Min-Max Relative Regret for Scheduling to Minimize Maximum Lateness

Imad Assayakh(✉), Imed Kacem, and Giorgio Lucarelli

LCOMS, University of Lorraine, Metz, France
{imad.assayakh,imed.kacem,giorgio.lucarelli}@univ-lorraine.fr

Abstract. We study the single machine scheduling problem under uncertain parameters, with the aim of minimizing the maximum lateness. More precisely, the processing times, the release dates and the delivery times of the jobs are uncertain, but an upper and a lower bound of these parameters are known in advance. Our objective is to find a robust solution, which minimizes the maximum relative regret. In other words, we search for a solution which, among all possible realizations of the parameters, minimizes the worst-case ratio of the deviation between its objective and the objective of an optimal solution over the latter one. Two variants of this problem are considered. In the first variant, the release date of each job is equal to 0. In the second one, all jobs are of unit processing time. In all cases, we are interested in the sub-problem of maximizing the (relative) regret of a given scheduling sequence. The studied problems are shown to be polynomially solvable.

Keywords: Scheduling · Maximum lateness · Min-max relative regret · Interval uncertainty

1 Introduction

Uncertainty is a crucial factor to consider when dealing with combinatorial optimization problems and especially scheduling problems. Thus, it is not sufficient to limit the resolution of a given problem to its deterministic version for a single realisation of the uncertain parameters, i.e., a scenario. In our study, we investigate a widely used method of handling uncertainty that relies on a set of known possible values of the uncertain parameters without any need of probabilistic description, namely the *robustness approach* or *worst-case approach* [12]. The aim of this approach is to generate solutions that will have a good performance under any possible scenario and particularly in the most unfavorable one.

The use of the robustness approach involves specifying two key components. The first component is the choice of the type of uncertainty set. Literature has proposed various techniques for describing the uncertainty set [6], with *the discrete uncertainty* and *the interval uncertainty* being the most well-examined. Indeed, the most suitable representation of uncertainty in scheduling problems is the interval uncertainty, where the value of each parameter is restricted within a

S.-Y. Hsieh et al. (Eds.): IWOCA 2023, LNCS 13889, pp. 49–61, 2023.
https://doi.org/10.1007/978-3-031-34347-6_5

specific closed interval defined by a lower and an upper bound. These bounds can be estimated through a data analysis on traces of previous problem executions.

The second component is the choice of the appropriate robustness criterion [1, 16]. The *absolute robustness* or *min-max* criterion seeks to generate solutions that provide the optimal performance in the worst case scenario. This criterion can be seen as overly pessimistic in situations where the worst-case scenario is unlikely, causing decision makers to regret not embracing a moderate level of risk. The *robust deviation* or *min-max regret* criterion aims at minimizing the maximum *absolute regret*, which is the most unfavorable deviation from the optimal performance among all scenarios. The *relative robust deviation* or *min-max relative regret* criterion seeks to minimize the maximum *relative regret*, which is the worst percentage deviation from the optimal performance among all possible scenarios. Averbakh [4] remarks that the relative regret objective is more appropriate compared to the absolute regret objective in situations where the statement "10% more expensive" is more relevant than "costs $30 more". However, the min-max relative regret criterion has a complicated structure and this may explain why limited knowledge exists about it.

The focus of this paper is to investigate the min-max relative regret criterion for the fundamental single machine scheduling problem with the maximum lateness objective. The interval uncertainty can involve the processing times, the release dates or the delivery times of jobs. In Sect. 2, we formally define our problem and the used criteria. In Sect. 3, we give a short review of the existing results for scheduling problems with and without uncertainty consideration. We next consider two variants of this problem.

In Sect. 4, we study the variant where all jobs are available at time 0 and the interval uncertainty is related to processing and delivery times. Kasperski [11] has applied the min-max regret criterion to this problem and developed a polynomial time algorithm to solve it by characterizing the worst-case scenario based on a single guessed parameter through some dominance rules. We prove that this problem is also polynomial for the min-max relative regret criterion. An iterative procedure is used to prove some dominance rules based on three guessed parameters in order to construct a partial worst-case scenario. To complete this scenario, we formulate a linear fractional program and we explain how to solve it in polynomial time.

In Sect. 5, we study the maximum relative regret criterion for the variant of the maximum lateness problem where the processing times of all jobs are equal to 1 and interval uncertainty is related to release dates and delivery times. For a fixed scenario, Horn [8] proposed an optimal algorithm for this problem. For the uncertainty version, we simulate the execution of Horn's algorithm using a guess of five parameters, in order to create a worst-case scenario along with its optimal schedule. Note that, we also give a much simpler analysis for the maximum regret criterion of this variant of our scheduling problem.

We conclude in Sect. 6.

2 Problem Definition and Notations

In this paper, we consider the problem of scheduling a set $\mathcal{J}$ of n non-preemptive jobs on a single machine. In the standard version of the problem, each job is characterized by a *processing time*, a *release date* and a *due date*. However, in this work we use a known equivalent definition of this problem in which the due dates are replaced by *delivery times*. In general, the values of the input parameters are not known in advance. However, an estimation interval for each value is known. Specifically, given a job $j \in \mathcal{J}$, let $[p_j^{\min}, p_j^{\max}]$, $[r_j^{\min}, r_j^{\max}]$ and $[q_j^{\min}, q_j^{\max}]$ be the uncertainty intervals for its characteristics.

