Computers & Industrial Engineering 149 (2020) 106838

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Robust (min–max regret) single machine scheduling with interval processing times and total tardiness criterion

Shijin Wang [a,*], Wenli Cui [a], Feng Chu [b,c], Jianbo Yu [d], Jatinder N.D. Gupta [e]

[a] School of Economics and Management, Tongji University, Shanghai 200092, China
[b] Laboratoire IBISC, Univ-Évry, Université Paris-Saclay, Évry 91025, France
[c] School of Economics and Management, Fuzhou University, Fuzhou 350116, China
[d] School of Mechanical Engineering, Tongji University, Shanghai 710049, China
[e] College of Business, The University of Alabama in Huntsville, Huntsville, AL 35899, USA

ARTICLE INFO

Keywords:
Robust single machine scheduling
Min–max regret
Uncertain interval processing times
Total tardiness
Approximation algorithm and heuristics

ABSTRACT

This paper studies a robust (min–max regret) single machine scheduling problem with uncertain processing times represented by interval data. The objective is to obtain robust sequences of jobs that minimize the absolute deviation of total tardiness from the optimal solution under the worst-case scenario. The problem is first formulated as a mixed-integer linear programming (MILP) model and assuming that the corresponding deterministic NP-hard problem for the mid-point scenario can be solved optimally, an optimal schedule under the mid-point scenario is then proved to be a 2-approximation solution to the problem. Next, the worst-case scenarios are proved to be not necessarily at the upper or lower limits of the interval processing times. Utilizing these results, a 2-approximation algorithm (2AA), a worst-case scenario-based heuristic (WSH), and an approximate worst-case-based heuristic (AWH) are proposed and empirically evaluated, in which an iterative procedure for evaluating the maximum regret of a solution is integrated. Finally, the paper is concluded by suggesting some fruitful directions for future research in this area.

1. Introduction

The single machine scheduling problem (SMSP) is one of the most important scheduling problems in various manufacturing and service industries since it is the basic building block for developing more comprehensive scheduling models. While most existing researches on the SMSP assume that job processing times are known and fixed numbers, many practical shop floors typically operate in highly uncertain environments where production schedules cannot be fully implemented according to plans due to various random interruptions, such as incomplete or inaccurate job data, job cancellations, machine failures, and emergency orders (Sabuncuoglu & Goren, 2009). Therefore, parameter uncertainty – like variability in job processing times – must be included in the development of approaches to solve SMSPs in particular and scheduling problems in general.

Broadly speaking, four main approaches exist to deal with parameter uncertainty to mitigate their negative impact on the shop floor performance (Chang et al., 2019, 2017; Niu et al., 2019; Yue et al., 2018). These four kinds of approaches are Stochastic Programming (SP) (e.g. Prekopa, 1995), fuzzy theory (FT) (e.g. Özelkan & Duckstein, 1999), robust optimization (RO) (e.g. Kouvelis & Yu, 1997), and distributionally robust optimization (DRO) (e.g. Delage & Ye, 2010; Zhang et al., 2018). Table 1 summarizes the characteristics, merits, and drawbacks of each of the four approaches (Wang et al., 2020).

When uncertain job processing times are represented by interval data, researchers suggest that the worst-case performance of the system is usually more important than the average performance (Kouvelis & Yu, 1997; Mulvey et al., 1995; Yue et al., 2018). Therefore, the robust optimization (RO) approach is often used to study SMSPs. It has been implemented in many practical cases, including operating room planning in hospitals (Addis et al., 2014; Denton et al., 2010), project scheduling in a large IT services delivery center (Coban et al., 2016), and demand-driven scheduling of trains under uncertain arrival rates (Rajabighamchi et al., 2019).

Furini et al. (2015) and Kouvelis and Yu (1997) suggested using one of the three main objective functions. The min–max criterion (absolute robustness) minimizes the worst-case solution over all scenarios as it aims at constructing solutions having good performance in the worst case. The min–max relative regret criterion (relative robustness) aims at obtaining a solution that minimizes the maximum relative deviation

* Corresponding author.
E-mail addresses: shijinwang@tongji.edu.cn (S. Wang), 1830374@tongji.edu.cn (W. Cui), feng.chu@univ-evry.fr (F. Chu), jbyu@tongji.edu.cn (J. Yu), guptaj@uah.edu (J.N.D. Gupta).

https://doi.org/10.1016/j.cie.2020.106838
Received 5 April 2020; Received in revised form 3 September 2020; Accepted 4 September 2020
Available online 7 September 2020

Table 1
Summary of existing approaches for dealing with parameter uncertainty.

Approach	Uncertainty description	Merits	Drawbacks
SP	Known distribution	Good expected performance in the long run	(a) Distribution is unknown in practice (b) Maybe the expectation are not enough
FT	Fuzzy numbers	(a) Exact distribution is not needed (b) The fuzzy rule can be used	(a) No stability analysis (b) Maybe only effective for specific set
RO	Uncertainty sets: Interval or discrete	Worst-case performance	No full information can be used
DRO	Distribution sets: Uni-modal, multi-modal, moment-based, β-robust	Expected performance under worst-case distribution	Some parameters need to be known beforehand: mean, variance, covariance

"SP": stochastic programming; "FT": fuzzy theory; "RO": robust optimization; "DRO": distributionally robust optimization.

and requires more computation time for the same algorithm. The min–max regret criterion (robust deviation) minimizes the maximum regret and is suitable in situations where the decision-maker may feel regret if he/she makes a wrong decision.

The existing RO literature considers single machine scheduling problems (SMSPs) with uncertain parameters for minimizing various objective functions, such as total (weighted) completion (flow) time (Kasperski & ZielińSki, 2008; Lebedev & Averbakh, 2006; Lu et al., 2012; Pereira, 2016; Sotskov et al., 2009; Wang et al., 2020), maximum waiting time (Yang & Yu, 2002), makespan (Lu et al., 2014), maximum lateness (Kasperski, 2005), and (weighted) number of late jobs (Drwal, 2018; Drwal & Józefczyk, 2020). However, the total tardiness criterion has not received enough research attention.

Motivated by existing studies and the need to consider the total tardiness performance measure, this paper considers the single machine scheduling problem with interval processing times to obtain a robust job sequence with a minimum worst-case absolute deviation of the total tardiness from an optimal solution (i.e., minimize the maximum absolute regret called min–max regret). The considered problem is represented as IDMR-SMSPTT where IDMR denotes "interval data min–max regret" and SMSPTT denotes "single machine scheduling problem with total tardiness". Since the SMSP with the objective of minimizing total tardiness is NP-hard, it follows that the IDMR-SMSPTT considered in this paper is also NP-hard as it involves solving a set of corresponding deterministic NP-hard SMSPTTs.

Our main contributions in this paper are as follows: first, a mixed-integer linear programming (MILP) model to optimally solve the considered IDMR-SMSPTT problem is formulated. Second, an optimal schedule under the mid-point scenario is proved to be a 2-approximation solution to the problem provided that the deterministic NP-hard problem for the mid-point scenario can be solved optimally. Third, the worst-case scenario is proved to be not necessarily at the upper or lower limits of the interval processing times. Finally, a 2-approximate algorithm and two heuristics are proposed and empirically evaluated to obtain robust schedules within a reasonable computation time, in which a procedure to effectively estimate the maximum regret of a given solution is embedded.

The remainder of the paper is organized as follows. In Section 2, the existing literature related to min–max regret single machine scheduling problems is briefly reviewed. In Section 3, a MILP model of the IDMR-SMSPTT is formulated and an algorithm to estimate the maximum regret value of a given schedule is developed. In Section 4, the optimal schedule under the mid-point scenario is proved to be a 2-approximation solution for the problem provided that the deterministic counterpart can be solved optimally. Then, the worst-case scenarios are proved to be not necessarily at the upper or lower limits of the interval processing times. Based on these results, a 2-approximation algorithm and two heuristics are developed to solve the considered problem. In Section 5, the effectiveness of the proposed algorithm and heuristics is empirically evaluated to solve the IDMR-SMSPTT. Finally, in Section 6 the paper is concluded and some future research directions are suggested.

2. Literature review

Robustness analysis is a theoretical framework that enables the decision-maker to take into account uncertainty or imprecision in order to obtain decisions that will behave reasonably under any likely input data. The min–max regret objective function is less conservative as its purpose is to obtain a solution minimizing the maximum deviation between the value of the solution and the optimal value of the corresponding scenario over all possible scenarios (Aissi et al., 2009). In this paper, the min–max regret optimization criterion is considered.

The interval data min–max regret (IDMR) versions of the classical combinatorial optimization and machine scheduling problems have been studied widely in the literature. In general, the min–max regret versions of classical combinatorial optimization problems are harder than their classical counterparts, since uncertainty increases the search space to find an optimal solution (c.f. Aissi et al., 2005; Averbakh, 2001; Feizollahi & Averbakh, 2014; Kasperski & ZielińSki, 2008; Montemanni et al., 2007).

It is relatively easy to determine the worst-case scenario for a feasible solution for many combinatorial optimization problems with interval data such as the 0-1 knapsack problem (Aissi et al., 2009; Furini et al., 2015) whose worst-case scenario of a solution has the lower bound profits for the selected items, and the upper bound profits for the non-selected items, and the traveling salesman problem (Montemanni et al., 2007) whose worst-case scenario of a solution has the upper bound costs for the selected arcs, and the lower bound costs for the non-selected arcs. However, it is not easy to determine the worst-case scenario of a feasible solution for the SMSP to minimize total tardiness, which renders the IDMR-SMSPTT more complicated. In this section, the existing literature most related to the IDMR-SMSPs is reviewed.

In terms of total completion time, Yang and Yu (2002) proved that the SMSPs with the min–max, min–max regret and the min–max relative regret criteria are NP-hard even in the case of two processing time scenarios. They developed a dynamic programming (DP) algorithm to optimally solve the problem. Kasperski (2005) studied the SMSP with interval processing times and due dates to minimize the maximal regret of the maximum lateness criterion and proposed an $O(n^4)$ algorithm to solve the problem. Lebedev and Averbakh (2006) considered the IDMR version of the SMSP with interval processing times of jobs to minimize the total flow time. They proved that for the case where all intervals of uncertainty have the same center, the problem can be solved in $O(nlogn)$ time if the number of jobs is even, and the problem is NP-hard if the number of jobs is odd. Kasperski and ZielińSki (2008) presented a 2-approximation algorithm for the IDMR version of the SMSP with precedence constraints and interval processing times to minimize the total flow time.

