In: Sequencing and Scheduling with Inaccurate Data ISBN: 978-1-61209-579-0
Editors: Yuri N. Sotskov and Frank Werner

Chapter 8

MINMAX (REGRET) SCHEDULING PROBLEMS

Adam Kasperski[1]*** and Paweł Zieliński***[2]†
[1]Institute of Industrial Engineering and Management,
Wrocław University of Technology,
Wybrzeże Wyspiańskiego 27, 50-370, Wrocław, Poland
[2]Institute of Mathematics and Computer Science,
Wrocław University of Technology,
Wybrzeże Wyspiańskiego 27, 50-370, Wrocław, Poland

Abstract

In this chapter a class of scheduling problems with uncertain parameters is discussed. The uncertainty is modeled by specifying a scenario set containing all possible vectors of the problem parameters which may occur. No additional information for the scenario set, such as a probability distribution, is provided. In order to choose a solution, the robust optimization framework is applied. The goal is to compute a schedule with the best worst-case performance over all scenarios. This performance is measured by the minmax and minmax regret criteria. The complexity of various minmax (regret) scheduling problems and some algorithms for solving them are described.

Keywords: Scheduling, minmax, minmax regret, robust optimization

AMS Subject Classification: 90B35, 90C05, 68W40.

1. Introduction

Scheduling problems arise in many applications, in particular in many manufacturing and service industries. There are a lot of scheduling models with different set of parameters and various objectives and a comprehensive description of them can be found, for example, in books [9, 46]. Deterministic scheduling models assume that all parameters, in particular job characteristics, are known in advance. Hence the cost of each schedule is precisely known and the objective is to compute a schedule of minimum cost. However, in practical

*E-mail address: adam.kasperski@pwr.wroc.pl
†E-mail address: pawel.zielinski@pwr.wroc.pl

applications decision makers often face problems with uncertain data. In many cases it is not possible to provide the exact values of parameters which may result from a lack of knowledge or a varying nature of the world. Moreover, a schedule which is optimal with respect to estimated parameters may be poor after the true realization of the parameters occurs.

In decision situations under uncertainty, a set of all possible realizations of parameters is a part of the input. This set is called *scenario set* and each particular realization of the parameters is called a *scenario*. Hence, a scenario represents a state of the world which may occur. Now the cost of a schedule depends on a scenario, so its value is also uncertain. Sometimes it is possible to provide an additional information with scenario set such as a probability distribution in this set. In this case the *stochastic optimization* framework can be used to compute a best schedule (see, e.g. [46]). In stochastic optimization, we typically seek a schedule minimizing the expected cost. However, the stochastic approach has several drawbacks. Namely, it may be hard or expensive to estimate probability distributions of unknown parameters. Also, minimizing the expected performance is questionable when a solution is not used repeatedly, but only once.

In practice the risk-averse decision makers are more interested in hedging against the worst possible scenarios. Furthermore, a solution obtained is often evaluated ex post and compared with an optimal solution that could have been obtained if the true realization of the parameters had been available. In order to meet these requirements, the *robust optimization* framework has been proposed [8,37]. In most robust optimization models, we compute a solution minimizing the largest cost (*minmax criterion*) or the largest deviation from the optimum (*minmax regret criterion*) over all scenarios. Both criteria are well known in theory of decision making under uncertainty, where no probability distribution over the states of the world is provided (see, e.g., [42, 51]).

Several methods of defining the scenario set have been proposed in the existing literature. Among the simplest and most popular ones are the discrete and interval uncertainty representations. In the *discrete uncertainty representation*, we define a scenario set by simply enumerating all possible scenarios. So, in this case, the scenario set is finite. In the *interval uncertainty representation*, the value of each parameter is known to fall within a closed interval and scenario set is the Cartesian product of all these intervals. In the discrete uncertainty representation the scenarios may correspond to some events which globally influence the problem parameters. On the other hand, the interval uncertainty representation allows us to model a local uncertainty connected with a single parameter which may vary independently on the values of the other parameters. For a deeper discussion on both uncertainty representations we refer the reader to [37], where the robust optimization framework has been also compared with the stochastic one.

In this chapter, we discuss several basic scheduling models in which a set of jobs must be executed on a set of machines. The characteristic of each job is specified by three parameters, namely a processing time, a due date and a weight. Each of these three parameters may be uncertain. We use the discrete and interval uncertainty representations to define scenario sets. In order to choose a solution, the minmax and minmax regret criteria are applied. For all particular scheduling models, we present complexity results and methods of solving them which are known to date. We also present some open problems, which are interesting subjects of further research.

This chapter is organized as follows. In Section 2, we formulate a deterministic scheduling problem and set up the notation and terminology used in the subsequent sections. In Section 3, we describe the minmax (regret) approach to solving scheduling problems with uncertain parameters. We describe two popular methods of uncertainty representations, namely the discrete and interval ones. Sections 3.1–3.5 are devoted to particular scheduling models. For each such a model, we present the complexity of its minmax (regret) version and show some methods of solving it known to date.

2. Deterministic Scheduling Problems

Let $J = \{J_1, \ldots, J_n\}$ be a set of jobs which must be processed on $m \geq 1$ machines from a set $M = \{1, \ldots, m\}$. For simplicity of notations, we will identify job J_j with its index j. If a job requires a number of processing steps (operations), then the pair (i, j) refers to the operation of job j on machine i. The set of jobs may be partially ordered by some precedence constraints. The notation $k \to l$ means that processing of job l cannot start before processing of job k is finished (job l is called a *successor* of job k). Several special cases of the precedence constraints such as chains, out-tree, or in-tree can be considered (see, e.g., [9, 46]). For each job j the following data may be specified: a nonnegative *processing time* p_{ij} on machine i (the subscript i is omitted when $m = 1$ or the processing time does not depend on the machine), a nonnegative *due date* d_j and a positive *weight* w_j. The due date d_j expresses a desired completion time of j and the weight w_j expresses the importance of job j relative to the other jobs in the system. In all scheduling models discussed in this chapter we assume that all the jobs are ready for processing at time 0, in other words, each job has a release date equal to 0. For each job j, let $f_j(t)$ denote the cost of completing j at time t. The following cost functions are commonly used: a *weighted completion time* $f_j(t) = w_j t$, a *weighted tardiness* $f_j(t) = w_j \max\{0, t - d_j\}$ and a *weighted unit penalty* $f_j(t) = w_j$ if $t > d_j$ and $f_j(t) = 0$ otherwise. We can also consider the unweighted versions of these functions (when all the weights are equal to 1) obtaining the completion time, tardiness and unit penalty, respectively.

A *schedule* π is an assignment of jobs to machines so that the precedence constraints among the jobs are not violated, no machine processes more than one job at a time and no job is processed on more than one machine at a time. A precise description of the schedule depends on detailed problem characteristics. In most scheduling problems discussed in this chapter each schedule can be represented by a feasible permutation of the jobs. In this case, we will use the standard notation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$, where $\pi(i)$ is the ith job in schedule π. The set of all feasible schedules will be denoted by Π. Let $C_{ij}(\pi)$ be the completion time of job j on machine i in schedule π. The completion time of j on the last machine in π is denoted by $C_j(\pi)$. We denote by $f(\pi)$ the cost of schedule π. Two types of the cost function are typically used: the *linear sum cost function* $f(\pi) = \sum_{j \in J} f_j(C_j(\pi))$, and the *bottleneck cost function* $f(\pi) = \max_{j \in J} f_j(C_j(\pi))$. In a deterministic scheduling problem, we seek a feasible schedule which minimizes the cost $f(\pi)$, that is:

$$\mathcal{P}: \min_{\pi \in \Pi} f(\pi).$$

Scheduling problems are typically denoted by the standard three-field notation of Graham et al. [17], i.e. $\mathcal{P} = \alpha|\beta|\gamma$, where α describes the machine environment, β describes the

job characteristics and γ describes the cost function. For example, $1|prec|\sum C_j$ is the single machine scheduling problem in which the jobs are partially ordered by arbitrary precedence constraints and the cost function is the sum of completion times. For more details, we refer the reader to [9].

3. Minmax (regret) Scheduling Problems

Let Γ be a *scenario set* containing all possible realizations of the problem parameters (processing times, due dates, and weights). These realizations are called *scenarios* and represent all states of the world which may occur. We will use $p_{ij}(S), d_j(S), w_j(S)$ to denote a processing time, a due date and a weight of job j under scenario $S \in \Gamma$, respectively. The completion times of job j in schedule π and the cost of schedule π depend on scenario $S \in \Gamma$ and will be denoted by $C_{ij}(\pi,S)$, $C_j(\pi,S)$, and $f(\pi,S)$, respectively. We will also use $f^*(S)$ to denote the cost of an optimal schedule under a fixed scenario S. In order to compute $f^*(S)$, we need to solve a deterministic problem for the fixed parameters specified by S. In this chapter, we study the following two robust scheduling problems:

$$\text{MINMAX } \mathcal{P}: \min_{\pi\in\Pi} F(\pi) = \min_{\pi\in\Pi} \max_{S\in\Gamma} f(\pi,S) = OPT_1,$$

$$\text{MINMAX REGRET } \mathcal{P}: \min_{\pi\in\Pi} Z(\pi) = \min_{\pi\in\Pi} \max_{S\in\Gamma} \{f(\pi,S) - f^*(S)\} = OPT_2.$$

The value $F(\pi)$ stands for the *maximum cost* of π over all scenarios and the value of $Z(\pi)$ is the *maximum regret* of schedule π over all scenarios. An optimal solution to MINMAX (REGRET) $\mathcal{P}$ is called an *optimal minmax (regret) schedule*. A scenario maximizing the cost or regret of a given schedule π is called a *worst case scenario* for π and an optimal schedule under this worst case scenario is called a *worst case alternative* for π.

We now describe two methods of defining the scenario set Γ which we will use in this chapter. In the *discrete uncertainty representation*, Γ is finite and contains $K \geq 1$ explicitly listed scenarios, i.e. $\Gamma = \{S_1, \ldots, S_K\}$. Notice that in this representation, the value of $F(\pi)$ can be computed in polynomial time. The value of $Z(\pi)$ can also be computed in polynomial time, provided that $\mathcal{P}$ is polynomially solvable. As we will see in the next sections, the MINMAX (REGRET) $\mathcal{P}$ problem may be NP-hard, even if $\mathcal{P}$ is polynomially solvable and the number of scenarios equals 2. We will distinguish two cases while describing complexity results for a particular problem. In the first case the number of scenarios is constant and in the second one the number of scenarios is unbounded (it is a part of the input).

In the *interval uncertainty representation* the value of each parameter may fall within a closed interval, namely $p_j \in [\underline{p}_j, \overline{p}_j]$, $d_j \in [\underline{d}_j, \overline{d}_j]$, $w_j \in [\underline{w}_j, \overline{w}_j]$ for $j \in J$, and Γ is the Cartesian product of all these intervals. In this case Γ is infinite, however the finite number of *extreme scenarios*, in which the values of the parameters take their maximal or minimal values will play an important role. For the interval uncertainty representation, MINMAX $\mathcal{P}$ is typically not harder to solve than its deterministic counterpart $\mathcal{P}$. In all scheduling problems discussed in this chapter, the cost function is nondecreasing in processing times and weights and nonincreasing in due dates. In consequence, for the minmax problem, the worst case scenario S is the same for all schedules, and is such that $p_j(S) = \overline{p}_j$, $w_j(S) = \overline{w}_j$ and $d_j(S) = \underline{d}_j$ for all $j \in J$. Hence, the minmax problem reduces to computing an optimal solution for S. On the other hand, the minmax regret version of the problem may be much more

complex. The first nontrivial task is to provide a characterization of a worst case scenario for a given schedule π and to compute the maximum regret $Z(\pi)$. Contrary to the class of combinatorial optimization problems, described for instance in [27], there is no general characterization of the worst case scenarios for interval parameters and the maximum regret criterion. Furthermore, for some problems there may be no extreme worst case scenario and an efficient method of computing the value of $Z(\pi)$ for a fixed π is unknown.

There are only few general properties valid for all MINMAX (REGRET) $\mathcal{P}$ problems. It is clear that both robust problems cannot be easier than $\mathcal{P}$, because we get the deterministic problem $\mathcal{P}$ when $|\Gamma| = 1$. Hence all negative complexity results known for $\mathcal{P}$ remain true for MINMAX $\mathcal{P}$. For the MINMAX REGRET $\mathcal{P}$ a stronger general result holds. It is easy to see that when the deterministic problem $\mathcal{P}$ is NP-hard, then MINMAX REGRET $\mathcal{P}$ is not at all approximable unless P=NP. This property follows from the fact that it is NP-hard to compute a schedule π such that $Z(\pi) \leq 0$ in the deterministic case.

3.1. Single Machine Scheduling with the Maximum Weighted Tardiness

In this section, we discuss the minmax (regret) version of $1|prec|\max w_j T_j$. In this problem the jobs are to be processed on one machine. For each job $j \in J$: a processing time p_j, a due date d_j and a weight w_j are specified and we seek a feasible schedule (permutation of the jobs) which minimizes the maximum weighted tardiness. The deterministic problem can be solved in $O(n^2)$ time by using the well known Lawler's algorithm [38]. Let $T_j(\pi,S) = [C_j(\pi,S) - d_j(S)]^+$ be the *tardiness* of job j in schedule π under scenario S (we use the notation $[x]^+ = \max\{0,x\}$). The cost of π under S is $f(\pi,S) = \max_{j\in J} w_j(S)T_j(\pi,S)$, so it is the maximum weighted tardiness under S. The maximum cost of π can be expressed as follows:

$$\begin{aligned} F(\pi) &= \max_{S\in\Gamma}\max_{j\in J} w_j(S)T_j(\pi,S) = \max_{j\in J}\max_{S\in\Gamma} w_j(S)T_j(\pi,S) \\ &= \max_{j\in J}\max_{S\in\Gamma}[w_j(S)(C_j(\pi,S) - d_j(S))]^+. \end{aligned} \tag{1}$$

Similarly, the maximum regret of π can be expressed as follows:

$$\begin{aligned} Z(\pi) &= \max_{S\in\Gamma}\{\max_{j\in J} w_j(S)T_j(\pi,S) - f^*(S)\} \\ &= \max_{j\in J}\max_{S\in\Gamma}\{[w_j(S)(C_j(\pi,S) - d_j(S))]^+ - f^*(S)\} \\ &= \max_{j\in J}\max_{S\in\Gamma}[w_j(S)(C_j(\pi,S) - d_j(S)) - f^*(S)]^+, \end{aligned} \tag{2}$$

where the last equality follows from the fact that $Z(\pi) \geq 0$ and $f^*(S) \geq 0$ for any $S \in \Gamma$. Fix a nonempty subset of jobs $D \subseteq J$ and define

$$F_j(D) = \max_{S\in\Gamma}[w_j(S)(\sum_{i\in D} p_i(S) - d_j(S))]^+, \tag{3}$$

$$Z_j(D) = \max_{S\in\Gamma}[w_j(S)(\sum_{i\in D} p_i(S) - d_j(S)) - f^*(S)]^+. \tag{4}$$

The following proposition is immediate:

Proposition 1. *If $D_2 \subseteq D_1$, then for any $j \in J$ it holds $Z_j(D_1) \geq Z_j(D_2)$ and $F_j(D_1) \geq F_j(D_2)$.*

Let $pred(\pi, j)$ be the set of jobs containing job j and all the jobs that precede j in π. Since $C_j(\pi, S) = \sum_{i \in pred(\pi,j)} p_i(S)$, the maximum cost and the maximum regret of π can be expressed as follows (see (1) and (2)):

$$F(\pi) = \max_{j \in J} F_j(pred(\pi, j)), \;\; Z(\pi) = \max_{j \in J} Z_j(pred(\pi, j)). \tag{5}$$

We now show that Algorithm 1 correctly solves both robust problems.

Algorithm 1 Algorithm for solving MINMAX (REGRET) $1|prec|\max w_j T_j$.

```
D ← {1,...,n}
for r ← n downto 1 do
    Find j ∈ D, which has no successor in D and has the minimum value of F_j(D) (Z_j(D))
    π(r) ← j
    D ← D \ {j}
end for
return π
```

Theorem 1. *Algorithm 1 computes an optimal minmax (regret) schedule for* MINMAX (REGRET) $1|prec|\max w_j T_j$.

Proof. We prove the theorem for the minmax version of the problem. The proof for the minmax regret version is just the same. Let π be the schedule returned by the algorithm. It is clear that π is feasible. Let us renumber the jobs so that $\pi = (1, 2, \ldots, n)$. Let σ be an optimal minmax schedule. Assume that $\sigma(j) = j$ for $j = k+1, \ldots, n$, where k is the smallest position among all optimal minmax schedules. If $k = 0$, then we are done, because $\sigma = \pi$ is optimal. Assume that $k > 0$, and so $k \neq \sigma(k) = i$. Let us move the job k just after i in σ and denote the resulting schedule as σ' (see Figure 3.1). Schedule σ' is feasible, because π is feasible. We now consider three cases:

1. If $j \in P \cup R$, then $pred(\sigma', j) = pred(\sigma, j)$ and $F_j(pred(\sigma', j)) = F_j(pred(\sigma, j))$.
2. If $j \in Q \cup \{i\}$, then $pred(\sigma', j) \subseteq pred(\sigma, j)$ and, according to Proposition 1, $F_j(pred(\sigma', j)) \leq F_j(pred(\sigma, j))$.
3. If $j = k$, then $F_j(D) \leq F_i(D)$ from the construction of Algorithm 1. Since $pred(\sigma, i) = pred(\sigma', j) = D$, we have $F_j(pred(\sigma', j)) \leq F_i(pred(\sigma, i))$.

From the above three cases and equality (5), we conclude that

$$F(\sigma') = \max_{j \in J} F_j(pred(\sigma', j)) \leq \max_{j \in J} F_j(pred(\sigma, j)) = F(\sigma),$$

so σ' is also optimal, which contradicts the minimality of k. □

Observe that the running time of the algorithm depends on the complexity of computing $F_j(D)$ and $Z_j(D)$ for given j and $D \subseteq J$. This complexity depends on the way in which scenario set Γ is defined and will be discussed in the next two sections.

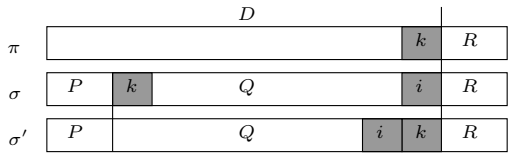

Figure 1. Illustration of the proof of Theorem 1.

3.1.1. Discrete uncertainty representation

If Γ contains a finite number of K scenarios, then Algorithm 1 can be implemented to run in $O(Kn^2)$ time. It is enough to introduce additional array containing the current values $(\sum_{i\in D} p_i(S_1),\ldots,\sum_{i\in D} p_i(S_K))$, which is initialized for $D=J$ just after line 1 and modified just after line 5 of Algorithm 1. For the minmax regret version we also need to introduce the array $(f^*(S_1),\ldots,f^*(S_K))$ containing the costs of the optimal schedules under all scenarios. These costs can be computed only once in $O(Kn^2)$ time just after line 1 of Algorithm 1. Having these two arrays, the job j in line 3 can be found in $O(Kn)$ time. We thus get the following result:

Theorem 2. MINMAX (REGRET) $1|prec|\max w_jT_j$ *can be solved in* $O(Kn^2)$ *time.*

3.1.2. Interval uncertainty representation

For the interval uncertainty representation the minmax problem reduces to solving its deterministic counterpart, which follows from the general remark presented in Section 3. We thus focus only on the minmax regret version, for which we need to compute the value of $Z_j(D)$. Let us define

$$Z_j(D,S) = [w_j(S)(\sum_{i\in D} p_i(S) - d_j(S)) - f^*(S)]^+. \tag{6}$$

Hence $Z_j(D) = \max_{S\in\Gamma} Z_j(D,S)$ and we have to find scenario $S\in\Gamma$ maximizing (6). Observe that in order to maximize (6), one can fix $w_i(S)=\underline{w}_i$, $d_i(S)=\overline{d}_i$ for all $i\in J\setminus\{j\}$ and $p_i(S)=\underline{p}_i$ for all $i\notin D$. It thus remains to fix $d_j(S)$, $w_j(S)$ and $p_i(S)$ for all $i\in D$.