A *scenario* $s = (p_1^s, ..., p_n^s, r_1^s, ..., r_n^s, q_1^s, ..., q_n^s)$ is a possible realisation of all values of the instance, such that $p_j^s \in [p_j^{\min}, p_j^{\max}]$, $r_j^s \in [r_j^{\min}, r_j^{\max}]$ and $q_j^s \in [q_j^{\min}, q_j^{\max}]$, for every $j \in \mathcal{J}$. The set of all scenarios is denoted by $\mathcal{S}$. A solution is represented by a *sequence* of jobs, $\pi = (\pi(1), ..., \pi(n))$ where $\pi(j)$ is the jth job in the sequence π. The set of all sequences is denoted by Π.

Consider a schedule represented by its sequence $\pi \in \Pi$ and a scenario $s \in \mathcal{S}$. The *lateness* of a job $j \in \mathcal{J}$ is defined as $L_j^s(\pi) = C_j^s(\pi) + q_j^s$, where $C_j^s(\pi)$ denotes the *completion time* of j in the schedule represented by π under the scenario s. The *maximum lateness* of the schedule is defined as $L(s, \pi) = \max_{j \in J} L_j^s(\pi)$. The job $c \in \mathcal{J}$ of maximum lateness in π under s is called *critical*, i.e., $L_c^s(\pi) = L(s, \pi)$. The set of all critical jobs in π under s is denoted by $Crit(s, \pi)$. We call *first critical job*, the critical job which is processed before all the other critical jobs. By considering a given scenario s, the optimal sequence is the one leading to a schedule that minimizes the maximum lateness, i.e., $L^*(s) = \min_{\pi \in \Pi} L(s, \pi)$. This is a classical scheduling problem, denoted by $1|r_j|L_{max}$ using the standard three-field notation, and it is known to be NP-hard [15].

In this paper, we are interested in the min-max regret and the min-max relative regret criteria whose definitions can be illustrated by a game between two agents, Alice and Bob. Alice selects a sequence π of jobs. The problem of Bob has as input a sequence π chosen by Alice, and it consists in selecting a scenario s such that the regret R of Alice $R(s, \pi) = L(s, \pi) - L^*(s)$ or respectively the relative regret RR of Alice $RR(s, \pi) = \frac{L(s,\pi) - L^*(s)}{L^*(s)} = \frac{L(s,\pi)}{L^*(s)} - 1$ is maximized. The value of $Z(\pi) = \max_{s \in S} R(s, \pi)$ (resp. $ZR(\pi) = \max_{s \in S} RR(s, \pi)$) is called *maximum regret* (resp. *maximum relative regret*) for the sequence π. In what follows, we call the problem of maximizing the (relative) regret, given a sequence π, as the *Bob's problem*. Henceforth, by slightly abusing the definition of the relative regret, we omit the constant -1 in $RR(s, \pi)$, since a scenario maximizing the fraction $\frac{L(s,\pi)}{L^*(s)}$ maximizes also the value of $\frac{L(s,\pi)}{L^*(s)} - 1$. Then, Alice has to find a sequence π which minimizes her maximum regret (resp. maximum relative regret), i.e., $\min_{\pi \in \Pi} Z(\pi)$ (resp. $\min_{\pi \in \Pi} ZR(\pi)$). This problem is known as the *min-max (relative) regret* problem and we call it as *Alice's problem*.

Given a sequence π, the scenario that maximises the (relative) regret over all possible scenarios is called *the worst-case scenario* for π. A *partial (worst-case) scenario* is a scenario defined by a fixed subset of parameters and can be *extended*

to a (worst-case) scenario by setting the remaining unknown parameters. For a fixed scenario s, any schedule may consist of several *blocks*, i.e., a maximal set of jobs, which are processed without any *idle time* between them. A job u_j is said to be *first-block* for the job j if it is the first job processed in the block containing j in a given schedule.

3 Related Work

In the deterministic version, the problem $1|r_j|L_{\max}$ has been proved to be strongly NP-hard [15]. For the first variant where all release dates are equal, the problem can be solved in polynomial time by applying *Jackson's rule* [9], i.e., sequencing the jobs in the order of non-increasing delivery times. For the second variant with unit processing time jobs, the rule of scheduling, at any time, an available job with the biggest delivery time is shown to be optimal by Horn [8].

For the discrete uncertainty case, the min-max criterion has been studied for several scheduling problems with different objectives. Kouvelis and Yu [12] proved that the min-max resource allocation problem is NP-hard and admits a pseudo-polynomial algorithm. Aloulou and Della Croce [2] showed that the min-max $1||\sum U_j$ problem of minimizing the number of late jobs is NP-hard, while the min-max problem of the single machine scheduling is polynomially solvable for many objectives like makespan, maximum lateness and maximum tardiness even in the presence of precedence constraints. The only scheduling problem studied under discrete uncertainty for min-max (relative) regret is the $1||\sum C_j$ for which Yang and Yu [17] have proved that it is NP-hard for all the three robustness criteria.