Lu et al. (2012) studied the IDMR version of the SMSP with interval job processing times and sequence-dependent family setup times (SDFSTs) to minimize the total flow time. They reformulated the problem as a robust constrained shortest path problem and developed a simulated annealing (SA)-based algorithm. Lu et al. (2014) reformulated

the IDMR version of the SMSP with interval processing times and sequence-dependent setup times to minimize the makespan as a robust TSP problem. Sotskov et al. (2009) developed precedence–dominance relations and a heuristic for the SMSP with a min–max relative regret criterion and interval processing times to minimize the total weighted flow time. Pereira (2016) developed a branch-and-bound (B&B) method for solving the problem considered by Sotskov et al. (2009).

Choi and Chung (2016) considered the IDMR version of the SMSP to determine which jobs should be processed by outsourcing under interval and discrete scenarios to minimize total cost. They derived the complexity results when the cost of in-house jobs is treated as the makespan or the total completion time. Drwal (2018) considered a class of SMSPs with interval due dates to find a solution that minimizes the min–max regret of the weighted number of late jobs.

Kacem and Kellerer (2019) derived complexity results and developed dominance properties for the IDMR version of the SMSP to maximize the number of early jobs with a common due date. Drwal and Józefczyk (2020) studied the IDMR version of the SMSP to minimize the weighted number of late jobs with interval processing times. Both common and job-dependent due-dates were considered. A branch and bound based exact solution method was developed for the problem. Silva et al. (2020) investigated the IDMR-SMSPTT where the processing times take any value in a budgeted uncertainty set. They developed the row-and-column generation algorithm and the branch-and-bound based exact algorithm for solving the problem.

Based on the above review on the representative existing research to solve various IDMR-SMSPs summarized in Table 2, it is evident that the performance measure of total tardiness has not received as much attention as compared to other performance metrics as only one paper by Silva et al. (2020) focused on developing an exact optimization method for solving the IDMR-SMSPTT.

In this paper, unlike Silva et al. (2020), an approximation algorithm and two heuristics for the IDMR-SMSP with total tardiness criterion are developed.

3. Problem formulation and maximum regret estimation

In this section, the IDMR-SMSPTT is first formulated as a mixed-integer linear programming (MILP) model. Then, an algorithm to estimate the maximum regret value of a given solution is described and numerically illustrated.

3.1. Problem definition

The problem under consideration can be formally described as follows. A given set $\mathcal{N} = \{1, 2, \ldots, n\}$ of n jobs is to be processed non-preemptively on a single machine. Each job has its due date d_j and its processing time p_j, $j \in \mathcal{N}$. The classical deterministic single machine scheduling problem with the objective of minimization of total tardiness, denoted by $1 \parallel \sum_{j\in\mathcal{N}} T_j$, is to find a schedule to minimize the total tardiness, where $T_j = \max\{0, C_j - d_j\}$ is the tardiness of job j and C_j is the completion time of job j. This classical $1 \parallel \sum_{j\in\mathcal{N}} T_j$ problem can be formulated with different variables, like completion time variables, time-indexed variables, linear ordering variables, and assignment and positional data variables (Keha et al., 2009). Koulamas (1994, 2010) review the extensive early literature on SMSPs with total tardiness minimization.

Suppose that the job due dates given by customers are assumed to be fixed, whereas the processing times vary due to production uncertainties in manufacturing (e.g., the machine conditions, worker skill levels, or some other accidental factors). Hence, in a more general situation, the processing time of job j, $j \in \mathcal{N}$ can be a priori unknown, since the exact processing times may not be known until all jobs are processed. Suppose that two integer numbers p_j^- and p_j^+ are given with $0 < p_j^- \le p_j^+$.

Assume that the processing time of job j, p_j can take any integer value from its uncertainty interval $[p_j^-, p_j^+]$, regardless of the values taken by other jobs. No assumption is made about the probability distribution of the processing times in this range. This degree of uncertainty introduces additional complexity to the original NP-hard problem. In this context, the set of scenarios S is defined as the Cartesian product of the uncertainty intervals $[p_j^-, p_j^+]$, $j \in \mathcal{N}$. A vector s of n processing times p_j^s satisfying $p_j^s \in [p_j^-, p_j^+]$ for $j \in \mathcal{N}$ is called a scenario. Let $P^+ = \sum_{j\in\mathcal{N}} p_j^+$ and $P^- = \sum_{j\in\mathcal{N}} p_j^-$.

Let $\mathcal{X}$ be the set of all possible sequences of jobs on the machine. Let $z^s(x)$ be the total tardiness value given for scenario $s \in S$ with sequence $x \in \mathcal{X}$ and let $z^s(*)$ be the optimal total tardiness value under scenario s. Then, the regret associated with a solution x for scenario s, $r^s(x)$ is given by

$$r^s(x) = z^s(x) - z^s(*) \tag{1}$$

The maximum regret of a solution x is the maximum $r^s(x)$ value over all scenarios, which is denoted by

$$r(x) = \max_{s\in S} r^s(x) \tag{2}$$

The objective is to find the minimum value of maximum regrets for all possible sequences, Rt defined as

$$Rt = \min_{x\in\mathcal{X}} \max_{s\in S} r^s(x) \tag{3}$$

Also, for $x \in \mathcal{X}$, a maximizer y in $r(x) = \max_{y\in\mathcal{X}} \max_{s\in S}(z^s(x) - z^s(y))$ is a worst-case alternative for x.

3.2. MILP formulation

To formulate IDMR-SMSPTT as a mixed-integer linear programming (MILP) model, the following decision variables are defined.

- x_{ji}^s: 1 if job j is processed before job i in solution $x = \{x_{ji}\}$ for scenario s, $i, j \in \mathcal{N}, i \ne j$, 0 otherwise;
- X_j^s: the completion time of job $j \in \mathcal{N}$ in solution x for scenario s;
- T_j^{xs}: the tardiness of job j in solution x for scenario s;
- y_{ji}^s: 1 if job j is processed before job i in solution $y = \{y_{ji}\}$ for scenario s, $i, j \in \mathcal{N}, i \ne j$, 0 otherwise;
- Y_j^s: the completion time of job j in solution y for scenario s;
- T_j^{ys}: the tardiness of job j in solution y for scenario s;
- p_j^s: the processing time of job j for scenario s.

Then, the MILP model of the robust (min–max regret) single machine scheduling problem with interval processing times to minimize total tardiness (denoted by IDMR-SMSPTT), called [RSST] is described below.

$$[\mathbf{RSST}] \quad \min_{x\in\mathcal{X}} \max_{s\in S} \max_{y\in\mathcal{X}} \Big(\sum_{j\in\mathcal{N}} T_j^{xs} - \sum_{j\in\mathcal{N}} T_j^{ys}\Big) \tag{4a}$$

$$X_j^s \ge p_j^s \qquad \forall\, j \in \mathcal{N}, s \in S; \tag{4b}$$

$$X_j^s + p_i^s \le X_i^s + P^+(1 - x_{ji}^s) \qquad \forall\, i, j \in \mathcal{N}, i \ne j, s \in S; \tag{4c}$$

$$X_i^s + p_j^s \le X_j^s + P^+ x_{ji}^s \qquad \forall\, i, j \in \mathcal{N}, i \ne j, s \in S; \tag{4d}$$

$$X_j^s \le \sum_{i\in\mathcal{N}} p_i^s \qquad \forall\, j \in \mathcal{N}, s \in S; \tag{4e}$$

$$T_j^{xs} \ge X_j^s - d_j \qquad \forall\, j \in \mathcal{N}, s \in S; \tag{4f}$$

$$Y_j^s \ge p_j^s \qquad \forall\, j \in \mathcal{N}, s \in S; \tag{4g}$$

Table 2
Representative existing research on IDMR versions of the SMSP.

Authors & Year	Parameter in interval	Minimizing objective for the deterministic problem	Methods
Yang and Yu (2002)	Processing time	Total completion times	DP & heuristics
Kasperski (2005)	Processing time, due date	Maximum lateness	Polynomial-time algorithm
Lebedev and Averbakh (2006)	Processing time	Total flow times	Computational complexity, polynomial-time algorithm
Kasperski and ZielińSki (2008)	Processing time	Total flow times	Approximation algorithm
Sotskov et al. (2009)	Processing time	Total weighted flow times	Dominance rules, heuristic
Lu et al. (2012)	Processing time, SDFST	Total flow times	SA
Lu et al. (2014)	Processing time, setup time	Makespan	SA-based heuristic
Pereira (2016)	Processing time	Weighted total completion times	B&B, heuristic
Choi and Chung (2016)	Processing time	Total cost	Approximation algorithm
Drwal (2018)	Due date	Weighted number of late jobs	Polynomial-time algorithm
Kacem and Kellerer (2019)	Processing time	Number of non-early jobs	Dominance rules
Drwal and Józefczyk (2020)	Processing time	Weighted number of late jobs	B&B
Silva et al. (2020)	Processing time	Total tardiness	B&B, row-and-column Generation algorithms
This paper	**Processing time**	**Total tardiness**	**Heuristics, approximation algorithm**

$$Y_j^s + p_i^s \le Y_i^s + P^+(1 - y_{ji}^s) \qquad \forall\, i,j \in \mathcal{N}, i \ne j, s \in \mathcal{S}; \tag{4h}$$

$$Y_i^s + p_j^s \le Y_j^s + P^+ y_{ji}^s \qquad \forall\, i,j \in \mathcal{N}, i \ne j, s \in \mathcal{S}; \tag{4i}$$

$$Y_j^s \le \sum_{i \in \mathcal{N}} p_i^s \qquad \forall\, j \in \mathcal{N}, s \in \mathcal{S}; \tag{4j}$$

$$T_j^{ys} \ge Y_j^s - d_j \qquad \forall\, j \in \mathcal{N}, s \in \mathcal{S}; \tag{4k}$$

$$x_{ji}^s, y_{ji}^s \in \{0,1\} \qquad \forall\, i,j \in \mathcal{N}, i \ne j, s \in \mathcal{S}; \tag{4l}$$

$$X_j^s, T_j^{xs}, Y_j^s, T_j^{ys} \ge 0 \qquad \forall\, j,k \in \mathcal{N}, s \in \mathcal{S}; \tag{4m}$$

$$p_i^s \in [p_i^-, p_i^+],\ \text{integer} \qquad \forall\, i \in \mathcal{N}, s \in \mathcal{S}. \tag{4n}$$

The objective function (4a) aims to find the minimum value of maximum regrets for all possible sequences. Constraint (4b) ensures that the completion time of each job is equal or greater than its processing time in solution x. Constraints (4c) and (4d) are disjunctive constraints which enforce that either job j is processed before job i or job i is processed before job j for any pair of jobs in solution x. Constraint (4e) ensures that the completion time of job j is not larger than the sum of the processing times of all the jobs in solution x. Constraint (4f) defines the tardiness of a job in solution x. Correspondingly, constraints (4g)–(4k) define the same constraints for solution y. Finally, constraints (4l), (4m) and (4n) give the domain of the decision variables.