Proposition 2. *In order to maximize (6) we can fix* $d_j(S)=\underline{d}_j$ *and* $w_j(S)=\overline{w}_j$.

Proof. It is enough to show that $Z_j(D,S)$ is nonincreasing with respect to $d_j(S)$ and nondecreasing with respect to $w_j(S)$. Let S be a fixed scenario and σ be such that $f^*(S)=f(\sigma,S)$. Define $C=\sum_{i\in D} p_i(S)$. If $Z_j(D,S)=0$, then $Z_j(D,S')\geq Z_j(D,S)$ for all $S'\in\Gamma$. We can thus assume that $Z_j(D,S)>0$ which implies $C>d_j(S)$. Let us subtract $\delta>0$ from $d_j(S)$ and denote the resulting scenario by S'. It holds $w_j(S')(C-d_j(S'))=w_j(S)(C-d_j(S))+\delta w_j(S)$ and $f^*(S')\leq f(\sigma,S')\leq f(\sigma,S)+\delta w_j(S)$. Hence $Z_j(D,S')\geq Z_j(D,S)$. Let us add $\delta>0$ to $w_j(S)$ and denote the resulting scenario by S'. It holds $w_j(S')(C-d_j(S'))=w_j(S)(C-d_j(S))+\delta(C-d_j(S))$. Suppose that the job j completes at time C_1 in σ under S. If $C_1>C$, then $f^*(S)=f(\sigma,S)\geq w_j(S)(C_1-d_j(S))>w_j(S)(C-d_j(S))$ and $Z_j(D,S)=0$ which contradicts the assumption that $Z_j(D,S)>0$. If $C_1\leq C$, then $f^*(S')\leq f(\sigma,S')\leq f(\sigma,S)+\delta(C_1-d_j(S))\leq f(\sigma,S)+\delta(C-d_j(S))$. Consequently, $Z_j(D,S')\geq Z_j(D,S)$. □

Proposition 3. *Assume that $\underline{w}_i = 1$ for all $i \in J$. Then there is an extreme scenario S maximizing (6) under which $d_j(S) = \underline{d}_j$, $w_j(S) = \overline{w}_j$, $d_i(S) = \overline{d}_i$, $w_i(S) = 1$ for $i \in J \setminus \{j\}$, $p_i(S) = \overline{p}_i$ for $i \in D$ and $p_i(S) = \underline{p}_i$ for $i \in J \setminus D$.*

Proof. Let S be a scenario maximizing (6) and σ be a schedule such that $f^*(S) = f(\sigma, S)$. We can assume that $Z_j(D,S) > 0$ since otherwise any scenario maximizes (6). According to Proposition 2 we can fix in S the due dates and weights as in the proposition. We can also assume that $p_i(S) = \underline{p}_i$ for all $i \notin D$. It remains to show that increasing any processing time $p_i(S)$, $i \in D$, cannot decrease (6). Let us add $\delta > 0$ to some $p_i(S)$, $i \in D$, and denote the resulting scenario as S'. It holds $\overline{w}_j(\sum_{i \in D} p_i(S') - \underline{d}_j) = \overline{w}_j(\sum_{i \in D} p_i(S) - \underline{d}_j) + \delta\overline{w}_j$. On the other hand, this modification cannot increase the completion time of any job in σ by more than δ and the tardiness by more than $\delta\overline{w}_j$, since $\overline{w}_j$ is the largest weight in S. Consequently $f^*(S') \leq f(\sigma, S') \leq f(\sigma, S) + \delta\overline{w}_j$ and $Z_j(D,S') \geq Z_j(D,S)$. □

Using Propositions 2 and 3, we can identify some special cases of the problem which are polynomially solvable. If all the processing times are precisely known or $\underline{w}_j = 1$ for all $j \in J$, then we can easily find a scenario maximizing (6) and compute $Z_j(D)$. Notice that the computation of $Z_j(D)$ requires $O(n^2)$ time, because we have to execute Lawler's algorithm to compute $f^*(S)$. We thus get the following corollary:

Corollary 1. *If all the processing times are precisely known or $\underline{w}_j = 1$ for all $j \in J$, then* MINMAX REGRET $1|prec|\max w_j T_j$ *is polynomially solvable.*

We now estimate the running time of the algorithm. The general case described in Corollary 1 can be solved in $O(n^4)$ time. Suppose that job processing times are deterministic. In this case, in order to compute $Z_j(D)$, we need the values of $f^*(S)$ for only n scenarios S_j, $j \in J$. The scenario S_j is such that $w_j(S_j) = \overline{w}_j$, $d_j(S_j) = \underline{d}_j$, $w_i(S_j) = \underline{w}_i$, $d_i(S_j) = \overline{d}_i$ for all $i \in J \setminus \{j\}$. We can compute all $f^*(S_j)$ in $O(n^3)$ time before executing Algorithm 1. An optimal minmax regret schedule can be then computed in $O(n^2)$ time. So the overall computation time is $O(n^3)$. Finally, consider the case when all due dates are deterministic and all the weights are equal to 1 (only job processing times are uncertain). Proposition 3 provides then an extreme scenario S which can be used to compute $Z_j(D)$ and $Z_j(D) = [\sum_{i \in D} \overline{p}_i - f^*(S) - d_j]^+$. The value of $\sum_{i \in D} \overline{p}_i - f^*(S)$ is the same for each $j \in D$ and line 3 of Algorithm 1 can be replaced with the following line: find $j \in D$, which has no successor in D and has the maximal value of d_j. Consequently, the computation of an optimal minmax regret schedule does not depend on job processing times and can be performed in $O(n^2)$ time.

The cases described in Corollary 1 are easy, because we can find efficiently extreme scenarios maximizing (6). Observe that we have no characterization of a scenario maximizing (6), when the processing times are uncertain and jobs have deterministic, but distinct weights. To see that such a characterization may be more complex consider an example with two jobs and the following parameters (the example is from [59]): $p_1 \in [2,4]$, $d_1 = 10$, $w_1 = 100$, $p_2 = 7$, $d_2 = 7$ and $w_2 = 1$. We would like to compute $Z_2(\{1,2\})$. There are exactly two extreme scenarios S_1 and S_2 that differ from each other in the processing time of job 1. In the scenario S_1 job 1 has processing time equal to 2 and in scenario S_2 its processing time is 4. It is easy to check that $f^*(S_1) = 0$ and $f^*(S_2) = 4$. So, if only these two scenarios are considered, formula (6) gives us $Z_2(\{1,2\}) = \max\{9-7-0, 11-7-4\} = 2$.

But if we consider scenario S in which the processing time of job 1 equals 3, then we get $f^*(S) = 0$ and $Z_2(\{1,2\}) = \max\{10-7-0\} = 3$. Hence there is no extreme scenario maximizing (6). In fact, we have just shown that if the processing times are uncertain and the weights of jobs may be different, then there may be no extreme worst case scenario for a given schedule. This makes the analysis of the problem more complex and challenging. Up to date, the complexity of the general problem is open.

3.1.3. Notes and references

The deterministic $1|prec|\max w_jT_j$ problem is a special case of more general problem $1|prec|f_{\max}$, where $f_{\max}$ is a regular bottleneck cost function. Every such a problem can be solved in $O(n^2)$ time by a simple algorithm designed by Lawler [38]. The minmax regret sequencing problem with interval processing times, interval due dates and the maximum lateness criterion was discussed by Kasperski in [26]. The algorithm constructed in [26] is similar to Algorithm 1 and also runs in $O(n^4)$ time. The MINMAX REGRET $1|prec|\max w_jT_j$ problem with interval weights, deterministic processing times and deterministic due dates was discussed by Averbakh in [5]. An $O(n^3)$ algorithm was proposed to solve this problem, which can be viewed as a special case of Algorithm 1. The problem with discrete uncertainty representation was first investigated by Aloulou and Croce in [4]. Namely, the authors discussed the problem with uncertain processing times, uncertain due dates and deterministic job weights equal to 1 under all scenarios. They showed that the minmax version of the problem can be solved in $O(Kn^2)$ time. In this chapter we generalize the algorithm for arbitrary nonnegative weights. Proposition 3 was first proven by Volgenant and Duin [59]. Algorithm 1 is a generalization of the algorithms proposed in [4, 26, 59]. It is similar to Lawler's algorithm and remains valid for any uncertainty representation. Its implementation requires a method of solving the optimization problems (3) and (4), which depends on the definition of scenario set Γ.

The most interesting open problem is to characterize the complexity of the minmax regret version of the problem with interval processing times and distinct job weights. An efficient method of computing the maximum regret of a given schedule should be first designed. The example showed in Section 3.1.2 (see also [59]) suggests that this task may be nontrivial.

3.2. Single Machine Scheduling with Weighted Sum of Completion Times

In this section we discuss the minmax (regret) versions of $1|prec|\sum w_jC_j$. In this problem, for each job $j \in J$ a processing time p_j and a weight w_j are specified. The jobs are to be processed on one machine and we seek a feasible permutation of the jobs minimizing the weighted sum of completion times. If the precedence constraints are arbitrary, then $1|prec|\sum C_j$ is strongly NP-hard [40]. However, if the precedence constraints are out-tree, in-tree or series parallel, then $1|prec|\sum w_jC_j$ is polynomially solvable [9]. Without precedence constraints an optimal schedule can be obtained in $O(n\log n)$ time by applying the Smith rule, i.e. by ordering the jobs with respect to nondecreasing ratios p_j/w_j [57].

Under uncertainty both processing times and weights may be uncertain, and the cost of a given schedule π under scenario S is expressed as $f(\pi,S) = \sum_{j\in J} w_j(S)C_j(\pi,S)$. The

maximum cost and the maximum regret of π are then as follows:

$$F(\pi) = \max_{S\in\Gamma} \sum_{j\in J} w_j(S)C_j(\pi,S), \;\; Z(\pi) = \max_{S\in\Gamma}\{\sum_{j\in J} w_j(S)C_j(\pi,S) - f^*(S))\}.$$

The deterministic $1|prec|\sum w_jC_j$ problem has an interesting property which will be exploited in the next sections. Consider a problem with job processing times p_j and weights w_j, $j \in J$. Let us create a new problem by inverting the role of processing times and weights, namely w_j is now the processing time of job j and p_j is its weight. We also invert the precedence constraints, i.e. if $k \to l$ then $l \to k$ in the new problem. Let π' be the inverted schedule π, i.e. $\pi' = (\pi(n),\ldots,\pi(1))$. An easy computation shows that the cost of π in the original problem is the same as the cost of π' in the new one. The transformation just described allows us to convert any problem with deterministic processing times and uncertain weights into an equivalent problem with uncertain processing times and deterministic weights (this observation was first established in [43]). It is enough to transform the scenario set Γ by inverting the role of processing times and weights under each scenario $S \in \Gamma$ and invert the precedence constraints. In particular, the transformation implies the following result, which remains valid for both discrete and interval uncertainty representations:

Proposition 4. MINMAX (REGRET) $1|prec|\sum C_j$ *and* MINMAX (REGRET) $1|prec, p_j = 1|\sum w_jC_j$ *are equivalent.*

3.2.1. Discrete uncertainty representation

We first present the complexity of the simplest version of the problem in which there are no precedence constraints among jobs and all job weights are equal to 1. So, Γ contains K processing times scenarios. The following result demonstrates that the minmax (regret) version of this problem becomes NP-hard even when the number of scenarios is 2:

Theorem 3 ([61]). MINMAX (REGRET) $1||\sum C_j$ *is NP-hard for 2 processing time scenarios and strongly NP-hard when the number of processing time scenarios is unbounded.*

Theorem 4 ([43]). MINMAX $1||\sum C_j$ *is hard to approximate within a factor less then* $6/5$ *when the number of processing time scenarios is unbounded.*

From Proposition 4, it follows that we can replace problem $1||\sum C_j$ with $1|p_j = 1|\sum w_jC_j$ in Theorems 3 an 4 and we get the same complexity results for the problem with unit processing times and uncertain weights. Finally, the following theorem states that the minmax version of the problem, in which both processing times and weights can be uncertain, is hard to approximate within any constant factor:

Theorem 5 ([43]). *If the number of scenarios is unbounded, then for every* $\varepsilon > 0$*, the* MINMAX $1||\sum w_jC_j$ *problem cannot be approximated within a ratio of* $O(\log^{1-\varepsilon} n)$*, where* n *is the input size unless the problems in NP have quasi polynomial algorithms.*

We now design 0-1 programming formulation to solve the general problem. Let $\delta_{ij} \in \{0,1\}$ be a binary variable such that $\delta_{ij} = 1$ if job j is processed after job i in a schedule

constructed. The minmax version of the problem can be then formulated as follows:

$$\begin{array}{lll} \text{ILP:} & \min t & \\ & \sum_{j\in J} p_j(S)w_j(S) + \sum_{j\in J}\sum_{i\in J\setminus\{j\}} \delta_{ij}p_i(S)w_j(S) \le t & S\in\Gamma \\ & \delta_{ij}+\delta_{ji}=1 & i,j\in J \\ & \delta_{ij}+\delta_{jk}+\delta_{ki}\ge 1 & i,j,k\in J \\ & \delta_{ij}=1 & i\to j \\ & \delta_{ij}\in\{0,1\} & i,j\in J \end{array} \tag{7}$$

We obtain the formulation for the minmax regret version after subtracting the constant $f^*(S)$ from the left hand side of the first constraint. However, we have to assume that the deterministic problem is polynomially solvable, since otherwise we are not able to compute efficiently the values of $f^*(S)$. The obtained 0-1 programming formulation is compact, i.e. the number of variables and constraints is bounded by a polynomial in n and K. In order to solve it a standard solver, such as CPLEX [22], can be applied. We describe the efficiency of this formulation and some its refinements in Notes and References at the end of this section. Formulation ILP allows us also to design an approximation algorithm for a special case of the problem. This algorithm will be presented in the next theorem, which is due to [43].

Theorem 6. *If the job processing times (weights) are deterministic, then* MINMAX $1|prec|\sum w_jC_j$ *with K weight (processing time) scenarios is approximable within 2.*

Proof. Assume that the job processing times are deterministic, i.e. $p_j(S)=p_j$ for each $S\in\Gamma$, and Γ contains K weight scenarios. Notice that this problem is equivalent to the one in which the job weights are deterministic and Γ contains K processing time scenarios. Since the job processing times are deterministic, each feasible schedule can be specified by a vector of job completion times (C_j). Using the binary variables δ_{ij}, we can define C_j as follows: $C_j = p_j + \sum_{i\in J\setminus\{j\}}\delta_{ij}p_i,\ j\in J$. By relaxing the constraints $\delta_{ij}\in\{0,1\}$, we get the following system of inequalities:

$$\begin{array}{ll} C_j = p_j + \sum_{i\in J\setminus\{j\}}\delta_{ij}p_i & j\in J \\ \delta_{ij}+\delta_{ji}=1 & i,j\in J \\ \delta_{ij}+\delta_{jk}+\delta_{ki}\ge 1 & i,j,k\in J \\ \delta_{ij}=1 & i\to j \\ \delta_{ij}\ge 0 & i,j\in J \end{array} \tag{8}$$

It has bee proved in [52,53] (see also [20]) that each feasible vector (C_j) in (8) satisfies the following inequalities:

$$\sum_{j\in I} p_jC_j \ge \frac{1}{2}\left((\sum_{j\in I}p_j)^2 + \sum_{j\in I}p_j^2\right) \text{ for all } I\subseteq J \tag{9}$$

Consider now the following relaxation of (7) for deterministic processing times:

$$\begin{array}{lll} \text{ILPR:} & \min t & \\ & \sum_{j\in J} C_j w_j(S) \le t & S \in \Gamma \\ & C_j = p_j + \sum_{i\in J\setminus\{j\}} \delta_{ij} p_i & j \in J \\ & \delta_{ij} + \delta_{ji} = 1 & i,j \in J \\ & \delta_{ij} + \delta_{jk} + \delta_{ki} \ge 1 & i,j,k \in J \\ & \delta_{ij} = 1 & i \to j \\ & \delta_{ij} \ge 0 & i,j \in J \end{array}$$

We have explicitly defined the job completion times in ILP and relaxed the constraints $\delta_{ij} \in \{0,1\}$. Since ILPR is a linear programming problem, it can be solved in polynomial time. Let (C_j^*) and t^* be an optimal solution to ILPR. Let us label the jobs so that $C_1^* \le C_2^* \le \cdots \le C_n^*$. The vector (C_j^*) must satisfy (9). Hence, by setting $I = \{1,\ldots,j\}$ we get

$$\sum_{i=1}^{j} p_i C_i^* \ge \frac{1}{2}\left((\sum_{i=1}^{j} p_i)^2 + \sum_{i=1}^{j} p_i^2 \right) \ge \frac{1}{2}\left((\sum_{i=1}^{j} p_i)^2 \right).$$

Since $C_j^* \ge C_i^*$ for each $i \in \{1\ldots j\}$, we get $C_j^* \sum_{i=1}^{j} p_i \ge \sum_{i=1}^{j} p_i C_i^* \ge \frac{1}{2}(\sum_{i=1}^{j} p_i)^2$ and, finally $C_j = \sum_{i=1}^{j} p_j \le 2C_j^*$ for each $j \in J$. Consider schedule $\pi = (1,2,\ldots,n)$. For each scenario $S \in \Gamma$, it holds $f(\pi,S) = \sum_{j\in J} C_j w_j(S) \le 2\sum_{j\in J} C_j^* w_j(S) \le 2t^*$, which implies $F(\pi) = \max_{S\in\Gamma} f(\pi,S) \le 2t^*$. From the fact that t^* is a lower bound on the maximum cost of an optimal schedule, we conclude that π is a 2-approximate solution. The bound of 2 is tight and the example illustrating this fact can be found in [43]. □

We now show that if there are no precedence constraints among the jobs, then an optimal schedule can be constructed by a dynamic programming algorithm. Observe that we can treat the minmax (regret) problem with K scenarios as a multicriteria scheduling problem. Namely, each schedule π is evaluated by the vector $(f(\pi,S_1),\ldots,f(\pi,S_K))$ of the costs under K scenarios. We can define the set of *efficient schedules*, denoted by Π^e, which has the following property: for any schedule $\sigma \in \Pi$ there exists a schedule $\pi \in \Pi^e$ such that $f(\pi,S_i) \le f(\sigma,S_i)$ for each $i = 1,\ldots,K$. We associate with Π^e the set Ω containing all vectors of the costs of the schedules in Π^e. The set Ω is called *efficient*. It is easy to check that Π^e contains an optimal minmax (regret) schedule. An optimal schedule for the minmax problem is the one in Π^e for which the largest cost over all scenarios is minimal. In a similar way we can find an optimal schedule for the minmax regret problem. It is enough to subtract first $f^*(S_i)$ from each $f(\pi,S_i)$, $\pi \in \Pi^e$, $i = 1,\ldots,K$. The idea of the dynamic algorithm, which follows from [43], is to compute recursively the sets Π^e and Ω. We will illustrate this algorithm by using a sample problem with 4 jobs and 3 scenarios shown in Table 1. Consider a vector $\alpha = ((w_1,p_1),(w_2,p_2),\ldots,(w_K,p_K))$ of K pairs of weight/processing time called a *job profile*. Suppose that in a given instance of the problem there are L distinct job profiles $\alpha_1,\ldots,\alpha_L$, where the profile α_i appears n_i times. In the sample problem there are 3 distinct job profiles $\alpha_1 = ((6,1),(2,1),(2,6))$, $\alpha_2 = ((4,2),(2,5),(1,5))$, $\alpha_3 = ((6,5),(1,1),(4,3))$, where profile α_2 appears twice. We thus have $n_1 = 1$, $n_2 = 2$, and $n_3 = 1$. Consider a $(n_1+1)\times(n_2+1)\times\cdots\times(n_L+1)$ dimensional table. A cell $c = (c_1,\ldots,c_L)$ represents a subinstance $I(c)$ of the problem in which

Table 1. A sample problem.

