Appl. Set-Valued Anal. Optim. 6 (2024), No. 3, pp. 263-273
Available online at http://asvao.biemdas.com
https://doi.org/10.23952/asvao.6.2024.3.02

AN ALGORITHMIC APPROACH TO THE ROBUST DOWNGRADING MAKESPAN SCHEDULING PROBLEM

LAM QUOC ANH[1], HUY MINH LE[2,*], KIEN TRUNG NGUYEN[1], LE XUAN THANH[3]

[1]*Department of Mathematics, Teacher College, Can Tho University, Can Tho, Vietnam*
[2]*Faculty of Fundamental Sciences, Van Lang University, Ho Chi Minh City, Vietnam*
[3]*Institute of Mathematics, Vietnam Academy of Science and Technology, Ha Noi, Vietnam*

Dedicated to Professor Le Dung Muu on the occasion of his 75th birthday

Abstract. This paper addresses the robust downgrading (single-machine) makespan scheduling problem, a scheduling challenge where processing times are augmented at a minimum cost to achieve a specified downgrading level in the makespan. The associated costs are modeled as intervals, and we employ the minmax regret criterion to handle uncertainty. The deterministic case is initially examined with a linear-time algorithm. Subsequently, the robust version of the problem is transformed into a single-variable objective, characterized by a piecewise linear function. Then, we develop a combinatorial algorithm with polynomial time complexity to solve the corresponding problem.

Keywords. Downgrading problem; Makespan; Robust optimization; Scheduling.
2020 Mathematics Subject Classification. 90B35.

1. INTRODUCTION

In a single-machine makespan scheduling problem, a collection of jobs is slated for execution. Each job is characterized by a release date and processing time. The objective is to determine a viable job sequence that minimizes the makespan, representing the completion time of the last job. Readers can refer to [5, 17, 18] for scheduling problems in general and makespan scheduling in particular. Up- and downgrading combinatorial optimization problems involve adjusting the parameters to either decrease or increase the optimal objective of the problem. Fulkerson and Harding [11] delved into maximizing the reduction in the length of the shortest and longest paths, respectively, through modifications of edge lengths. Meanwhile, Frederickson and Solis-Oba [10], Drangmeister et al. [9], and Krumke et al. [16] examined variations for network improvement and degradation in the context of the minimum spanning tree and Steiner tree problems. A broader investigation into the up- and downgrading versions of 0-1 combinatorial optimization problems was conducted by Burkard et al. [6, 7] and Zhang, Yang and Cai [23]. Currently, the up- and downgrading problems concerning facility location involve modifying parameters so that the 1-median or 1-center objective, with respect to the modified parameters,

*Corresponding author.
E-mail address: quocanh@ctu.edu.vn (L.Q. Anh), huy.lm@vlu.edu.vn (H.M. Le), trungkien@ctu.edu.vn (K.T. Nguyen), lxthanh@math.ac.vn (X.T. Le).
Received 14 March 2023; Accepted 30 April 2024; Published online 1 August 2024.
©2024 Applied Set-Valued Analysis and Optimization

should be minimized or maximized under a budget constraint; see, e.g., [1, 12, 20, 22] and the references therein.

Many real-life situations involve uncertain data stemming from changes in the environment, market volatility, errors in measurement, and so on. Since there is no distribution rule for uncertain inputs, robust optimization tools are applied in such cases instead of stochastic ones. For a comprehensive understanding of general problem settings and techniques, we refer to the pioneering works of Bel-Tal et al. [4] and Kouvelis and Yu [15]. According to the best of our knowledge, the inverse robust optimization problem was first investigated by Chassein and Goerigk [8], where the size of robustness was changed at a minimum cost, ensuring that a prespecified solution becomes an optimal solution with respect to the new uncertainty. Ghobadi et al. [13] studied the robust inverse linear programming problem with an uncertain optimal solution. They reduced the problem to finitely many classical inverse optimization subproblems, thus solving the robust version by addressing these corresponding subproblems. Nhan et al. [19] investigated the minmax regret reverse 1-median problem involving uncertain vertex weights. They developed a greedy algorithm capable of solving the corresponding problem in $O(n^3)$ time, with n representing the size of the input data. Concerning the uncertain costs, Averbakh [2] solved the minmax regret linear resource allocation problem in $O(n \log n)$ time. In another study, Nguyen and Hung [21] explored the minmax regret inverse maximum weight problem with interval costs. They demonstrated that the maximum regret function can be reduced to a convex function, allowing them to solve the problem efficiently in linear time. Bachtler et al. [3] considered the robust makespan scheduling problem with interval release dates and developed polynomial time algorithms for both absolute and minmax regret criteria.

In this paper, we investigate the problem of augmenting the processing times of jobs in a manner of minimum cost, ensuring that the new makespan equals a given downgrading level. Furthermore, each of the cost coefficients is not precisely known but can assume any value within a given interval. We term this problem the robust downgrading makespan scheduling problem. This problem can be encountered in practice. For example, if the resources needed for a particular task within the system are unavailable at the scheduled time, the completion time may need to be extended to prevent machine idle time. In the realm of security, additional time is essential beyond the current system setup to debug harmful features. Therefore, the completion time should be prolonged. In the case of job dependencies, if a task relies on the completion of another and the dependent task encounters delays, it may be necessary to lengthen the completion time for the subsequent job. These situations can be mathematically modeled as a downgrading makespan scheduling problem.