While the solution to the above [RSST] model finds a solution x such that the maximum regret is minimized, it is useful to find the maximum regret of a given solution $x = \{\tilde{x}_{ji}\}$ where $\tilde{x}_{ji} = 1$ if job $j \in \mathcal{N}$ is processed before job $i \in \mathcal{N}$ and 0 otherwise. The following additional decision variables are defined.

- T_j^x: the tardiness of job j in solution x;
- C_j^x: the completion time of job j in solution x;
- p_j: the processing time of job j in a scenario;
- y_{ji}: 1 if job j is processed before job i in solution $y = \{y_{ji}\}$, $i,j \in \mathcal{N}, i \ne j$, 0 otherwise;
- C_j^y: the completion time of job j in solution y;
- T_j^y: the tardiness of job j in solution y.

Intuitively, the maximum regret for a given schedule $\tilde{x}$ can be obtained by solving the following [XMR] model.

$$[\mathbf{XMR}] \quad \max_{y \in \mathcal{X}} \Big(\sum_{j \in \mathcal{N}} T_j^x - \sum_{j \in \mathcal{N}} T_j^y \Big) \tag{5a}$$

$$\text{subject to} \quad p_j^- \le p_j \le p_j^+ \qquad \forall j \in \mathcal{N}; \tag{5b}$$

$$C_j^x \ge p_j \qquad \forall\, j \in \mathcal{N}; \tag{5c}$$

$$C_j^x + p_i \le C_i^x + P^+(1 - \tilde{x}_{ji}) \qquad \forall\, i,j \in \mathcal{N}, i \ne j; \tag{5d}$$

$$C_i^x + p_j \le C_j^x + P^+ \tilde{x}_{ji} \qquad \forall\, i,j \in \mathcal{N}, i \ne j; \tag{5e}$$

$$C_j^x \le \sum_{i \in \mathcal{N}} p_i \qquad \forall\, j \in \mathcal{N}; \tag{5f}$$

$$T_j^x \ge C_j^x - d_j \qquad \forall\, j \in \mathcal{N}; \tag{5g}$$

$$C_j^y \ge p_j \qquad \forall\, j \in \mathcal{N}; \tag{5h}$$

$$C_j^y + p_i \le C_i^y + P^+(1 - y_{ji}) \qquad \forall\, i,j \in \mathcal{N}, i \ne j; \tag{5i}$$

$$C_i^y + p_j \le C_j^y + P^+ y_{ji} \qquad \forall\, i,j \in \mathcal{N}, i \ne j; \tag{5j}$$

$$C_j^y \le \sum_{i \in \mathcal{N}} p_i \qquad \forall\, j \in \mathcal{N}; \tag{5k}$$

$$T_j^y \ge C_j^y - d_j \qquad \forall\, j \in \mathcal{N}; \tag{5l}$$

$$C_j^x, T_j^x, C_j^y, T_j^y \ge 0 \qquad \forall\, j,k \in \mathcal{N}; \tag{5m}$$

$$y_{ji} \in \{0,1\} \qquad \forall\, i,j \in \mathcal{N}, i \ne j; \tag{5n}$$

$$p_j \text{ is integer} \qquad \forall j \in \mathcal{N}. \tag{5o}$$

The objective function (5a) is to maximize the regret of total tardiness for a given solution x. Constraint (5b) determines the processing time of job j, which can take any integer value from the interval $[p_j^-, p_j^+]$. Constraint (5c) ensures that the completion time of each job is equal or greater than its processing time in solution x. Constraints (5d) and (5e) are disjunctive constraints that enforce that either job j is processed before job i or job i is processed before job j for any pair of jobs in solution x. Constraint (5f) ensures that the completion time of job j is not larger than the sum of the processing times of all the jobs in solution x. Constraint (5g) defines the tardiness of a job in solution x. Correspondingly, constraints (5h)–(5l) define the same constraints in solution y. Finally, constraints (5m), (5n) and (5o) give the domain of the decision variables.

There are $n!$ [XMR] models for a problem with n jobs. The more jobs there are, the greater the number of [XMR] models. Therefore, some effective procedures are needed to reduce the search space to find the maximum regret value of a given solution.

3.3. Maximum regret estimation

To obtain the maximum regret value $r(x)$ of a given solution x through the [XMR] model requires too much computation time since even if a scenario s of processing times is given, a NP-hard problem $1 \parallel \sum_{j\in\mathcal{N}} T_j$ has to be solved. To significantly reduce the computational effort, the following procedure (called Algorithm 1) is developed to estimate the maximum regret value $r(x)$ of solution x.

Algorithm 1 Procedure to estimate the maximum regret value of a given solution x

1: **Input:** The problem information including p_j^+, p_j^-, d_j and n, a feasible solution x, the running time $tused = 0$, the time limit $tlim$, $tlim1$, and the set of initial regret $\mathcal{R} = \emptyset$.
2: Compute the total tardiness ST_x with processing times $p_j^-, j \in \mathcal{N}$ for the solution x.
3: Apply the M-NBR method to obtain a feasible schedule y and its corresponding total tardiness ST_y with processing times $p_j^-, j \in \mathcal{N}$. Let $r_c(x) = ST_x - ST_y$.
4: **while** $tused \le tlim$ **do**
5: $\mathcal{R} \cup \{r_c(x)\} \to \mathcal{R}$.
6: Solve the model [XMR1] with CPLEX (set the time IloCplex: TiLim as $tlim1$) to obtain a scenario of processing times p_j^s $(j \in \mathcal{N})$, $ST1_x = \sum_{j\in\mathcal{N}} T_j^x$, $ST1_{y1} = \sum_{j\in\mathcal{N}} T_j$ and $R0 = ST1_x - ST1_{y1}$.
7: **if** $R0 < \max\{\mathcal{R}\}$ **then**
8: $\mathcal{R} \setminus r_c(x) \to \mathcal{R}$. Go to Line 17.
9: **end if**
10: Compute the total tardiness ST_x with the scenario p_j^s $(j \in \mathcal{N})$ for the solution x.
11: Apply the M-NBR method to obtain the optimal schedule y under the scenario p_j^s $(j \in \mathcal{N})$, and also obtain its total tardiness ST_y with the scenario p_j^s $(j \in \mathcal{N})$ for solution y.
12: **if** $ST1_{y1} \le ST_y$ **then**
13: $ST_y = ST1_{y1} - 1$.
14: **end if**
15: $r_c(x) = ST_x - ST_y$.
16: **end while**
17: **Output:** The current maximum regret value $R_{max} = \max\{\mathcal{R}\}$.

To begin the solution process, the problem is initialized and the algorithm parameters are given. Let $\mathcal{R} = \emptyset$ be the set of regret values, which will be updated iteratively. The limit of total computation time $tlim$ is given (Line 1).

The total tardiness ST_x for a given solution x with the scenario of $p_j^-, j \in \mathcal{N}$ is first computed (Line 2) and the modified Net Benefit of Relocation (M-NBR) method (Russell & Holsenback, 1997) is applied to obtain a feasible schedule y under the scenario of $p_j^-, j \in \mathcal{N}$. Note that other effective methods can also be applied for the deterministic single machine scheduling problem of minimizing total tardiness. The corresponding total tardiness ST_y with processing times $p_j^-, j \in \mathcal{N}$ for solution y can be obtained. $r_c(x) = ST_x - ST_y$ is the current regret value (Line 3). Then the main loop begins (Line 4). Let $\mathcal{R} = \mathcal{R} \cup \{r_c(x)\}$ (Line 5). With the updated $\mathcal{R}$, new constraints $\sum_{j\in\mathcal{N}} T_j^x - \sum_{j\in\mathcal{N}} T_j \le \max\{\mathcal{R}\}$ are added into the original [XMR] model, which forms the following [XMR1] model,

$$[\text{XMR1}] \quad \max\Big(\sum_{j\in\mathcal{N}} T_j^x - \sum_{j\in\mathcal{N}} T_j\Big) \tag{6a}$$

$$\text{subject to} \sum_{j\in\mathcal{N}} T_j^x - \sum_{j\in\mathcal{N}} T_j \le \max\{\mathcal{R}\}; \tag{6b}$$

$$constraints\ (5b) - (5o). \tag{6c}$$

Then, the model [XMR1] is solved to obtain the corresponding optimal scenario of processing times $p_j^s, j \in \mathcal{N}$, which maximizes the objective (6a) of [XMR1] with the constraints (6b). By solving [XMR1] with CPLEX, $ST1_x = \sum_{j\in\mathcal{N}} T_j^x$ and $ST1_{y1} = \sum_{j\in\mathcal{N}} T_j$ are obtained (Line 6). The time limit for solving [XMR1] with CPLEX is set as $tlim1$ since preliminary computational tests showed that if $\max\{\mathcal{R}\} > r(x)$, [XMR1] is equivalent to [XMR]. Therefore, for solving problem instances with a large number of jobs n using [XMR1] with CPLEX for this case requires a lot of computational effort. However, finding an optimal solution of [XMR1] using CPLEX is much easier if $\max\{\mathcal{R}\} \le r(x)$ even for problem instances with $n \ge 20$. Hence, in this algorithm, $\max\{\mathcal{R}\}$ is gradually updated by adding cuts iteratively.

Once the desired scenario of processing times $p_j^s, j \in \mathcal{N}$ is identified, the total tardiness of the given solution x, ST_x, and the total tardiness of the solution y, ST_y, are computed by the M-NBR method (Lines 10–11). Clearly, $ST1_x = ST_x$ since both are computed for $p_j^s, j \in \mathcal{N}$ and the given solution x. To ensure the continuation of the iteration, $ST_y = ST1_{y1} - 1$ is given if $ST1_{y1} \le ST_y$. Then, let $r_c(x) = ST_x - ST_y$ (Lines 12–15).

Algorithm 1 terminates once the total computation time $tused$ exceeds the predefined time limit $tlim$, or $r_c(x) \in \mathcal{R}$, or the time for solving [XMR1] with CPLEX exceeds the limit $tlim1$. The output from Algorithm 1 is the maximum regret value $R_{max} = \max\{\mathcal{R}\}$ (Line 17).

To illustrate Algorithm 1, a numerical 10-job example problem is considered, which has the job processing time limits as $\mathbf{p}^- = [5, 10, 6, 12, 9, 5, 8, 7, 11, 6]$, $\mathbf{p}^+ = [18, 26, 26, 41, 19, 25, 31, 15, 32, 21]$ and the job due dates as $\mathbf{d} = [21, 12, 30, 20, 35, 26, 21, 36, 10, 26]$, respectively. Then, Table 3 shows the detailed iterations for a given solution $x = [6, 3, 7, 8, 5, 1, 2, 4, 9, 10]$.