	S_1		S_2		S_3	
J	$w(S_1)$	$p(S_1)$	$w(S_2)$	$p(S_2)$	$w(S_3)$	$p(S_3)$
1	6	1	2	1	2	6
2	4	2	2	5	1	5
3	3	2	1	1	4	3
4	4	2	2	5	1	5

there are c_i jobs with profile α_i. For example, $c = (1,1,0)$ represent the instance with jobs 1 and 2, and $c = (0,2,0)$ represents an instance with jobs 2 and 4 (which are the same). The number of jobs in $I(c)$ is denoted by $n(c) = c_1 + \cdots + c_L$. Of course, the cell $(n_1,\ldots,n_L)$ represents the original instance with all jobs. For each partial schedule π' in the instance $I(c)$ we can determine the vector $(f(\pi',S_1),\ldots,f(\pi',S_K))$ of the costs under scenarios in Γ. Let $\Omega(c)$ be the set of *efficient* cost vectors for the instance specified by cell c. This set has the following property: for any schedule π' in the instance $I(c)$, there is a schedule σ' in the instance $I(c)$ with $(f(\sigma'),S_1),\ldots,f(\sigma',S_K)) \in \Omega(c)$, such that $f(\sigma',S_j) \leq f(\pi',S_j)$ for each scenario $j = 1,\ldots,K$. The idea of the algorithm is to compute recursively the sets $\Omega(c)$ in order of increasing $n(c)$. Then the last set corresponding to the cell with $n(c) = n$, will contain the cost of an optimal solution to the minmax (regret) problem and this solution can be retrieved from the computation path. The computations for the sample problem are illustrated in Figure 2.

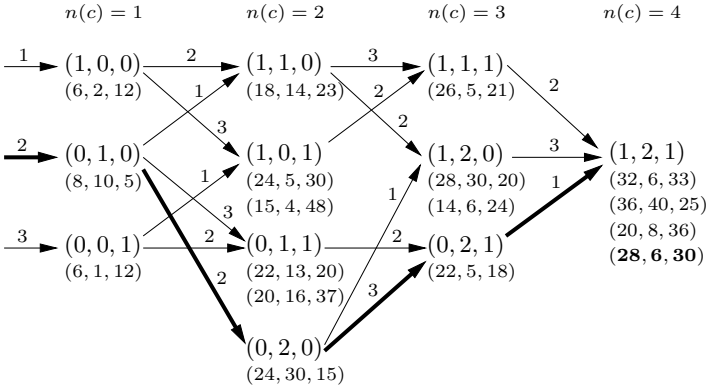

Figure 2. Illustration of the dynamic algorithm.

We first consider the cells with $n(c) = 1$. These cells correspond to the instances with one job. For example, $(1,0,0)$ corresponds to the instance containing only job 1 So, $\Omega(c)$ has exactly one schedule $\pi' = (1)$ with $f(\pi',S_1) = 6$, $f(\pi',S_2) = 2$ and $f(\pi',S_3) = 12$. The vector of the costs $(6,2,12)$ is shown below c in Figure 2. After computing the sets $\Omega(c)$ for all c with $n(c) = 1$, we can determine the efficient sets for all cells c with $n(c) = 2$. For example, cell $(1,1,0)$ can be reached from the cells $(1,0,0)$ and $(0,1,0)$. The former correspond to partial schedule $(1,2)$ while the latter to the partial schedule $(2,1)$. After computing the cost vectors for both schedules, we get $(26,22,27)$ and $(18,14,23)$,

respectively. Since $(18,14,23)$ dominates $(26,22,27)$, we remove $(26,22,27)$ from $\Omega(c)$. A similar computations can be performed for cell $(1,0,1)$, in which we need to consider the cost vectors for two partial schedules $(1,3)$ and $(3,1)$. Because none of these cost vectors dominates the other, we must keep both of them in $\Omega(c)$. Now it is clear how to determine the remaining sets $\Omega(c)$ and the computations are presented in Figure 2. The last cell $(1,2,1)$ contains 4 vectors and the vector $(28,6,30)$ corresponds to optimal schedules $(2,4,3,1)$ and $(4,2,3,1)$ (jobs 4 and 2 are just the same).

Let us now examine the running time of the dynamic algorithm. Let $p_{\max}$ and $w_{\max}$ be the largest processing time and the largest weight in a given instance of the problem. Since the cost of every schedule π under each scenario is bounded by $n^2 p_{\max} w_{\max}$, the number of cost vectors in each $\Omega(c)$ is bounded by $(n^2 p_{\max} w_{\max})^K$. The computation of $\Omega(c)$ for each cell c requires at most $L(n^2 p_{\max} w_{\max})^K + (n^2 p_{\max} w_{\max})^{2K}$ time, where the second term in this sum represents the time required to remove dominated vectors from $\Omega(c)$ by using a pairwise comparison. The number of cells is bounded by n^L, so the whole table can be filled in time $n^L(L(n^2 p_{\max} w_{\max})^K + (n^2 p_{\max} w_{\max})^{2K})$. The number of different job profiles L is bounded by $(p_{\max} \cdot w_{\max})^K$. The running time of the algorithm is exponential. Notice, however, that when the number of scenarios K is constant and the job processing times and due dates in all scenarios are bounded by a constant, then the algorithm is polynomial in n.

The dynamic programming approach presented in this section is general and can also be applied to other minmax (regret) scheduling problems in which there are no precedence constraints between jobs and the cost function is regular. An algorithm obtained in this way will be, however, exponential so its applicability is limited only to small instances.

3.2.2. Interval uncertainty representation

For the interval uncertainty representation the minmax regret problem is NP-hard, which is due to the following result:

Theorem 7 ([39]). MINMAX REGRET $1||\sum C_j$ *with interval processing times is NP-hard.*

According to Proposition 4, MINMAX REGRET $1|p_j = 1|\sum w_j C_j$ with interval weights is also NP-hard. In the existing literature, exact methods of solving the problem are known only for the special case in which there are no precedence constraints among jobs and all job weights are equal to 1, i.e. for MINMAX REGRET $1||\sum C_j$. Using Proposition 4, all these exact methods can also be applied to MINMAX REGRET $1|p_j = 1|\sum w_j C_j$ with interval weights. Let us start by recalling an important result which allows us to construct a partial optimal minmax regret schedule. For any two jobs i and j, we write $i \preceq j$ if $\underline{p}_i \leq \underline{p}_j$ and $\overline{p}_i \leq \overline{p}_j$. The relation $\preceq$ establishes a partial order in the set of jobs and the following theorem holds:

Theorem 8 ([13]). *There exists an optimal minmax regret schedule π, in which for each pair of jobs i and j, if $i \preceq j$, then i is processed before j in π.*

Using Theorem 8 we can immediately determine the relative positions in an optimal minmax regret schedule for every pair of jobs i, j such that $i \preceq j$. In particular, if it is possible to totally order all jobs in J so that $\pi(1) \preceq \pi(2) \preceq \cdots \preceq \pi(n)$, then π is an optimal minmax regret schedule. It is not possible if there are some pairs of nested processing

time intervals, that is when for some two jobs i and j it holds $[\underline{p}_i,\overline{p}_i] \subset [\underline{p}_j,\overline{p}_j]$. Hence the complexity of the problem increases with the number of nested processing time intervals. We will apply Theorem 8 to refine some exact and approximation algorithms for solving the problem.

Our goal now is to construct a mixed integer programming formulation for the problem. The formulation ILP, shown in the previous section, is not appropriate, because the number of scenarios is infinite. As we will see later, for each schedule π there exists an extreme worst case scenario. So, it is possible to transform Γ into a discrete scenario set containing all extreme scenarios and use ILP to solve the problem. However, the resulting problem will have an exponential number of constraints. In order to solve the problem we will use a different representation of a schedule. Let us define binary variables x_{ij}, $i,j = 1,\ldots,n$, where $x_{ij} = 1$ if job j occupies position i (i.e. $j = \pi(i)$). These binary variables must satisfy assignment constraints which ensure that each job is assigned to exactly one position and each position is occupied by exactly one job. Suppose that a binary vector $\boldsymbol{x} = (x_{ij})$ represents schedule π and a binary vector $\boldsymbol{y} = (y_{kj})$ represents schedule σ. Then for each scenario $S \in \Gamma$ it holds

$$\sum_{j\in J} C_j(\pi,S) = \sum_{j=1}^{n}\sum_{i=1}^{n}(n-i+1)p_j(S)x_{ij}, \tag{10}$$

$$\sum_{j\in J} C_j(\sigma,S) = \sum_{j=1}^{n}\sum_{k=1}^{n}(n-k+1)p_j(S)x_{kj}, \tag{11}$$

$$\sum_{j\in J} C_j(\pi,S) - \sum_{j\in J} C_j(\sigma,S) = \sum_{j=1}^{n} p_j(S)\sum_{k=1}^{n}\sum_{i=1}^{n}(k-i)x_{ij}y_{kj}. \tag{12}$$

Equalities (10) and (11) are easy to verify. To see that (12) is true, suppose that job j occupies position i in π and position k in σ. Then $x_{ij} = 1$, $y_{kj} = 1$ and, according to (12), job j contributes $p_j(S)(k-i)$. We get exactly the same result for job j when we subtract (11) from (10). Using (12), the maximum regret $Z(\pi)$ of schedule π described by (x_{ij}) can be expressed as follows:

$$Z(\pi) = \max_{\boldsymbol{y}} \max_{S\in\Gamma} \sum_{j=1}^{n} p_j(S)\sum_{k=1}^{n}\sum_{i=1}^{n}(k-i)x_{ij}y_{kj}. \tag{13}$$

Given a vector (y_{kj}), the scenario maximizing (13) can be obtained in the following way: $p_j(S) = \overline{p}_j$ if $k-i \geq 0$ and $p_j(S) = \underline{p}_j$ if $k-i < 0$. So, it holds

$$Z(\pi) = \max_{\boldsymbol{y}} \sum_{j=1}^{n}\sum_{k=1}^{n}\left(\overline{p}_j\sum_{i=1}^{k}(k-i)x_{ij} + \underline{p}_j\sum_{i=k}^{n}(k-i)x_{ij}\right)y_{kj}. \tag{14}$$

Let $\boldsymbol{x} = (x_{ij})$ describes schedule π and let us define $c_{kj}(\boldsymbol{x}) = \overline{p}_j\sum_{i=1}^{k}(k-i)x_{ij} + \underline{p}_j\sum_{i=k}^{n}(k-i)x_{ij}$. Since $c_{kj}(\boldsymbol{x})$ are constant for a fixed $\boldsymbol{x}$, we can compute the value of $Z(\pi)$ by solving the following assignment problem:

$$\begin{array}{lll} \max & \sum_{j=1}^{n}\sum_{k=1}^{n} c_{kj}(\boldsymbol{x})y_{kj} & \\ & \sum_{k=1}^{n} y_{kj} = 1 & j = 1,\ldots,n \\ & \sum_{j=1}^{n} y_{kj} = 1 & k = 1,\ldots,n \\ & y_{kj} \geq 0 & j,k = 1,\ldots,n \end{array} \tag{15}$$

Observe that we can relax the constraints $y_{kj} \in \{0,1\}$ without changing the optimal objective value, which follows from the well known fact that the matrix of the assignment constraints is totally unimodular. The following proposition summarizes the results presented so far:

Proposition 5. *For each schedule π, there exists an extreme worst case scenario and the maximum regret of π can by computed in polynomial time by solving the assignment problem.*

In order to find an optimal schedule we need to find $\boldsymbol{x}$ minimizing (15). The dual problem to (15) has the following form:

$$\begin{array}{ll} \min & \sum_{j=1}^{n} \alpha_j + \sum_{k=1}^{n} \beta_k \\ & \alpha_j + \beta_k \geq c_{kj}(\boldsymbol{x}) \qquad j,k = 1,\ldots,n \end{array} \tag{16}$$

From the definition of $c_{kj}(\boldsymbol{x})$, the fact that $\boldsymbol{x}$ must satisfy the assignment constraints and the equivalence of (15) and (16), we have the following MIP formulation for the problem:

$$\begin{array}{lll} \min & \sum_{j=1}^{n} \alpha_j + \sum_{k=1}^{n} \beta_k & \\ & \alpha_j + \beta_k \geq \overline{p}_j \sum_{i=1}^{k} (k-i) x_{ij} + \underline{p}_j \sum_{i=k}^{n} (k-i) x_{ij} & j,k = 1,\ldots,n \\ & \sum_{i=1}^{n} x_{ij} = 1 & j = 1,\ldots,n \\ & \sum_{j=1}^{n} x_{ij} = 1 & i = 1,\ldots,n \\ & x_{ij} \in \{0,1\} & i,j = 1,\ldots,n \end{array} \tag{17}$$

The size of the resulting MIP formulation is polynomial in n. It can be additionally refined by applying Theorem 8. The idea consists in adding additional constraints, which use relation $\preceq$ to reduce the solution space. Let $a_j = |\{i : i \neq j, i \preceq j\}|$ and $b_j = |\{i : i \neq j, j \preceq i\}|$. Thus a_j is the number of jobs that are known to precede j and b_j is the number of jobs that are known to follow j in an optimal schedule. For each pair of jobs u and v such that $u \preceq v$ we add the following constraint to the MIP formulation (17):

$$\sum_{i=1}^{n} i x_{iv} \geq \sum_{i=1}^{n} i x_{iu} + 1. \tag{18}$$

Constraint (18) expresses the fact that the position occupied by job u is less than the position occupied by job v. Also, for each job j we add the following two constraints:

$$\sum_{i=1}^{a_j} x_{ij} = 0 \tag{19}$$

$$\sum_{i=n-b_j+1}^{n} x_{ij} = 0 \tag{20}$$

Constraints (19) and (20) assure that at least a_j jobs precede j (therefore first a_j positions cannot be occupied by j) and at least b_j jobs follow j (therefore the last $n-b_j+1$ positions cannot be occupied by j). Constraints (18)-(20) may significantly reduce the solution space and consequently speed up calculations. Computational tests performed in [44] suggest that we can solve efficiently the problem with up to 45 jobs by using CPLEX solver to solve the MIP model. We believe that a little bit larger problems can be solved by using faster computers and the latest version of optimization software.

The next theorem shows that the minmax regret problem with interval processing times can also be approximated within 2 if only the underlying deterministic problem is polynomially solvable.

Theorem 9. *If the deterministic* $1|prec|\sum C_j$ *problem is polynomially solvable, then* MINMAX REGRET $1|prec|\sum C_j$ *with interval processing times is approximable within 2.*

Proof. Let us denote by j_π and j_σ the positions occupied by job j in schedules π and σ, respectively. The following equality for any $S \in \Gamma$ is easy to verify:

$$f(\pi,S) - f(\sigma,S) = \sum_{j\in J} C_j(\pi,S) - \sum_{j\in J} C_j(\sigma,S) = \sum_{j\in J} (j_\sigma - j_\pi) p_j(S). \tag{21}$$

Using (21) and the definition of the maximum regret one can easily prove that for any two feasible schedules π and σ the following inequality holds:

$$Z(\pi) \geq \sum_{\{j:j_\sigma > j_\pi\}} (j_\sigma - j_\pi)\overline{p}_j + \sum_{\{j:j_\sigma < j_\pi\}} (j_\sigma - j_\pi)\underline{p}_j. \tag{22}$$

We now show that any feasible schedules π and σ satisfy the inequality:

$$Z(\sigma) \leq Z(\pi) + \sum_{\{j:j_\pi > j_\sigma\}} (j_\pi - j_\sigma)\overline{p}_j + \sum_{\{j:j_\pi < j_\sigma\}} (j_\pi - j_\sigma)\underline{p}_j. \tag{23}$$

Indeed, the following inequality follows from (21), after interchanging π and σ:

$$f(\sigma,S) \leq f(\pi,S) + \sum_{\{j:j_\pi > j_\sigma\}} (j_\pi - j_\sigma)\overline{p}_j + \sum_{\{j:j_\pi < j_\sigma\}} (j_\pi - j_\sigma)\underline{p}_j, \tag{24}$$

If S is a worst case scenario for σ, then subtracting $f^*(S)$ from both sides of (24) yields $Z(\sigma) \leq f(\pi,S) - f^*(S) + \sum_{\{j:j_\pi > j_\sigma\}} (j_\pi - j_\sigma)\overline{p}_j + \sum_{\{j:j_\pi < j_\sigma\}} (j_\pi - j_\sigma)\underline{p}_j$, which, together with $Z(\pi) \geq f(\pi,S) - f^*(S)$, gives (23). Let S^m be the *midpoint scenario*, i.e. $p_j^{S^m} = \frac{1}{2}(\underline{p}_j + \overline{p}_j)$ for all $j \in J$, and let σ be an optimal schedule under S^m. By the assumption that the deterministic problem is polynomially solvable, we can compute σ in polynomial time. Using (21) we can see that for any schedule π, it holds $\sum_{j\in J} (j_\sigma - j_\pi)(\underline{p}_j + \overline{p}_j) \geq 0$ which is equivalent to the following inequality:

$$\sum_{\{j:j_\sigma > j_\pi\}} (j_\sigma - j_\pi)\overline{p}_j + \sum_{\{j:j_\sigma < j_\pi\}} (j_\sigma - j_\pi)\underline{p}_j \geq \sum_{\{j:j_\pi > j_\sigma\}} (j_\pi - j_\sigma)\overline{p}_j + \sum_{\{j:j_\pi < j_\sigma\}} (j_\pi - j_\sigma)\underline{p}_j. \tag{25}$$

Now applying formula (22) to (25) we obtain

$$Z(\pi) \geq \sum_{\{j:j_\pi > j_\sigma\}} (j_\pi - j_\sigma)\overline{p}_j + \sum_{\{j:j_\pi < j_\sigma\}} (j_\pi - j_\sigma)\underline{p}_j. \tag{26}$$

Inequalities (23) and (26) yield $Z(\sigma) \leq Z(\pi) + Z(\pi) = 2Z(\pi)$. It is easy to show that the bound of 2 obtained is tight. Consider a problem with three jobs and with no precedence constraints. The interval processing times of jobs are $\tilde{p}_1 = [0,2]$, $\tilde{p}_2 = [1,1]$ and $\tilde{p}_3 = [1,1]$. The midpoint scenario assigns to all jobs the processing times equal to 1. An easy computation shows that schedule $\sigma = (1,3,2)$ has the maximum regret equal to 2, while the optimal minmax regret schedule $\pi = (3,1,2)$ has the maximum regret equal to 1, which implies $Z(\sigma)/Z(\pi) = 2$. □

Theorem 9, together with Proposition 4, imply that MINMAX REGRET $1|prec, p_j = 1|\sum w_jC_j$ with interval weights is also approximable within 2, if only its deterministic counterpart is polynomially solvable. Theorem 9 suggests an approximation algorithm for solving the problem even if some precedence constraints among jobs are allowed. However, in this case no efficient method of computing the maximum regret of a given feasible schedule is known. In order to compute $Z(\pi)$ we can apply formulation (15) with constraints $y_{kj} \in \{0,1\}$ and additional constraints which force precedence constraints among the jobs. But it is not clear that the resulting model can be solved in polynomial time. Observe, however, that even with presence of precedence constraints, there is still an extreme worst case scenario for each feasible schedule π.

If there are no precedence constraints among jobs, then one can try to improve a solution returned by the 2-approximation algorithm by applying a simple local search. For a given schedule π, we define the neighborhood $N(\pi)$ consisting of all schedules π' resulting from interchanging the positions of two jobs i and j in π. We can restrict the neighborhood by using Theorem 8. Namely, if i is processed before j in π and $i \preceq j$, then the positions of i and j are not interchanged, since the maximum regret of the resulting schedule π' cannot be less than the maximum regret of π. We can use the neighborhood N to construct a simple iterative improvement algorithm which returns a local minimum with respect to N. Recall that the maximum regret of a given schedule can be computed in polynomial time by solving an assignment problem. Thus each step of the algorithm can be done in polynomial time.

