7 | 11.8 | 11.6 | 128.6 | 162 | 4,150.5 | 186.2 | 2657.8 | 448.4 | 14,295.3 |
| 25 | 2.2 | 40.3 | 89.5 | 2659.2 | 734.4 | 21,649.3 | 2070.8 | 24,435.1 | 3139 | 31,035 | 6211.2 | 45,008.1 |
| 30 | 26.5 | 243.1 | 375.3 | 3384.6 | 3219.9 | 26,832.7 | 7878.4 | 37,100.5 | 9393.9 | 33,313.1 | 12,766.5 | 32,838.3 |
| 35 | 116.6 | 1123.9 | 4280.9 | 20,515.8 | 7882.6 | 22,609.1 | 8639 | 15,408.5 | 17,258.5 | 17,258.5 | | |
| 40 | 1153.7 | 9752.3 | 6407.9 | 16,627.9 | 10,859.5 | 19,025.5 | 4460.5 | 4460.5 | | | | |

and verifying optimality within reduced running times and acceptable standard deviations. For medium and large sized instances, both the average running time and the standard deviation increase, thus reinforcing our previous conclusion on the limits of applicability of the proposed exact method.

A different metric to evaluate the total computing effort required to solve an instance corresponds to the average number of nodes explored by the branch-and-bound procedure before verifying optimality. The metric benefits from being agnostic to the computer used for the experimentation and can also be used as a reference on the advantages of using a branch-and-bound algorithm over the raw enumeration of the partial solutions. Tables 6 and 7 report results on both the average and the maximum number of nodes explored by the branch-and-bound procedure before verifying optimality.

The results of the table highlight the improvement of the branch-and-bound method over the enumeration of alternatives. For example, the number of alternative sequences for a Sotskov instance with $\delta = 100$ and $n=20$ is $20! \simeq 2.43 \cdot 10^{18}$, whereas the proposed method only evaluates an average of $1.13 \cdot 10^{7}$ partial solutions for the same group of instances. The results of Tables 6 and 7 show that, as the number of jobs and variability

ratio increases, so does the number of nodes that need to be examined. Nevertheless, the trend is reversed in the most difficult instances (larger number of jobs and processing time variability), as the time required to analyze each partial solution increases (mainly due to the time spent on the lower bounding procedure and the time spent evaluating the regret of non fathomed solutions) and the algorithm does not solve all of the instances; thus the average is calculated considering only the instances solved to optimality.

In order to evaluate the performance of the bounding method proposed in Section 3.2, Table 8 provides the average ratio of nodes pruned by the lower bound. The lower bound is applied only to those partial solutions that are feasible according to the dominance rules described in Section 2.4, hence the results should be analyzed taking into account that if no dominance rule was used, a larger ratio of pruned nodes would have been found.

The results depicted in Table 8 show that the number of partial solutions pruned by the bounding procedure increases as the variability of processing times increases, since its relative role on the performance of the algorithm increases with respect to the dominance relations. For those instances with small variability (as the *Sotskov* set with $\delta = 1$), the percentage of partial solutions fathomed by the bound is small, as the dominance relations are able to avoid the exploration of partial solutions which cannot lead to improved solutions, but for those instances in which the variability is high (as the *Sotskov* set with $\delta = 100$) the bounding method is able to fathom large areas of the search space, and up to 92% (on average) of the explored partial solutions are discarded due to the lower bound. While the ratio shows a positive correlation with the number of jobs of the instance (the ratio of fathomed nodes increases with the number of jobs), the increase is not significant enough to enable the resolution of larger instances.

### 4.4. Evaluation of the midpoint scenario heuristic

If regret evaluation is not deemed practical or a faster solution is needed, the midpoint scenario heuristic seems to be a viable option. Consequently, and in order to evaluate the quality of the midpoint scenario heuristic, Table 9 provides the average gap between the branch-and-bound solution within a 600 s run time limit and the solution of the heuristic. The gap is calculated using formula (14), in which $UB_h$ corresponds to the regret of the midpoint heuristic and $UB$ corresponds to the upper bound provided by the branch-and-bound. Please note that the average gap is calculated considering all of the instances and not only those instances for which the optimal solution was found, and thus it also provides the improvement achieved by the branch-and-bound with respect to the midpoint heuristic