For the interval uncertainty case, the min-max criterion has the same complexity as the deterministic problem since it is equivalent to solve it for an extreme well-known scenario. Considerable research has been dedicated to the min-max regret criterion for different scheduling problems. Many of these problems have been proved to be polynomially solvable. For instance, Averbakh [3] considered the min-max regret $1||\max w_j T_j$ problem to minimize the maximum weighted tardiness, where weights are uncertain and proposed a $O(n^3)$ algorithm. He also presented a $O(m)$ algorithm for the makespan minimization for a permutation flow-shop problem with 2 jobs and m machines with interval uncertainty related to processing times [5]. The min-max regret version of the first variant of our problem has been considered by Kasperski [11] under uncertain processing times and due dates. An $O(n^4)$ algorithm has been developed which works even in the presence of precedence constraints. On the other hand, Lebedev and Averbakh. [14] showed that the min-max regret $1||\sum C_j$ problem is NP-hard. Kacem and Kellerer [10] considered the single machine problem of scheduling jobs with a common due date with the objective of maximizing the number of early jobs and they proved that the problem is NP-hard.

Finally, for the min-max relative regret criterion for scheduling problems with interval uncertainty, the only known result is provided by Averbakh [4]

who considered the problem $1||\sum C_j$ with uncertain processing times and he proved that it is NP-hard.

4 Min-Max Relative Regret for 1 || L_{max}

In this section, we consider the min-max relative regret criterion for the maximum lateness minimization problem, under the assumption that each job is available at time 0, i.e., $r_j^s = 0$ for all jobs $j \in \mathcal{J}$ and all possible scenarios $s \in \mathcal{S}$. For a fixed scenario, this problem can be solved by applying the Jackson's rule, i.e., sequencing the jobs in order of non-increasing delivery times.

4.1 The Bob's Problem

We denote by $B(\pi, j)$ the set of all the jobs processed before job $j \in \mathcal{J}$, including j, in the sequence π and by $A(\pi, j)$ the set of all the jobs processed after job j in π. The following lemma presents some properties of a worst-case scenario for a given sequence of jobs.

Lemma 1. *Let π be a sequence of jobs. There exists (1) a worst case scenario s for π, (2) a critical job $c_\pi \in Crit(s, \pi)$ in π under s, and (3) a critical job $c_\sigma \in Crit(s, \sigma)$ in σ under s, where σ is the optimal sequence for s, such that:*

i for each job $j \in A(\pi, c_\pi)$, it holds that $p_j^s = p_j^{\min}$,
ii for each job $j \in \mathcal{J} \backslash \{c_\pi\}$, it holds that $q_j^s = q_j^{\min}$,
iii for each job $j \in B(\pi, c_\pi) \cap B(\sigma, c_\sigma)$, it holds that $p_j^s = p_j^{\min}$, and
iv c_σ is the first critical job in σ under s.

Consider the sequence π chosen by Alice. Bob can guess the critical job c_π in π and the first critical job c_σ in σ. Then, by Lemma 1 (i)–(ii), he can give the minimum processing times to all jobs in $A(\pi, c_\pi)$, and the minimum delivery times to all jobs except for c_π. Since the delivery times of all jobs except c_π are determined and the optimal sequence σ depends only on the delivery times according to the Jackson's rule, Bob can obtain σ by guessing the position $k \in [\![1, n]\!]$ of c_π in σ. Then, by Lemma 1 (iii), he can give the minimum processing times to all jobs in $B(\pi, c_\pi) \cap B(\sigma, c_\sigma)$. We denote by the triplet (c_π, c_σ, k) the guess made by Bob. Based on the previous assignments, Bob gets a partial scenario $\bar{s}^\pi_{c_\pi, c_\sigma, k}$. It remains to determine the exact value of q_{c_π} and the processing times of jobs in $B(\pi, c_\pi) \cap A(\sigma, c_\sigma)$ in order to extend $\bar{s}^\pi_{c_\pi, c_\sigma, k}$ to a scenario $s^\pi_{c_\pi, c_\sigma, k}$. At the end, Bob will choose, among all the scenarios $s^\pi_{c_\pi, c_\sigma, k}$ created, the worst case scenario s_π for the sequence π, i.e., $s_\pi = \arg\max_{i,j,k} \left\{ \frac{L(s^\pi_{i,j,k}, \pi)}{L^*(s^\pi_{i,j,k})} \right\}$.

In what follows, we propose a *linear fractional program* (P) in order to find a scenario $s^\pi_{c_\pi, c_\sigma, k}$ which extends $\bar{s}^\pi_{c_\pi, c_\sigma, k}$ and maximizes the relative regret for the given sequence π. Let p_j, the processing time of each job $j \in B(\pi, c_\pi) \cap A(\sigma, c_\sigma)$, and q_{c_π}, the delivery time of job c_π, be the continuous decision variables in (P).