Some experiments on the 2-approximation algorithm and the iterative improvement were performed in [27], where problems with up to 40 jobs were examined (those problems can be solved by using the MIP formulation). The simple 2-approximation algorithm, that computes an optimal solution for the midpoint scenario, returns solutions whose average percentage deviations from the optimum are about 8%. By applying the iterative improvement, we can achieve the average percentage deviations from the optimum less than 1%. So, the iterative improvement algorithm seems to be an attractive method of solving large problems, for which the MIP formulation is too slow. It is worth pointing out that in both MIP formulation and the iterative improvement, the dominance property described in Theorem 8 significantly reduces the computation time.

3.2.3. Notes and references

The deterministic $1|prec|\sum w_jC_j$ problem is one of the most basic and extensively studied scheduling problems. A comprehensive description of its complexity and known solution algorithms can be found, for instance, in the books by Brucker [9] and Pinedo [46]. The MINMAX REGRET $1||\sum C_j$ problem with interval processing times was first discussed in the paper by Daniels and Kouvelis [13], where Theorem 8 and the characterization of the worst case scenario of a given schedule (Proposition 5) can be found. The complexity of the problem was characterized by Lebedev and Averbakh in [39], who proved that MINMAX REGRET $1||\sum C_j$ with interval processing times is NP-hard (Theorem 7). Historically, the branch and bound algorithm proposed by Daniels and Kouvelis in [13] was the first exact method of solving the problem. Another approach, which is partially based on the results obtained in [13], was proposed by Montemanni in [44], who constructed a com-

pact mixed integer programming formulation for the problem (this formulation is shown in Section 3.2.2). Since the MIP formulation is simpler and allows us to solve larger problems, we do not describe the branch and bound method in this chapter (we refer the reader to [13] for details). The 2-approximation algorithm described in Theorem 9 was constructed by Kasperski and Zieliński in [33]. Some heuristics based on local search were first suggested in [13] and the iterative improvement algorithm from Section 3.2.2 was proposed by Kasperski in [27]. The local minimum returned by the iterative improvement is at most a factor of 2 away from the global one. However, no example is known for which the worst case ratio of 2 is achieved.

There is a number of interesting open problems on the minmax regret version of the problem with interval data. We know that MINMAX REGRET $1||\sum C_j$ with interval processing times is weakly NP-hard (the reduction shown in [39] is from a variant of the partition problem). It is thus possible that this problem has a pseudopolynomial time algorithm and even admits a fully polynomial time approximation scheme (FPTAS). Computing a worst case scenario of a given schedule requires solving an assignment problem which is much more time consuming than solving the deterministic problem. No faster method of computing the worst case scenario is known. In particular no efficient method of computing the maximum regret is known if some precedence constraints between jobs are allowed (a 0-1 programming model can be constructed for that purpose). Also, no method of characterizing a worst case scenario is known when both job processing times and weights are imprecise. The deterministic $1||\sum C_j$ problem can be generalized to $R||\sum C_j$, in which the jobs can be processed on m unrelated machines. Problem $R||\sum C_j$ is polynomially solvable (see, e.g. [9]). It is thus interesting to extend the minmax regret model to this multimachine case.

The MINMAX (REGRET) $1||\sum C_j$ problem with discrete uncertainty representation was first discussed by Yang and Yu [61]. They proved that this problem is NP-hard for two processing time scenarios and strongly NP-hard if the number of processing time scenarios is unbounded. A dynamic programming algorithm was also proposed in [61] to solve the problem. Theorems 4, 5 and 6 as well as the idea of the dynamic programming, described in Section 3.2.1, is due to Mastrolilli et al. [43]. The MIP formulation (7) is based on an idea proposed by Potts [47]. The 2-approximation algorithm shown in Theorem 7 is based on the results obtained by Schulz [52, 53] and Hall et al. [20].

The MIP formulation (7) was used by Regis de Farias et al. [49] to solve the MINMAX $1||\sum w_j C_j$ problem with deterministic weights. The authors proposed to refine the formulation by adding some additional constraints to reduce the search space (for details we refer the reader to [49]). This refined formulation was then solved by using CPLEX 10 solver and it was shown that problems with up to 140 jobs and 10 processing time scenarios can be solved within reasonable time.

3.3. Single Machine Scheduling Problem with Weighted Sum of Late Jobs

In this section we consider the minmax (regret) versions of $1||\sum w_j U_j$. In this problem, the jobs are to be processed on one machine. For each job $j \in J$ a processing time p_j a due date d_j and a weight w_j are given and the objective is to compute a permutation of the jobs minimizing the weighted number of late jobs. We will assume that there are no precedence

constraints among the jobs, so each schedule is feasible. The problem $1||\sum w_jU_j$ is weakly NP-hard and can be solved in pseudopolynomial time [25]. However, its special cases $1||\sum U_j$ and $1|p_j=1|\sum w_jU_j$ with unit weights and unit processing times, respectively, are polynomially solvable. The former can be solved in $O(n\log n)$ time by using well known Moore's algorithm [45] and the latter has a matroidal structure and can be solved in $O(n\log n)$ time by using a greedy algorithm [9, 12].

We will also discuss the minmax (regret) version of the $1|p_j=1,d_j=d|\sum w_jU_j$ problem, in which all jobs have unit processing times and a deterministic common due date, i.e. $d_j=d$ for all $j\in J$. Observe that in this case the deterministic problem reduces to computing a subset $J'\subseteq J$ of exactly $n-d$ jobs of the smallest weights. An optimal schedule can be then constructed by first processing the jobs in $J\setminus J'$ in any order and then the jobs in J' in any order. It is easily seen that this schedule minimizes the sum of the weights of late jobs. We thus can see that the problem is equivalent to the following SELECTING ITEMS problem: given a set of items E with a weight w_i for each $e_i\in E$, and a number p; compute a subset of precisely p items out of E of the minimum total weight. The minmax (regret) version of SELECTING ITEMS was discussed in a number of papers and all results obtained for this problem remain valid for MINMAX (REGRET) $1|p_j=1,d_j=d|\sum w_jU_j$. We will use this fact later in this section. Define $U_j(\pi,S)=1$ if $C_j(\pi,S)>d_j$ and $U_j(\pi,S)=0$, otherwise. The cost of schedule π under scenario S is $f(\pi,S)=\sum_{j\in J}w_jU_j(\pi,S)$. The maximum cost and the maximum regret of schedule π are then expressed as follows:

$$F(\pi)=\max_{S\in\Gamma}\sum_{j\in J}w_jU_j(\pi,S),\ \ Z(\pi)=\max_{S\in\Gamma}\{\sum_{j\in J}w_jU_j(\pi,S)-f^*(S)\}.$$

3.3.1. Discrete uncertainty representation

The following theorems establish all the known complexity results for the discrete uncertainty representation:

Theorem 10 ([4]). MINMAX $1||\sum U_j$ *is NP-hard even if job due dates are deterministic and there are two processing time scenarios.*

Theorem 11 ([1]). MINMAX $1|p_j=1|\sum U_j$ *is strongly NP-hard and not approximable with a ratio less than 2 if the number of due date scenarios is unbounded.*

Since MINMAX (REGRET) $1|p_j=1,d_j=d|\sum w_jU_j$ is equivalent to MINMAX (REGRET) SELECTING ITEMS the following theorems holds:

Theorem 12 ([6, 34]). MINMAX (REGRET) $1|p_j=1,d_j=d|\sum w_jU_j$ *is NP-hard for 2 weight scenarios and strongly NP-hard if the number of weight scenarios is unbounded.*

Theorem 13 ([30]). *If the number of weight scenarios is unbounded, then* MINMAX $1|p_j=1,d_j=d|\sum w_jU_j$ *is not approximable within any constant factor unless P=NP.*

For the discrete uncertainty representation, the general MINMAX (REGRET) $1||\sum w_jU_j$ problem can be solved by a dynamic algorithm whose idea is just the same as for the problem with the weighted sum of completion times (see Section 3.2.1). A job profile is now a vector of triples $\alpha=((p_1,d_1,w_1),\ldots,(p_K,d_K,w_K))$. Let $p_{\max},d_{\max}$ and $w_{\max}$ be the largest

processing time, due date, and weight in the input instance. The number of different job profiles is bounded by $(p_{\max} \cdot d_{\max} \cdot w_{\max})^K$. The cost of each schedule under any scenario is bounded by $nw_{\max}$. Hence, an analysis similar to that in Section 3.2.1, shows that the number of steps required to fill the dynamic programming table is bounded by $n^L(L(nw_{\max})^K + (nw_{\max})^{2K})$. Again, if the number of scenarios K and the values of the problem parameters are bounded by a constant, then the running time of the dynamic algorithm is polynomial in n.

If all jobs have unit processing times under all scenarios, then the problem with K scenarios is a special case of MINMAX (REGRET) ASSIGNMENT. To see this, let us introduce binary variables $x_{ij} \in \{0,1\}$ such that $x_{ij} = 1$ if job j appears at position i in a schedule constructed. The variables x_{ij} must satisfy assignment constraints, which ensure that each job is processed at exactly one position and each position is occupied by exactly one job. Under each scenario $S \in \Gamma$, the cost $c_{ij}(S)$ of assigning job j to position i is $w_j(S)$ if $i > d_j(S)$ and 0 otherwise. We can now formulate the minmax problem as follows:

$$\begin{array}{ll} \min t & \\ \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}(S)x_{ij} \le t & S \in \Gamma \\ \sum_{i=1}^{n} x_{ij} = 1 & j = 1,\ldots,n \\ \sum_{j=1}^{n} x_{ij} = 1 & i = 1,\ldots,n \\ x_{ij} \in \{0,1\} & i,j = 1,\ldots,n \end{array} \tag{27}$$

For the minmax regret version, we should subtract $f^*(S)$ from the left hand side of the first constraint. Formulation (27) models the MINMAX (REGRET) ASSIGNMENT problem. Since this problem is known to be approximable within K (see, e.g., [2]), we get the following result:

Theorem 14. MINMAX (REGRET) $1|p_j = 1|\sum w_j U_j$ *is approximable within* K.

The K-approximation algorithm simply outputs an optimal assignment for the average costs $\hat{c}_{ij} = \frac{1}{K}\sum_{S\in\Gamma} c_{ij}(S)$. We now use formulation (27) to prove the following result, which is due to [1]:

Theorem 15. *If the number of due date scenarios is constant, then* MINMAX $1|p_j = 1|\sum U_j$ *is approximable within 3.*

Proof. We will use the fact that MINMAX $1|p_j = 1|\sum U_j$ is a special case of MINMAX ASSIGNMENT, where additionally all costs $c_{ij}(S)$ are 0 or 1 (see (27)). Let $\Gamma = \{S_1,\ldots,S_K\}$ and let v be a number in $\{0,\ldots,n\}$. Consider the following problem:

$$CA(v): \quad \begin{array}{ll} \min \ \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}(S_K)x_{ij} & \\ \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}(S_k)x_{ij} \le v & k = 1,\ldots,K-1 \\ \sum_{i=1}^{n} x_{ij} = 1 & j = 1,\ldots,n \\ \sum_{j=1}^{n} x_{ij} = 1 & i = 1,\ldots,n \\ x_{ij} \in \{0,1\} & i,j = 1,\ldots,n \end{array} \tag{28}$$

There exists a value of $v \in \{0,\ldots,n\}$ for which an optimal solution to $CA(v)$ is an optimal solution to MINMAX ASSIGNMENT. Indeed, suppose that the cost of an optimal assignment under scenario S_k is v_k. Since v_k is the number of late jobs under scenario S_k, it

is always an integer between 0 and n. Let $v^* = \max_{k\in\{1,\ldots,K\}} v_k$. It is clear that by solving $CA(v^*)$ we get an optimal assignment. Consequently, we can reduce the problem to solving $n+1$ problems $CA(v)$. Unfortunately, when K is unbounded, then the problem of solving $CA(v)$ is NP-hard and hard to approximate within a ratio less than 2 even if $v = 1$ (see [1]). It has been also shown in [1] that when the costs are 0 or 1 under all scenarios, then the problem of finding an assignment $\boldsymbol{x}$ such that $F(\boldsymbol{x}) \leq v$ is solvable in $O(n^{2vK+3})$ time ($F(\boldsymbol{x})$ is the maximum cost of $\boldsymbol{x}$ over all scenarios in Γ). Let us apply this algorithm for $v = 0,\ldots,K-1$. Consequently, if there is an assignment with the maximum cost less than or equal to $K-1$, then we can obtain an optimal assignment in polynomial time, since the number of scenarios K is assumed to be constant. Otherwise, we know that the maximum cost of an optimal assignment is not less than $K-2$, in other words $v^* \in \{K-2,\ldots,n\}$. We now focus on constructing an approximate solution to $CA(v^*)$. Consider the relaxation of (28) for $v = v^*$, denoted by $CAR(v^*)$, in which the constraints $x_{ij} \in \{0,1\}$ are replaced with $x_{ij} \geq 0$. Denote an optimal fractional extreme solution to $CAR(v^*)$ by (x^*_{ij}). Let us define $F = \{(i,j) : 0 < x^*_{ij} < 1\}$, $I_0 = \{(i,j) : x^*_{ij} = 0\}$ and $I_1 = \{(i,j) : x^*_{ij} = 1\}$. Fix variables, whose indices are in I_0 or I_1 according, to their values in (x^*_{ij}). As the result we get a smaller problem $CAR'(v)$ with variables whose indices are in F, where $(x^*_{ij})_{(i,j)\in F}$ is an extreme solution to $CAR'(v^*)$. It is well known that the number of positive variables in every extreme solution to linear programming problems equals the number of linearly independent constraints. Consequently, each extreme solution to $CAR'(v^*)$ has at most $2(n-|I_1|)+K-2$ positive variables, because one assignment constraint is redundant. Consequently $|F| \leq 2n+K-2$. We now form set F_2 as follows. For each position i such that exactly two pairs (i,j') and (i,j'') appear in F, we add to F_2 a pair, say (i,j') such that $x^*_{ij'} \geq 1/2$. The position i is called *assigned*. We break the ties arbitrarily so that F_2 represents a partial assignment. Finally, we form $F_{>2}$ by choosing an arbitrary pair $(i,j) \in F$ for each unassigned position i so that $F_{>2}$ also represents a partial assignment. It holds $2|F_2|+3|F_{>2}| \leq |F| \leq 2(n-|I_1|)+K-2$ and $n-|I_1| = |F_2|+|F_{>2}|$, which implies $|F_{>2}| \leq K-2$. Let $\boldsymbol{x} = (x_{ij})$ be the assignment represented by $I_1 \cup F_2 \cup F_{>2}$. For each $k = 1,\ldots,K$, it holds $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}(S_k)x_{ij} \leq \sum_{(i,j)\in I_1} c_{ij}(S_k)x^*_{ij} + 2\sum_{(i,j)\in F_2} c_{ij}(S_k)x^*_{ij} + K - 2 \leq 3v^*$. Thus the constructed assignment $\boldsymbol{x}$ is a 3-approximate solution for the problem. □

The approximability of the problem with arbitrary deterministic processing times is established by the next theorem, which is due to [1].

Theorem 16. MINMAX $1||\sum U_j$ *with deterministic processing times and K due date scenarios is approximable within K.*

Proof. Define the fictitious scenario $\hat{S}$ such that $d_j(\hat{S}) = \min_{S\in\Gamma} d_j(S)$ for $j \in J$. Let us denote by $L(\sigma,S)$ the set of late jobs in σ under S. We now show that for any schedule σ it holds

$$L(\sigma,\hat{S}) \subseteq \bigcup_{S\in\Gamma} L(\sigma,S). \tag{29}$$

Let $j \in L(\sigma,\hat{S})$. Then job j completes after $d_j(\hat{S})$ in σ. Since $d_j(\hat{S}) = d_j(S)$ for some scenario $S \in \Gamma$ and all processing times are deterministic, job j is also late under S and $j \in$

$\bigcup_{S\in\Gamma} L(\sigma,S)$. Let π be an optimal schedule under $\hat{S}$, which can by computed in $O(n\log n)$ time by Moor's algorithm. For any schedule σ, (29) implies

$$F(\sigma)=\max_{S\in\Gamma}|L(\sigma,S)|\geq\frac{1}{K}\sum_{S\in\Gamma}|L(\sigma,S)|\geq\frac{1}{K}|L(\sigma,\hat{S})|\geq\frac{1}{K}|L(\pi,\hat{S})|\geq\frac{1}{K}F(\pi),$$

where the last inequality follows from the fact that $|L(\pi,S)|\leq|L(\pi,\hat{S})|$ for each $S\in\Gamma$. Hence $F(\pi)\leq K\cdot F(\sigma)$ for any σ and π is a K-approximate schedule. The bound of K is tight and the corresponding example can be found in [1]. □

It turns out that a better approximation algorithm can be designed for the minmax version of the problem, when all jobs have a common due date. We first construct a MIP model for this special case. Let $x_j\in\{0,1\}$ be a binary variable such that $x_j=1$ if job j is completed after d in the schedule constructed (i.e. it is late). Consider the following MIP formulation:

$$\begin{array}{ll}\min\ t & \\ \sum_{j=1}^{n} w_j(S)x_j\leq t & S\in\Gamma \\ \sum_{j=1}^{n} x_j=n-d & \\ x_j\in\{0,1\} & j=1,\ldots,n\end{array} \tag{30}$$

Let (x_j^*) be an optimal solution to (30) with the objective value of t^*. We construct a schedule π as follows: we first process all the jobs j such that $x_j^*=0$ in any order and then all the jobs j such that $x_j^*=1$ in any order. Clearly, π is an optimal minmax schedule with $F(\pi)=t^*$. Notice that (30) is a MIP formulation for the MINMAX SELECTING ITEMS problem and the following theorem holds:

Theorem 17 ([30, 34]). MINMAX $1|p_j=1,d_j=d|\sum w_jU_j$ *with K weight scenarios is approximable within $O(\log K)$.*

We now briefly describe the idea of the algorithm (all details can be found in [30, 34]). We first consider a relaxation of (30) in which the constraints $x_j\in\{0,1\}$ are replaced with $0\leq x_j\leq 1$. Let $LP(t)$ be the set of constraints of the relaxation of (30) in which we additionally fix $x_j=0$ if $w_j(S)>t$ for some scenario $S\in\Gamma$. Applying binary search we can find in polynomial time the smallest value of t for which $LP(t)$ is feasible. Denote this smallest value by $\hat{t}$ and the corresponding solution to $LP(\hat{t})$ by $(\hat{x}_j)$. We can now design a simple randomized algorithm, which selects job j (fixes $x_j=1$) with probability $\hat{x}_j$. After repeating this selection a polynomial number of times we are guaranteed to obtain an $O(\log K)$ approximate solution with a high probability. This simple randomized algorithm can be converted into a deterministic one by using a method of conditional probabilities and pessimistic estimators (see [30]).

3.3.2. Interval uncertainty representation

In this section we study the MINMAX REGRET $1|p_j=1|\sum w_jU_j$ problem with deterministic due dates d_j and interval weights $[\underline{w}_j,\overline{w}_j]$, $j\in J$. This is the only minmax regret version of the problem with interval data discussed to date. Consider first the case when all jobs have a common due date. The problem is then equivalent to MINMAX REGRET SELECTING

ITEMS with interval weights, which is known to be polynomially solvable. Therefore, the following theorem holds:

Theorem 18 ([11]). MINMAX REGRET $1|p_j = 1, d_j = d|\sum w_j U_j$ *with interval weights is solvable in* $O(n\min\{d, n-d\})$ *time.*

The details of the polynomial time algorithm can be found in [11]. We now discuss the more general case in which job due dates can be distinct. The complexity of this problem is unknown and describing it is an interesting open problem. Our goal is to construct a compact mixed integer programming formulation and an approximation algorithm to solve the problem. Since all the processing times are deterministic (they are equal to 1), the job completion times do not depend on scenarios in Γ. A job j is said to be *on time* in schedule π if $C_j(\pi) \leq d_j$; otherwise it is called *late*. In order to solve the problem it is sufficient to consider only some particular schedules, namely the canonical ones. A schedule π is said to be in a *canonical form* if all on time jobs are processed before all late jobs, and all on-time jobs are ordered with respect to nondecreasing due dates. Using a simple interchange argument, it is easy to show that for any schedule π, there exists a canonical schedule π' such that $f(\pi', S) \leq f(\pi, S)$ for all $S \in \Gamma$. This fact easily implies that there is an optimal minmax regret schedule in the canonical form.