This paper is structured as follows. In Section 2, we introduce the scheduling problem and its downgrading version with the augmentation of processing times. This section also explores a linear-time solution approach for the downgrading makespan scheduling problem. In Section 3, we delve into the robust problem under interval costs. We apply the minmax regret criterion to address the robust model. The problem is transformed into a univariate optimization problem with a piecewise linear objective function. Finally, we present a combinatorial algorithm that solves the problem in $O(n^2)$ time.

2. PRELIMINARIES

The (single-machine) makespan scheduling problem consists of one machine and n jobs, denoted as $J_i = (r_i, p_i)$, where r_i and p_i are positive, for $i \in \mathcal{N} := \{1, \ldots, n\}$. Here, r_i and p_i represent the release date (before which the corresponding job cannot be scheduled) and processing time of job J_i for $i \in \mathcal{N}$. A schedule is defined as a permutation $\pi : \mathcal{N} \to \mathcal{N}$. In a given schedule π, the machine processes jobs in the order $J_{\pi(1)}, J_{\pi(2)}, \ldots, J_{\pi(n)}$, where $\pi(i)$ is a job processed at position i. Consequently, we can identify a sequence of completion times.

$$C_{\pi(1)} = r_{\pi(1)} + p_{\pi(1)},$$
$$C_{\pi(i)} = \max\left\{C_{\pi(i-1)}, r_{\pi(i)}\right\} + p_{\pi(i)} \text{ for } i = 2, 3, \cdots, n.$$

The makespan for a schedule π is the completion time of the last job, say $C_{\pi(n)}$. In a makespan scheduling problem, the objective is to find a schedule π that minimizes the makespan. The following result states the rule for determining such an optimal scheduling.

Lemma 2.1. *(Earliest Release Date (ERD) rule; see Lenstra et al. [18]) The schedule π^* in which $r_{\pi^*(1)} \leq r_{\pi^*(2)} \leq \cdots \leq r_{\pi^*(n)}$ yields the minimum makespan.*

We now define the setting of the *downgrading makespan scheduling problem* (DMSP). The processing time of each job J_i can be augmented by an amount x_i, where $x_i \geq 0$, for $i \in \mathcal{N}$. In other words, the adjusted processing time is $\hat{p}_i = p_i + x_i$ for $i \in \mathcal{N}$. Additionally, we incur a positive cost c_i associated with a unit modification of the processing time of job J_i for $i \in \mathcal{N}$. Let a downgrading level Γ be given, where Γ is strictly larger than the current minimum makespan. The objective of the DMSP is to find a strategy to augment the processing times of jobs, minimizing the total cost, while ensuring that the makespan of the system becomes Γ.

In the context of the DMSP, the release dates of the jobs are fixed. Furthermore, according to Lemma 2.1, the optimal scheduling is dependent on the order of release dates. Hence, for the sake of simplicity, we assume that $r_1 \leq r_2 \leq \ldots \leq r_n$, meaning the schedule π^* satisfying $\pi^*(i) = i$ for $i \in \mathcal{N}$ is an optimal schedule. The DMSP can be formulated as follows:

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{N}} c_i x_i \\
\text{s.t.} \quad & \hat{p}_i = p_i + x_i \quad \forall i \in \mathcal{N}, && (2.1)\\
& \hat{C}_1 = r_1 + \hat{p}_1, && (2.2)\\
& \hat{C}_i = \max\left\{\hat{C}_{i-1}, r_i\right\} + \hat{p}_i \ \ \forall i = 2, 3, \cdots, n, && (2.3)\\
& \hat{C}_n = \Gamma, && (2.4)\\
& x_i \geqslant 0 \quad \forall i \in \mathcal{N}. && (2.5)
\end{aligned}$$

To present the original makespan C_n (the makespan with no augmentation of processing times), we identify the critical index

$$i_{cri} = \max\left\{i \in \mathcal{N} : r_i > C_{i-1}\right\}.$$

Such an index is unique and always exists for a given schedule. If the machine processes all jobs without idle time, the first job satisfies this property. Then, we know that

$$C_n = r_{i_{cri}} + \sum_{j=i_{cri}}^{n} p_j.$$

We attain the following result.

Proposition 2.1. *There exists an optimal solution x^* of the DMSP such that exactly one index m satisfies $x^*_m > 0$, and $x^*_i = 0$ for $i \in \mathcal{N}\backslash\{m\}$.*

Proof. Let x' be an optimal solution to the DMSP. We prove that there always exists a feasible solution x^* satisfying the conditions in the proposition, and the objective value at x^* is less than or equal to the objective value at x'. Therefore, x^* is also an optimal solution.