All other processing and delivery times are constants and their values are defined by $\bar{s}^{\pi}_{c_\pi,c_\sigma,k}$. Recall that $\sigma(j)$ denotes the j-th job in the sequence σ. To simplify our program, we consider two fictive values $q_{\sigma(n+1)} = q^{\min}_{c_\pi}$ and $q_{\sigma(0)} = q^{\max}_{c_\pi}$.

$$\text{maximize} \quad \frac{\sum_{i\in B(\pi,c_\pi)} p_i + q_{c_\pi}}{\sum_{i\in B(\sigma,c_\sigma)} p_i + q_{c_\sigma}} \qquad (P)$$

$$\text{subject to} \quad \sum_{i\in B(\pi,j)} p_i + q_j \le \sum_{i\in B(\pi,c_\pi)} p_i + q_{c_\pi} \qquad \forall j \in \mathcal{J} \quad (1)$$

$$\sum_{i\in B(\sigma,j)} p_i + q_j \le \sum_{i\in B(\sigma,c_\sigma)} p_i + q_{c_\sigma} \qquad \forall j \in \mathcal{J} \quad (2)$$

$$p_j \in [p^{\min}_j, p^{\max}_j] \quad \forall j \in B(\pi,c_\pi) \cap A(\sigma,c_\sigma) \quad (3)$$

$$q_{c_\pi} \in [\max\{q^{\min}_{c_\pi}, q_{\sigma(k+1)}\}, \min\{q^{\max}_{c_\pi}, q_{\sigma(k-1)}\}] \quad (4)$$

The objective of (P) maximizes the relative regret for the sequence π under the scenario $s^{\pi}_{c_\pi,c_\sigma,k}$ with respect to the hypothesis that c_π and c_σ are critical in π and σ, respectively, i.e.,

$$ZR(\pi) = \frac{L(s^{\pi}_{c_\pi,c_\sigma,k},\pi)}{L^*(s^{\pi}_{c_\pi,c_\sigma,k})} = \frac{L^{s^{\pi}_{c_\pi,c_\sigma,k}}_{c_\pi}(\pi)}{L^{s^{\pi}_{c_\pi,c_\sigma,k}}_{c_\sigma}(\sigma)}$$

Constraints (1) and (2) ensure this hypothesis. Constraints (3) and (4) define the domain of the continuous real variables p_j, $j \in B(\pi,c_\pi) \cap A(\sigma,c_\sigma)$, and q_{c_π}. Note that, the latter one is based also on the guess of the position of c_π in σ. The program (P) can be infeasible due to the Constraints (1) and (2) that impose jobs c_π and c_σ to be critical. In this case, Bob ignores the current guess of (c_π, c_σ, k) in the final decision about the worst case scenario s_π that maximizes the relative regret.

Note that the Constraint (1) can be safely removed when considering the whole procedure of Bob for choosing the worst case scenario s_π. Indeed, consider a guess (i,j,k) which is infeasible because the job i is not critical in π due to the Constraint (1). Let s be the scenario extended from the partial scenario $\bar{s}^{\pi}_{i,j,k}$ by solving (P) without using the Constraint (1). Let c_π be the critical job under the scenario s. Thus, $L^s_{c_\pi}(\pi) > L^s_i(\pi)$. Consider now the scenario s' of maximum relative regret in which c_π is critical. Since s_π is the worst case scenario chosen by Bob for the sequence π and by the definition of s' we have

$$\frac{L(s_\pi,\pi)}{L^*(s_\pi)} \ge \frac{L(s',\pi)}{L^*(s')} \ge \frac{L(s,\pi)}{L^*(s)} = \frac{L^s_{c_\pi}(\pi)}{L^*(s)} > \frac{L^s_i(\pi)}{L^*(s)}$$

In other words, if we remove the Constraint (1), (P) becomes feasible while its objective value cannot be greater than the objective value of the worst case scenario s_π and then the decision of Bob with respect to the sequence π is not affected. This observation is very useful in Alice's algorithm. However, a similar observation cannot hold for Constraint (2) which imposes c_σ to be critical in σ.

As mentioned before, the program (P) is a linear fractional program, in which all constraints are linear, while the objective function corresponds to a fraction of linear expressions of the variables. Moreover, the denominator of the objective function has always a positive value. Charnes and Cooper [7] proposed a polynomial transformation of such a linear fractional program to a linear program. Hence, (P) can be solved in polynomial time.

Note also that, in the case where $c_\pi \neq c_\sigma$, the value of the maximum lateness in the optimal sequence $(\sum_{j \in B(\sigma, c_\sigma)} p_j^s + q_{c_\sigma}^s)$ is fixed since the processing times of jobs processed before the job c_σ in σ, as well as, the delivery time $q_{c_\sigma}^s$ of the job c_σ are already determined in the partial scenario $\bar{s}_{c_\pi, c_\sigma, k}^{\pi}$. Therefore, if $c_\pi \neq c_\sigma$ then (P) is a linear program. Consequently, the Charnes-Cooper transformation is used only in the case where $c_\pi = c_\sigma$.

Theorem 1. *Given a sequence π, there is a polynomial time algorithm that returns a worst case scenario s_π of maximum relative regret $ZR(\pi)$ for the problem $1 \mid\mid L_{\max}$.*

4.2 The Alice's Problem

In this section, we show how Alice constructs an optimal sequence π minimizing the maximum relative regret, i.e., $\pi = \text{argmin}_{\sigma \in \Pi} ZR(\sigma)$. Intuitively, by starting from the last position and going backwards, Alice searches for an unassigned job that, if placed at the current position and it happens to be critical in the final sequence π, will lead to the minimization of the maximum relative regret for π.