Let us now define set $\boldsymbol{I} \subseteq 2^J$ such that $A \in \boldsymbol{I}$ if and only if all the jobs in A can be scheduled on time. It is not difficult to decide whether $A \in \boldsymbol{I}$. We can proceed as follows: schedule the jobs in A in order of nondecreasing due dates; if at least one job becomes late then $A \notin \boldsymbol{I}$, otherwise all the jobs in A are on time and $A \in \boldsymbol{I}$. It turns out that the system $(J, \boldsymbol{I})$ is a matroid (see e.g. [12]) and we will call it a *sequencing matroid*. A *base* of this matroid is a maximal (under inclusion) subset of jobs which can be scheduled on time. Let $\mathcal{B}$ be the set of all bases of the sequencing matroid $(J, \boldsymbol{I})$. From the properties of the matroidal problems, it follows that all the bases have the same cardinality and we will denote it by l. Hence $|B| = l$ for all $B \in \mathcal{B}$. Let Γ' be the Cartesian product of the intervals $[M - \overline{w}_j, M - \underline{w}_j]$ for $j \in J$, where M is a sufficiently large constant, say $M = \max_{j \in J} \overline{w}_j$. The cost of $B \in \mathcal{B}$ under scenario $S \in \Gamma'$ is $f(B, S) = \sum_{j \in B} w_j(S)$ and the maximum regret of B is $Z(B) = \max_{S \in \Gamma'} \{f(B, S) - f^*(S)\}$, where $f^*(S)$ is the cost of an optimal base under S. For each base $B \in \mathcal{B}$ we can construct the corresponding canonical schedule by first processing the jobs in B in order of nondecreasing due dates and then all the remaining jobs in any order. The following proposition holds:

Proposition 6. *If* $B^* \in \mathcal{B}$ *is a base of the minimum value of the maximum regret over scenario set* Γ'*, then the canonical schedule* π *corresponding to* B^* *is an optimal minmax regret schedule for scenario set* Γ.

Proof. It is easy to see that there is an optimal minmax regret schedule π and a worst case alternative σ for π in which the sets of all on time jobs are maximal. Denote these sets by B and B', $B, B' \in \mathcal{B}$. It holds $f(\pi, S) - f(\sigma, S) = \sum_{j \in B'} w_j(S) - \sum_{j \in B} w_j(S)$. Since each scenario S has the corresponding scenario S', such that $w_j(S) = M - w_j(S')$ and $|B| = |B'| = l$, we get

$$f(\pi, S) - f(\sigma, S) = \sum_{j \in B} w_j(S') - \sum_{j \in B'} w_j(S') = f(B, S') - f(B', S'). \tag{31}$$

Equality (31) implies that the maximum regret of π under scenario set Γ equals the maximum regret of B under scenario set Γ' and the proposition follows. □

We now design a MIP for the following problem: given a sequencing matroid $(J,\boldsymbol{I})$ with interval weights $[\underline{w}_j,\overline{w}_j]$, $j\in J$, compute a base $B\in\mathcal{B}$ minimizing the maximum regret $Z(B)$. Assume that jobs in J are numbered so that $d_1\le d_2\le\cdots\le d_n$. Recall that l is the unique cardinality of any base $B\in\mathcal{B}$. The value of l can easily be obtained by computing any base $B\in\mathcal{B}$ in a greedy way. Let us introduce a binary variable $x_j\in\{0,1\}$ for each $j\in J$, where $x_j=1$ if j belongs to a base constructed. Consider the following set of constraints:

$$\begin{array}{ll}
x_1\le d_1 & \\
x_1+x_2\le d_2 & \\
\vdots & \\
x_1+x_2+\cdots+x_n\le d_n & \\
x_1+x_2+\cdots+x_n=l & \\
x_j\in\{0,1\} & j\in J
\end{array} \tag{32}$$

Clearly, constraints (32) describe characteristic vectors of the solutions (bases) from $\mathcal{B}$. Suppose that (x_j) is a characteristic vector of a base $B\in\mathcal{B}$. Since $|B|=l$, the equality $x_1+\cdots+x_n=l$ is satisfied. Moreover, all jobs from B can be scheduled on time in some canonical schedule π. In this schedule all jobs $\pi(1),\ldots,\pi(l)$ are on time and they are processed according to nondecreasing due dates. Hence $x_{\pi(1)}+\cdots+x_{\pi(j)}\le d_{\pi(j)}$ for $j=1,\ldots l$. Since $x_{\pi(l+1)}=\cdots=x_{\pi(n)}=0$, we can see that the characteristic vector (x_j) satisfies constraints (32). Conversely, suppose that vector (x_j) satisfies (32). Then (x_j) contains precisely l variables which take the value of 1. Furthermore, it is easy to check that the subset of jobs corresponding to these positive variables can be scheduled on time. In consequence (x_j) is a characteristic vector of some base $B\in\mathcal{B}$.

Computing the maximum regret $Z(B)$ for any base $B\in\mathcal{B}$ is not difficult, because we can use the following well known characterization of a worst case scenario for B (see, e.g. [31]): scenario S_B such that $w_j(S_B)=\overline{w}_j$ if $j\in B$ and $w_j(S_B)=\underline{w}_j$ if $j\notin B$ is a worst case scenario for B. Hence, if vector (x_j) represents base B, then the maximum regret $Z(B)$ can be computed as follows:

$$Z(B)=f(B,S_B)-f^*(S_B)=\sum_{j=1}^{n}\overline{w}_jx_j-\min_{(y_j)\in\mathcal{B}}\sum_{j=1}^{n}(\overline{w}_jx_j+\underline{w}_j(1-x_j))y_j. \tag{33}$$

We can express the problem of computing $f^*(S_B)$, given (x_j), as follows:

$$\begin{array}{lll}
\min & \sum_{j=1}^{n}(\overline{w}_jx_j+\underline{w}_j(1-x_j))y_j & \\
 & \sum_{i=1}^{j}-y_i\ge -d_j & j=1,\ldots,n \\
 & y_1+y_2+\cdots+y_n=l & \\
 & -y_j\ge -1 & j=1,\ldots,n \\
 & y_j\ge 0 & j=1,\ldots,n
\end{array} \tag{34}$$

We have multiplied some inequalities by -1 to obtain a standard form of the linear program. One can easily verify that the constraints matrix of (32) is totally unimodular. The dual

to (34) has the following form:

$$\begin{array}{ll} \max & -\sum_{j=1}^{n} d_j\alpha_j - \sum_{j=1}^{n}\beta_j + l\gamma \\ & -\sum_{i=j}^{n}\alpha_i - \beta_j + \gamma \leq \overline{w}_j x_j + \underline{w}_j(1-x_j) \quad j=1,\ldots,n \\ & \alpha_j,\beta_j \geq 0 \quad j=1,\ldots,n \end{array} \tag{35}$$

From (33), (32) and (35), we get the following MIP formulation for the problem:

$$\begin{array}{ll} \min & \sum_{j=1}^{n}\overline{w}_j x_j + \sum_{j=1}^{n} d_j\alpha_j + \sum_{j=1}^{n}\beta_j - l\gamma \\ & \sum_{i=1}^{j} x_i \leq d_j \quad j=1,\ldots,n \\ & x_1 + x_2 + \cdots + x_n = l \\ & -\sum_{i=j}^{n}\alpha_i - \beta_j + \gamma \leq \overline{w}_j x_j + \underline{w}_j(1-x_j) \quad j=1,\ldots,n \\ & \alpha_j,\beta_j \geq 0 \quad j=1,\ldots,n \\ & x_j \in \{0,1\} \quad j=1,\ldots,n \end{array} \tag{36}$$

The formulation (36) can be additionally refined. Namely, some jobs can be added to a solution constructed while some other jobs can be excluded from the solution constructed before solving (36). A job is called *possibly optimal* if it is a part of an optimal base under some scenario $S \in \Gamma$ and it is called a *necessarily optimal* if it is a part of an optimal base under each scenario $S \in \Gamma$. It turns out that job $j \in J$ which is not possibly optimal cannot be a part of any optimal solution to the minmax regret problem and we can fix $x_j = 0$ before solving the problem. On the other hand, if all weight intervals are nondegenerate, i.e. $\underline{w}_j < \overline{w}_j$ for all $j \in J$, then all necessarily optimal jobs can be added to a solution constructed. In other words, we can fix $x_j = 1$ for each necessarily optimal job. Furthermore, all nonpossibly and necessarily optimal jobs can be detected efficiently, which follows from the following property: a job j is possibly optimal if and only if it is a part of an optimal base under scenario S such that $w_j(S) = \underline{w}_j$ and $w_i(S) = \overline{w}_i(S)$ for all $i \neq j$; a job j is necessarily optimal if and only if it is a part of an optimal base under scenario S such that $w_j(S) = \overline{w}_j$ and $w_i(S) = \underline{w}_i(S)$ for all $i \neq j$.

The next theorem shows that the problem is approximable within 2.

Theorem 19. MINMAX REGRET $1|p_j = 1|w_jU_j$ *with deterministic due dates and interval weights is approximable within 2.*

Proof. We have shown that the problem is equivalent to computing an optimal minmax regret base in a sequencing matroid with interval weights. Since that problem belongs to the class of combinatorial optimization problems studied in [31], the 2-approximation algorithm designed in [31] can be applied to solve it. The idea of this algorithm consists in solving the deterministic problem for the midpoint scenario S such that $w_j(S) = \frac{1}{2}(\underline{w}_j + \overline{w}_j)$, $j \in$J. □

The efficiency of the MIP formulation (36) and the 2-approximation algorithm were tested in [27]. Instances with up to 200 jobs were solved to optimality within less than 220 seconds by using the CPLEX 8.0 solver. The preprocessing significantly reduced the problem size as the average number of nonpossibly optimal jobs reported was between 12% - 17% and the average number of necessarily optimal jobs reported was between 18% - 21%. The 2-approximation algorithm performs quite well. For the tested instances it returned solutions whose average deviations from the optimum were less than 2%.

3.3.3. Notes and references

A review of all known complexity results and solution algorithms for the deterministic $1||\sum w_jU_j$ problem can be found, for example, in the book by Brucker [9]. A description of the greedy algorithm for $1|p_j=1|\sum w_jU_j$ can be found in the book by Cormen et al. [12]. The MINMAX $1||\sum U_j$ problem under discrete uncertainty representation was discussed by Aloulou et al. [4] and Aissi et al. [1]. The NP-hardness of this problem for two processing time scenarios (Theorem 10) was established in [4] and the NP-hardness of the problem for unbounded number of due date scenarios (Theorem 11) was proven in [1]. We do not know if the problem is strongly NP-hard when the number of the processing time scenarios is unbounded or NP-hard when the number of due date scenarios is constant. The approximation algorithms shown in Theorems 15 and 16 are due to Aissi et al. [1]. In [1] also some dynamic programming algorithms for solving some special cases of the problem can be found. The MINMAX (REGRET) SELECTING ITEMS problem, which is equivalent to MINMAX (REGRET) $1|p_j=1,d_j=d|\sum w_jU_j$, was first discussed by Averbakh [6], who showed that it is NP-hard for 2 weight scenarios. This result was extended by Kasperski et al. [30], who proved that MINMAX (REGRET) SELECTING ITEMS is not approximable within any constant factor if the number of weight scenarios is unbounded, unless P=NP. In [30, 34] randomized and deterministic $O(\ln K)$- approximation algorithms for MINMAX SELECTING ITEMS were proposed, which can be applied to the corresponding scheduling problem when the number of weight scenarios is unbounded.

The only version of the problem with interval data, namely MINMAX REGRET $1|p_j=1|\sum w_jU_j$ with deterministic due dates and interval weights, was discussed by Kasperski [27]. When all jobs have a common due date, then this problem is equivalent to MINMAX REGRET SELECTING ITEMS with interval weights which is known to be polynomially solvable. The polynomial algorithms for this problem were constructed by Averbakh [6] and Conde [11]. However, the complexity of the problem, when the jobs have deterministic but distinct due dates, is open. In this case the problem has still a matroidal structure and this fact was used by Kasperski [27] to construct the MIP formulation and the 2-approximation algorithm, shown in Section 3.3.2. The approximation algorithm is based on the previous results obtained by Kasperski and Zieliński in [31] for the class of minmax regret combinatorial optimization problems with interval weights. The preprocessing consisting in identifying nonpossibly and necessarily optimal jobs can be done efficiently due to the results obtained by Kasperski and Zieliński in [32].

3.4. Permutation Flow Shop Problem

In this section, we discuss the minmax (regret) versions of the permutation flow shop problem denoted by $F||C_{\max}$. In this problem, the jobs in J are to be processed on m machines from the set $M=\{1,\ldots,m\}$. Each job $j\in J$ is first processed on machine 1, then on machine 2 through machine m. A processing time of job j on machine i is equal to p_{ij}. We seek a permutation of the jobs minimizing the *makespan*, i.e. the completion time of the last job on the last machine (machine m). We will use $Fm||C_{\max}$ to denote the problem in which the number of machines is fixed and equals m. We will also discuss the $F|n=2|C_{\max}$ problem, where the number of jobs equals 2. The deterministic $F2||C_{\max}$ problem is solvable in $O(n\log n)$ time by Johnson's algorithm [24], but $F3||C_{\max}$ is strongly NP-hard [15].

Let $C_{\max}(\pi, S)$ be the makespan of schedule π under scenario S. The maximum cost and the maximum regret of a given schedule π are defined as follows:

$$F(\pi) = \max_{S\in\Gamma} C_{\max}(\pi, S), \;\; Z(\pi) = \max_{S\in\Gamma}\{C_{\max}(\pi, S) - C^*_{\max}(S)\}, \tag{37}$$

where $C^*_{\max}(S)$ is the minimum makespan under S.

In this section we will also use the following additional notation: $L_i(S) = \sum_{j\in J} p_{ij}(S)$ is the total load of machine i under scenario S, $L_{\max}(S) = \max_{i\in M} L_i(S)$ stands for the maximum machine load under S, $L_{\max} = \max_{S\in\Gamma} L_{\max}(S)$ denotes the maximum machine load over the set of scenarios, $l_j(S) = \sum_{i\in M} p^S_{ij}$ is the length of job $j \in J$ in scenario S, $l^{\max}_j = \max_{S\in\Gamma} l_j(S)$ denotes the maximum length of job $j \in J$ over the set of scenarios, $p_{\max}$ is the largest job processing time in an input instance.

3.4.1. Interval uncertainty representation

We now discuss MINMAX REGRET $F|n = 2|C_{\max}$, i.e. the minmax regret version of the permutation flow shop with only two jobs, m machines and interval job processing times, which is known to be polynomially solvable. The presented results are due to [7]. This problem has only two solutions $\pi_{12} = (1,2)$, and $\pi_{21} = (2,1)$, so $\Pi = \{\pi_{12}, \pi_{21}\}$. Therefore, in order to find a schedule π that minimizes the maximum regret $Z(\pi)$, it suffices to determine the values of $Z(\pi_{12})$ and $Z(\pi_{21})$. We now show that this task can be done in polynomial time. The makespans of π_{12} and π_{21} and the minimum makespan under scenario $S \in \Gamma$ are as follows:

$$C_{\max}(\pi_{12}, S) = C_{m2}(\pi_{12}, S), \;\; C_{\max}(\pi_{21}, S) = C_{m1}(\pi_{21}, S),$$
$$C^*_{\max}(S) = \min\{C_{\max}(\pi_{12}, S), C_{\max}(\pi_{21}, S)\}.$$

Determining the value of $C^*_{\max}(S)$ or equivalently solving the deterministic $F|n = 2|C_{\max}$ problem can be done in $O(m)$ time by an algorithm based on a geometric approach, adapted from [9], to solving the deterministic $F|n = 2|C_{\max}$. It turns out that $F|n = 2|C_{\max}$ can be formulated as a shortest path problem in a plane with rectangular objects as obstacles (see Figure 3). The processing times of job 1 (job 2) on machine 1 through machine m under S are represented by intervals on the x-axis (y-axis). A feasible schedule π corresponds to a path from O to F that avoids the obstacles M_i, where point O has coordinates $(0,0)$ and F is the point with coordinates $(\sum_{i=1}^m p_{i1}(S), \sum_{i=1}^m p_{i2}(S))$. The path has the following properties:

(i) It consists of segments which are either parallel to one of the axes (only one job is processed) or diagonal, coordinates of diagonal parts are such that $x - y = \text{const}$, (both jobs are processed in parallel). The length of the path, which corresponds to a maximum completion time of jobs in schedule π, is the sum of the length of horizontal parts, the length of vertical parts and the length of diagonal parts divided by $\sqrt{2}$.

(ii) It avoids the interior of any rectangular obstacle M_i that corresponds to machine i, two jobs cannot be processed simultaneously on the same machine. The horizontal and vertical sides of the rectangle equal to $p_{i1}(S)$ and $p_{i2}(S)$, respectively.

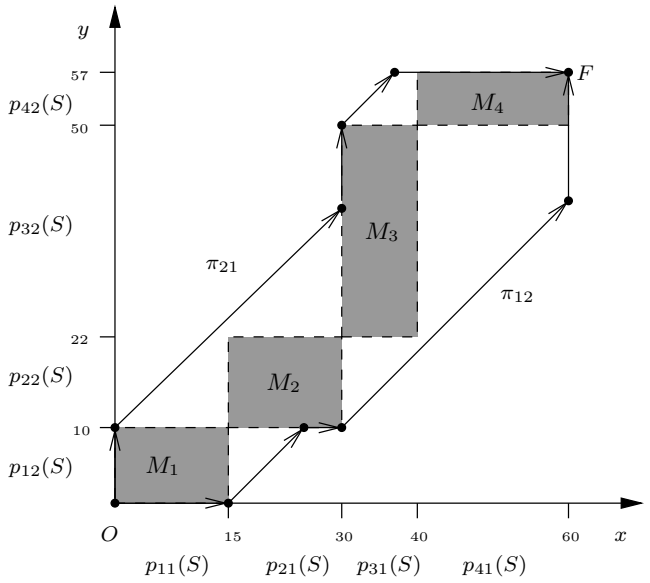

Figure 3. The geometric formulation of an instance of $F|n=2|C_{\max}$ under S.

(iii) All the rectangular obstacles M_i are placed either to the North and West or the South and East from the path which is due to the fact that in a permutation flow shop the order of processing jobs must be the same for all machines.

Observe that there are two paths in Figure 3, which correspond to no idle time feasible schedules π_{12} and π_{21}. The lengths of the paths are equal to $C_{\max}(\pi_{12},S)$ and $C_{\max}(\pi_{21},S)$, respectively. Using the geometric formulation one can explicitly express $C_{\max}(\pi_{12},S)$ and $C_{\max}(\pi_{21},S)$:

$$C_{\max}(\pi_{12},S)=\sum_{i=1}^{m}p_{i1}(S)+\sum_{i=1}^{m}p_{i2}(S)-\sum_{i=1}^{i_{12}(S)-1}p_{i2}(S)-\sum_{i=i_{12}(S)+1}^{m}p_{i1}(S), \tag{38}$$

$$C_{\max}(\pi_{21},S)=\sum_{i=1}^{m}p_{i1}(S)+\sum_{i=1}^{m}p_{i2}(S)-\sum_{i=1}^{i_{21}(S)-1}p_{i1}(S)-\sum_{i=i_{21}(S)+1}^{m}p_{i2}(S), \tag{39}$$

where $i_{12}(S)$ is the index of the obstacle with the smallest value of $y-x$ of its South-East corner and $i_{21}(S)$ is the index of the obstacle with the largest value of $y-x$ of its North-West corner. In case of tie one takes the smallest (the largest) such an index, formally:

$$i_{12}(S)=\arg\min_{i\in M}\{\sum_{k=1}^{i-1}p_{k2}(S)-\sum_{k=1}^{i}p_{k1}(S)\},\ \ i_{21}(S)=\arg\max_{i\in M}\{\sum_{k=1}^{i}p_{k2}(S)-\sum_{k=1}^{i-1}p_{k1}(S)\}.$$

For the example depicted in Figure 3, we get $i_{12}(S)=2$, $i_{21}(S)=3$, $C_{\max}(\pi_{12},S)=60+57-10-30=77$, $C_{\max}(\pi_{21},S)=60+57-30-7=80$ and $C^{*}_{\max}(S)=77$. The following theorem is crucial in determining $Z(\pi_{12})$:

Theorem 20. *Let $\Gamma' = \{S_k : k \in M\} \subset \Gamma$, where S_k is scenario defined as follows:*

$$p_{i1}(S_k) = \begin{cases} \overline{p}_{i1} & \text{if } i \leq k, \\ \underline{p}_{i1} & \text{if } i > k, \end{cases} \quad p_{i2}(S_k) = \begin{cases} \underline{p}_{i2} & \text{if } i < k, \\ \overline{p}_{i2} & \text{if } i \geq k, \end{cases} \quad i \in M.$$

Then a worst case scenario for π_{12} belongs to Γ'.