We can assume that $\hat{i}_{cri}$ is the critical index with respect to the instance $J_i = (r_i, \hat{p}_i)$ of modified processing time $\hat{p}_i = p_i + x'_i$ for $i \in \mathcal{N}$, and we can present

$$\hat{C}_n = r_{\hat{i}_{cri}} + \sum_{j=\hat{i}_{cri}}^{n} \hat{p}_j = \Gamma.$$

For any index $j < \hat{i}_{cri}$, we can set $x^*_j = 0$ as the completion time of job j is less than $r_{\hat{i}_{cri}}$ for all $j < \hat{i}_{cri}$, which implies

$$\sum_{j=1}^{\hat{i}_{cri}-1} c_j x^*_j \leq \sum_{j=1}^{\hat{i}_{cri}-1} c_j x'_j. \tag{2.6}$$

The rest is to consider the program

$$\min\left\{ \sum_{j=\hat{i}_{cri}}^{n} c_j x_j : r_{\hat{i}_{cri}} + \sum_{j=\hat{i}_{cri}}^{n} p_j + \sum_{j=\hat{i}_{cri}}^{n} x_j = \Gamma, x_j \geq 0 \text{ for } j = \hat{i}_{cri}, \ldots, n \right\}.$$

This program is a continuous knapsack problem. Hence, there exists an optimal solution such that $x^*_m = \Gamma - \left(r_{\hat{i}_{cri}} + \sum_{j=\hat{i}_{cri}}^{n} p_j\right) > 0$ and $x^*_j = 0$ for $j \in \{\hat{i}_{cri}, \ldots, n\}\backslash\{m\}$, where $m = \arg\min\left\{c_j : j = \hat{i}_{cri}, \ldots, n\right\}$ (see Kellerer et al. [14]). Then, this implies

$$\sum_{j=\hat{i}_{cri}}^{n} c_j x^*_j \leq \sum_{j=\hat{i}_{cri}}^{n} c_j x'_j. \tag{2.7}$$

We conclude that $\sum_{i\in\mathcal{N}} c_i x^*_i \leq \sum_{i\in\mathcal{N}} c_i x'_i$, which is due to (2.6) and (2.7). Thus x^* is an optimal solution. This proves the proposition. □

We now partition the set $\mathcal{N}$ into *blocks of jobs*. We start with $i^1 = 1$ and identify the smallest index i^2 (if it exists) such that $r_{i^1} + \sum_{j=i^1}^{i^2-1} p_j < r_{i^2}$. Then, we define the block $B_1 = \{i^1, i^1 + 1, \ldots, i^2 - 1\}$. If there is no index i^2, then $B_1 = \{i^1, i^1 + 1, \ldots, n\}$, and we stop. To continue, we identify i^3 such that $r_{i^2} + \sum_{j=i^2}^{i^3-1} p_j < r_{i^3}$. The next block is defined as $B_2 = \{i^2, i^2 + 1, \ldots, i^3 - 1\}$. If there is no index i^3, then $B_2 = \{i^2, i^2 + 1, \ldots, n\}$, and we stop. Using a similar argument, we can identify k blocks $B_1, B_2, \ldots, B_k$.

Remark 2.1. The current critical index is $i_{cri} = i^k$.

Let us denote *the range of a block* B_l for $l = 1, \ldots, k$ as

$$\mathcal{R}(B_l) := \sum_{j=i^l}^{i^{l+1}-1} p_j.$$

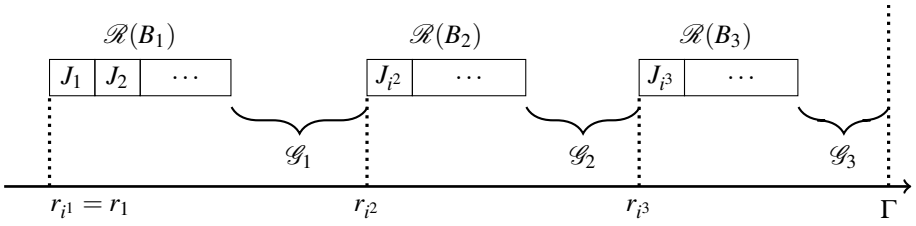

FIGURE 1. The index set is partitioned into blocks.

We also define *the gap between block l and block $l+1$* for $l = 1, \ldots, k-1$ as

$$\mathscr{G}_l := r_{i^{l+1}} - (r_{i^l} + \mathscr{R}(B_l))$$

and, by the convention,

$$\mathscr{G}_k := \Gamma - (r_{i^k} + \mathscr{R}(B_k)).$$

Example 2.1. In Figure 1, the blocks and gaps are illustrated.

One can compute the blocks and gaps for a given instance of release dates and processing times in linear time by starting from the smallest index and checking the conditions for constructing a new block until all indices $1, \ldots, n$ are taken into account.

Due to Proposition 2.1, we augment the processing time of exactly one job to achieve the optimal solution. Let us assume that job J_i with $i \in B_{l_i}$ (B_{l_i} means that block l containing job i) is the one augmented in processing time. Then, the equation $r_{i^{l_i}} + \sum_{j=i^{l_i}}^{n} \mathscr{R}(B_l) + x_i = \Gamma$ yields

$$x_i = \Gamma - (r_{i^{l_i}} + \sum_{j=i^{l_i}}^{n} \mathscr{R}(B_l)) = \sum_{j=l_i}^{k} \mathscr{G}_j.$$

We can find the optimal solution of the DMSP as follows.