In order to formalize this procedure we need some additional definitions. Assume that Alice has already assigned a job in each position $n, n-1, \ldots, r+1$ of π. Let B_r be the set of unassigned jobs and consider any job $i \in B_r$. If i is assigned to the position r in π, then the sets $B(\pi, i)$ and $A(\pi, i)$ coincide with B_r and $\mathcal{J} \backslash B_r$, respectively, and are already well defined, even though the sequence π is not yet completed (recall that $B(\pi, i)$ includes i). Indeed, $B(\pi, i)$ and $A(\pi, i)$ depend only on the position r, i.e., $B(\pi, i) = B(\pi, j) = B_r$ and $A(\pi, i) = A(\pi, j)$ for each couple of jobs $i, j \in B_r$. Hence, Alice can simulate the construction in Bob's procedure in order to decide which job to assign at position r. Specifically, for a given job $i \in B_r$, Alice considers all scenarios $s_{i,j,k}^{B_r}$, where $j \in \mathcal{J}$ is the first critical job in the optimal sequence σ for this scenario and $k \in [\![1, n]\!]$ is the position of i in σ, constructed as described in Bob's algorithm. Note that, we slightly modified the notation of the scenario constructed by Bob for a guess (i, j, k) to $s_{i,j,k}^{B_r}$ instead of $s_{i,j,k}^{\pi}$, since a partial knowledge $(B_r = B(\pi, i))$ of π is sufficient for his procedure. Moreover, the reason of omitting Constraint (1) in the program (P) is clarified here, since the job i is not imposed to be necessarily critical in π. For a job $i \in B_r$, let

$$f_i(\pi) = \max_{j \in \mathcal{J}, k \in [\![1,n]\!]} \left\{ \frac{L(s_{i,j,k}^{B(\pi,i)})}{L^*(s_{i,j,k}^{B(\pi,i)})} \right\}$$

Then, Alice assigns to position r the job i which minimizes $f_i(\pi)$, and the following theorem holds.

Theorem 2. *There is a polynomial time algorithm which constructs a sequence π that minimizes the maximum relative regret for the problem* $1 \mid\mid L_{\max}$.

5 Min-Max Relative Regret for $1 \mid r_j, p_j = 1 \mid L_{\max}$

In this section, we consider the case of unit processing time jobs, i.e., $p_j^s = 1$ for all jobs $j \in \mathcal{J}$ and all possible scenarios $s \in \mathcal{S}$. In contrast to the previous section, the jobs are released on different dates whose values are also imposed to uncertainties. For a fixed scenario, Horn [8] proposes an extension of the Jackson's rule leading to an optimal schedule for this problem: at any time t, schedule the available job, if any, of the biggest delivery time, where a job j is called available at time t if $r_j \leq t$ and j is not yet executed before t.

5.1 The Bob's Problem

Since all jobs are of unit the processing times, a scenario s is described by the values of the release dates and the delivery times of the jobs, i.e., by $r_j^s \in [r_j^{\min}, r_j^{\max}]$ and $q_j^s \in [q_j^{\min}, q_j^{\max}]$, for each $j \in \mathcal{J}$. Recall that, in the presence of different release dates, the execution of the jobs is partitioned into blocks without any idle time, while, given a sequence π and a scenario s, the first job in the block of a job $j \in \mathcal{J}$ is called first-block job for j in π under s. The following lemma characterizes a worst case scenario for a given sequence of jobs π.

Lemma 2. *Let π be a sequence of jobs. There exists a worst case scenario s, a critical job $c \in Crit(s, \pi)$ and its first-block job u_c in π under s such that:*

i for each job $j \in \mathcal{J} \backslash \{c\}$, it holds that $q_j^s = q_j^{\min}$,
ii for each job $j \in \mathcal{J} \backslash \{u_c\}$, it holds that $r_j^s = r_j^{\min}$.

Consider the sequence π chosen by Alice. Bob can guess the critical job c in π and its first-block job u_c. Using Lemma 2, we get a partial scenario $\bar{s}$ by fixing the delivery times of all jobs except for c as well as the release dates of all jobs except for u_c to their minimum values. It remains to determine the values of q_c and r_{u_c} in order to extend the partial scenario $\bar{s}$ to a scenario s. At the end, Bob will choose, among all scenarios created, the worst case one for the sequence π, i.e., the scenario with the maximum value of relative regret.

In what follows, we explain how to construct a sequence σ which will correspond to an optimal schedule for the scenario s when the values of q_c and r_{u_c} will be fixed. The main idea of the proposed algorithm is that, once a couple of σ and s is determined, then σ corresponds to the sequence produced by applying Horn's algorithm with the scenario s as an input. The sequence σ is constructed from left to right along with an associated schedule which determines the starting time B_j and the completion time $C_j = B_j + 1$ of each job $j \in \mathcal{J}$. The assignment of a job j to a position of this schedule (time B_j) introduces additional constraints in order to respect the sequence produced by Horn's algorithm:

(C1) there is no idle time in $[r_j, B_j)$,

(C2) at time B_j, the job j has the biggest delivery time among all available jobs at this time, and
(C3) the delivery times of all jobs scheduled in $[r_j, B_j)$ should be bigger than q_j^s.