Before the main loops (i.e., Iteration No. 0, denoted by "Iter. No". in column 1), $ST_x = 215$, $ST_y = 168$ in Line 3 (denoted by "L3"), then $r_c(x) = 215 - 168 = 47$. In iteration No.13, since $ST_y = 512 \ge ST1_{y1} = 512$, $ST_y = ST1_{y1} - 1 = 512 - 1 = 511$, $r_c(x) = ST_x - ST_y = 931 - 511 = 420$. In iteration No.15, since $R0 = 420 < r_c(x) = 421$, $\max\{\mathcal{R}\}$ after Line 8 (denoted by "L8") is 420. Algorithm 1 terminates with the current maximum regret value $R_{max} = 420$ as its output.

Table 3
An illustrative example for Algorithm 1.

Iter. No.	$ST1_x$	$ST1_{y1}$	$R0$	$\max\{\mathcal{R}\}$ in L7	$\max\{\mathcal{R}\}$ after L8	ST_x	ST_y in L3/L11	$r_c(x)$
0	–	–	–	–	–	215	168	47
1	299	252	47	47	47	299	207	92
2	1102	1010	92	92	92	1102	843	259
3	642	383	259	259	259	642	345	297
4	690	393	297	297	297	690	369	321
5	864	543	321	321	321	864	494	370
6	841	471	370	370	370	841	437	404
7	789	385	404	404	404	789	384	405
8	847	442	405	405	405	847	433	414
9	849	435	414	414	414	849	434	415
10	844	429	415	415	415	844	427	417
11	918	501	417	417	417	918	501	418
12	931	513	418	418	418	931	512	419
13	931	512	419	419	419	931	512	420
14	931	511	420	420	420	931	512	421
15	931	511	**420**	**421**	**420**	–	–	–

"L": represents Line in Algorithm 1.

4. 2-approximation algorithm and heuristics

In this section, the discussion of the 2-approximation algorithm and heuristics is started by describing a procedure inspired by Kasperski and Zieliński (2008) to find the completion time of a tardy job in a given sequence. For a given π, the following notations are defined:

- Y: the set of all tardy jobs, i.e. $Y = \{i : C_i > d_i\ (i \in \mathcal{N})\}$;
- X: the set of all non-tardy jobs, i.e. $X = \{i : C_i \le d_i\ (i \in \mathcal{N})\}$;
- Y_π: the job sequence in π of set Y;
- $|Y_\pi|$: the total number of tardy jobs in sequence Y_π;
- X_π: the job sequence in π of set X;

- i_{Y_π}: the position that is occupied by job i in the sequence Y_π;
- i'_{Y_π}: the position of a tardy job i' in the sequence Y_π, which is in the nearest right side position of non-tardy job i in schedule π.

Then, the total tardiness of the schedule π is:

$$z^s(\pi) = \sum_{i\in Y_\pi}(|Y_\pi| - i_{Y_\pi} + 1)p_i^s + \sum_{i\in X_\pi}(|Y_\pi| - i'_{Y_\pi} + 1)p_i^s - \sum_{i\in Y_\pi} d_i \tag{7}$$

where $\sum_{i\in Y_\pi}(|Y_\pi| - i_{Y_\pi} + 1)p_i^s + \sum_{i\in X_\pi}(|Y_\pi| - i'_{Y_\pi} + 1)p_i^s$ is the completion time of a tardy job i in the sequence π.

For example, consider a 6-job problem where $Y = \{1, 5, 4\}$ and $X = \{2, 3, 6\}$ and $\pi = [2, 5, 6, 4, 3, 1]$, then $Y_\pi = \{5, 4, 1\}$ and $X_\pi = \{2, 6, 3\}$, and

$5_{Y_\pi} = 1$; $4_{Y_\pi} = 2$; $1_{Y_\pi} = 3$.

$2'_{Y_\pi} = 5_{Y_\pi} = 1$; $6'_{Y_\pi} = 4_{Y_\pi} = 2$; $3'_{Y_\pi} = 1_{Y_\pi} = 3$.

Hence, the total tardiness is calculated as:

$$\begin{aligned} z^s(\pi) =& (3-1+1)p_5^s + (3-2+1)p_4^s + (3-3+1)p_1^s + (3-1+1)p_2^s \\ &+ (3-2+1)p_6^s + (3-3+1)p_3^s - (d_5 + d_4 + d_1) \end{aligned}$$

4.1. Properties of the mid-point and worse case scenarios

For describing the proposed approximation and heuristics in this section, some properties of the mid-point and worse case scenarios are first proved. For this purpose, let L, U and M be the scenarios for processing times with p_j^-, p_j^+ and $\lfloor 0.5 \times (p_j^- + p_j^+) \rfloor$ for all $j \in \mathcal{N}$, respectively. Then, the following lemma related to the property of the optimal mid-point solution can be proved.

Lemma 1. *Let x^-, x^+, and x be an optimal schedule under scenarios L, U, and M, respectively, then for any feasible schedule π, $r(x) \le 2r(\pi)$ and this bound is tight. However, $r(x^-) \le 2r(\pi)$ and $r(x^+) \le 2r(\pi)$ do not necessarily hold.*

Proof. For a given schedule π, if all jobs are tardy, the total tardiness is equivalent to the total completion time. Then using Theorem 1 in Kasperski and ZielińSki (2008), it follows that $r(x) \le 2r(\pi)$. On the other hand, if no jobs in the given schedule π are tardy, the objective function is 0. Hence, $r(x) \le 2r(\pi)$. Therefore, Lemma 1 is only needed to prove for the cases where the given schedule π has at least one tardy job. To do so, let:

$$\begin{aligned} A_1 =& \sum_{\{i: i\in Y_\pi, |Y_\pi|-i_{Y_\pi} > |Y_x|-i_{Y_x}\}} (|Y_\pi| - i_{Y_\pi} - |Y_x| + i_{Y_x})p_i^+ \\ &+ \sum_{\{i: i\in Y_\pi, |Y_\pi|-i_{Y_\pi} > |Y_x|-i'_{Y_x}\}} (|Y_\pi| - i_{Y_\pi} - |Y_x| + i'_{Y_x})p_i^+ \\ &+ \sum_{\{i: i\in X_\pi, |Y_\pi|-i'_{Y_\pi} > |Y_x|-i_{Y_x}\}} (|Y_\pi| - i'_{Y_\pi} - |Y_x| + i_{Y_x})p_i^+ \\ &+ \sum_{\{i: i\in X_\pi, |Y_\pi|-i'_{Y_\pi} > |Y_x|-i'_{Y_x}\}} (|Y_\pi| - i'_{Y_\pi} - |Y_x| + i'_{Y_x})p_i^+ \\ &+ \sum_{\{i: i\in Y_\pi, |Y_\pi|-i_{Y_\pi} < |Y_x|-i_{Y_x}\}} (|Y_\pi| - i_{Y_\pi} - |Y_x| + i_{Y_x})p_i^- \\ &+ \sum_{\{i: i\in Y_\pi, |Y_\pi|-i_{Y_\pi} < |Y_x|-i'_{Y_x}\}} (|Y_\pi| - i_{Y_\pi} - |Y_x| + i'_{Y_x})p_i^- \\ &+ \sum_{\{i: i\in X_\pi, |Y_\pi|-i'_{Y_\pi} < |Y_x|-i_{Y_x}\}} (|Y_\pi| - i'_{Y_\pi} - |Y_x| + i_{Y_x})p_i^- \\ &+ \sum_{\{i: i\in X_\pi, |Y_\pi|-i'_{Y_\pi} < |Y_x|-i'_{Y_x}\}} (|Y_\pi| - i'_{Y_\pi} - |Y_x| + i'_{Y_x})p_i^- \end{aligned}$$

$$\begin{aligned} A_2 =& \sum_{\{i: i\in Y_\pi, |Y_\pi|-i_{Y_\pi} > |Y_x|-i_{Y_x}\}} (|Y_x| + i_{Y_x} - |Y_\pi| - i_{Y_\pi})p_i^- \\ &+ \sum_{\{i: i\in Y_\pi, |Y_\pi|-i_{Y_\pi} > |Y_x|-i'_{Y_x}\}} (|Y_x| + i'_{Y_x} - |Y_\pi| - i_{Y_\pi})p_i^- \\ &+ \sum_{\{i: i\in X_\pi, |Y_\pi|-i'_{Y_\pi} > |Y_x|-i_{Y_x}\}} (|Y_x| + i_{Y_x} - |Y_\pi| - i'_{Y_\pi})p_i^- \\ &+ \sum_{\{i: i\in X_\pi, |Y_\pi|-i'_{Y_\pi} > |Y_x|-i'_{Y_x}\}} (|Y_x| + i'_{Y_x} - |Y_\pi| - i'_{Y_\pi})p_i^- \\ &+ \sum_{\{i: i\in Y_\pi, |Y_\pi|-i_{Y_\pi} < |Y_x|-i_{Y_x}\}} (|Y_x| + i_{Y_x} - |Y_\pi| - i_{Y_\pi})p_i^+ \\ &+ \sum_{\{i: i\in Y_\pi, |Y_\pi|-i_{Y_\pi} < |Y_x|-i'_{Y_x}\}} (|Y_x| + i'_{Y_x} - |Y_\pi| - i_{Y_\pi})p_i^+ \\ &+ \sum_{\{i: i\in X_\pi, |Y_\pi|-i'_{Y_\pi} < |Y_x|-i_{Y_x}\}} (|Y_x| + i_{Y_x} - |Y_\pi| - i'_{Y_\pi})p_i^+ \\ &+ \sum_{\{i: i\in X_\pi, |Y_\pi|-i'_{Y_\pi} < |Y_x|-i'_{Y_x}\}} (|Y_x| + i'_{Y_x} - |Y_\pi| - i'_{Y_\pi})p_i^+ \end{aligned}$$

$$B_1 = \sum_{i\in Y_x\setminus Y_\pi} d_i - \sum_{i\in Y_\pi\setminus Y_x} d_i.$$

and

$$B_2 = \sum_{i\in Y_\pi\setminus Y_x} d_i - \sum_{i\in Y_x\setminus Y_\pi} d_i.$$

Now, from Eq. (7), for any two schedules π, x and a scenario s of processing times, the following equation holds:

$$\begin{aligned} z^s(\pi) - z^s(x) =& \sum_{i\in Y_\pi}(|Y_\pi| - i_{Y_\pi} + 1)p_i^s + \sum_{i\in X_\pi}(|Y_\pi| - i'_{Y_\pi} + 1)p_i^s \\ &- \Big(\sum_{i\in Y_x}(|Y_x| - i_{Y_x} + 1)p_i^s + \sum_{i\in X_x}(|Y_x| - i'_{Y_x} + 1)p_i^s\Big) \\ &+ \sum_{i\in Y_x\setminus Y_\pi} d_i - \sum_{i\in Y_\pi\setminus Y_x} d_i \end{aligned} \tag{8}$$