Theorem 2.1. *The optimal solution of the DMSP is x^*, satisfying $x^*_m = \sum_{j=l_m}^{k} \mathscr{G}_j$ and $x^*_i = 0$ for $i \in \mathscr{N} \setminus \{m\}$, where $m := \arg\min\{c_i \sum_{j=l_i}^{k} \mathscr{G}_j : i \in \mathscr{N}, i \in B_{l_i}\}$.*

Proof. If an index i is in block B_{l_i}, we know that the processing time of J_i is augmented by $\sum_{j=l_i}^{k} \mathscr{G}_j$ to attain the makespan of Γ. The augmenting cost is $c_i \sum_{j=l_i}^{k} \mathscr{G}_j$. Hence, the optimal objective is the smallest value of $c_i \sum_{j=l_i}^{k} \mathscr{G}_j$ among all jobs. This proves the result of the theorem. □

The optimal solution x^* of the DMSP can be found in linear time. Initially, we identify the blocks B_j, compute the ranges $\mathscr{R}(B_j)$ and gaps $\mathscr{G}_j$ for $j = 1, \ldots, k$ in linear time, as discussed previously. We can recursively compute all $X^l := \sum_{j=l}^{k} \mathscr{G}_j$ for $l = 1, \ldots, k$ in linear time. For each index $i \in \mathscr{N}$, we assume that $i \in B_{l_i}$. Then, one compares all values $c_i X^{l_i}$ for $i \in \mathscr{N}$ to get the smallest value in linear time. Altogether, this naive algorithm can solve the DMSP in linear time provided that all release dates are already sorted.

Remark 2.2. We denote by $OPT(J)$ and $SOL(J)$ the optimal objective and the value x^*_m of the optimal solution x^* in Theorem 2.1 concerning jobs $J_1, J_2, \ldots, J_n$.

3. The Robust Downgrading Makespan Scheduling Problem

In many real-life situations, the costs related to augmenting processing times of jobs are not precisely known; instead, they are uncertain. In the scope of this paper, we consider interval costs, denoted as $c_i \in [\underline{c}_i, \overline{c}_i]$ for $i = 1, \ldots, n$. A scenario is regarded as the assignment of a modifying cost to a fixed value. Specifically, let us call $\mathscr{S}$ the set of all scenarios, the cost c_i^s is a fixed value within $[\underline{c}_i, \overline{c}_i]$.

Let $\mathscr{F} = \{x : (2.1) - (2.5)\}$ be the set of all feasible solutions of the DMSP. For a scenario $s \in \mathscr{S}$ and a feasible solution $x \in \mathscr{F}$, the corresponding cost is $\sum_{i=1}^{n} c_i^s x_i$. Moreover, the optimal objective to scenario s is denoted by $opt(s)$. The regret function corresponding to scenario s and a solution x is defined as:

$$R^s(x) := \sum_{i=1}^{n} c_i^s x_i - opt(s).$$

The Robust Downgrading Makespan Scheduling Problem (RobDMSP) aims to find a feasible solution x that minimizes the maximum regret across all scenarios. The formulation of RobDMSP is as follows:

$$\min_{x \in \mathscr{F}} \max_{s \in \mathscr{S}} R^s(x).$$

Let $R(x) = \max_{s \in \mathscr{S}} R^s(x)$ be the maximum regret function over all scenarios for $x \in \mathscr{F}$, and the RobDMSP is then reformulated as $\min_{x \in \mathscr{F}} R(x)$.

Let us now analyze the property of the maximum regret function $R(x)$ for $x \in \mathscr{F}$. By Theorem 2.1, we can represent $opt(s) := c_{i_s}^s X^{l_{i_s}}$, where $i_s := \arg\min_{i \in \mathscr{N}} c_i^s X^{l_i}$. Subsequently, we can rewrite $R^s(x) = \sum_{i=1}^{n} c_i^s x_i - c_{i_s}^s X^{l_{i_s}}$.

Lemma 3.1. *Any feasible solution x in $\mathscr{F}$ satisfies $x_i \leq X^{l_i}$ for $i \in \mathscr{N}$ and i in block B_{l_i}.*

Proof. For any feasible solution x, if there exists an index i such that $x_i > X^{l_i}$ for $i \in \mathscr{N}$ and $i \in B_{l_i}$, then, due to the fact that $r_l + \mathscr{R}(B_{l_i}) + x_i > \Gamma$, this contradicts the feasibility of x. □

By Lemma 3.1, the regret function $R^s(x) = \sum_{i \in \mathscr{N} \setminus \{i_s\}} c_i^s x_i + c_{i_s}^s (x_{i_s} - X^{l_{i_s}})$. As $x_i \geq 0$ and $x_{i_s} - X^{l_{i_s}} \leq 0$, the function attains its maximum value at the scenario s that satisfies

$$c_i^s := \begin{cases} \overline{c}_i & \text{if } i \neq i_s, \\ \underline{c}_i & \text{if } i = i_s. \end{cases}$$

From this point onward, we assume that any scenario s in $\mathscr{S}$ corresponds to an assignment $c_i^s = \overline{c}_i$ for $i \neq i_s$ and $c_{i_s}^s = \underline{c}_{i_s}$ to ensure that the regret function $R^s(x)$ attains its maximum value. As i_s is an index in $\mathscr{N}$, we consider the maximum regret function

$$\begin{aligned} R(x) &= \max_{m \in \mathscr{N}} \left\{ \sum_{i \neq m} \overline{c}_i x_i + \underline{c}_m \left(x_m - X^{l_m} \right) \right\} \\ &= \max_{m \in \mathscr{N}} \left\{ \sum_{i \in \mathscr{N}} \overline{c}_i x_i - (\overline{c}_m - \underline{c}_m) x_m - \underline{c}_m X^{l_m} \right\} \\ &= \sum_{i \in \mathscr{N}} \overline{c}_i x_i - \min_{m \in \mathscr{N}} \left\{ (\overline{c}_m - \underline{c}_m) x_m + \underline{c}_m X^{l_m} \right\}. \end{aligned}$$