$$gap = \frac{UB_h - UB}{UB} \cdot 100 \tag{14}$$

Table 9 shows that the midpoint scenario heuristic provides very good upper bounds. While its behavior deteriorates when the variability increases in terms of number of reported best-found solutions, the optimality gaps remain low, leading us to conjecture that the variability does not greatly affect the quality of the obtained solutions. Note that the instances from the *Sotskov* set provide lower gaps than the instances from the *Allahverdi* set. This behavior can be attributed to the differences between both instances, as the processing time intervals of the *Sotskov* instances are centered around a middle value (the first value used to generate the instance), while the *Allahverdi* instance set does not show such a pattern. Hence, the performance of the midpoint scenario heuristic depends on the structure of the processing time intervals, offering better solutions when the intervals are centered around a predefined value (as it would be the case if the interval data was generated considering a percentage deviation from an average observed value).

#### 4.4.1. Analysis of the variability on the regret of the optimal solutions

In order to evaluate the effect of the variability on the maximum regret, we calculated the average maximum regret among all of the solution for which the instances are solved to optimality (when not all of the instances have been solved to optimality, the average among solved instances is reported). Note that these results are dependent on the range of values for the processing

**Table 8**
Average percentage of nodes fathomed by the lower bounding method proposed in Section 3.3 for the *Sotskov* and *Allahverdi* instance sets. For each instance size (row) and variability value $\delta$ or $D$ (column), the percentage of pruned partial solution with respect to the total number of explored nodes is reported.

| $n$ | Sotskov | | | | | | | | | Allahverdi | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 5 | 10 | 15 | 25 | 50 | 75 | 100 | 10 | 20 | 30 | 40 | 50 | 60 |
| 10 | 43.9 | 33 | 35.2 | 39.3 | 50.2 | 54 | 69.3 | 79.9 | 83.6 | 39.4 | 49.3 | 50.7 | 57.5 | 66.5 | 67.5 |
| 15 | 25.2 | 28.1 | 23.6 | 39.4 | 42.6 | 69.1 | 81.8 | 87.2 | 88.1 | 39.5 | 58 | 61.9 | 68.6 | 71.8 | 75.8 |
| 20 | 21 | 20.7 | 20.2 | 43.3 | 64.1 | 71.2 | 84.5 | 89.4 | 92.2 | 43.1 | 61.2 | 68.7 | 77.1 | 76.8 | 82.1 |
| 25 | 14.7 | 16.3 | 42.2 | 59.7 | 70.8 | 79.6 | 87.5 | 91.6 | | 49.1 | 68.2 | 78.3 | 75.8 | 81.7 | 83.4 |
| 30 | 14.7 | 17 | 44.3 | 64.3 | 75.1 | 81.9 | 88.8 | | | 54.8 | 69.4 | 74.9 | 81.1 | 81.4 | 82.8 |
| 35 | 13.9 | 29.1 | 44.1 | 72.7 | 77.4 | 85 | | | | 58.2 | 73.3 | 76.7 | 80 | 80.2 | |
| 40 | 14.2 | 16.5 | 52.6 | 71.2 | 79.7 | | | | | 61.2 | 71.5 | 80.8 | 70.1 | | |

**Table 9**
Results of the midpoint scenario heuristic for the *Sotskov* and *Allahverdi* instance set. For each instance size (row) and variability value, $\delta$ or $D$, (column), the average gap between the midpoint scenario and the best solution found by the branch-and-bound algorithm after a 600 s time limit is reported.