These constraints are mainly translated to a refinement of the limits of q_c or of r_{u_c}, i.e., updates on $q_c^{\min}$, $q_c^{\max}$, $r_{u_c}^{\min}$ and $r_{u_c}^{\max}$. If at any point of our algorithm the above constraints are not satisfied, then we say that the assumptions/guesses made become infeasible, since they cannot lead to a couple (σ, s) respecting Horn's algorithm. Whenever we detect an infeasible assumption/guess, we throw it and we continue with the next one.

Let $\ell[x]$ be the x-th job which is released after the time $r_{u_c}^{\min}$, that is, $r_{u_c}^{\min} \leq r_{\ell[1]}^{\bar{s}} \leq r_{\ell[2]}^{\bar{s}} \leq \ldots \leq r_{\ell[y]}^{\bar{s}}$. By convention, let $r_{\ell[0]}^{\bar{s}} = r_{u_c}^{\min}$ and $r_{\ell[y+1]}^{\bar{s}} = +\infty$. To begin our construction, we guess the positions k_c and k_{u_c} of the jobs c and u_c, respectively, in σ as well as the interval $[r_{\ell[x]}^{\bar{s}}, r_{\ell[x+1]}^{\bar{s}})$, $0 \leq x \leq y$, of B_{u_c} in the optimal schedule s for σ. Let $k_{\min} = \min\{k_c, k_{u_c}\}$. We start constructing σ and its corresponding schedule by applying Horn's algorithm with input the set of jobs $\mathcal{J}\backslash\{c, u_c\}$ for which all data are already determined by the partial scenario $\bar{s}$, until $k_{\min} - 1$ jobs are scheduled. Then, we set $\sigma(k_{\min}) = \arg\min\{k_c, k_{u_c}\}$. We now need to define the starting time of $\sigma(k_{\min})$ and we consider two cases:

Case 1: $k_c < k_{u_c}$. We set $B_c = \max\{C_{\sigma(k_{\min}-1)}, r_c^{\bar{s}}\}$. If $B_c = r_c^{\bar{s}}$ and there is an idle time and an available job $j \in \mathcal{J}\backslash\{c, u_c\}$ in $[C_{\sigma(k_{\min}-1)}, B_c)$, then we throw the guess k_c, k_{u_c}, $[r_{\ell[x]}^{\bar{s}}, r_{\ell[x+1]}^{\bar{s}})$ since we cannot satisfy constraint (C1) for j, and hence our schedule cannot correspond to the one produced by Horn's algorithm.

Let $q_a^{\bar{s}} = \max\{q_j^{\bar{s}} : j \in \mathcal{J}\backslash\{c, u_c\}$ is available at $B_c\}$. Then, in order to satisfy constraint (C2) we update $q_c^{\min} = \max\{q_c^{\min}, q_a^{\bar{s}}\}$. Let $q_b^{\bar{s}} = \min\{q_j : j \in \mathcal{J}\backslash\{c, u_c\}$ is executed in $[r_c, B_c)\}$. Then, in order to satisfy constraint (C3) we update $q_c^{\max} = \min\{q_c^{\max}, q_b^{\bar{s}}\}$. If $q_c^{\max} < q_c^{\min}$, then we throw the guess k_c, k_{u_c}, $[r_{\ell[x]}^{\bar{s}}, r_{\ell[x+1]}^{\bar{s}})$ since we cannot get a feasible value for q_c^s.

It remains to check if there is any interaction between c and u_c. Since $k_c < k_{u_c}$, u_c is not executed in $[r_c, B_c)$. However, u_c may be available at B_c, but we cannot be sure for this because the value of $r_{u_c}^s$ is not yet completely determined. For this reason, we consider two opposite assumptions. Note that B_c is already fixed by the partial scenario $\bar{s}$ in the following assumptions, while $r_{u_c}^s$ is the hypothetical release date of u_c in the scenario s.

Assumption 1.1: $r_{u_c}^s \leq B_c$. In order to impose this assumption, we update $r_{u_c}^{\max} = \min\{r_{u_c}^{\max}, B_c\}$.
Assumption 1.2: $r_{u_c}^s > B_c$. In order to impose this assumption, we update $r_{u_c}^{\min} = \max\{r_{u_c}^{\min}, B_c + 1\}$.

If in any of these cases we have that $r_{u_c}^{\max} < r_{u_c}^{\min}$, then we throw the corresponding assumption, since there is no feasible value for $r_{u_c}^s$. For each non-thrown assumption, we continue our algorithm separately, and we eventually get two different couples of sequence/scenario if both assumptions are maintained. More specifically, for each assumption, we continue applying Horn's algorithm with input the set of jobs $\mathcal{J}\backslash\{\sigma(1), \sigma(2), \ldots, \sigma(k_{\min} - 1), c, u_c\}$ starting from time

$C_c = B_c + 1$, until $k_{u_c} - k_c - 1$ additional jobs are scheduled. Then, we set $\sigma(k_{u_c}) = u_c$ and $B_{u_c} = \max\{C_{\sigma(k_{u_c}-1)}, r^{\min}_{u_c}, r^{\bar{s}}_{\ell[x]}\}$. Note that B_{u_c} depends for the moment on the (updated) $r^{\min}_{u_c}$ and not on the final value of $r^{s}_{u_c}$ which has not been determined at this point of the algorithm. If $B_{u_c} \geq r^{\bar{s}}_{\ell[x+1]}$, then we throw the guess on $[r^{\bar{s}}_{\ell[x]}, r^{\bar{s}}_{\ell[x+1]})$. We next check if the constraints (C1)-(C3) are satisfied for all jobs in $\mathcal{J}\backslash\{u_c\}$ with respect to the assignment of the job u_c at the position k_{u_c} of σ with starting time B_{u_c}. If not, we throw the current assumption. Otherwise, Horn's algorithm with input the jobs in $\mathcal{J}\backslash\{\sigma(1), \sigma(2), \ldots, \sigma(k_{u_c})\}$ and starting from time $B_{u_c} + 1$ is applied to complete σ.