Proof. Consider a worst case scenario $S \in \Gamma$ for π_{12}. By (38) and (39), and the fact that Π contains only two schedules, the maximum regret of π_{12} can be expressed as follows: $C_{\max}(\pi_{12},S) - C_{\max}(S) = \max\{C_{\max}(\pi_{12},S) - C_{\max}(\pi_{21},S),0\} = \max\{\sum_{i=1}^{i_{21}(S)-1} p_{i1}(S) + \sum_{i=i_{21}(S)+1}^{m} p_{i2}(S) - \sum_{i=1}^{i_{12}(S)-1} p_{i2}(S) - \sum_{i=i_{12}(S)+1}^{m} p_{i1}(S),0\}$. Observe that the above regret cannot decrease if we replace S with $S_{i_{12}(S)} \in \Gamma'$. Hence $S_{i_{12}(S)} \in \Gamma'$ is also a worst case scenario for π_{12}. □

An analogous theorem holds for the schedule π_{21}. Theorem 20 shows that in order to compute $Z(\pi_{12})$ and $Z(\pi_{21})$, it suffices to consider only m extreme scenarios from Γ'. Furthermore, for any scenario S, the values of $i_{12}(S)$, $i_{21}(S)$, $C_{\max}(\pi_{12},S)$ and $C_{\max}(\pi_{21},S)$ can be computed in $O(m)$ time. This leads to an $O(m^2)$-algorithm for the problem. This running time can be reduced to $O(m)$ by computing $C_{\max}(\pi_{12},S_k)$ and $C_{\max}(\pi_{21},S_k)$ for all $S_k \in \Gamma'$ in $O(m)$ time. The key observation is that $i_{12}(S) = \arg\max_{i \in M}\{\sum_{k=i}^{m} p_{k2}(S) - \sum_{k=i+1}^{m} p_{k1}(S)\}$. Set $c_i(S) = \sum_{k=i}^{m} p_{k2}(S) - \sum_{k=i+1}^{m} p_{k1}(S)$. From this and (38), we have:

$$C_{\max}(\pi_{12},S) = \sum_{i=1}^{m} p_{i1}(S) + c_{i_{12}(S)}(S).$$

The values of $\sum_{i=1}^{m} p_{i1}(S)$ for all $S \in \Gamma'$ can be determined in $O(m)$ time. Computing $c_{i_{12}(S)}(S)$ for all $S_k \in \Gamma'$ is more tricky, but can also be done in $O(m)$ time by an incremental technique, i.e. the technique that uses the values already computed (each increment can be done in a constant time), avoiding in this way unnecessary enumeration (see [7] for details). The values of $C_{\max}(\pi_{21},S)$ for all $S \in \Gamma'$ can be computed in $O(m)$ time in a similar way. We have thus shown the following result:

Theorem 21. MINMAX REGRET $F|n=2|C_{\max}$ *is solvable in $O(m)$ time.*

We now study MINMAX REGRET $F2||C_{\max}$, i.e. the minmax regret version of the permutation flow shop with 2 machines. Let us point out that the complexity of this problem is unknown and it is still an interesting and unresolved open problem. We first address the problem of computing the maximum regret $Z(\pi)$ of given schedule π. Let S_π^j be a scenario under which the job processing times are as follows: $\overline{p}_{1\pi(i)}$ for $i \leq j$; $\overline{p}_{2\pi(i)}$ for $i \geq j$; $\underline{p}_{1\pi(i)}$ for $i > j$; and $\underline{p}_{2\pi(i)}$ for $i < j$. Define $\Gamma' = \{S_\pi^1, \ldots, S_\pi^n\}$.

Theorem 22. *Scenario set Γ' contains a worst case scenario for π.*

Proof. Let S be a worst case scenario for schedule π and let $\pi(j^*)$ be a critical job in π under S, i.e. the last job whose completion time on machine 1 is equal to its starting time on machine 2 in schedule π under S. We will show that $S_\pi^{j^*}$ is also a worst case scenario for π. Indeed, when we replace S with $S_\pi^{j^*}$, the makespan of π will increase by exactly

$\Delta = \sum_{i \le j}(\overline{p}_{1\pi(i)} - p_{1\pi(i)}(S)) + \sum_{i \ge j}(\overline{p}_{2\pi(i)} - p_{2\pi(i)}(S))$. On the other hand, the makespan of an optimal schedule under $S_\pi^{j^*}$ cannot increase by more than Δ in comparison with S and the theorem follows. □

Theorem 22 allows us to express the maximum regret of π as follows:

$$Z(\pi) = \max_{j \in J}\{\sum_{i=1}^{j} \overline{p}_{1\pi(i)} + \sum_{i=j}^{n} \overline{p}_{2\pi(i)} - F^*(S_\pi^j)\}. \tag{40}$$

A job j maximizing (40) is called a *critical* in π. In consequence, the value of $Z(\pi)$ can be computed in $O(n^2 \log n)$, since the makespan of an optimal schedule can be determined in $O(n \log n)$ time by Johnson's algorithm. This running time can be improved to $O(n^2)$ by observing that the subsequent scenarios in Γ' differ only in a few processing times (see [36] for details).

We now present a branch and bound algorithm and some heuristics to solve the problem. The branch and bound algorithm consists in systematical augmenting partial schedules to capture all feasible assignments of jobs to positions in the partial schedules. Each node at level j of the search tree corresponds to the situation after assigning the first j jobs in a partial schedule. The leaves of the search tree represent complete schedules. The partial schedules, whose associated lower bounds exceed the best upper bound are fathomed. Furthermore, the partial schedules which violate certain dominance conditions are eliminated from examinations. The complete schedules, which are not eliminated by the dominance conditions and the bounding process are evaluated by a method for computing the maximum regret for the associated schedule. We now present all the elements of the branch and bound algorithm.

Computing a lower bound for the maximum regret of any partial schedule is similar in spirit to the algorithm for determining the maximum regret of a given schedule. Consider a partial schedule $\pi = (\pi(1), \ldots, \pi(l), \underbrace{\cdot, \ldots, \cdot}_{n-l})$, where jobs $\pi(1), \ldots, \pi(l)$ are fixed and the positions of the remaining $n-l$ jobs, denoted by J', are unknown. A lower bound is determined in three stages. The first stage consists of l iterations. For iteration j, $j = 1, \ldots, l$, the job placed at position j in π is assumed to be critical. The scenario S_π^j is then completely determined and we can compute the lower bound $LB = \sum_{i=1}^{j} \overline{p}_{1\pi(i)} + \overline{p}_{2\pi(j)} + \sum_{i \in J'} \overline{p}_{2i} - C^*_{\max}(S_\pi^j)$. The largest value of LB over all $j = 1, \ldots, l$ is the first stage lower bound. In the second stage, an impact of each unassigned job $j \in J'$ being the critical job while occupying the position $l+1$ is evaluated. In this case scenario S_π^j is also completely determined and we can compute a lower bound in the same way as in the first stage. The smallest value of LB obtained over all $j \in J'$ is the second stage lower bound. The third stage is similar to the second one with the exception that each unsigned job is assumed to be critical job at position n. Finally, a lower bound for the maximum regret of the partial schedule π is the best lower bound obtained in the three stages. The overall time complexity of computing this lower bound is $O(n^2)$, if we additionally make use of the observation about similarity of two consecutive scenarios (see [36] for details). The next two propositions establish some dominance properties.

Proposition 7. *If for two jobs j and h, $\overline{p}_{1j} \le \underline{p}_{1h}$ and $\underline{p}_{2j} \ge \overline{p}_{2h}$, then there exists an optimal minmax regret schedule in which j precedes h.*

Proof. Suppose, on the contrary, that in every optimal minmax regret schedule job h precedes job j. Choose one such a schedule, denote it by π, and construct schedule π' from π by swapping j and h. Let S_π and $S_{\pi'}$ be some worst case scenarios for π and π', respectively. Thus, $C_{\max}(\pi', S_{\pi'}) - C_{\max}(S_{\pi'}) > C_{\max}(\pi, S_\pi) - C_{\max}(S_\pi)$. Let k denote the position of jobs j and h in schedules π and π', respectively. Then, from Johnson's algorithm, the time required to complete the first k jobs is minimized by partitioning this set into the subset L containing the jobs with $p_{1l}(S_{\pi'}) \leq p_{2l}(S_{\pi'})$ and the subset R containing the jobs with $p_{1l}(S_{\pi'}) > p_{2l}(S_{\pi'})$ and constructing a schedule in which the jobs from L are processed first in nondecreasing order of $p_{1l}(S_{\pi'})$, and the jobs from R are processed next in nonincreasing order of $p_{2l}(S_{\pi'})$. Now $p_{1j}(S_{\pi'}) \leq p_{1h}(S_{\pi'})$ and $p_{2j}(S_{\pi'}) \geq p_{2h}(S_{\pi'})$, due to $\overline{p}_{1j} \leq \underline{p}_{1h}$ and $\underline{p}_{2j} \geq \overline{p}_{2h}$, which implies $C_{2j}(\pi, S_{\pi'}) \geq C_{2h}(\pi', S_{\pi'})$. Hence, $C_{\max}(\pi, S_{\pi'}) - C_{\max}(S_{\pi'}) \geq C_{\max}(\pi', S_{\pi'}) - C_{\max}(S_{\pi'})$, which contradicts our assumption. □

Proposition 8. *If for two jobs j and h, $\min\{\overline{p}_{1j}, \overline{p}_{2h}\} \leq \min\{\underline{p}_{1h}, \underline{p}_{2j}\}$, then an optimal minmax regret schedule can be determined without considering schedules in which job h immediately precedes job j.*

Proof. Suppose, on the contrary, that in every optimal minmax regret schedule job h immediately precedes job j. Take such a schedule π and construct schedule π' from π by swapping j and h. If S_π and $S_{\pi'}$ are worst case scenarios for π and π', respectively, then $C_{\max}(\pi', S_{\pi'}) - C_{\max}(S_{\pi'}) > C_{\max}(\pi, S_\pi) - C_{\max}(S_\pi)$. Consider schedule π and fix scenario $S_{\pi'}$. From $\min\{\overline{p}_{1j}, \overline{p}_{2h}\} \leq \min\{\underline{p}_{1h}, \underline{p}_{2j}\}$, we have $\min\{p_{1j}(S_{\pi'}), p_{2h}(S_{\pi'})\} \leq \min\{p_{1h}(S_{\pi'}), p_{2j}(S_{\pi'})\}$. Hence and from [9, Lemma 6.10], it follows that h and j can be swapped without increasing $C_{\max}(\pi, S_{\pi'})$. Thus, $C_{\max}(\pi, S_{\pi'}) \geq C_{\max}(\pi', S_{\pi'})$ and $C_{\max}(\pi, S_{\pi'}) - C_{\max}(S_{\pi'}) \geq C_{\max}(\pi', S_{\pi'}) - C_{\max}(S_{\pi'})$, which contradicts our assumption. □

We now discuss a heuristic algorithm, based on a local search, for computing a close approximation schedule with a little computation effort. It can also be used to determine a good initial upper bound in the branch and bound algorithm. The heuristic algorithm starts with an initial schedule π, computes its maximum regret and tries to improve it by performing a local search. The neighborhood of π is defined as the set of all schedules that can be reached by generating $n(n-1)$ insertions of jobs in π and $n(n-1)/2$ pairwise interchanges of jobs in π. Only those schedules are examined that do not violate the dominance conditions shown in Propositions 7 and 8 and their maximum regrets are computed. If none of the schedules in the neighborhood of π is better than π, then π is an approximation schedule for the problem and the algorithm terminates. Otherwise, a new schedule π' with the smallest maximum regret in the neighborhood of π replaces π and the process is repeated. It remains to show how the initial schedule is constructed. Let us first observe that each job j contributes at least $\min\{\overline{p}_{1j}, \overline{p}_{2j}\}$ to the makespan of an optimal minmax regret schedule. Therefore, for each job j, the value of $\min\{\overline{p}_{1j}, \overline{p}_{2j}\}$ is computed and two job sets L and R are formed in the following way: $j \in L$ if $\overline{p}_{1j} \leq \overline{p}_{2j}$, and $j \in R$ otherwise (job j such that $\overline{p}_{1j} \leq \overline{p}_{2j}$ has more chances to provide its minimum contribution to the makespan of an approximate schedule if it is placed early in the schedule). Then an initial schedule is constructed by arranging jobs $i \in L$ in the Johnson's order with respect to $\overline{p}_{1j}$ and $\underline{p}_{2j}$ followed by jobs $i \in L$ arranged in the Johnson's order with respect to $\underline{p}_{1j}$ and $\overline{p}_{2j}$.

We now discuss the computational performance of the branch and bound and the heuristic algorithms, by describing the results of experiments carried out in [36]. All the tests were performed for 810 test problems, involving $n = 9, 12, 15$ jobs. The bounds of job processing times were generated randomly from a uniform distribution on intervals with controlled variability. Each instance was solved by the branch and bound algorithm and the heuristic algorithm. Additionally, for each instance an optimal expected makespan schedule was determined. The worst case performance of optimal expected makespan schedules and the expected makespan performance of optimal minmax regret schedules were tested. The experiments illustrated that the running time of the branch and bound algorithm grown fast with the problem size n. Thus its performance may be poor for lager-sized problems. This justifies the construction of the heuristic algorithm which was very fast for every instance. The heuristic algorithm correctly returned optimal solutions in 95% of the 810 test problems, while the average and worst errors in approximating optimal solutions were 0.2% and 16.7%, respectively. Furthermore, the experiments shown the worst case performance of optimal expected makespan schedules were poor (the maximum error of 289.1%) in contrast to optimal minmax regret schedules which closely approximated optimal expected makespan schedules (the maximum error of 3.9%). Thus, optimal minmax regret schedules simultaneously hedged against worst case realization of job processing times and had good the expected makespan performance.

3.4.2. Discrete uncertainty representation

We start by showing the known results on the complexity and hardness of approximation of MINMAX (REGRET) $F2||C_{\max}$ under discrete uncertainty representation.

Theorem 23 ([28]). MINMAX (REGRET) $F2||C_{\max}$ *is strongly NP-hard for two processing time scenarios. Furthermore,* MINMAX REGRET $F2||C_{\max}$ *is not at all approximable for two processing time scenarios unless P=NP.*

Theorem 24 ([28]). *If the number of scenarios is unbounded, then* MINMAX $F2||C_{\max}$ *is strongly NP-hard not approximable within* $(4/3-\varepsilon)$ *for any* $\varepsilon > 0$*, unless P=NP.*

Theorems 23 and 24 imply that MINMAX $F2||C_{\max}$ for two scenarios does not admit an FPTAS and does not admit a PTAS if the number of scenarios is unbounded.

We now briefly describe a branch and bound algorithm and a heuristic algorithm for MINMAX REGRET $F2||C_{\max}$. They are similar in spirit to the corresponding algorithms for MINMAX REGRET $F2||C_{\max}$ with interval processing times described in Section 3.4.1. All the presented results are also valid for MINMAX $F2||C_{\max}$, because it suffices to set $C^*_{\max}(S) = 0$ for each $S \in \Gamma$. A method for determining the maximum regret of given schedule π is as follows. For each scenario $S \in \Gamma$, we compute the value of $C^*_{\max}(S)$ by using Johnson's algorithm, which requires $(Kn \log n)$ time. The maximum regret of each schedule $\pi \in \Pi$ can be then computed in $O(Kn)$ time, which is necessary to compute the makespans of π under all scenarios in Γ. Determining a lower bound on the maximum regret of any partial schedule $\pi = (\pi(1), \ldots, \pi(l), \cdot, \ldots, \cdot)$ can be done in K iterations. Namely, for each scenario $S \in \Gamma$, the $n-l$ unknown jobs are arranged in the Johnson's order with respect to $p_{ij}(S)$, and append to the partial schedule π. The resulting schedule is denoted by π'. Then $LB = C_{\max}(\pi', S) - C^*_{\max}(S)$ is a lower bound on the maximum regret of the partial schedule

$\pi = (\pi(1), \ldots, \pi(l), \cdot, \ldots, \cdot)$. Clearly, we choose the smallest lower bound computed over all K scenarios in Γ. If the optimal makespan for each $S \in \Gamma$ is determined and stored prior to the above bounding process, then computing a lower bound on the maximum regret of any partial schedule can be done in $O(Kn)$ time.

The following dominance conditions are analogous to these in Propositions 7 and 8.

Proposition 9 ([36]). *If for two jobs j and h, $p_{1j}(S) \leq p_{1h}(S)$ and $p_{2j}(S) \geq p_{2h}(S)$ for all $S \in \Gamma$, then there exists an optimal minmax regret schedule in which j precedes h.*

Proposition 10 ([36]). *If for two jobs j and h, $\min\{p_{1j}(S), p_{2h}(S)\} \leq \min\{p_{1h}(S), p_{2j}(S)\}$ for all $S \in \Gamma$, then an optimal minmax regret schedule can be determined without considering schedules in which job h immediately precedes job j.*

The heuristic algorithm consists in performing a local search for each scenario $S \in \Gamma$ and choosing a schedule with the smallest value of the maximum regret. Consider a scenario $S \in \Gamma$. The local search algorithm starts with an initial schedule π which is optimal under S. This schedule is constructed by Johnson's algorithm. The neighborhood of π is defined as the set of all schedules that can be reached by generating $n(n-1)$ insertions of jobs in π and $n(n-1)/2$ pairwise interchanges of jobs in π. If none of the schedules in the neighborhood of π has smaller maximum regret than π, then the algorithm restarts with a next scenario in Γ. Otherwise, a new schedule $\pi'$with the smallest value of the maximum regret in the neighborhood π replaces π and the process is repeated. A schedule with the smallest value of the maximum regret obtained by the local search algorithm in iterations 1 through K becomes an approximation schedule for the problem and the heuristic algorithm terminates.

The branch and bound and the heuristic algorithms were tested in [36]. All the tests were performed for 810 test problems with $n = 9, 12, 15$ jobs and $K = 4, 8, 12$ scenarios. The job processing times were randomly generated for each scenario from a uniform distribution on intervals with controlled variability. Each instance was solved by the branch and bound algorithm and the heuristic algorithm. Additionally, for each instance an optimal expected makespan schedule was computed by the branch and bound algorithm assuming that each scenario is equally probable. The performance of the algorithms and conclusions drawn from the experiments were similar to these for the branch and bound and the heuristic algorithms under the interval uncertainty representation (see Section 3.4.1). For more details, we refer the reader to [36].