| $n$ | Sotskov | | | | | | | | | Allahverdi | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 5 | 10 | 15 | 25 | 50 | 75 | 100 | 10 | 20 | 30 | 40 | 50 | 60 |
| 10 | 0 | 0 | 0 | 0 | 1.38 | 0.37 | 0.59 | 0.27 | 0.27 | 2.03 | 4.15 | 5.47 | 6.94 | 3.54 | 5.02 |
| 15 | 0 | 0.4 | 0.47 | 0.44 | 0.53 | 0.64 | 0.4 | 0.12 | 0.13 | 2.73 | 6.98 | 5.4 | 8.12 | 6.53 | 7.13 |
| 20 | 0 | 0.03 | 0.41 | 0.61 | 0.55 | 0.51 | 0.28 | 0.19 | 0.12 | 3.97 | 5.15 | 5.01 | 6.95 | 7.75 | 6.89 |
| 25 | 0 | 0.25 | 0.5 | 0.48 | 0.4 | 0.44 | 0.22 | 0.07 | 0.01 | 4.85 | 6.04 | 7.99 | 7.27 | 7.72 | 9.82 |
| 30 | 0.02 | 0.48 | 0.51 | 0.49 | 0.64 | 0.24 | 0.19 | 0.02 | | 4.86 | 6.32 | 7.67 | 9.08 | 8.49 | 7.8 |
| 35 | 0.08 | 0.43 | 0.3 | 0.57 | 0.52 | 0.25 | 0.06 | | | 5.05 | 6.59 | 7.5 | 7.56 | 9.39 | 9.39 |
| 40 | 0.19 | 0.43 | 0.25 | 0.57 | 0.34 | 0.23 | 0.05 | | | 4.75 | 5.94 | 7.45 | 7.56 | 8.21 | 7.82 |

**Table 10**
Average maximum regret for the optimal solutions of the Sotskov and Allahverdi instance sets. For each instance size (row) and variability value $\delta$ or $D$ (column), the average maximum regret of the optimal solutions is reported. For those instance sizes and variability indices in which no optimal solution was verified, no values are reported.

| $n$ | Sotskov | | | | | | | | | Allahverdi | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 5 | 10 | 15 | 25 | 50 | 75 | 100 | 10 | 20 | 30 | 40 | 50 | 60 |
| 10 | 15 | 50 | 210 | 1085 | 2188 | 5029 | 15,955 | 34,490 | 62,074 | 255 | 642 | 1249 | 2113 | 2716 | 3738 |
| 15 | 35 | 100 | 549 | 2153 | 3889 | 10,820 | 38,177 | 86,610 | 147,861 | 461 | 1374 | 2615 | 4816 | 5540 | 7421 |
| 20 | 53 | 213 | 1163 | 3467 | 7509 | 15,899 | 61,256 | 127,864 | 258,852 | 643 | 2301 | 4371 | 7866 | 9922 | 12,205 |
| 25 | 78 | 313 | 1658 | 5082 | 10,087 | 25,956 | 88,070 | 183,288 | 442,396 | 643 | 2301 | 4371 | 7866 | 9922 | 12,205 |
| 30 | 113 | 411 | 2448 | 7517 | 15,015 | 34,732 | 108,485 | 203,308 | | 1641 | 4636 | 9466 | 13,235 | 19,644 | 25,349 |
| 35 | 169 | 548 | 2744 | 9654 | 17,361 | 39,236 | | | | 1902 | 6153 | 12,000 | 17,228 | 23,823 | 39,732 |
| 40 | 201 | 624 | 3737 | 11,661 | 21,494 | | | | | 2409 | 7195 | 12,986 | 17,589 | | |

times (which contribute to the differences in the maximum regret observed in Table 10) and the weights, and can only be used as an indication of the expected grow rate if the number of jobs and/or variability ratio changes.

Both tables show that the maximum regret of the optimal solution increases as the variability increases. Furthermore, the change seems to be greater than linear. Therefore, we are inclined to conclude that a linear reduction of the variability on a system is to have a more than linear impact on the ability to avoid extreme undesirable situations.

## 5. Conclusions

In this paper we have studied the robust (minmax regret) total weighted completion time problem with interval processing times, we have put forward different valid methods to evaluate the maximum regret, and we have proposed a branch-and-bound method to obtain the sequence that minimizes the maximum regret. Our main recommendations and findings are the following:

1. The difficulty of the instances depends not only on the instance size but also on the degree of uncertainty (the variability of the possible realizations of processing times). This behavior was already detected in other robust optimization problems, like the robust assignment problem [24]. In this case, the origin of the additional difficulty comes from the lack of dominance rules among jobs, which increases the number of worst-case alternatives that need to be evaluated during the calculation of the maximum regret. Furthermore, the dominance rule reduces the branching factor of the branch-and-bound algorithm, increasing the number of alternatives that need to be considered.
2. As in other problems [24,25,8], the midpoint scenario heuristic provides very good solutions. Furthermore, it provides a 2-approximation of the optimal solution, see Appendix A. Consequently, the midpoint scenario is to be the recommended approach when the resolution of large size instances is required, or the evaluation of the quality of the solution is considered as unfeasible (e.g. under short computation times).
3. The branch-and-bound algorithm is shown to be able to verify the optimal solution within relatively short running times for instances with up to 25–40 jobs depending on the variability of the instance. While the size of the instances which can be optimally solved is small when compared to other IDMR machine scheduling problems, we note that the non-linear nature of the objective function (the total cost depends not only on the assignment of a job to a position in the sequence but also on the jobs sequenced after the said position) is the responsible of the added difficulty.
4. While the proposed lower bound does not provide an optimality gap, it is very fast and it is able to fathom large areas of the branch-and-bound tree. Furthermore, the method is very general and may be useful for other regret optimization problems in which we can obtain the optimal continuation of a partial solution for any given scenario and evaluate the regret of the continuation easily.
5. The effect on the maximum regret of increasing the variability of the processing time intervals is more than linear. Therefore, it is to be expected that reducing variability on the processing times will have a more-than-linear improvement on the performance against extreme undesirable situations.

The previous comments highlight the positive findings of this research. We would also like to point up the main shortcomings of the proposed methods.

The results show that medium-sized instances can be optimally solved, but the computational experiment outlines the boundaries of applicability of the proposed methods to evaluate the maximum regret; that is, the optimization problem that for any given sequence tries to find the sequence and the scenario in which the difference between the objective value of both sequences is maximized.

We conjecture that the regret evaluation is the bottleneck element of the resolution method restricting the capability of the proposed branch-and-bound to solve larger instances, and the structure of the regret evaluation problem (the objective function contains multiple interactions among jobs and with their respective processing times), is the source of much of the complexity of the RWCTP. Note that this work does study the computational complexity of the regret evaluation problem. Moreover the result from previous works on IDMR problems [1] are not applicable for this regret evaluation problem.

While computational complexity is not the focus of this research, we would like to highlight that most of the previously studied IDMR problems require the resolution of a deterministic version of the problem in order to evaluate the regret of a given solution. Some exceptions exist like the unweighted version of the RWCTP, in which the resolution of an assignment problem to construct a sequence substitutes the SPT rule (see [11]) required to solve the deterministic case. We suspect that a similar difficulty increase also appears in the problem in hand, in which the WSPT rule is substituted by a problem that takes into account multiple interactions among jobs and with their respective processing times.

The previous conjecture led us to regard the regret evaluation problem as non polynomially solvable, even when we have not been able to reduce any NP-hard problem to the regret evaluation problem (nor reduce the problem to any polynomially solvable problem), and thus the complexity issue remains open. Nonetheless, if an efficient algorithm for the regret evaluation problem is found, it could be easily integrated with the proposed dominance rules, lower bound and branching rule in order to obtain a branch-and-bound algorithm which could be used to solve larger instances.

**Acknowledgments**

The author would like to express his gratitude to the reviewers for their suggestions and opinions which have greatly improved the overall quality of the paper.

## Appendix A. The midpoint scenario heuristic

In this appendix we show that the optimal solution of the WCTP using the processing times of the midpoint scenario provides a 2-approximation of the optimal solution for the interval data RWCTP. The proof follows a previous proof provided in Kasperski and Zielinski [15] for the unweighted case and differs in the treatment of the objective function.

Let $F(\sigma, p)$ be the total weighted completion time of sequence $\sigma$ under scenario $p$, and let $F(\sigma_p^*, p)$ be the total weighted completion time of the optimal sequence under scenario $p$. Then, the maximum regret for a given sequence $\sigma$, Eq. (3), can be rewritten as (A.1), and consequently (A.2) holds

$$Z(\sigma) = \max_{p\in P}\left(F(\sigma, p) - F(\sigma_p^*, p)\right) \tag{A.1}$$

$$Z(\sigma) \geq F(\sigma, p) - F(\sigma_p^*, p) \quad \forall\, p \in P \tag{A.2}$$

For any two schedules $\sigma$, $\sigma'$ the difference between their objectives on scenario $p$ can be expressed as (A.3), in which $p_i^p$ corresponds to the processing time of job $i$ under scenario $p$

$$F(\sigma, p) - F(\sigma', p) = \sum_{i\in J}\left(\sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j\right)p_i^p \tag{A.3}$$