Case 2: $k_c > k_{u_c}$. We set $B_{u_c} = \max\{C_{\sigma(k_{\min}-1)}, r^{\min}_{u_c}, r^{\bar{s}}_{\ell[x]}\}$. As before, B_{u_c} depends on $r^{\min}_{u_c}$ and not on the final value of $r^{s}_{u_c}$. If $B_{u_c} \geq r^{\bar{s}}_{\ell[x+1]}$ then we throw the current guess on $[r^{\bar{s}}_{\ell[x]}, r^{\bar{s}}_{\ell[x+1]})$. We need also to check if the constraints (C1)-(C3) are satisfied for all jobs in $\mathcal{J}\backslash\{u_c\}$ with respect to the assignment of the job u_c at the position k_{u_c} of σ with starting time B_{u_c}. If not, we throw the current guess k_c, k_{u_c}, $[r^{\bar{s}}_{\ell(q)}, r^{\bar{s}}_{\ell(q)+1})$. Note that the last check is also applied for c and eventually leads to update $q^{\max}_c = \min\{q^{\max}_c, q^{\bar{s}}_{u_c}\}$ if c is available at B_{u_c}. This can be easily verified because of the guess of the interval of B_{u_c}.

Next, we continue applying Horn's algorithm with input the set of jobs $\mathcal{J}\backslash\{\sigma(1), \sigma(2), \ldots, \sigma(k_{\min}-1), c, u_c\}$ starting from time $B_{u_c}+1$, until $k_c - k_{u_c} - 1$ additional jobs are scheduled. Then, we set $\sigma(k_c) = c$, $B_c = \max\{C_{\sigma(k_c-1)}, r^{\bar{s}}_c\}$, and we check if the constraints (C1)-(C3) are satisfied for all jobs in $\mathcal{J}\backslash\{c\}$ with respect to the assignment of the job c at the position k_c of σ with starting time B_c. If not, we throw the current guess k_c, k_{u_c}, $[r^{\bar{s}}_{\ell[x]}, r^{\bar{s}}_{\ell[x+1]})$. Moreover, an update on $q^{\min}_c$ and $q^{\max}_c$ is possible here, like the one in the begin of case 1. Finally, Horn's algorithm with input the jobs in $\mathcal{J}\backslash\{\sigma(1), \sigma(2), \ldots, \sigma(k_c)\}$ and starting from time $C_c = B_c + 1$ is applied to complete σ.

Note that after the execution of the above procedure for a given guess c, u_c, k_c, k_{u_c}, $[r^{\bar{s}}_{\ell[x]}, r^{\bar{s}}_{\ell[x+1]})$, and eventually an assumption 1.1 or 1.2, we get a sequence σ and its corresponding schedule, while the values of q^{s}_c and $r^{s}_{u_c}$ are still not defined but their bounds are probably limited to fit with this guess. Then, we apply the following three steps in order to get the scenario s:

1. Extend the partial scenario $\bar{s}$ to a scenario $s_{\min}$ by setting $q^{s_{\min}}_c = q^{\min}_c$ and $r^{s_{\min}}_{u_c} = r^{\min}_{u_c}$.
2. Extend the scenario $s_{\min}$ to the scenario s_1 by increasing the delivery time of c to its maximum without increasing the maximum lateness and without exceeding $q^{\max}_c$, i.e., $q^{s_1}_c = q^{s_{\min}}_c + \min\{q^{\max}_c - q^{s_{\min}}_c, L^*(s_{\min}) - L^{s_{\min}}_c(\sigma)\}$.
3. Extend the scenario s_1 to the scenario s by increasing the release date of u_c to its maximum without increasing the maximum lateness, without exceeding $r^{\max}_{u_c}$ and without violating the constraint (C1) and the current guess.

The following theorem holds since in an iteration of the above algorithm, the guess corresponding to an optimal sequence σ for the worst case scenario s will be considered, while Horn's algorithm guarantees the optimality of σ.