Let $\gamma_m := \underline{c}_m X^{l_m}$ and $\Delta_m := \overline{c}_m - \underline{c}_m$ for $m \in \mathscr{N}$. We define that $t := \min_{m\in\mathscr{N}}\{\Delta_m x_m + \gamma_m\}$. We can then rewrite $R(x) = \sum_{i\in\mathscr{N}} \overline{c}_i x_i - t$. Note that $t \le \Delta_m x_m + \gamma_m$ and $t \ge \gamma_{\min} := \min_{m\in\mathscr{N}} \gamma_m$ as $x_m \ge 0$ for $m \in \mathscr{N}$. Consequently, we can imply that $x_m \ge \frac{t-\gamma_m}{\Delta_m}$ for $m \in \mathscr{N}$.

Let us denote by $A^<(t) := \{i \in \mathscr{N} : \gamma_i < t\}$ and $A^\ge(t) := \{i \in \mathscr{N} : t \le \gamma_i\}$. For any $i \in A^<(t)$, we obtain the constraint $x_i \ge \frac{t-\gamma_i}{\Delta_i}$ as $\frac{t-\gamma_i}{\Delta_i} > 0$. For any $i \in A^\ge(t)$, we obtain $x_i \ge 0$ as $\frac{t-\gamma_i}{\Delta_i} \le 0$. Thus, the RobDMSP is reformulated as the following program:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \overline{c}_i x_i - t \\ \text{s.t.} \quad & (2.1) - (2.5), \\ & x_i \ge 0 \quad \forall i \in A^\ge(t), \\ & x_i \ge \frac{t-\gamma_i}{\Delta_i} \quad \forall i \in A^<(t), \\ & t \ge \gamma_{\min}. \end{aligned} \tag{3.1}$$

For a fixed value t, let us denote by

$$\mathscr{F}(t) = \{x : (2.1) - (2.5), x_i \geqslant 0 \ \forall i \in A^\ge(t), \text{ and } x_i \ge \frac{t-\gamma_i}{\Delta_i} \ \forall i \in A^<(t)\}$$

the set of feasible solution of (3.1) with respect to t. We consider the function

$$g(t) = \min_{x\in\mathscr{F}(t)} \sum_{i=1}^{n} \overline{c}_i x_i - t.$$

Then, (3.1) is equivalent to $\min_{t\ge\gamma_{\min}} g(t)$. In order to compute the function $g(t)$ for $t \ge 0$, we first set the modified processing time of jobs

- $p_i^t = p_i$ for $i \in A^\ge(t)$.
- $p_i^t = p_i + \frac{t-\gamma_i}{\Delta_i}$ for $i \in A^<(t)$.

The current blocks and gaps for new processing time p^t are denoted by $B_1(t), B_2(t), \ldots, B_{k'}(t)$, and $\mathscr{G}_1(t), \mathscr{G}_2(t), \ldots, \mathscr{G}_{k'}(t)$. Due to Theorem 2.1, we have the value

$$g(t) = \sum_{i\in A^<(t)} \overline{c}_i \frac{t-\gamma_i}{\Delta_i} - t + \min_{i\in\mathscr{N}} \overline{c}_i \sum_{j=l_i}^{k'} \mathscr{G}_j(t). \tag{3.2}$$

Applying the denotations, we have the following result.

Proposition 3.1. *The function $g(t)$ is a piecewise linear function for $t \ge \gamma_{\min}$.*

Proof. We first set $x_i = \frac{t-\gamma_i}{\Delta_i}$ for $i \in A^<(t)$ and $x_i = 0$ for $i \in A^\ge(t)$. Based on the current blocks $B_1(t), B_2(t), \cdots, B_{k'}(t)$, we can find the minimum cost to augment the current processing times p^t. The minimum cost is $\min_{i\in\mathscr{N}} \overline{c}_i \sum_{j=l_i}^{k'} \mathscr{G}_j(t)$, where

$$\begin{aligned} \mathscr{G}_j(t) = r_{ij+1} - \mathscr{R}(B_j(t)) &= r_{ij+1} - \sum_{q\in B_j(t)} p_q - \sum_{q\in B_j(t)\cap A^<(t)} \frac{t-\gamma_i}{\Delta_i} \\ &= \kappa_j t + \omega_j \end{aligned}$$

with $\kappa_j = -\sum_{q\in B_j(t)\cap A^<(t)} \frac{1}{\Delta_i}$ and $\omega_j = r_{ij+1} - \sum_{q\in B_j(t)} p_q + \sum_{q\in B_j(t)\cap A^<(t)} \frac{\gamma_i}{\Delta_i}$ for $j = 1, \ldots, k'$. Here, we convent that $r_{ik'+1} = \Gamma$. Assume that t takes the sufficient small value such that the

blocks remain the same, then we know that $\mathscr{G}_j(t)$ are linear functions for $j = 1,\ldots,k'$. Hence, the function in (3.2) is a linear function. The presentation of $g(t)$ changes if there is an alternative presentation in the set of blocks, the set $A^<(t)$, or the representation of $\min_{i\in\mathscr{N}} \overline{c}_i \sum_{j=l_i}^{k'} \mathscr{G}_j(t)$. Hence, we conclude that $g(t)$ is a piecewise linear function for $t \geq \min_{i\in\mathscr{N}} \gamma_i$. □