We now present some approximation results for MINMAX $F2||C_{\max}$. The following observation is immediate:

Observation 1. MINMAX $Fm||C_{\max}$ *is approximable within m.*

Proof. It is clear that $OPT_1 \geq L_{\max}$. On the other hand, for any schedule π and scenario S, $C_{\max}(\pi, S) \leq m \cdot L_{\max}(S)$. Therefore, $\max_{S \in \Gamma} C_{\max}(\pi, S) \leq m \cdot L_{\max} \leq m \cdot OPT_1$. Hence a trivial algorithm that outputs any schedule yields the approximation bound of m. □

Observation 1 implies that the unbounded version of MINMAX $F2||C_{\max}$ is approximable within 2. It is interesting to compare this fact to the negative approximation result from Theorem 24. In the remaining part of this section we show how to construct a polynomial time approximation scheme (PTAS) for the bounded version of the problem, i.e. when

the number of scenarios K is assumed to be constant. Let us fix $\varepsilon \in (0,1)$ and let $\alpha > 0$ be a fixed number fulfilling the following inequality

$$\varepsilon^{\lceil 2/\varepsilon \rceil} \leq \alpha \leq \varepsilon. \tag{41}$$

The precise value of α will be specified later (in Step 2). A job $j \in J$ is *big* if $l_j^{\max} \geq \alpha L_{\max}$, *small* if $\alpha\varepsilon L_{\max} < l_j^{\max} < \alpha L_{\max}$ and *tiny* if $l_j^{\max} \leq \alpha\varepsilon L_{\max}$. Accordingly, we partition the set of jobs J into the three disjoint sets: *big jobs*, *small jobs* and *tiny jobs*, denoted by $\mathcal{B}$, $\mathcal{S}$ and $\mathcal{T}$, respectively. The PTAS is built in three steps.

Step 1. Let $UB = 2L_{\max}$ be the upper bound on OPT_1. Define $\delta = \alpha\varepsilon L_{\max}$ and let us assign the time interval $[0, UB + 2\delta|\mathcal{B}|]$ to machines 1 and 2 under each scenario $S \in \Gamma$. We will refer to the interval on i under S as $\langle M_i, S\rangle$. Each interval $\langle M_i, S\rangle$ is partitioned into $UB/\delta + 2|\mathcal{B}|$ intervals of the same length equal to δ and these intervals will be called the *intervals of the first type*. Consider a permutation π of the big jobs. We place each big job $j \in \mathcal{B}$ in each $\langle M_i, S\rangle$ so that j starts at the beginning of some interval of the first type and the order of the big jobs in all $\langle M_i, S\rangle$ is the same as in π. The resulting partial schedule will be called an *outline* and we will denote it by O. An outline is feasible if under any scenario: the jobs do not overlap on any machine, the precedence constraints between the jobs on machines 1 and 2 are not violated, and no job is completed after $UB + 2\delta|\mathcal{B}|$. A sample feasible outline is shown in Figure 4. This outline corresponds to the permutation of big jobs $\pi = (1, 2, \ldots, |\mathcal{B}|)$. Notice that many feasible outlines may correspond to the permutation π, but the number of all such outlines is finite and we can easily enumerate all of them. Now the key observation is that we can generate all the feasible outlines corresponding to all the permutations of big jobs in constant time, provided that the number of scenarios K is constant. This follows from the fact that the number of big jobs, $|\mathcal{B}|$, is bounded by $K \cdot UB/(\alpha L_{\max}) = 2K/\alpha$ and the number of intervals of the first type is bounded by $2/\alpha\varepsilon + 4K/\alpha$. Therefore, the number of all possible allocations of the big jobs to the beginning of intervals of the first type depends only on K, ε and α which are constant.

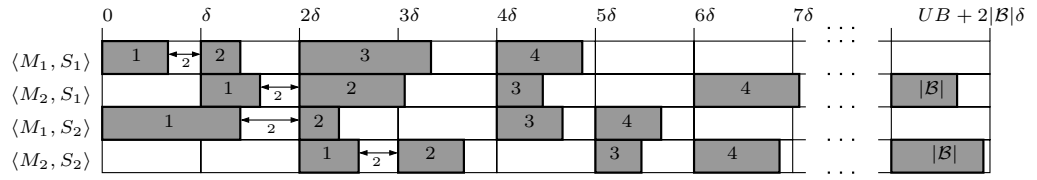

Figure 4. A sample outline for two scenarios corresponding to the sequence of big jobs $(1, 2, \ldots, |\mathcal{B}|)$. One sample interval of the second type indexed by 2 is also shown.

Step 2. Consider now the small jobs. The parameter α can be chosen so that the following inequality holds:

$$\sum_{j \in \mathcal{S}} l_j^{\max} \leq \varepsilon K L_{\max}. \tag{42}$$

The reasoning is similar to that in [23, 55]. Consider a sequence of real numbers $(\alpha_1, \alpha_2, \ldots, \alpha_{\lceil 2/\varepsilon \rceil})$, where $\alpha_k = \varepsilon^k$. Each α_k defines a set of small jobs $\mathcal{S}_k = \{j \in J \mid \alpha_k\varepsilon L_{\max} < l_j^{\max} < \alpha_k L_{\max}\}$. Clearly $\mathcal{S}_i \cap \mathcal{S}_k = \emptyset$ for $i \neq k$. If the inequality (42) would be

violated for all $\mathcal{S}_k$, $k = 1, \ldots, \lceil 2/\varepsilon \rceil$, then

$$\sum_{j\in J}\sum_{S\in\Gamma}(p_{1j}(S)+p_{2j}(S)) \geq \sum_{j\in J}\max_{S}(p_{1j}(S)+p_{2j}(S)) = \sum_{j\in J} l_j^{\max} \geq \sum_{k=1}^{\lceil 2/\varepsilon\rceil}\sum_{j\in\mathcal{S}_k} l_j^{\max} > 2KL_{\max},$$

which is a contradiction. Hence, inequality (42) must hold for at least one $\mathcal{S}_k$ and we can fix $\alpha = \alpha_k$. Inequality (42) shows that the total length of the small jobs is upper bounded by $\varepsilon KL_{\max}$ and the total contribution of them to the makespan is at most $\varepsilon KL_{\max}$. Thus, we can simply append the small jobs at the end of the schedule constructed below.

Step 3. In the third and most involved step we add to each feasible outline O all the tiny jobs from the set $\mathcal{T}$. Let us number the big jobs with respect to their positions in O, i.e. the first big job is indexed by 1 and the last one by $|\mathcal{B}|$. We denote by $\sigma_{kj}(O,S)$ and $C_{ik}(O,S)$ the starting and completion time, respectively, of the big job k on machine i under scenario S. For ease of notation, we will also define $C_{i0}(O,S) = 0$ and $\sigma_{i|\mathcal{B}|+1}(O,S) = UB + 2|\mathcal{B}|\delta$ for each S and $i = 1,2$. Making use of these starting and completion times we create in each $\langle M_i, S\rangle$ additional intervals $[C_{ik-1}(O,S), \sigma_{ik}(O,S)]$ indexed by $k = 1, \ldots, |\mathcal{B}|+1$, called the *intervals of the second type*. One sample interval of the second type, labeled by 2, is shown in Figure 4. The time contained in an interval of the second type in $\langle M_i, S\rangle$ will be used for processing tiny jobs in this interval on machine i under S. In order to determine an assignment of the tiny jobs from $\mathcal{T}$ to the intervals of the second type, corresponding to the outline O, we use a linear programming formulation. We define the following decision variables: x_{jk}, $j \in \mathcal{T}$, $k = 1, \ldots, |\mathcal{B}|+1$ and C, where $x_{jk} = f$, $0 \leq f \leq 1$, means that the same fraction f of a tiny job j is processed in the interval k of the second type on each machine under each scenario and the value of C is the maximal length of an assignment (a schedule) of the big and tiny jobs over all scenarios. The linear programming formulation is the following:

$$C_{\min}(O) = \min C \tag{43}$$

$$\sum_{k=1}^{|\mathcal{B}|+1} x_{jk} = 1 \qquad j \in \mathcal{T} \tag{44}$$

$$C_{ik-1}(O,S) + \sum_{j\in\mathcal{T}} p_{ij}(S)x_{jk} \leq \sigma_{ik}(O,S) \qquad k = 1, \ldots, |\mathcal{B}|, \quad i = 1,2, \ S \in \Gamma \tag{45}$$

$$C_{i|\mathcal{B}|}(O,S) + \sum_{j\in\mathcal{T}} p_{ij}(S)x_{j|\mathcal{B}|+1} \leq C \qquad i = 1,2, \ S \in \Gamma \tag{46}$$

$$C \leq UB + 2|\mathcal{B}|\delta \tag{47}$$

$$C, x_{jk} \geq 0 \qquad j \in \mathcal{T}, \ k = 1, \ldots, |\mathcal{B}|+1 \tag{48}$$

Constraints (44) assure that each tiny job j is fully assigned. Constraints (45) and (46) assure that the total sums of the processing times of tiny jobs assigned to the intervals of the second type do not exceed the lengths of these intervals. Constraints (46) together with the objective function minimize the maximal length, over all scenarios, of the assignment computed. Constraint (47) assures that this length is not too large, i.e. it does not exceed $UB + 2|\mathcal{B}|\delta$. We now show that the linear program (43)-(48) is feasible. Namely, we show

that there is a feasible outline O^* such that

$$C_{\min}(O^*) \leq OPT_1 + 2|\mathcal{B}|\delta. \tag{49}$$

Let π^* be an optimal minmax schedule with the value OPT_1. We transform π^* into $\tilde{\pi}^*$ using the method which is illustrated in Figure 5. Consider intervals $\langle M_1, S\rangle$ and $\langle M_2, S\rangle$ for some scenario S. In Figure 5, the first big job 1 is placed properly (i.e. at the beginning of some interval of the first type) on 1 but not properly on 2. So, we delay the starting times of all the jobs on 2 by a, so that job 1 starts processing at time δ on 2. The next big job is 2. It is not placed properly on machine 1, so we delay the starting times of 2 and all the jobs succeeding 2 on machines 1 and 2 by b. Then job 2 is still not placed properly on 2, so we delay the starting times of 2 and all the jobs succeeding 2 by c on 2. It is easy to see that delaying the starting times of these two big jobs increases the makespan under S by at most 4δ. We proceed in this way for all the subsequent big jobs and, as the result, we get the schedule $\tilde{\pi}^*$ in which all the big jobs start processing at the beginning of some intervals of the first type and the makespan under S increases by at most $2\delta|\mathcal{B}|$. We can repeat

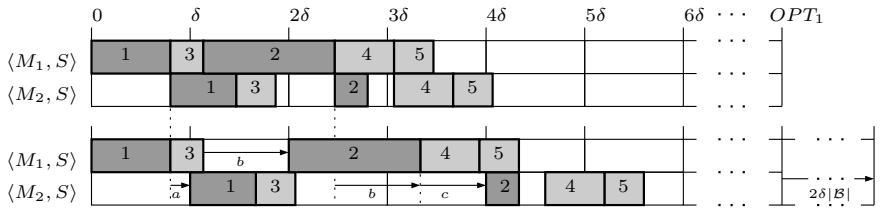

Figure 5. A sample transformation of the schedule π^* into $\tilde{\pi}^*$.

this independently for each scenario $S \in \Gamma$ and the makespan of the resulting schedule $\tilde{\pi}^*$ increases under each scenario by at most $2\delta|\mathcal{B}|$. Now, if we remove all the small and tiny jobs from $\tilde{\pi}^*$, then we obtain a feasible outline O^* with the corresponding intervals of the second type. Observe that in $\tilde{\pi}^*$ each tiny job is fully assigned to some interval of the second type with respect to O^*. Consider the set of constraints (44)-(48) built with respect to O^*. Let us fix $C = OPT_1 + 2\delta|\mathcal{B}|$. Note that $OPT_1 \leq UB = 2L_{\max}$ and $C \leq UB + 2\delta|\mathcal{B}|$. Making use of the starting times of the tiny jobs in $\tilde{\pi}^*$, we accommodate them in the intervals of the second type in O^* and set $x_{jk} = 1$ if and only if job $j \in \mathcal{T}$ is located in the k-th interval of the second type, $k = 1, \ldots, |\mathcal{B}| + 1$. It is easily seen that the binary assignment to variables x_{jk} satisfies all the constraints (44)-(48). In consequence, there is at least one feasible solution for $C = OPT_1 + 2\delta|\mathcal{B}|$, which implies (49).

In an optimal solution to the linear program (43)-(48) at most $2K(|\mathcal{B}|+1)+1$ tiny jobs receive fractional assignment. The linear program has $|\mathcal{T}| + 2K(|\mathcal{B}|+1)+1$ constraints and $|\mathcal{T}|(|\mathcal{B}|+1)+1$ variables. Thus a basic feasible solution has at most $|\mathcal{T}| + 2K(|\mathcal{B}|+1)+1$ positive variables. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the sets of tiny jobs that receive a unique assignment and a fractional assignment in this solution, respectively. Hence, and from the fact that each job has at least one positive variable associated to it, which is due to (44), we have:

$$|\mathcal{T}_1| + |\mathcal{T}_2| = |\mathcal{T}|, \;\; |\mathcal{T}_1| + 2|\mathcal{T}_2| \leq |\mathcal{T}| + 2K(|\mathcal{B}|+1) + 1.$$

Combining these inequalities yields $|\mathcal{T}_2| \leq 2K(|\mathcal{B}|+1)+1$. This implies that the tiny jobs $\mathcal{T}_2$, receiving fractional assignment, are such that:

$$\sum_{j\in\mathcal{T}_2} l_j^{\max} \leq (2K(|\mathcal{B}|+1)+1)\alpha\varepsilon L_{\max}. \tag{50}$$

We will treat the jobs from $\mathcal{T}_2$ similarly to the small jobs and append them at the end of the schedule constructed.

Now let us focus on the remaining tiny jobs $\mathcal{T}_1$ that receive an integral assignment to the intervals of the second type in an optimal solution to (43)-(48). Notice that this assignment says nothing about the order of the tiny jobs within the intervals of the second type. Hence, we must establish a permutation schedule for the tiny jobs in each interval of the second type. In order to do this we use Sevastianov's algorithm [54] for the *Compact Vector Summation* problem. We recall the following result on this problem:

Lemma 1 ([18,54])**.** *Consider a set of vectors $\boldsymbol{v}_1,\ldots,\boldsymbol{v}_n \in \mathbb{R}^d$ such that $\sum_{k=1}^n \boldsymbol{v}_k = \boldsymbol{0}$. Then it is possible to find, in polynomial time (in $O(n^2d^2)$ time), a permutation ρ of $1,\ldots,n$ such that, for all $j=1,\ldots,n$, $\left\|\sum_{k=1}^j \boldsymbol{v}_{\rho(k)}\right\|_\infty \leq d\max_{1\leq k\leq n}\|\boldsymbol{v}_k\|_\infty$.*

We now prove the following lemma that will be used to construct a permutation schedule of the tiny jobs $\mathcal{T}_1$ in each interval of the second type. Later $p_{\max}$ will be replaced by $p^{\mathcal{T}}_{\max}$, where $p^{\mathcal{T}}_{\max}$ is the largest processing time among all the processing times of the tiny jobs.

Lemma 2. *For any instance of the* MINMAX $F2||C_{\max}$ *problem, there is an $O(n^2K^2)$-time algorithm which outputs a permutation schedule $\rho\in\Pi$ such that*

$$C_{\max}(\rho,S) \leq L^S_{\max} + (K+1)p_{\max} \quad \textit{for all } S\in\Gamma. \tag{51}$$

Proof. The proof is adapted from [18, Sect. 1.5.2]. We first transform the instance of the problem so that the loads of the two machines under each scenario S are equal, i.e. $L_{\max}(S) = L_1(S) = L_2(S)$. We do this by increasing the processing times of jobs of the less loaded machine, stopping the increase of a processing time as soon as it reaches $p_{\max}$, until both machines are equally loaded. It is clear that if there is a permutation schedule ρ satisfying (51) for the modified instance, then ρ also satisfies (51) for the original one. The following equality holds for any permutation schedule π:

$$C_{\max}(\pi,S) = I_2(S) + \sum_{j=1}^n p_{2\pi(j)}(S) = I_2(S) + L_2(S) \quad \text{for all } S\in\Gamma, \tag{52}$$

where $I_2(S)$ is the total idle time on machine 2 under scenario S in π. The amount of the idle time on machine 2 under scenario S, before it starts processing job $\pi(j)$ can be determined by the following formula:

$$I_{2\pi(j)}(S) = \begin{cases} p_{1\pi(1)}(S) & \text{if } j=1, \\ I_{2\pi(j-1)}(S) & \text{if } C_{2\pi(j)}(\pi,S) = C_{2\pi(j-1)}(\pi,S) \\ & \qquad +p_{2\pi(i)}(S), \\ \sum_{k=1}^j p_{1\pi(k)}(S) - \sum_{k=1}^{j-1} p_{2\pi(k)}(S) & \text{if } C_{2\pi(j)}(\pi,S) = C_{1\pi(j)}(\pi,S) \\ & \qquad +p_{2\pi(j)}(S). \end{cases} \tag{53}$$

We can rewrite the third case of (53) as $p_{2\pi(j)}(S)+\sum_{k=1}^{j}(p_{1\pi(k)}(S)-p_{2\pi(k)}(S))$. Now, if we can find a permutation ρ such that

$$\sum_{k=1}^{j}(p_{1\rho(k)}(S)-p_{2\rho(k)}(S))\leq Kp_{\max},\ j=1,\ldots,n,\ \text{for all } S\in\Gamma \tag{54}$$

then we conclude from (53) that $I_{2\rho(j)}(S)\leq\max\{I_{2\rho(i-1)}(S),(K+1)p_{\max}\}$ and, since $I_{2\rho(1)}(S)\leq p_{\max}$, $I_2(S)=I_{2\rho(n)}(S)\leq(K+1)p_{\max}$. Because $\sum_{j=1}^{n}p_{2\rho(j)}(S)=L_2(S)=L_{\max}(S)$, equality (52) implies (51).

It remains to show that a permutation ρ fulfilling (54) can be constructed in $O(n^2K^2)$ time. In order to do this we apply Lemma 2. Let us define a set of K-dimensional vectors $\boldsymbol{v}_1,\ldots,\boldsymbol{v}_n$, where $\boldsymbol{v}_k=(p_{1k}(S_1)-p_{2k}(S_1),p_{1k}(S_2)-p_{2k}(S_2),\ldots,p_{1k}(S_K)-p_{2k}(S_K))$, $k=1,\ldots,n$. This set satisfies the assumptions of Lemma 1. Indeed, $\sum_{k=1}^{n}\boldsymbol{v}_k=(L_1(S_1)-L_2(S_1),L_1(S_2)-L_2(S_2),\ldots,L_1(S_K)-L_2(S_K))=\boldsymbol{0}$ by equality $L_{\max}(S)=L_1(S)=L_2(S)$. Furthermore $\|\boldsymbol{v}_k\|_\infty\leq p_{\max}$ for all k. Lemma 1 now shows that the required permutation ρ can be determined in $O(n^2K^2)$ time. □

Consider any interval of the second type, say the k-th. There is a subset of the tiny jobs $\mathcal{T}_1$ assigned integrally to k by the linear program. We construct a permutation schedule for these tiny jobs using the algorithm from Lemma 2. This algorithm produces a permutation schedule with the length under each scenario S at most $\max\{\sigma_{1k}(O,S)-C_{1k-1}(O,S),\sigma_{2k}(O,S)-C_{2k-1}(O,S)\}+(K+1)p^{\mathcal{T}}_{\max}$, see (51). Hence, in order to accommodate the tiny jobs according to the computed permutation schedule to the intervals $[C_{1k-1}(O,S),\sigma_{1k}(O,S)]$ and $[C_{2k-1}(O,S),\sigma_{2k}(O,S)]$, $C_{1k-1}(O,S)\leq C_{2k-1}(O,S)$, $\sigma_{1k}(O,S)\leq\sigma_{2k}(O,S)$, for all S without violating its feasibility, it suffices to increase the length of the k-th interval of the second type on machine 2 under each scenario S by at most $(K+1)p^{\mathcal{T}}_{\max}$, i.e. we shift the big job k and all the jobs starting after this big job to the right on machine 2 under each scenario S by at most $(K+1)p^{\mathcal{T}}_{\max}$. We can apply the algorithm from Lemma 2 to each interval of the second type and the total increase of the length of the intervals of the second type is at most $(|\mathcal{B}|+1)(K+1)p^{\mathcal{T}}_{\max}$.