Theorem 3. *There is a polynomial time algorithm which, given a sequence π, constructs a worst case scenario s_π of maximum relative regret for the problem $1|r_j, p_j = 1|L_{\max}$.*

5.2 The Alice's Problem

In this section, we describe Alice's algorithm in order to construct an optimal sequence minimizing the maximum relative regret for $1|r_j, p_j = 1|L_{\max}$. Since Alice knows how Bob proceeds, she can do a guess g of the five parameters c, u_c, k_c, k_{u_c}, $[r_{\ell[x]}, r_{\ell[x+1]})$ in order to construct an optimal sequence σ_g for a scenario s_g corresponding to this guess. Then, she assumes that σ_g is provided as input to Bob. Bob would try to maximize its relative regret with respect to σ_g by eventually doing a different guess $\hat{g}$, obtaining a scenario $s_{\hat{g}}$, i.e.,

$$RR(s_{\hat{g}}, \sigma_g) = \max_{g'} \frac{L(s_{g'}, \sigma_g)}{L^*(s_{g'})}$$

Note that, if $g = \hat{g}$, then $RR(s_{\hat{g}}, \sigma_g) = 1$ since by definition σ_g is the optimal sequence for the scenario $s_g = s_{\hat{g}}$. Therefore, Alice can try all possible guesses in order to find the one that minimizes her maximum relative regret by applying Bob's algorithm to the sequence obtained by each guess, and hence the following theorem holds.

Theorem 4. *There is a polynomial time algorithm which constructs a sequence π minimizing the maximum relative regret for the problem $1|r_j, p_j = 1|L_{\max}$.*

Note that, Bob's guess for this problem defines almost all parameters of a worst case scenario, without really using the input sequence provided by Alice. This is not the case in Sect. 4 where, according to Lemma 1, the jobs that succeed the critical job in Alice's sequence should be known. For this reason Alice's algorithm is simpler here compared to the one in Sect. 4.2.

6 Conclusions

We studied the min-max relative regret criterion for dealing with interval uncertain data for the single machine scheduling problem of minimizing the maximum lateness. We considered two variants and we proved that they can be solved optimally in polynomial time. Our main technical contribution concerns the sub-problem of maximizing the relative regret for these variants. The complexity of our results justifies in a sense the common feeling that the min-max relative criterion is more difficult than the min-max regret criterion.

Note that our result for the variant without release dates can be extended even in the case where the jobs are subject to precedence constraints. Indeed, Lawler [13] proposed an extension of Jackson's rule for the deterministic version of this problem, while the monotonicity property still holds. Thus, the corresponding lemma describing a worst case scenario holds, and the determination

of the optimal sequence depends only on the guess of the position of the critical job in this sequence which should be imposed to respect the precedence constraints.

In the future, it is interesting to clarify the complexity of the general maximum lateness problem with respect to min-max relative regret when all parameters are subject to uncertainty. We believe that this problem is NP-hard. If this is confirmed, the analysis of an approximation algorithm is a promising research direction.

Acknowledgement. This research has been partially supported by the ANR Lorraine Artificial Intelligence project (ANR-LOR-AI).

References

1. Aissi, H., Bazgan, C., Vanderpooten, D.: Min-max and min-max regret versions of combinatorial optimization problems: a survey. Eur. J. Oper. Res. **197**(2), 427–438 (2009)
2. Aloulou, M., Della Croce, F.: Complexity of single machine scheduling problems under scenario-based uncertainty. Oper. Res. Lett. **36**, 338–342 (2008)
3. Averbakh, I.: Minmax regret solutions for minimax optimization problems with uncertainty. Oper. Res. Lett. **27**(2), 57–65 (2000)
4. Averbakh, I.: Computing and minimizing the relative regret in combinatorial optimization with interval data. Discret. Optim. **2**(4), 273–287 (2005)
5. Averbakh, I.: The minmax regret permutation flow-shop problem with two jobs. Eur. J. Oper. Res. **169**(3), 761–766 (2006)
6. Buchheim, C., Kurtz, J.: Robust combinatorial optimization under convex and discrete cost uncertainty. EURO J. Comput. Optim. **6**(3), 211–238 (2018). https://doi.org/10.1007/s13675-018-0103-0
7. Charnes, A., Cooper, W.W.: Programming with linear fractional functionals. Nav. Res. Logistics Q. **9**(3–4), 181–186 (1962)
8. Horn, W.: Some simple scheduling algorithms. Nav. Res. Logistics Q. **21**(1), 177–185 (1974)
9. Jackson, J.: Scheduling a production line to minimize maximum tardiness. Research report, Office of Technical Services (1955)
10. Kacem, I., Kellerer, H.: Complexity results for common due date scheduling problems with interval data and minmax regret criterion. Discret. Appl. Math. **264**, 76–89 (2019)
11. Kasperski, A.: Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion. Oper. Res. Lett. **33**(4), 431–436 (2005)
12. Kouvelis, P., Yu, G.: Robust Discrete Optimization and Its Applications. Kluwer Academic Publishers, Amsterdam (1997)
13. Lawler, E.L.: Optimal sequencing of a single machine subject to precedence constraints. Manage. Sci. **19**(5), 544–546 (1973)
14. Lebedev, V., Averbakh, I.: Complexity of minimizing the total flow time with interval data and minmax regret criterion. Discret. Appl. Math. **154**, 2167–2177 (2006)
15. Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P.: Complexity of machine scheduling problems. In: Studies in Integer Programming, Annals of Discrete Mathematics, vol. 1, pp. 343–362. Elsevier (1977)

16. Tadayon, B., Smith, J.C.: Robust Offline Single-Machine Scheduling Problems, pp. 1–15. Wiley, Hoboken (2015)
17. Yang, J., Yu, G.: On the robust single machine scheduling problem. J. Comb. Optim. **6**, 17–33 (2002)