Let us define the set $A_{\sigma,\sigma'}$ as the subset of jobs $J$ fulfilling (A.4). Note that the set $J\backslash A_{\sigma,\sigma'} = A_{\sigma',\sigma}$

$$\sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j > 0 \tag{A.4}$$

Using the results derived from Theorem 2, see Section 3.1, the scenario $p$ maximizing $F(\sigma, p) - F(\sigma', p)$ can be obtained by setting $p_i^p = p_i^+$ if $i$ belongs to $A_{\sigma,\sigma'}$ and $p_i^p = p_i^-$ if any other case. In fact, for those jobs in which (A.4) holds in equality, the realization of the processing time is irrelevant to the calculation of the maximum regret, and could be ignored throughout the computation.

Consequently, inequality (A.5) can be derived from (A.3), and inequality (A.6) holds for any pair of sequences $\sigma$, $\sigma'$

$$\sum_{i\in J}\left(\sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j\right)p_i^p \leq \sum_{i\in A_{\sigma,\sigma'}}\left(\sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j\right)p_i^+ + \sum_{i\in J\backslash A_{\sigma,\sigma'}}\left(\sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j\right)p_i^- \tag{A.5}$$

$$Z(\sigma) \geq \sum_{i\in A_{\sigma,\sigma'}}\left(\sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j\right)p_i^+ + \sum_{i\in J\backslash A_{\sigma,\sigma'}}\left(\sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j\right)p_i^- \tag{A.6}$$

Let $p_{\sigma'}$ be the worst case scenario for sequence $\sigma'$, then inequality (A.7) can be derived from (A.3) and (A.5)

$$F(\sigma', p_{\sigma'}) \leq F(\sigma, p_{\sigma'}) + \sum_{i\in A_{\sigma',\sigma}}\left(\sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j\right)p_i^+ + \sum_{i\in J\backslash A_{\sigma',\sigma}}\left(\sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j\right)p_i^- \tag{A.7}$$

Subtracting $F(\sigma_p^*, p_{\sigma'})$ from both sides of (A.7), and using (A.2) we get

$$Z(\sigma') \leq Z(\sigma) + \sum_{i\in A_{\sigma',\sigma}}\left(\sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j\right)p_i^+ + \sum_{i\in J\backslash A_{\sigma',\sigma}}\left(\sum_{\substack{j\in J:\\ \pi_{\sigma'}(i)\leq\pi_{\sigma'}(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j\right)p_i^- \tag{A.8}$$

Using inequalities (A.6) and (A.8) and equality (A.3), we are not ready to prove the claim.

**Theorem 3.** *Let $\sigma^=$ be the optimal solution for the midpoint scenario ($p_i^= = 1/2(p_i^- + p_i^+)$ for all jobs $i \in J$). Then, for every feasible solution $\sigma$, $Z(\sigma^=) \leq 2Z(\sigma)$ holds.*

**Proof.** From (A.3) and the optimality of $\sigma^=$ for the midpoint scenario, (A.9) holds

$$\sum_{i\in J}\left(\sum_{\substack{j\in J:\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j - \sum_{\substack{j\in J:\\ \pi_{\sigma^=}(i)\leq\pi_{\sigma^=}(j)}} w_j\right)\left(p_i^+ + p_i^-\right) \geq 0 \tag{A.9}$$

Inequality (A.9) is equivalent to

$$\sum_{i\in A_{\sigma,\sigma^=}}\left(\sum_{\substack{j\in J\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j - \sum_{\substack{j\in J\\ \pi_{\sigma^=}(i)\leq\pi_{\sigma^=}(j)}} w_j\right)p_i^+ + \sum_{i\in J\backslash A_{\sigma,\sigma^=}}\left(\sum_{\substack{j\in J\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j - \sum_{\substack{j\in J\\ \pi_{\sigma^=}(i)\leq\pi_{\sigma^=}(j)}} w_j\right)p_i^- \geq \sum_{i\in J\backslash A_{\sigma^=,\sigma}}\left(\sum_{\substack{j\in J\\ \pi_{\sigma^=}(i)\leq\pi_{\sigma^=}(j)}} w_j - \sum_{\substack{j\in J\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j\right)p_i^+ + \sum_{i\in A_{\sigma^=,\sigma}}\left(\sum_{\substack{j\in J\\ \pi_{\sigma^=}(i)\leq\pi_{\sigma^=}(j)}} w_j - \sum_{\substack{j\in J\\ \pi_\sigma(i)\leq\pi_\sigma(j)}} w_j\right)p_i^- \tag{A.10}$$