Simplifying Eq. (8) by using the definitions of A_1 and B_1, the following formula holds:

$$r(\pi) = z^{s_\pi}(\pi) - z^{s_\pi}(*) \ge A_1 + B_1 \tag{9}$$

Similarly, using Eq. (8) for the worst-case scenario of a solution x, s_x, the following inequality holds:

$$z^{s_x}(x) - z^{s_x}(\pi) \le A_2 + B_2 \tag{10}$$

Subtracting $z^{s_x}(*)$ from both sides of (10) together with the definition of maximum regret of any schedule π, $r(\pi) \ge z^{s_\pi}(\pi) - z^{s_\pi}(*)$ and noting that $z^{s_x}(*) \ge z^{s_\pi}(*)$, the following inequality can be obtained:

$$r(x) \le r(\pi) + A_2 + B_2 \tag{11}$$

Since x is an optimal schedule under the mid-point scenario M, $z^M(\pi) - z^M(x) \ge 0$. Therefore, the following relationship can be obtained:

$$\begin{aligned} &\sum_{i\in Y_\pi}(|Y_\pi| - i_{Y_\pi} + 1)(p_i^+ + p_i^-) + \sum_{i\in X_\pi}(|Y_\pi| - i'_{Y_\pi} + 1)(p_i^+ + p_i^-) \\ &- \Big(\sum_{i\in Y_x}(|Y_x| - i_{Y_x} + 1)(p_i^+ + p_i^-) + \sum_{i\in X_x}(|Y_x| - i'_{Y_x} + 1)(p_i^+ + p_i^-)\Big) \\ &+ 2\sum_{i\in Y_x\setminus Y_\pi} d_i - 2\sum_{i\in Y_\pi\setminus Y_x} d_i \ge 0 \end{aligned} \tag{12}$$

Then, expanding and simplifying the left-hard side of inequality (12), using the definitions of A_1, A_2, B_1, and B_2 shows that:

$$A_1 + B_1 \ge A_2 + B_2 \tag{13}$$

Combining Eq. (9) together with inequality (13) and noting that $r(\pi) \ge z^{s_\pi}(\pi) - z^{s_\pi}(x)$ yields:

$$r(\pi) \ge A_1 + B_1 \ge A_2 + B_2 \tag{14}$$

Table 4
The pattern of solutions and worst-case scenarios for an illustrative example.

j	n = 3			n = 4			n = 5			n = 6			n = 7		
	p_j^-	p_j^+	d_j	p_j^-	p_j^+	d_j	p_j^-	p_j^+	d_j	p_j^-	p_j^+	d_j	p_j^-	p_j^+	d_j
1	2	9	16	2	9	16	2	9	16	2	9	16	2	9	16
2	4	11	15	4	11	15	4	11	15	4	11	15	4	11	15
3	3	6	10	3	6	10	3	6	10	3	6	10	3	6	10
4				3	6	13	3	6	13	3	6	13	3	6	13
5							4	10	12	4	10	12	4	10	12
6										5	13	18	5	13	18
7													5	14	13
x^*	[3, 1, 2]			[3, 4, 1, 2]			[3, 4, 1, 5, 2]			[3, 4, 1, 5, 2, 6]			[3,4,1,5,2,6,7]		
$p_j^{\sigma(x)}$	{9, 4, 6}			{9, 4, 6, 6}			{9, 4, 6, 6, 4}			{9, 4, 6, 6, 10, 5}			{9, 4, 6, 6, 10, 5, 5}		
$Rt(x^*)$	1			5			12			20			31		

Now, inequalities (11) and (14) imply $r(x) \le 2r(\pi)$. Since the optimal schedule $(*)$ could be any feasible schedule π, its maximum regret value of x is at most double of $r(*)$.

To show that this bound is tight, consider the problem instance with three jobs and let $\mathbf{p}^- = [2,3,2]$, $\mathbf{p}^+ = [4,4,2]$ and the due dates of the jobs $\mathbf{d} = [3,3,3]$. Under the mid-point scenario M, the schedule $x = [1,3,2]$ has the maximal regret value $r(x) = 2$. For the problem, the minimum maximal regret value is $r(*) = 1$, which is obtained by the schedule $\pi = [3,1,2]$. Thus, $r(x) = 2r(\pi)$.

Now the scenarios L and U are considered. Using Eqs. (9) and (10) for scenario L, it follows that $r(x^-) \le 2r(\pi) + (A_2 + B_2) - (A_1 + B_1) = 2r(\pi) + z^M(x^-) - z^M(\pi)$. Therefore, $r(x^-) \le 2r(\pi)$ for some schedules that satisfy $z^M(x^-) - z^M(\pi) \le 0$. However, $r(x^-) \le 2r(\pi)$ does not necessarily hold when $z^M(x^-) - z^M(\pi) > 0$ for other schedules. As an example, let $\mathbf{p}^- = [2,3,3]$, $\mathbf{p}^+ = [4,5,3]$ and $\mathbf{d} = [3,3,3]$. The optimal schedule $x^- = [1,2,3]$ under scenario L has the maximal regret value $r(x^-) = 3$ and the optimal schedule $x^+ = [1,3,2]$ under scenario U has the maximal regret value $r(x^+) = 2$, while the optimal schedule $[3,1,2]$ has the min–max regret value $r(*) = 1$. Therefore $r(x^-) \le 2r(*)$ does not necessarily hold.

Similar arguments are also valid for scenario U and its corresponding optimal sequence x^+. Therefore, $r(x^-) \le 2r(\pi)$ and $r(x^+) \le 2r(\pi)$ do not necessarily hold for scenarios L and U. ■

Theorem 1. *If the deterministic total tardiness single machine problem is polynomially solvable, then the optimal schedule under the mid-point scenario is a polynomial-time 2-approximation for the IDMR-SMSPTT.*

Proof. Let x^* be the optimal solution to the problem IDMR-SMSPTT and let x be an optimal schedule under the mid-point scenario M. Then from Lemma 1 above, it follows that $r(x) \le 2r(x^*)$. ■

Theorem 2. *For the IDMR-SMSPTT problem, the worst-case scenario $\sigma(x)$ of any solution $x \in \mathcal{X}$ does not necessarily occur at the upper or lower limits of the interval processing times.*

Proof. This theorem can be proved by contradiction with a counter example containing $n = 6$ jobs. Let $\mathbf{p}^- = [6,6,5,4,3,8]$, $\mathbf{p}^+ = [12,9,11,8,15,18]$ and the due dates $\mathbf{d} = [16,30,21,41,10,13]$. Let a solution $x = [6,3,5,1,2,4]$. If the worst-case scenario is only possible on one of the extreme scenarios with job processing times only at the lower or upper limits of the interval, it can be found that for the scenario $\sigma(x) = \{12,6,11,4,15,18\}$ among all extreme scenarios, the solution x has the maximum regret $r^{\sigma(x)}(x) = 51$. However, for the same solution x, regret $r^s(x) = 55 > 51$ can be obtained for the scenario $s = \{6,9,11,5,10,18\}$, which contradicts the assumption, since $p_4^s = 5, 4 = p_4^- < 5 < p_4^+ = 8$ and $p_5^s = 10, 3 = p_5^- < 10 < p_5^+ = 15$. ■

4.2. A 2-approximation algorithm

The results of Theorem 1 can be used to describe an algorithm to find an approximate solution to the IDMR-SMSPTT. However, since the corresponding problem for the mid-point scenario is NP-hard, such an algorithm may require excessive computational effort. Therefore, in the proposed 2-approximation algorithm (called Algorithm 4), the M-NBR method developed by Russell and Holsenback (1997) is used to obtain the feasible schedule x under the mid-point scenario M.

Algorithm 2 The 2-approximation algorithm (2AA)

1: **Input:** Problem data including p_j^+, p_j^-, d_j and n.
2: Let $p_j^M = \lfloor 0.5 \times (p_j^- + p_j^+) \rfloor$, $j \in \mathcal{N}$.
3: Apply the M-NBR method to obtain the feasible schedule x under the mid-point scenario M of processing times.
4: Call Algorithm 1.
5: **Output:** The maximum regret R_{max} and the optimal maximum regret Rt.

After inputting the problem data (Line 1), the mid-point scenario of processing times of jobs can be set (Line 2) and a feasible solution x by using the M-NBR method can be obtained (Line 3). Then Algorithm 1 is used to obtain the maximum regret R_{max} for the solution x (Line 4). Finally, according to Lemma 1, an estimated value for the optimal maximum regret Rt can be obtained.

4.3. A worst-case scenario-based heuristic

Although, as shown by Theorem 2, the worst-case scenario is not necessarily on p_j^- or p_j^+, it may have some pattern. To obtain some observations about the patterns of solutions and worst-case scenarios, preliminary computational experiments were conducted. Problem instances with $n = \{3,4,5,6,7\}$ are solved optimality with model [XMR] by CPLEX 12.8. Table 4 presents the pattern of solutions and worst-case scenarios for an illustrative example, in which the interval of processing times $[p_j^-, p_j^+]$, due date d_j, optimal schedule x^*, worst-case scenario $p_j^{\sigma(x)}$, and the corresponding min–max regret $Rt(x^*)$ are listed.

For example, $\mathcal{X} = \{\{1,2,3\},\{1,3,2\},\{2,3,1\},\{2,1,3\},\{3,1,2\},\{3,2,1\}\}$, represents 6 solutions for $n = 3$, i.e., p_1 can take 8 values (i.e., 2, 3, ⋯, 9), p_2 can take 8 values (i.e., 4, 5, ⋯, 11), and p_3 can take 4 values (i.e., 3, 4, 5, 6). There are $8 \times 8 \times 4 = 256$ scenarios in total. Let a schedule $x_1 = \{1,2,3\}$ and a scenario $s_1 = [9,11,6]$. Then the total tardiness of x_1 under s_1 is $0+(9+11-15)+(9+11+6-10) = 21$ due to $\mathbf{d} = [16,15,10]$. Since the optimal total tardiness of s_1 is 11 corresponding to the solution $\{3,1,2\}$, then the regret of x_1 under s_1 is $21 - 11 = 10$. Next, the regret values of x_1 in all 256 scenarios are calculated, such that the maximum regret value of the solution x_1 can be obtained. The same method is used to calculate the maximum regret values of other solutions in all scenarios. Finally, the solution corresponding to the minimum maximum regret value can be obtained by comparing the maximum regret values of all solutions. This is an optimal solution.