As the function $g(t)$ is piecewise linear, we find the set of breakpoints of the function, where the blocks and the set $A^<(t)$ change. First of all, the sets $A^<(t)$ and $A^\geq(t)$ change at the values $\gamma_1,\gamma_2,\ldots,\gamma_n$. Without loss of generality, we assume that $\gamma_1 \leq \gamma_2 \leq \ldots \leq \gamma_{n-1} \leq \gamma_n$. Let us consider the change of blocks in each interval $(\gamma_1,\gamma_2),\ldots,(\gamma_{n-1},\gamma_n)$.

We start with $t \in (\gamma_1,\gamma_2)$. Let us consider the current set of blocks $B_1^0, B_2^0, \ldots, B_{k^0}^0$ with respect to $t = 0$, i.e., the processing times take the original values. Let $r_{i_0^l}$ be the corresponding release dates in block B_l^0 for $l = 1,\ldots,k^0$, i.e., $\mathscr{R}(B_l^0) = \sum_{j\in B_l^0} p_j$. We consider the gap between block l and $l+1$:

$$\begin{aligned}\mathscr{G}_l^0(t) &= r_{i_0^{l+1}} - (r_{i_0^l} + \mathscr{R}(B_l^0(t))) \\ &= r_{i_0^{l+1}} - (r_{i_0^l} + \sum_{j\in B_l^0} p_j) - \sum_{j\in A^<(\gamma_2)\cap B_l^0} \frac{t-\gamma_j}{\Delta_j},\end{aligned}$$

where $r_{i_0^{k^0+1}} = \Gamma$. Let $\mathscr{G}_l^0(t) = 0$, we attain

$$t := \xi_l^0 = \frac{r_{i_0^{l+1}} - (r_{i_0^l} + \sum_{j\in B_l^0} p_j) + \sum_{j\in A^<(\gamma_2)\cap B_l^0} \frac{\gamma_j}{\Delta_j}}{\sum_{j\in A^<(\gamma_2)\cap B_l^0} \frac{1}{\Delta_j}}$$

for $l = 1,\ldots,k^0$. As it costs $O(|B_l^0|)$ time to find each ξ_l^0, the total complexity for computing all ξ_l^0 for $l = 1,\ldots,k^0$ is $O(\sum_{l=1}^n |B_l^0|) = O(n)$. If $\min_{l=1}^{k^0} \xi_l^0$ is in the interval (γ_1,γ_2), we set $\psi_1 = \min_{l=1}^{k^0} \xi_l^0$. Otherwise, we consider the next interval (γ_2,γ_3). We have found ψ_1. Let us consider the updated blocks $B_1^1, B_2^1, \ldots, B_{k^1}^1$ and compute the values $\xi_1^1, \xi_2^1, \ldots, \xi_{k^1}^1$ by similar arguments. If $\min_{l=1}^{k^1} \xi_l^1$ is in (γ_1,γ_2), we set $\psi_2 = \min_{l=1}^{k^0} \xi_l^1$. Otherwise, we consider the next interval (γ_2,γ_3). We continue the process until the next interval (γ_2,γ_3) is considered. For other intervals $(\gamma_2,\gamma_3),\ldots,(\gamma_{n-1},\gamma_n)$, we also calculate the values such that two or several blocks are merged. Note that, if the gap $\mathscr{G}_{k^i}^i$ between the critical bock and Γ attains 0, we stop the process as the value t must be less than such a value.

We discuss the complexity of finding all values ψ_1, ψ_2, …, ψ_h. We know that $h \leq n$ as there are at most n times the blocks are merged. Furthermore, each value ψ_i for $i = 1,\ldots,h$ can be found in linear time. This leads to the total complexity of $O(n^2)$.

We denote the set $\{\gamma_i : i = 1,\ldots,n\} \cup \{\psi_j : j = 1,\ldots,h\}$ by the following set $\mathscr{B} := \{t_1,t_2,\ldots,t_M\}$, where $M \leq 2n$.

Proposition 3.2. *There exists an optimal solution t^* of $\min_{t\geq 0} g(t)$ such that t^* is in $\mathscr{B}$.*

Proof. Let us consider $t \in [t_i,t_{i+1}]$. As the set of blocks does not change for t in such an interval, the function $g(t)$ as in (3.2) is presented as

$$g(t) = \sum_{i\in A^<(t)} \overline{c}_i \frac{t-\gamma_i}{\Delta_i} - t + \min_{i\in\mathscr{N}} \overline{c}_i \sum_{j=l_i}^{k'} \mathscr{G}_j(t).$$

Since each function $\mathscr{G}_j(t)$ is a linear function for $j = 1,\ldots,k'$, the lower envelop of these linear functions $\min_{i\in\mathscr{N}} \overline{c}_i \sum_{j=l_i}^{k'} \mathscr{G}_j(t)$ is a concave function for $t \in [t_i, t_{i+1}]$ and it attains minimum value at either $t = t_i$ or $t = t_{i+1}$. These two points are in $\mathscr{B}$. □