Substituting (A.6) on the left side and (A.8) on the right side, we

get

$$Z(\sigma) \geq Z(\sigma^{=}) - Z(\sigma) \tag{A.11}$$

Hence, $2Z(\sigma) \geq Z(\sigma^{=})$ and the proof is completed.□

The bound of 2 is tight, as the bound is also tight for the unweighted case, which is a special case of the problem, see Kasperski and Zielinski [15].

## Appendix B. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cor.2015.08.010.

## References


[1] Aissi H, Bazgan C, Vanderpooten D. Minmax and minmax regret versions of combinatorial optimization problems: a survey. Eur J Oper Res 2009;197:427–38. http://dx.doi.org/10.1016/j.ejor.2008.09.012.

[2] Allahverdi A, Aydilek H, Aydilek A. Single machine scheduling problem with interval processing times to minimize mean weighted completion time. Comput Oper Res 2014;51:200–7. http://dx.doi.org/10.1016/j.cor.2014.06.003.

[3] Averbakh I. Minmax regret solutions for minimax optimization problems with uncertainty. Oper Res Lett 2000;27:57–65. http://dx.doi.org/10.1016/S0167-6377(00)00025-0.

[4] Averbakh I. The minmax regret permutation flow-shop problem with two jobs. Eur J Oper Res 2006;169:761–6. http://dx.doi.org/10.1016/j.ejor.2004.07.073.

[5] Baker KR. Minimizing earliness and tardiness costs in stochastic scheduling. Eur J Oper Res 2014;236:445–52. http://dx.doi.org/10.1016/j.ejor.2013.12.011.

[6] Bellman RE. Dynamic programming. Princeton, NJ, USA: Princeton University Press; 1957.

[7] Conde E. A 2-approximation for minmax regret problems via a mid-point scenario optimal solution. Oper Res Lett 2010;38:326–7. http://dx.doi.org/10.1016/j.orl.2010.03.002.

[8] Conde E. A MIP formulation for the minmax regret total completion time in scheduling with unrelated parallel machines. Optim Lett 2014;8:1577–89. http://dx.doi.org/10.1007/s11590-013-0655-0.

[9] Conway RW, Maxwell WL, Miller L. Theory of scheduling. Reading. USA: Addison Welsey; 1967.

[10] Daniels RL, Carrillo JE. $\beta$-Robust scheduling for single-machine systems with uncertain processing times. IIE Trans 1997;29:977–85. http://dx.doi.org/10.1023/A:1018500319345.

[11] Daniels RL, Kouvelis P. Robust scheduling to hedge against processing time uncertainty in single-stage production. Manag Sci 1995;41:363–76. http://dx.doi.org/10.1287/mnsc.41.2.363.

[12] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discrete Math. 1979;5:287–326. http://dx.doi.org/10.1016/S0167-5060(08)70356-X.

[13] Held M, Karp RM. A Dynamic programming approach to sequencing problems. J Soc Ind Appl Math 1962;10:196–210. http://dx.doi.org/10.1137/0110015.

[14] Iranpoor M, Fatemi Ghomi S, Zandieh M. Due-date assignment and machine scheduling in a low machine-rate situation with stochastic processing times. Comput Oper Res 2013;40:1100–8. http://dx.doi.org/10.1016/j.cor.2012.11.013.

[15] Kasperski A, Zielinski P. A 2-approximation algorithm for interval data minmax regret sequencing problems with the total flow time criterion. Oper Res Lett 2008;36:343–4. http://dx.doi.org/10.1016/j.orl.2007.11.004.

[16] Keha AB, Khowala K, Fowler JW. Mixed integer programming formulations for single machine scheduling problems. Comput Ind Eng 2009;56:357–67. http://dx.doi.org/10.1016/j.cie.2008.06.008.

[17] Kooli A, Serairi M. A mixed integer programming approach for the single machine problem with unequal release dates. Comput Oper Res 2014;51:323–30. http://dx.doi.org/10.1016/j.cor.2014.06.013.