Based on the results in Table 4, the following pattern can be obtained: (i) the optimal schedule of $m \le n$ jobs is related to the optimal schedule of m-1 jobs (i.e., the sequence of m-1 jobs are kept

in m jobs), and (ii) the worst-case scenarios have processing times at the lower and upper limits of the interval. Although it has been shown in Theorem 2 that the worst-case scenario is not necessarily on extreme values of processing times, the results of the preliminary computational experiments suggest that in many cases, the worst-case scenarios occur with extreme values of the interval processing times.

By combining the pattern of the optimal solution and the worst-case scenario mentioned above, a worst-case scenario-based heuristic (WSH) is proposed where, in order to restrict a scenario to take the extreme values of the interval processing times, the processing times of scenario s are defined as:

$$p_j^s = \alpha_j p_j^- + (1-\alpha_j)p_j^+ \quad j \in \mathcal{N}; \tag{15a}$$

$$\alpha_j \in \{0,1\}; \tag{15b}$$

The detailed steps of the WSH are shown in Algorithm 3.

After initialization (Line 1), the [RSST] model is solved directly for the first three jobs to obtain the optimal solution x^* with the min–max regret value for these three jobs (Line 2). Then, the loop begins (Lines 3–11). Combining the observation of the pattern of optimal solutions, insert an unscheduled job into x^*, which leads to $m = m+1$ possible solutions.

For example, for a problem with $n = 6$, when $m = 3$, $x^* = [1,2,3]$. Let $m \to m+1 = 4$, an unscheduled job, say job 5, is inserted into x^*, which will lead to four possible partial solutions: $[1,2,3,5]$, $[1,2,5,3]$, $[1,5,2,3]$ and $[5,1,2,3]$, by keeping the sequence of x^* for $m = 3$ unchanged. Then, put these partial solutions into $\mathcal{P}$ and use Lemmas A.1–A.4 in the Appendix to determine whether some partial solutions can be deleted (Lines 5–8). Next, for each solution in $\mathcal{P}$, call Algorithm 1 to obtain the maximum regret value R_{max}. Before calling Algorithm 1, the constraints (15a) and (15b) are added into [XMR1] to combine the observation of the pattern on the worst-case scenario mentioned above, and the optimal solution x^* for the current job set can be obtained. Finally, out of the loop, the minimum maximum regret value $Rt(x^*)$ and its corresponding solution x^* can be obtained.

Algorithm 3 The Worst-case Scenario-based heuristic (WSH)

1: **Input:** Problem data including p_j^+, p_j^-, d_j, and n. Let $m = 3$. Let $\mathcal{P} = \emptyset$ be the set of solutions and $\mathcal{R} = \emptyset$ be the set of regret values.
2: Solve the [RSST] model directly to obtain the optimal solution x^*, since it is easy to solve [RSST] with CPLEX for the first three jobs.
3: **while** $m \le n-1$ **do**
4: $m+1 \to m$.
5: Put an unscheduled job into x^*, which leads to m possible partial solutions and put these solutions into $\mathcal{P}$.
6: **if** $x \in \mathcal{P}$ does not satisfy Lemmas A.1–A.4 **then**
7: $\mathcal{P} \setminus x \to \mathcal{P}$.
8: **end if**
9: With each $x \in \mathcal{P}$, call Algorithm 1. Before calling Algorithm 1, the constraints (15a) and (15b) are added into the [XMR1] model. Obtain the regret value R_{max} as the output of Algorithm 1 of each x, $\mathcal{R} \cup R_{max} \to \mathcal{R}$. $Rt(x^*) = \min\{\mathcal{R}\}$, and the corresponding best solution x^*.
10: $\mathcal{R} = \emptyset$.
11: **end while**
12: **Output:** $Rt(x^*)$ and the corresponding best solution x^*.

4.4. An approximate worst-case-based heuristic

For a solution x, a maximizer y in $r(x) = \max_{y \in \mathcal{X}} \max_{s \in \mathcal{S}}(z^s(x) - z^s(y))$ is a worst-case approximation for x. Based on this, an approximate worst-case-based heuristic (Algorithm 4) can be developed. For a solution $x \in \mathcal{P}$, where $\mathcal{P}$ is the set of any random feasible solutions, the basic idea is to find an approximate worst-case y for x and the corresponding maximum regret R_{max} for x by Algorithm 1 (Line 3), then to find an approximate worst-case alternative for y and the corresponding maximum regret R_{max} for y by Algorithm 1, where $\mathcal{P}$ is a set of feasible solutions. The process iterates until the set of solutions $\mathcal{P}$ is empty (Lines 2–10).

Algorithm 4 Approximate worst-case-based heuristic (AWH)

1: **Input:** Problem data including p_j^+, p_j^-, d_j, and n, set of the initial feasible solutions $\mathcal{P}_0$. Let the set of regret $\mathcal{R} = \emptyset$ and the set of any random feasible solutions $\mathcal{P} \leftarrow \mathcal{P}_0$.
2: **while** $\mathcal{P}$ is not empty **do**
3: For a solution $x \in \mathcal{P}$, call Algorithm 1 to obtain the maximum regret R_{max} and the corresponding solution y.
4: **if** $y \notin \mathcal{P}$ **then**
5: $\mathcal{P} \cup y \to \mathcal{P}$.
6: **end if**
7: $\mathcal{R} \cup \{R_{max}\} \to \mathcal{R}$, $\mathcal{P} \setminus x \to \mathcal{P}$
8: Go to Line 3.
9: **end while**
10: **Output:** $Rt = \min\{\mathcal{R}\}$ and the corresponding solution x.

5. Computational experiments and results

In this section, the results from the empirical experiments are reported and they are used to evaluate the effectiveness of the proposed approximation and heuristic algorithms. To perform these experiments, MATLAB 2016a is used to implement the proposed methods (i.e., the 2AA, WSH, and AWH) on a computer with a 3.5 GHz AMD-5350M processor and 4 GB of RAM running the windows 7 operating system.

5.1. Experimental design

Problem instances with different sizes were randomly generated using a discrete uniform distribution since such problem instances contain some of the hardest ones to solve (Muth & Thompson, 1963). Further, the use of simulated job processing times and due dates may generate problem instances that are harder to solve than those found in real-life situations (Amar & Gupta, 1986). The generated problem instances are classified in two sets: (i) small problem instances with $n = \{10,12,14,15\}$, and (ii) large problem instances with $n = \{20,30,50\}$. The intervals of the job processing times and due dates were randomly generated as follows inspired by Lu et al. (2014):

(1) The lower bound of the processing time p_j^- of job j was randomly generated from the discrete uniform distribution of integers in the interval $[1, \lfloor 50\beta_1 \rfloor]$, where $\beta_1 \in \{0.2, 0.6\}$.
(2) The upper bound of the processing time p_j^+ of job j was randomly generated from the discrete uniform distribution of integers over $[p_j^- + 1, \lfloor p_j^-(1+\beta_2) \rfloor]$, where $\beta_2 \in \{0.2, 0.6\}$.
(3) The due date d_j of job j was randomly generated from the discrete uniform distribution of integers over $[\lfloor \beta_3 p_j^- \rfloor, \lfloor \beta_3 p_j^+ \rfloor]$, where $\beta_3 \in \{1,3\}$.

The intervals of job processing times and due dates are generated from the uniform distributions with parameters β_1, β_2, and β_3, so it is desired to learn the impact of different values of these parameters on the performance measure. Note that the magnitude and interval length of the job processing times are determined by β_1 and β_2, while all three parameters β_1, β_2 and β_3 affect the due dates. The impact of different degrees of uncertainty on the performance measure are explored.

In the following experiments, ten replicated tests were conducted for each combination of n, β_1, β_2, and β_3, resulting in a total of 560 ($7 \times 2 \times 2 \times 2 \times 10$) problem instances. There are 80 instances for each n. The average performance measures over the ten replications were computed for each combination of parameters.

To solve the problem instances using MILP, CPLEX 12.8 is used, and 3600 CPU seconds (1 h) are used as the time limit for each problem instance with $n = \{10,12,14,15\}$. Intuitively, as n increases, the CPU time required to solve a problem instance increases. The

Table 5
Comparative results with $n \leq 15$.

n	β_1	β_2	β_3	Average min–max regret ($\overline{Rt}$)			Average CPU time (s)			Std_{Rt}		
				2AA	WSH	AWH	2AA	WSH	AWH	2AA	WSH	AWH
10	0.2	0.2	1	**49.2**	50.3	50.0	410.2	1326.7	730.0	3.64	3.50	3.54
			3	**46.5**	48.2	47.9	423.2	1400.2	801.2	2.81	2.93	2.75
		0.6	1	**51.3**	53.4	52.0	451.6	1426.3	836.3	5.18	7.42	8.19
			3	**47.5**	48.9	48.5	462.3	1510.2	908.5	4.74	5.31	5.49
	0.6	0.2	1	**52.3**	53.3	53.0	446.3	1475.2	877.2	7.93	7.52	7.51
			3	**49.6**	52.2	50.2	473.5	1540.0	939.8	7.03	6.20	6.82
		0.6	1	**55.2**	59.9	56.3	495.5	1601.3	1021.2	7.52	7.56	7.45
			3	**53.2**	56.6	55.5	520.0	1689.9	1182.9	6.63	6.79	7.18
12	0.2	0.2	1	**61.1**	–	64.3	869.2	>1 h	>1 h	6.95	–	8.30
			3	**58.3**	–	62.3	900.3	>1 h	>1 h	6.88	–	7.68
		0.6	1	**63.3**	–	67.0	951.6	>1 h	>1 h	8.52	–	9.24
			3	**59.5**	–	63.5	978.2	>1 h	>1 h	8.46	–	9.31
	0.6	0.2	1	**64.7**	–	68.8	963.6	>1 h	>1 h	13.37	–	10.39
			3	**61.7**	–	65.5	992.3	>1 h	>1 h	9.66	–	9.62
		0.6	1	**67.9**	–	72.0	1059.4	>1 h	>1 h	11.16	–	11.95
			3	**65.5**	–	70.0	1127.0	>1 h	>1 h	10.62	–	11.12
14	0.2	0.2	1	**83.3**	–	87.8	1269.2	>1 h	>1 h	9.77	–	13.16
			3	**80.9**	–	85.6	1423.2	>1 h	>1 h	8.99	–	9.98
		0.6	1	**86.6**	–	91.3	1499.9	>1 h	>1 h	12.43	–	13.28
			3	**82.4**	–	86.5	1521.1	>1 h	>1 h	9.95	–	10.76
	0.6	0.2	1	**87.2**	–	93.3	1463.6	>1 h	>1 h	12.64	–	13.47
			3	**83.3**	–	88.5	1492.9	>1 h	>1 h	11.69	–	10.29
		0.6	1	**90.0**	–	98.3	1601.0	>1 h	>1 h	11.93	–	13.64
			3	**87.5**	–	92.4	1710.2	>1 h	>1 h	11.82	–	12.66
15	0.2	0.2	1	**106.0**	–	126.3	1699.3	>1 h	>1 h	17.31	–	17.97
			3	**96.6**	–	120.2	1800.3	>1 h	>1 h	11.21	–	15.67
		0.6	1	**108.2**	–	132.3	1929.3	>1 h	>1 h	18.66	–	19.44
			3	**99.3**	–	125.6	1896.7	>1 h	>1 h	14.09	–	14.99
	0.6	0.2	1	**117.1**	–	142.2	1888.9	>1 h	>1 h	16.48	–	17.50
			3	**110.1**	–	131.0	1900.0	>1 h	>1 h	6.13	–	9.07
		0.6	1	**120.5**	–	150.0	2010.2	>1 h	>1 h	10.16	–	13.65
			3	**115.5**	–	136.6	2152.3	>1 h	>1 h	11.38	–	12.37