Due to Proposition 3.2, we compute the value of $g(t)$ at the breakpoints in $\mathscr{B}$ and take the smallest value among them in order to obtain the optimal objective of (3.1). Hence, we can develop a combinatorial algorithm to solve the RobDMSP based on the computation of the objective values at the breakpoints, as outlined in Algorithm 3.1. Here, we denote by $OPT(J^t)$ and $SOL(J^t)$ the optimal objective and the value of x^*_m in the optimal solution with respect to jobs $J^t_i = (r_i, p^t_i)$, $p^t_i = p_i$ for $i \in A^{\geq}(t)$ and $p^t_i = p_i + \frac{t-\gamma_i}{\Delta_i}$ for $i \in A^{<}(t)$, as in Remark 2.2.

Algorithm 3.1 Solves the RobDMSP.

Input: An instance of the RobDMSP with costs $c_i \in [\underline{c}_i, \overline{c}_i]$ and jobs $J_i = (r_i, p_i)$ for $i \in \mathscr{N}$ $(r_1 \leq r_2 \leq \ldots \leq r_n)$.
Output: An optimal solution x^* of the problem.
Initialize the optimal solution $x^*_i = 0$ for $i \in \mathscr{N}$.
Identify the current blocks B^0_i, gaps $\mathscr{G}^0_i$, and X^{l_i} for $i = 1,\ldots,k$.
Compute $\gamma_i = \underline{c}_i X^{l_i}$ and $\Delta_i = \overline{c}_i - \underline{c}_i$ for $i \in \mathscr{N}$.
Compute the values $\psi_1, \psi_2, \ldots, \psi_h$ at which the blocks are changed.
Set $\mathscr{B} = \{t_1, t_2, \ldots, t_M\}$, it is $\{\gamma_1, \ldots, \gamma_n\} \cup \{\psi_1, \ldots, \psi_h\}$.
for $i = 1,\ldots,M$ **do**
  Compute $g(t_i) = \sum_{i\in A^{<}(t_i)} \overline{c}_i \frac{t_i-\gamma_i}{\Delta_i} - t_i + OPT(J^{t_i})$.
end for
Find $t^* = \arg\min\{g(t_i) : t_i \in \mathscr{B}\}$.
Set $x^*_i := \frac{t^*-\gamma_i}{\Delta_i}$ for $i \in A^{<}(t)$ and $x^*_{\hat{i}} := x^*_{\hat{i}} + SOL(J^{t^*})$ for $\hat{i}$ is the corresponding index of the optimal solution of DMSP with respect to J^{t^*}.

Theorem 3.1. *The RobDMSP can be solved in $O(n^2)$ time.*

Proof. The minimum value among all the values $g(t_i)$ for $t_i \in \mathscr{B}$ is the optimal objective of $\min_{t\geq\gamma_{\min}} g(t)$, as stated in Proposition 3.2. Subsequently, we trace back the optimal solution based on t^*. The corresponding solution x^* is optimal for (3.1), with $x^*_i := \frac{t^*-\gamma_i}{\Delta_i}$ for $i \in A^{<}(t)$, and $x^*_{\hat{i}} := x^*_{\hat{i}} + SOL(J^{t^*})$ for $\hat{i}$ being the corresponding index of the optimal solution of DMSP concerning J^{t^*}. Therefore, the algorithm correctly finds an optimal solution for the RobDMSP.

Let us analyze the complexity of the algorithm. As mentioned earlier, we first compute all γ_i and Δ_i in linear time. Additionally, we calculate all values ψ_i for $i = 1,\ldots,h$ with $h \leq n$ in $O(n^2)$ time through recursive computations. Subsequently, each value $g(t_i)$ can be computed in linear time. This results in an overall $O(n^2)$ complexity to obtain all values $g(t_i)$ for $i = 1,\ldots,n$. Finally, we find the minimum value among them and calculate the corresponding optimal solution in linear time. Therefore, Algorithm 3.1 runs in $O(n^2)$ time. □

We illustrate Algorithm 3.1 by the following numerical example.

Example 3.1. Let us consider the input of the RobDMSP with 6 jobs and interval modifying costs as in Table 1.

i	1	2	3	4	5	6
$J_i = (r_i, p_i)$	$(1,3)$	$(2,2)$	$(10,2)$	$(11,2)$	$(20,2)$	$(21,2)$
$c_i = [\underline{c}_i, \overline{c}_i]$	$[1,2]$	$[1,2]$	$[2,4]$	$[2,4]$	$[3,6]$	$[3,6]$

TABLE 1. An instance of the RobDMSP

Let the downgrading level be $\Gamma = 30$. We first identify the blocks $\mathscr{B}_1^0 = \{1,2\}$, $\mathscr{B}_2^0 = \{3,4\}$, $\mathscr{B}_3^0 = \{5,6\}$. The gaps are $\mathscr{G}_1^0 = 4$, $\mathscr{G}_2^0 = 6$, $\mathscr{G}_3^0 = 6$. Then, we can compute $X^{l_1} = 16$, $X^{l_2} = 12$, $X^{l_3} = 6$. Furthermore, we calculate γ_i and Δ_i for $i = 1,\ldots,6$ as in Table 2.