[18] Kouvelis P, Daniels RL, Vairaktarakis G. Robust scheduling of a two-machine flow shop with uncertain processing times. IIE Trans 2000;32:421–32. http://dx.doi.org/10.1023/A:1007640726040.

[19] Kouvelis P, Yu G. Robust discrete optimization and its applications. Boston: Academic Publishers; 1997.

[20] Lebedev V, Averbakh I. Complexity of minimizing the total flow time with interval data and minmax regret criterion. Discrete Appl Math 2006;154:2167–77. http://dx.doi.org/10.1016/j.dam.2005.04.015.

[21] Lu C-C, Lin S-W, Ying K-C. Robust scheduling on a single machine to minimize total flow time. Comput Oper Res 2012;39:1682–91. http://dx.doi.org/10.1016/j.cor.2011.10.003.

[22] Lu C-C, Lin S-W, Ying K-C. Minimizing worst-case regret of makespan on a single machine with uncertain processing and setup times. Appl Soft Comput 2014;23:144–51. http://dx.doi.org/10.1016/j.asoc.2014.06.006.

[23] Montemanni R. A mixed integer programming formulation for the total flow time single machine robust scheduling problem with interval data. J Math Model Algorithms 2007;6:287–96. http://dx.doi.org/10.1007/s10852-006-9044-3.

[24] Pereira J, Averbakh I. Exact and heuristic algorithms for the interval data robust assignment problem. Comput Oper Res 2011;38:1153–63. http://dx.doi.org/10.1016/j.cor.2010.11.009.

[25] Pereira J, Averbakh I. The robust set covering problem with interval data. Ann Oper Res 2013;207:217–35. http://dx.doi.org/10.1007/s10479-011-0876-5.

[26] Pinedo ML. Scheduling, Theory, Algorithms and Systems. 4th ed.. New York: Springer; 2012.

[27] Schrage L, Baker KR. Dynamic programming solution of sequencing problems with precedence constraints. Oper Res 1978;26:444–9. http://dx.doi.org/10.1287/opre.26.3.444.

[28] Siepak M, Jòzefczyk J. Solution algorithms for unrelated machines minmax regret scheduling problem with interval processing times and the total flow time criterion. Ann Oper Res 2014;222:517–33. http://dx.doi.org/10.1007/s10479-014-1538-1.

[29] Smith WE. Various optimizers for single-stage production. Nav Res Logist Q 1956;3:59–66. http://dx.doi.org/10.1002/nav.3800030106.

[30] Sotskov Y, Egorova N, Lai T.C. Minimizing total weighted flow time of a set of jobs with interval processing times. Math Comput Model 2009;50:556–73. http://dx.doi.org/10.1016/j.mcm.2009.03.006.

[31] Sotskov Y, Lai T-C. Minimizing total weighted flow time under uncertainty using dominance and a stability box. Comput Oper Res 2012;39:1271–89. http://dx.doi.org/10.1016/j.cor.2011.02.001.

[32] Sotskov YN, Egorova NG, Werner F. Minimizing total weighted completion time with uncertain data: a stability approach. Autom Remote Control 2010;71:2038–57. http://dx.doi.org/10.1134/S0005117910100048.

[33] Valente JM, Schaller JE. Dispatching heuristics for the single machine weighted quadratic tardiness scheduling problem. Comput Oper Res 2012;39:2223–31. http://dx.doi.org/10.1016/j.cor.2011.11.005.

[34] Vilà M, Pereira J. Exact and heuristic procedures for single machine scheduling with quadratic earliness and tardiness penalties. Comput Oper Res 2013;40:1819–28. http://dx.doi.org/10.1016/j.cor.2013.01.019.

[35] Wu CW, Brown KN, Beck JC. Scheduling with uncertain durations: modeling-robust scheduling with constraints. Comput Oper Res 2009;36:2348–56.

[36] Xu X, Cui W, Lin J, Qian Y. Robust makespan minimisation in identical parallel machine scheduling problem with interval data. Int J Prod Res 2013;51:3532–48.

[37] Yang J, Yu G. On the robust single machine scheduling problem. J Comb Optim 2002;6:17–33. http://dx.doi.org/10.1023/A:1013333232691.