–: No objective values can be obtained by WSH within the time limit of 1 h.

results of the preliminary computational experiments confirmed that the computation time of the methods for the problem instances with $n \leq 15$ is not enough for the ones with $n \geq 20$. Therefore, 7200 CPU seconds (2 h) are set as the time limit for each problem instance with $n \geq 20$. If a problem instance cannot be solved to optimality within its CPU time limit, the current best solution will be used for computation of the average minimum of maximum regret. The parameter $tlim1$ in Algorithm 1 is set as 900 CPU seconds for problem instances with $n \leq 15$, 1800 CPU seconds for problem instances with $n = \{20, 30\}$, and 3600 CPU seconds for problem instances with $n = 50$, respectively. The parameter $tlim$ in Algorithm 1 is set as 3600 CPU seconds for each problem instance.

5.2. Experimental results

Over 10 problem instances, for each combination of $n \leq 15$ and $\{\beta_1, \beta_2, \beta_3\}$, Table 5 presents the average min–max regret value (denoted by $\overline{Rt}$), the average CPU time, and the standard deviation (denoted by Std_{Rt}) for each of the three methods (i.e., 2AA, WSH, and AWH). These results show that the 2AA performs best compared to the WSH and AWH in terms of the average min–max regret value, the average CPU time, and the standard deviation. The WSH performs worst. For the problem instances with $n \geq 12$, the WSH could not obtain the objective value within the time limit of 1h. This may be because that the WSH starts with three jobs and adds jobs in the sequence iteratively, which takes too much computation effort before it can obtain a complete feasible solution. As for the AWH, it can obtain the min–max regret value for all problem instances even though for $n \geq 10$, the CPU time limit is exceeded.

The results in Table 5 also show that, for a given n, $\overline{Rt}$ is largest for the combination of $\{\beta_1, \beta_2, \beta_3\} = \{0.6, 0.6, 1\}$ for both 2AA and AWH. This is not unexpected as compared to the other combinations, since $\{\beta_1, \beta_2, \beta_3\} = \{0.6, 0.6, 1\}$ implies a higher uncertainty in processing times and tighter due dates of the jobs. This may lead to larger min–max regret values. Correspondingly, $\overline{Rt}$ is smallest for the combination of $\{\beta_1, \beta_2, \beta_3\} = \{0.2, 0.2, 3\}$.

Table 6 presents the comparisons of the 2AA and AWH for problem instances with $n = \{20, 30, 50\}$. Since the WSH cannot obtain the objective values within 1h for problem instances with $n \geq 12$, the results of the WSH are not listed in Table 6. As in Table 5, similar patterns of the average min–max regret ($\overline{Rt}$) and the standard deviation (Std_{Rt}) are found for problem instances with $n = \{20, 30, 50\}$. That is, for a given n, $\overline{Rt}$ is largest and smallest for the combination of $\{\beta_1, \beta_2, \beta_3\} = \{0.6, 0.6, 1\}$ and $\{\beta_1, \beta_2, \beta_3\} = \{0.2, 0.2, 3\}$, respectively for both 2AA and AWH. The 2AA outperforms the AWH in terms of the average min–max regret, the CPU time, and the standard deviation. The AWH exceeds the time limit of 2h for all problem instances with $n = \{20, 30, 50\}$, while the 2AA can obtain much better min–max regret values with much smaller computational effort.

To confirm the finding reported above, the problem instances generated in the computational experiments are used to test for any significant differences in the performance of the proposed methods. Since the heuristic WSH could not solve problem instances with more than 10 jobs, the statistical analysis are restricted to the 2AA and AWH only. The availability of the large number of problem instances (320 with $n \leq 15$ and 240 with $n \geq 20$), enables us to use the paired t-test where the null hypothesis, H_0 states that $\mu_{2AA} \geq \mu_{AWH}$ where μ_h represents the average min–max regret value obtained by using heuristic h. The alternative hypothesis, H_1 states that $\mu_{2AA} < \mu_{AWH}$.

Table 7 shows the results of the paired t-test with means, standard deviations, lower and upper limits of the $\mu_{2AA} - \mu_{AWH}$ values at the 99% confidence level, the calculated t values, and the critical t values at the 99% confidence level for $n \leq 15$ and $n \geq 20$, respectively. Since, the absolute calculated values of t in Table 7 are much greater than the critical t value, the null hypothesis can be rejected with $p = 0.001$.

Table 6
Comparative results with $n \geq 20$.

n	β_1	β_2	β_3	Average min–max regret ($\overline{Rt}$)		Average CPU time (s)		Std_{Rt}	
				2AA	AWH	2AA	AWH	2AA	AWH
20	0.2	0.2	1	**155.9**	188.8	2856.3	>2 h	15.77	18.41
			3	**145.2**	175.3	3000.6	>2 h	14.42	16.46
		0.6	1	**159.2**	193.3	3139.2	>2 h	21.41	27.29
			3	**149.7**	186.6	3226.2	>2 h	19.02	21.42
	0.6	0.2	1	**169.3**	210.3	3001.0	>2 h	26.73	28.24
			3	**161.7**	200.3	3120.4	>2 h	24.83	24.90
		0.6	1	**175.4**	219.9	3216.6	>2 h	28.24	30.21
			3	**166.6**	209.2	3300.1	>2 h	26.66	27.99
30	0.2	0.2	1	**205.3**	248.8	4521.3	>2 h	26.85	27.47
			3	**193.4**	236.7	4620.2	>2 h	24.74	26.73
		0.6	1	**210.2**	255.3	4712.2	>2 h	28.80	24.22
			3	**199.4**	246.6	4823.4	>2 h	14.02	16.23
	0.6	0.2	1	**215.6**	270.0	4659.6	>2 h	26.15	28.00
			3	**203.3**	261.2	4723.3	>2 h	22.97	20.05
		0.6	1	**229.7**	279.9	4853.9	>2 h	29.74	30.89
			3	**216.4**	266.6	4952.3	>2 h	22.77	28.38
50	0.2	0.2	1	**410.2**	531.4	6523.2	>2 h	23.82	30.91
			3	**402.2**	516.6	6720.4	>2 h	21.47	28.60
		0.6	1	**425.3**	560.2	6800.0	>2 h	26.17	29.66
			3	**420.0**	541.0	7010.0	>2 h	25.19	28.79
	0.6	0.2	1	**460.5**	570.3	6826.2	>2 h	27.64	32.61
			3	**435.2**	556.6	6953.5	>2 h	23.22	29.28
		0.6	1	**472.2**	576.6	7122.2	>2 h	28.94	38.95
			3	**440.2**	562.3	7156.7	>2 h	24.20	34.33

Table 7
Results of the t-Test: paired two sample for means.

Number of jobs n	Paired differences				Degrees of freedom	t-stat	Critical t	Sig. level (one-tail)
	Mean	Standard deviation	99% Confidence interval					
			Lower	Upper				
$n \leq 15$	−8.54	15.26	−10.75	−6.33	319	−10.02	1.65	0.001
$n \geq 20$	−68.42	52.99	−77.30	−59.54	239	−20.00	1.65	0.001

Thus, the results in Table 7 show that, statistically, the average performance of 2AA is significantly better than that of the AWH. Therefore, it can be concluded that the use of 2AA is relatively more effective in minimizing maximum regret than the use of AWH.

6. Conclusions

In this paper, a robust (min–max regret) single machine scheduling problem was studied, in which the objective is the minimization of the maximum regret value of total tardiness and the job processing times are uncertain and are represented by interval data. Given its NP-hard nature, the problem was first formulated as a mixed-integer linear program and then the worst-case scenario was proved to be not necessarily at the upper or lower limits of the interval processing times. The optimal schedule under the mid-point scenario was proved to be a 2-approximation algorithm for the problem provided that the deterministic counterpart problem can be solved optimally. Utilizing these results and integrating a procedure for estimating the maximum regret of a given solution, a 2-approximation algorithm and two heuristics were developed to solve the problem. The experimental results from empirical tests on a large set of randomly generated problem instances demonstrated that the proposed 2-approximation algorithm performs best in terms of both the average of the min–max regret values and the computation time.

The results reported in this paper suggest several issues for future research. First, additional solution approaches, especially some decomposition-based methods, could be explored to solve large-scale problem instances efficiently. Second, the proposed methods may be extended to the robust (min–max regret) versions of single machine scheduling problems with other objectives related to tardiness and earliness. Third, more complicated production scheduling problems, such as parallel-machine and flow-shop scheduling problems could be investigated in the form of robust optimization (the IDMR) for tardiness-related objective functions.

CRediT authorship contribution statement

Shijin Wang: Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing, Supervision, Funding acquisition. **Wenli Cui:** Conceptualization, Methodology, Formal analysis. **Feng Chu:** Writing - review & editing, Visualization, Formal analysis. **Jianbo Yu:** Writing - review & editing, Visualization. **Jatinder N.D. Gupta:** Writing - review & editing, Formal analysis.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 71571135 and 71971155. The work was also supported by the Fundamental Research Funds for the Central Universities, China.

Appendix. Dominance conditions

In this Appendix, the dominance conditions used to reduce the number of sequences examined in heuristic WSH are stated and proved. To do so, two solutions $x, y \in \mathcal{X}$ are considered, whose worst-case scenarios

are $\sigma(x)$ and $\sigma(y)$, respectively. Let the maximum total tardiness values of $x, y \in \mathcal{X}$ on the corresponding worst-case scenarios be $\bar{z}(x)$ and $\bar{z}(y)$, respectively.