i	1	2	3	4	5	6
γ_i	16	16	24	24	18	18
Δ_i	1	1	2	2	3	3

TABLE 2. The values γ_i and Δ_i for $i = 1,\ldots,6$

The set of breakpoints of the objective function is $\mathscr{B} = \{16, 18, 21, 22.5\}$. We can compute the corresponding objective at the breakpoints as in Table 3.

t_i	16	18	21	22.5
$g(t_i)$	16	14	19	21.5

TABLE 3. Objectives at the break points

Hence, the optimal solution of the problem is $t^* = 18$ and we can trace back the optimal solution of the RobDMSP as

$$(x_1^*, x_2^*, x_3^*, x_4^*, x_5^*, x_6^*) = (2, 14, 0, 0, 0, 0).$$

4. CONCLUSIONS

This paper addressed the problem of computing the minimum cost to downgrade the makespan of a given scheduling system by augmenting the processing times of jobs. We first proved that the deterministic problem could be solved in linear time. Then, we focused on the problem with interval cost coefficients and applied the minmax regret criterion to deal with robustness, calling it the robust downgrading makespan scheduling problem. The problem induced the corresponding program with a piecewise linear objective function. Finally, we solved the problem in $O(n^2)$ time, where n was the number of jobs. Further extensions of the corresponding problem with variable release dates and under different norm costs are considered promising for future research.

Acknowledgments
The authors would like to thank anonymous referees whose valuable suggestions helped to improve the paper. The authour Huy Minh Le would like to thank Van Lang University for funding this work.

REFERENCES

[1] E. Afrashteh, B. Alizadeh, F. Baroughi, Optimal approaches for upgrading selective obnoxious p-median location problems on tree networks, Annals of Operations Research 289 (2020), 153–172.

[2] I. Averbakh, Minmax regret linear resource allocation problems, Operations Research Letters 32 (2004), 174-180.

[3] O. Bachtler, S.O Krumke, H.M. Le, Robust single machine makespan scheduling with release date uncertainty, Operations Research Letters 48 (2020), 816-819.

[4] A. Ben-Tal, A. Nemirovski, L. El-Ghaoui, Robust Optimization, Princeton University Press, Princeton, New Jersey, 2009.

[5] P. Brucker, Scheduling Algorithms (Fifth Edition), Springer Verlag, Heidelberg, 2007.

[6] R.E. Burkard, B. Klinz, J. Zhang, Bottleneck capacity expansion problems with general budget constraints, RAIRO Recherche Operationelle 35 (2001), 1-20.

[7] R.E. Burkard, Y. Lin, J. Zhang, Weight reduction problems with certain bottleneck objectives, European Journal of Operational Research 153 (2004), 191–199.

[8] A. Chassein, M. Goerigk, Variable-sized uncertainty and inverse problems in robust optimization, European Journal of Operational Research 264 (2018), 17–28.

[9] K.U. Drangmeister, S.O. Krumke, M.V. Marathe, H. Noltemeier, S.S. Ravi, Modifying edges of a network to obtain short subgraphs, Theoretical Computer Science 203 (1998), 91–121.

[10] G.N. Frederickson, R. Solis-Oba, Increasing the weight of minimum spanning trees, Journal of Algorithms 33 (1999), 244–266.

[11] D.R. Fulkerson, G.C. Harding, Maximizing the minimum source-sink path subject to a budget constraint, Mathematical Programming 13 (1977), 116–118.

[12] E. Gassner, Up- and downgrading the 1-center in a network, European Journal of Operational Research 198 (2009), 370-377.

[13] K. Ghobadi, T. Lee, H. Mahmoudzadeh, D. Terekhov, Robust inverse optimization, Operations Research Letters 46 (2018), 339–344.

[14] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer Berlin, Heidelberg, 2010.

[15] P. Kouvelis, G. Yu, Robust Discrete Optimization and Its Applications, Kluwer Academic Publishers, 1997.

[16] S.O. Krumke, M.V. Marathe, H. Noltemeier, R. Ravi, S.S. Ravi, Approximation algorithms for certain network improvement problems, Journal of Combinatorial Optimization 2 (1998), 257–288.

[17] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D. Shmoys, "Sequencing and Scheduling: Algorithms and Complexity", in Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory, S.S. Graves, A.H.G. Rinnooy Kan and P. Zipkin, (eds.), (1993) pp. 445-522, North-Holland, New York.

[18] J. Lenstra, A. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics 1 (1997), 343-362.

[19] T.H.N. Nhan, K.T. Nguyen, H. Nguyen-Thu, The minmax regret reverse 1-median problem on trees with uncertain vertex weights, Asia-Pacific Journal of Operational Research 40 (2023), 2250033.

[20] K.T. Nguyen, Reverse 1-center problem on weighted trees, Optimization 65 (2016), 253-264.

[21] K.T. Nguyen, N.T. Hung, The minmax regret inverse maximum weight problem, Applied Mathematics and Computation 407 (2021), 126328.

[22] A.R. Sepasian, Upgrading the 1-center problem with edge length variables on a tree, Discrete Optimization 29 (2018), 1-17.

[23] J. Zhang, X.G. Yang and M.C. Cai, A network improvement problem under different norms, Computational Optimization and Applications 27 (2004), 305–319.