Lemma A.1. *If $\bar{z}(x) \leq \bar{z}(y)$, and $p_j \in \sigma(x) \geq p_j \in \sigma(y)$ for all $j \in \mathcal{N}$, then solution x dominates y in terms of their min–max regret.*

Proof. If $p_j \in \sigma(x) \geq p_j \in \sigma(y)$ for all $j \in \mathcal{N}$, the optimal total tardiness in $\sigma(x)$ and $\sigma(y)$ have the relationship $z^{\sigma(x)}(*) \geq z^{\sigma(y)}(*)$. Since $\bar{z}(x) \leq \bar{z}(y)$, the maximum regret for x is $r(x) = \bar{z}(x) - z^{\sigma(x)}(*)$, which is no larger than the maximum regret for y, i.e., $r(y) = \bar{z}(y) - z^{\sigma(y)}(*)$. ■

In the remainder of this appendix, the Emmons' job dominance conditions for the $1 \parallel \sum T_i$ problem are extended to the interval processing times case. For this purpose, let $\mathcal{B}_i$ and $\mathcal{A}_i$ denote the sets of jobs known to be processed before job i and after job i, respectively. Define notation $j \prec k$ to denote job j precedes job k in the optimal sequence. For any set $\mathcal{Q}$, $\mathcal{Q} \subseteq \mathcal{N}$, $\mathcal{Q}' = \mathcal{N} \setminus \mathcal{Q}$ is its complement of $\mathcal{Q}$ in $\mathcal{N}$. Let $P(\mathcal{Q}) = \sum_{i \in \mathcal{Q}} p_i$, $P^+(\mathcal{Q}) = \sum_{i \in \mathcal{Q}} p_i^+$, and $P^-(\mathcal{Q}) = \sum_{i \in \mathcal{Q}} p_i^-$ for any set of jobs $\mathcal{Q}$. With these notations and definitions, the following results can be stated and proved.

Lemma A.2. *For any two jobs, j, k, with $p_j^+ \leq p_k^-$, if $d_j \leq \max\{d_k, P^-(\mathcal{B}_k) - p_k^+\}$, then $j \prec k$.*

Proof. If $p_j^+ \leq p_k^-$, then $p_j \leq p_k$. Since $\max\{d_k, P^-(\mathcal{B}_k) - p_k^+\} \leq \max\{d_k, P(\mathcal{B}_k) - p_k\}$, $d_j \leq \max\{d_k, P(\mathcal{B}_k) - p_k\}$, use of Emmons' rule No.1 for $1 \parallel \sum T_j$ (Emmons, 1969; Rinnooy Kan et al., 1975) shows that $j \prec k$. ■

Lemma A.3. *For any two jobs, j, k, if $d_k \geq \max\{d_j, P^+(\mathcal{A}'_j) - p_k^-\}$, then $j \prec k$.*

Proof. Since $\max\{d_k, P^+(\mathcal{A}'_j) - p_k^-\} \geq \max\{d_k, P(\mathcal{A}'_j) - p_k\}$, $d(k) \geq \max\{d_j, P(\mathcal{A}'_j) - p_k\}$, use of Emmons' rule No.2 for $1 \parallel \sum T_j$ (Emmons, 1969; Rinnooy Kan et al., 1975) shows that $j \prec k$. ■

Lemma A.4. *For any two jobs, j, k, if $d_k \geq P^+(\mathcal{A}'_j)$, then $j \prec k$.*

Proof. Since $P^+(\mathcal{A}'_j) \geq P(\mathcal{A}'_j)$ and $d_k \geq P(\mathcal{A}'_j)$, use of Emmons' rule No.1 for $1 \parallel \sum T_j$ (Emmons, 1969; Rinnooy Kan et al., 1975) shows that $j \prec k$. ■

References

Addis, B., Carello, G., & Tànfani, E. (2014). A robust optimization approach for the advanced scheduling problem with uncertain surgery duration in operating room planning-an extended analysis. https://hal.archives-ouvertes.fr/hal-00936019v2. (Last Accessed 26 June 2020).

Aissi, H., Bazgan, C., & Vanderpooten, D. (2005). Complexity of the min–max and min–max regret assignment problems. *Operations Research Letters*, *33*, 634–640.

Aissi, H., Bazgan, C., & Vanderpooten, D. (2009). Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, *197*, 427–438.

Amar, A. D., & Gupta, J. N. D. (1986). Simulated versus real life data in testing the efficiency of scheduling algorithms. *IIE Transactions*, *18*, 16–25.

Averbakh, I. (2001). On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, *90*, 263–272.

Chang, Z., Ding, J. Y., & Song, S. (2019). Distributionally robust scheduling on parallel machines under moment uncertainty. *European Journal of Operational Research*, *272*, 832–846.

Chang, Z., Song, S., Zhang, Y., Ding, J. Y., Zhang, R., & Chiong, R. (2017). Distributionally robust single machine scheduling with risk aversion. *European Journal of Operational Research*, *256*, 261–274.

Choi, B. C., & Chung, K. (2016). Min–max regret version of a scheduling problem with outsourcing decisions under processing time uncertainty. *European Journal of Operational Research*, *252*, 367–375.

Coban, E., Heching, A., Hooker, J. N., & Scheller-Wolf, A. (2016). Robust scheduling with logic-based benders decomposition. In *Operations research proceedings 2014* (pp. 99–105). Springer.

Delage, E., & Ye, Y. (2010). Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations Research*, *58*, 595–612.

Denton, B. T., Miller, A. J., Balasubramanian, H. J., & Huschka, T. R. (2010). Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations Research*, *58*, 802–816.

Drwal, M. (2018). Robust scheduling to minimize the weighted number of late jobs with interval due-date uncertainty. *Computers & Operations Research*, *91*, 13–20.

Drwal, M., & Józefczyk, J. (2020). Robust min–max regret scheduling to minimize the weighted number of late jobs with interval processing times. *Annals of Operations Research*, *284*, 263–282.

Emmons, H. (1969). One-machine sequencing to minimize certain functions of job tardiness. *Operations Research*, *17*, 701–715.

Feizollahi, M. J., & Averbakh, I. (2014). The robust (minmax regret) quadratic assignment problem with interval flows. *INFORMS Journal on Computing*, *26*, 321–335.

Furini, F., Iori, M., Martello, S., & Yagiura, M. (2015). Heuristic and exact algorithms for the interval min–max regret knapsack problem. *INFORMS Journal on Computing*, *27*, 392–405.

Kacem, I., & Kellerer, H. (2019). Complexity results for common due date scheduling problems with interval data and minmax regret criterion. *Discrete Applied Mathematics*, *264*, 76–89.

Kasperski, A. (2005). Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion. *Operations Research Letters*, *33*, 431–436.

Kasperski, A., & ZieliŃski, P. (2008). A 2-approximation algorithm for interval data minmax regret sequencing problems with the total flow time criterion. *Operations Research Letters*, *36*, 343–344.

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

Koulamas, C. (1994). The total tardiness problem: review and extensions. *Operations Research*, *42*, 1025–1041.

Koulamas, C. (2010). The single-machine total tardiness scheduling problem: review and extensions. *European Journal of Operational Research*, *202*, 1–7.

Kouvelis, P., & Yu, G. (1997). Robust discrete optimization: past successes and future challenges. In *Robust discrete optimization and its applications* (pp. 333–356). Springer.

Lebedev, V., & Averbakh, I. (2006). Complexity of minimizing the total flow time with interval data and minmax regret criterion. *Discrete Applied Mathematics*, *154*, 2167–2177.

Lu, C. C., Lin, S. W., & Ying, K. C. (2012). Robust scheduling on a single machine to minimize total flow time. *Computers & Operations Research*, *39*, 1682–1691.

Lu, C. C., Lin, S. W., & Ying, K. C. (2014). Minimizing worst-case regret of makespan on a single machine with uncertain processing and setup times. *Applied Soft Computing*, *23*, 144–151.

Montemanni, R., Barta, J., Mastrolilli, M., & Gambardella, L. M. (2007). The robust traveling salesman problem with interval data. *Transportation Science*, *41*, 366–381.

Mulvey, J. M., Vanderbei, R. J., & Zenios, S. A. (1995). Robust optimization of large-scale systems. *Operations Research*, *43*, 264–281.

Muth, J. F., & Thompson, G. L. (1963). *Industrial scheduling*. Prentice-Hall.

Niu, S., Song, S., Ding, J. Y., Zhang, Y., & Chiong, R. (2019). Distributionally robust single machine scheduling with the total tardiness criterion. *Computers & Operations Research*, *101*, 13–28.

Özelkan, E. C., & Duckstein, L. (1999). Optimal fuzzy counterparts of scheduling rules. *European Journal of Operational Research*, *113*, 593–609.

Pereira, J. (2016). The robust (minmax regret) single machine scheduling with interval processing times and total weighted completion time objective. *Computers & Operations Research*, *66*, 141–152.

Prekopa, A. (1995). *Stochastic programming*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Rajabighamchi, F., Hajlou, E. M. H., & Hassannayebi, E. (2019). A multi-objective optimization model for robust skip-stop scheduling with earliness and tardiness penalties. *Urban Rail Transit*, *5*, 172–185.

Rinnooy Kan, A., Lageweg, B., & Lenstra, J. K. (1975). Minimizing total costs in one-machine scheduling. *Operations Research*, *23*, 908–927.

Russell, R., & Holsenback, J. (1997). Evaluation of greedy, myopic and less-greedy heuristics for the single machine, total tardiness problem. *The Journal of the Operational Research Society*, *48*, 640–646.

Sabuncuoglu, I., & Goren, S. (2009). Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. *International Journal of Computer Integrated Manufacturing*, *22*, 138–157.

Silva, M., Poss, M., & Maculan, N. (2020). Solution algorithms for minimizing the total tardiness with budgeted processing time uncertainty. *European Journal of Operational Research*, *283*, 70–82.

Sotskov, Y. N., Egorova, N. G., & Lai, T. C. (2009). Minimizing total weighted flow time of a set of jobs with interval processing times. *Mathematical and Computer Modelling*, *50*, 556–573.

Wang, S. J., Cui, W. L., Chu, F., & Yu, J. B. (2020). The interval min–max regret knapsack packing-delivery problem. *International Journal of Productions Research* http://dx.doi.org/10.1080/00207543.2020.1789235, (in press).

Yang, J., & Yu, G. (2002). On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, *6*, 17–33.

Yue, F., Song, S., Zhang, Y., Gupta, J. N. D., & Chiong, R. (2018). Robust single machine scheduling with uncertain release times for minimising the maximum waiting time. *International Journal of Productions Research*, *56*, 5576–5592.

Zhang, Y., Shen, Z. J. M., & Song, S. (2018). Exact algorithms for distributionally β-robust machine scheduling with uncertain processing times. *INFORMS Journal on Computing*, *30*, 662–676.