We are now ready to provide our PTAS which works as follows. First, we generate all the feasible outlines. There must be a feasible outline O^* which satisfies inequality (49). The linear program applied to O^* gives us the partition of the set of the tiny jobs into $\mathcal{T}_1$ and $\mathcal{T}_2$. Then, we apply the algorithm from Lemma 2 to schedule the jobs from $\mathcal{T}_1$ in each interval of the second type. Finally, we create any permutation schedule for $\mathcal{T}_2\cup\mathcal{S}$ and append it to the end of the schedule constructed. As the result we get a permutation schedule $\hat{\pi}\in\Pi$ for all the jobs in J with the maximum makespan over all scenarios that can be upper bounded as follows:

$$\max_{S\in\Gamma}C_{\max}(\hat{\pi},S)\leq OPT_1+2\delta|\mathcal{B}|+\sum_{j\in\mathcal{T}_2}l_j^{\max}+\sum_{j\in\mathcal{S}}l_j^{\max}+(|\mathcal{B}|+1)(K+1)p^{\mathcal{T}}_{\max},$$

which is due to (49) and the construction described this makespan. Now using (41), (42), (50) and the conditions: $\delta=\alpha\varepsilon L_{\max}$, $|\mathcal{B}|\leq 2K/\alpha$, $OPT_1\geq L_{\max}$ and $p^{\mathcal{T}}_{\max}\leq\alpha\varepsilon L_{\max}$, we get after easy computations:

$$\max_{S\in\Gamma}C_{\max}(\hat{\pi},S)\leq(1+O(K^2)\varepsilon)OPT_1.$$

For the constant values of K and ε, the above algorithm is polynomial with respect to the number of jobs n. We have thus proved the following theorem.

Theorem 25. *If the number of processing time scenarios K is constant, then* MINMAX $F2||C_{\max}$ *admits a PTAS.*

The same technique yields to a PTAS for the bounded version of MINMAX $Fm||C_{\max}$, when the number of machines m is a fixed constant.

Theorem 26. *If the number of processing time scenarios K and the number of machines are constant, then* MINMAX $Fm||C_{\max}$ *admits a PTAS.*

3.4.3. Notes and references

The two machine permutation flow shop problem under discrete uncertainty representation was first investigated by Kouvelis et al. in [36], where it was proven that its minmax regret version with only two scenarios is weakly NP-hard. This complexity result was strengthened by Kasperski et al. in [28], where the proofs of Theorems 23, and 24 can be found. The PTAS for the bounded version of the minmax problem, shown in Section 3.4.2 was also constructed in [28] and its idea is partially based on the previous works of Jansen et al. [23], Potts [48] and Hall [19] on the deterministic job shop problem. The branch and bound and heuristic algorithms, described in Section 3.4.2, were proposed by Kouvelis et al. [36]. Also the dominance properties shown in Propositions 9 and 10 were first established in [36]. The minmax regret version of the problem with two machines and interval job processing times was first discussed by Kouvelis et al. [36] and by Averbakh [7]. However, the computational complexity of this problem is still open. In [36] the characterization of the worst case scenario (Theorem 22) and the dominance properties (Proposition 7 and Proposition 8) were first proven. The $O(m^2)$ and $O(m)$ time algorithms for the problem with 2 jobs, shown in Section 3.4.1, are due to Averbakh [7]. The branch and bound algorithm and a local search based heuristic for MINMAX REGRET $F2||C_{\max}$ with interval processing times were proposed by Kouvelis et al. [36].

3.5. Parallel Machine Scheduling Problem

In this section, we discuss the minmax (regret) versions of the identical parallel machine scheduling problem denoted by $P||C_{\max}$. In this problem, the jobs in J are to be scheduled on m identical machines from the set $M = \{1,\ldots,m\}$. Each job $j \in J$ has a processing time p_j. A schedule π is an assignment of the jobs to the machines so that each job is assigned to exactly one machine. Notice that π is not a permutation of the jobs but we prefer to use the same notation for schedules as for the problems discussed in the previous sections. In the deterministic case, the objective is to compute a schedule minimizing the makespan, i.e. the time in which all the jobs in π are completed. The deterministic $P2||C_{\max}$ problem is weakly NP-hard [41] and becomes strongly NP-hard when the number of machines is unbounded [14]. However, when the number of machines is fixed, then the problem admits an FPTAS [50], and when the number of machines is unbounded a PTAS for the problem can be constructed [3, 21].

Let $L_i(\pi,S)$ be the *load* of machine $i \in M$ under S, i.e. the sum of the completion times of all jobs processed on i in π. Let $C_{\max}(\pi,S) = \max_{i\in M} L_i(\pi,S)$ be the makespan of schedule π under scenario S. The maximum cost and the maximum regret of π are then expressed as follows:

$$F(\pi) = \max_{S\in\Gamma} C_{\max}(\pi,S), \;\; Z(\pi) = \max_{S\in\Gamma}\{C_{\max}(\pi,S) - C^*_{\max}(S)\}, \tag{55}$$

where $C^*_{\max}(S)$ is the minimum makespan under S.

In this section, we will describe the results only for the discrete uncertainty representation. Some attempts to solve the minmax regret problem with interval processing times will be described in Notes and References. Notice that under the interval uncertainty representation the computation of $Z(\pi)$ for a given schedule π is NP-hard, since even the simplest deterministic counterpart of the problem is NP-hard. Furthermore, according to the general remark presented in Section 3, MINMAX REGRET $P2||C_{\max}$ with interval processing times is not at all approximable unless P=NP.

3.5.1. Discrete uncertainty representation

It turns out that MINMAX $P||C_{\max}$ is equivalent to the following VECTOR SCHEDULING problem discussed in [10]: we are given a set V of d-dimensional vectors $a_1,\ldots a_n$ from $[0,\infty)^d$ and a number m. A valid solution is a partition of V into m sets $A_1,\ldots,A_m$. The objective is to minimize $\max_{i\in\{1,\ldots,m\}} ||\overline{A}_i||_\infty$, where $\overline{A}_i$ is the sum of the vectors in A_i. To see that this problem is equivalent to MINMAX $P||C_{\max}$ it is enough to observe that we can treat each vector a_j as a job whose processing time under k-th scenario is the k-th component of a_j. A valid partition of the jobs (vectors) corresponds to a feasible schedule. The following theorem provides a hardness approximation result for the problem under consideration:

Theorem 27 ([10])**.** *If the number of scenarios and machines are unbounded, then no polynomial time algorithm exists that approximates* MINMAX $P||C_{\max}$ *within* γ*, for any constant* $\gamma \geq 1$*, unless P=NP.*

The next theorem establishes the complexity of the minmax problem when the number of machines equals 2 and the number of processing time scenarios is unbounded:

Theorem 28 ([29])**.** *If the number of processing time scenarios is unbounded, then* MINMAX $P2||C_{\max}$ *is strongly NP-hard and not approximable within* $(3/2-\varepsilon)$ *for any* $\varepsilon > 0$*, unless P=NP.*

From Theorem 28, It follows that MINMAX $P2||C_{\max}$ does not admit a PTAS for the unbounded number of scenarios. The general remark shown in Section 3 implies that MINMAX REGRET $P2||C_{\max}$ is not at all approximable even in the deterministic case (when $K = 1$).

It is not difficult to construct a mixed integer programming formulation for MINMAX (REGRET) $P||C_{\max}$. Let us introduce binary variables $x_{ij} \in \{0,1\}$ such that $x_{ij} = 1$ if job j

is assigned to machine i in the schedule constructed. Thus, the MIP formulation for MINMAX $P||C_{\max}$ is as follows:

$$\begin{array}{ll} \min & t \\ & \sum_{j=1}^{n} p_j(S)x_{ij} \leq t \quad i \in M, S \in \Gamma \\ & \sum_{i=1}^{m} x_{ij} = 1 \qquad\quad j = 1,\ldots n \\ & x_{ij} \in \{0,1\} \qquad\quad i = 1,\ldots,n, j = 1,\ldots,m \end{array} \tag{56}$$

The corresponding MIP formulation for the minmax regret version can be obtained by subtracting $C^*_{\max}(S)$ from the left hand side of the first constraint in (56). Notice, however that computing $C^*_{\max}(S)$ for a fixed S is NP-hard.

We now show an alternative exact algorithm for the minmax version of the problem, based on dynamic programming approach. Let us define a load vector $L = (L_{11},\ldots,L_{m1},L_{12},\ldots,L_{m2},\ldots,L_{1K},\ldots,L_{mK})$, where L_{ik} is the load of machine i under scenario S_k for some partial schedule, $k = 1,\ldots,K$. Let $L_{\max} = \max_{S\in\Gamma}\sum_{j\in J} p_j(S)$ and let $p_{\max} = \max_{j\in J, S\in\Gamma} p_j(S)$. Clearly $L_{ik} \leq L_{\max}$ for all $i = 1,\ldots,m$ and $k = 1,\ldots,K$. Therefore, the number of distinct load vectors is bounded by $l = (L_{\max})^{mK}$. We can now use a simple dynamic algorithm to compute an optimal schedule. An idea of that algorithm is illustrated in Figure 6.

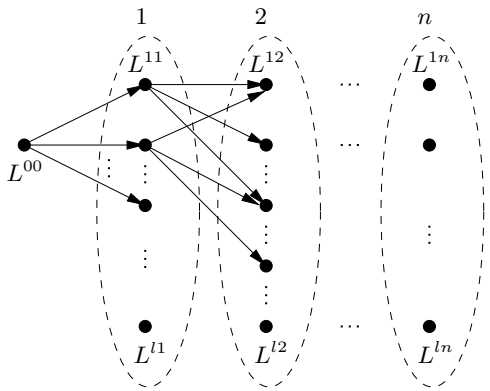

Figure 6. An illustration of the exact algorithm.

We first build a directed acyclic network composed of node $L^{00} = (0,\ldots,0)$ and n layers, where the j-th layer is composed of nodes $L^{1j},\ldots,L^{lj}$. The arc between L^{uj-1} and L^{vj} exists if and only if the load vector L^{vj} can be obtained from L^{uj-1} by placing the job j on some machine. This network can be built in $O(nml) = O(nm(L_{\max})^{mK})$ time, since it contains $nl+1$ nodes and each node has at most m outgoing arcs. The last, n-th, layer contains the load vectors of at most l complete schedules and the vector corresponding to schedule π with the minimum value of $\max_{i,k} L_{ik} = \max_{S\in\Gamma} C_{\max}(\pi,S)$ is optimal. This optimal schedule can be obtained by using simple backward computations in the network constructed. The overall running time of the algorithm is $O(nm(L_{\max})^{mK}) = O(nm(np_{\max})^{mK}) = O(n^{mK+1}m(p_{\max})^{mK})$. We thus get the following result:

Theorem 29. *When both the number of scenarios and the number of machines are constant, the* MINMAX $Pm||C_{\max}$ *is solvable in pseudopolynomial time* $O(n^{mK+1}m(p_{\max})^{mK})$.

Theorem 30. *Given any fixed* $\varepsilon \in (0,1)$*, there is an* $(1+\varepsilon)$*-approximation algorithm for* MINMAX $Pm||C_{\max}$ *which runs in* $O(n^{2mK+1}m(1/\varepsilon)^{mK})$ *time. Hence,* MINMAX $Pm||C_{\max}$ *with constant m and K admits an FPTAS.*

Proof. Let us fix $\varepsilon \in (0,1)$. For each scenario $S \in \Gamma$ define a *scaled scenario* $\widehat{S}$ under which $p_j(\widehat{S}) = \left\lfloor \frac{np_j(S)}{\varepsilon p_{\max}} \right\rfloor$, $j \in J$. We will denote by $\widehat{\Gamma}$ the set of scaled scenarios. If L_{ik} is the load of machine i under S_k for some schedule π, then $\widehat{L}_{ik}$ is the load of machine i under $\widehat{S}_k$ for π. Similarly, $\max_{\widehat{S}\in\widehat{\Gamma}} C_{\max}(\pi,\widehat{S})$ is the maximum makespan of over the set $\widehat{\Gamma}$. It holds: $\frac{\varepsilon p_{\max}}{n} p_j(\widehat{S}) \le p_j(S) \le \frac{\varepsilon p_{\max}}{n}(p_j(\widehat{S})+1)$. Thus, $\frac{\varepsilon p_{\max}}{n}\widehat{L}_{ik} \le L_{ik} \le \frac{\varepsilon p_{\max}}{n}\widehat{L}_{ik} + \varepsilon p_{\max}$, for each $i = 1,\ldots,m$ and $k = 1,\ldots,K$, which implies that for any schedule π:

$$\frac{\varepsilon p_{\max}}{n} \max_{\widehat{S}\in\widehat{\Gamma}} C_{\max}(\pi,\widehat{S}) \le \max_{S\in\Gamma} C_{\max}(\pi,S) \le \frac{\varepsilon p_{\max}}{n} \max_{\widehat{S}\in\widehat{\Gamma}} C_{\max}(\pi,\widehat{S}) + \varepsilon p_{\max}. \tag{57}$$

If $\widehat{\pi}$ is an optimal schedule for $\widehat{\Gamma}$ and π^* is an optimal schedule for the original scenario set Γ, then $\max_{\widehat{S}\in\widehat{\Gamma}} C_{\max}(\widehat{\pi},\widehat{S}) \le \max_{\widehat{S}\in\widehat{\Gamma}} C_{\max}(\pi^*,\widehat{S})$, $OPT_1 \ge p_{\max}$ and inequalities (57) imply:

$$\begin{aligned}\max_{S\in\Gamma} C_{\max}(\widehat{\pi},S) &\le \frac{\varepsilon p_{\max}}{n} \max_{\widehat{S}\in\widehat{\Gamma}} C_{\max}(\widehat{\pi},\widehat{S}) + \varepsilon p_{\max} \le \frac{\varepsilon p_{\max}}{n} \max_{\widehat{S}\in\widehat{\Gamma}} C_{\max}(\pi^*,\widehat{S}) + \varepsilon p_{\max} \\ &\le OPT_1 + \varepsilon p_{\max} \le (1+\varepsilon)OPT_1.\end{aligned}$$

Hence the cost of $\widehat{\pi}$ is within $(1+\varepsilon)OPT_1$. We can find $\widehat{\pi}$ by applying the exact pseudopolynomial time algorithm for the scaled scenarios (see Theorem 29). Since the largest scaled processing time is not greater than n/ε, the schedule $\widehat{\pi}$ can be computed in $O(n^{2mK+1}m(1/\varepsilon)^{mK})$ time. □

If the number of scenarios K is constant and the number of machines m is unbounded, then MINMAX $P||C_{\max}$ admits a PTAS.

Theorem 31 ([10])**.** *Given any fixed* $\varepsilon \in (0,1)$*, there is an* $(1+\varepsilon)$*-approximation algorithm for* MINMAX $P||C_{\max}$ *which runs in* $(nK/\varepsilon)^{O(s)}$ *time, where* $s = O((\ln(K/\varepsilon)/\varepsilon)^K)$*. Hence,* MINMAX $P||C_{\max}$ *with constant K admits a PTAS.*

We now consider the case when both the number of machines and the number of scenarios are unbounded. We show several approximation algorithms for that case. The first algorithm is trivial and returns any schedule. The following proposition provides its approximation ratio:

Proposition 11. *The unbounded version of* MINMAX $P||C_{\max}$ *is approximable within m.*

Proof. Observe that for any schedule $\pi \in \Pi$ under each scenario S, it holds $C_{\max}(\pi,S) \le \sum_{j\in J} p_j(S)$ and $C_{\max}(\pi^*,S) \ge \frac{1}{m}\sum_{j\in J} p_j(S)$, where π^* is an optimal minmax schedule. This implies $C_{\max}(\pi,S) \le m \cdot C_{\max}(\pi^*,S)$ for all $S \in \Gamma$. □

The idea of the second algorithm is to construct an artificial scenario $\hat{S}$ under which $p_j(\hat{S}) = \sum_{S\in\Gamma} p_j(S)$, $j \in J$, and apply the classical list scheduling algorithm under $\hat{S}$.

Theorem 32. *The list scheduling based algorithm returns a* $(K+1)$*-approximate schedule for the unbounded version of* MINMAX $P||C_{\max}$.

Proof. Let $\hat{\pi}$ be the schedule returned by the list scheduling algorithm. A standard analysis of the list scheduling algorithm yields: $C_{\max}(\hat{\pi},\hat{S}) \leq \sum_{j\in J,S\in\Gamma} p_j(S)/m + p_{\max}$. It holds $OPT_1 \geq \sum_{j\in J} p_j(S)/m$ for all $S\in\Gamma$, $OPT_1 \geq p_{\max}$ and the theorem follows. □

In order to give the next two approximation algorithms, we assume, without loss of generality, that job processing times have been scaled so that $OPT_1 = 1$. This yields the following forms of the lower bounds on OPT_1:

$$p_{\max} = \max_{j\in J,S\in\Gamma} p_j(S) \leq 1, \quad \frac{\sum_{j\in J}\sum_{S\in\Gamma} p_j(S)}{mK} \leq 1. \tag{58}$$

The second algorithm is very simple. It assigns each job to exactly one machine chosen uniformly at random. The random mechanism in the algorithm can be realized by casting for each job $j\in J$ a symmetric m-faced dice whose face probabilities are $1/m$. Since each job is assigned to exactly one machine, the randomized algorithm returns a feasible schedule. Before, we give an approximation bound for a schedule constructed by the randomized algorithm, we need a version of the standard Chernoff bound.

Proposition 12 ([10])**.** *Let $X_1,\ldots,X_n$, be independent binary random variables and let $X = \sum_{i=1}^{n} t_i X_i$. Let $T = \max_i t_i$ and $\mu = \mathbf{E}[X]$. Then for any sufficiently large h, $\Pr[X > (1+c\ln h/\ln\ln h)(\mu+T)] \leq h^{-c/2}$.*

Theorem 33. *The randomized algorithm returns an $O(\ln(Km)/\ln\ln(Km))$-approximate schedule with high probability.*

Proof. Let us fix a machine $i^*\in M$. Let X_j, $j\in J$, be independent binary random variables such $\Pr[X_j = 1] = 1/m$ and $X_j = 1$ if job j is assigned to machine i^*. Define $Y(S) = \sum_{j\in J} p_j(S)X_j$. Now $\mathbf{E}[Y(S)] = \sum_{j\in J} p_j(S)/m \leq 1$ and $\max_j p_j(S) \leq 1$, where the bounds follow from (58). Proposition 12 now yields $\Pr[Y(S) > 2(1+c\ln(Km)/\ln\ln(Km))] \leq (Km)^{-c/2}$. In general, if $\xi_i(S)$ is the event that the load of machine i under scenario S is greater than $2(1+c\ln(Km)/\ln\ln(Km))$ then $\Pr[\xi_i(S)] \leq (Km)^{-c/2}$. By the union bound, $\Pr[\xi = \cup_{i\in M}\cup_{S\in\Gamma}\xi_i(S)] \leq Km(Km)^{-c/2}$. By choosing c sufficiently large, one can ensure that $\Pr[\xi]$ is less than the inverse of a polynomial factor. Hence, for a schedule $\hat{\pi}$ returned by the randomized algorithm the equality $\max_{S\in\Gamma} C_{\max}(S,\hat{\pi}) \leq O(\ln(Km)/\ln\ln(Km))$ holds with high probability. □

We now present a deterministic $O(\ln^2 K)$- approximation algorithm for the problem. This algorithm will call a known approximation algorithm for the *packing integer program* (PIP) problem as a subroutine. The special case of PIP, which we will use can be stated follows: given a set of jobs J with $p_j(S)\in[0,1]$, $j\in J$, $S\in\Gamma$, the *largest volume packing problem* is the problem of finding a subset $J^*\subset J$ such that $\max_{S\in\Gamma}\sum_{j\in J^*} p_j(S) \leq 1$ and $\sum_{j\in J^*}\sum_{S\in\Gamma} p_j(S)$ is maximized. Let $V_{\max}$ denote the value of an optimal solution. A subset $\hat{J}$ of jobs in J is (α,β)-*approximation* for the largest volume packing problem if it satisfies the conditions: $\max_{S\in\Gamma}\sum_{j\in\hat{J}} p_j(S) \leq \alpha$ and $\sum_{j\in\hat{J}}\sum_{S\in\Gamma} p_j(S) \geq \beta V_{\max}$, $\alpha,\beta\leq 1$. Consider an algorithm shown in the form of Algorithm 2.

We now give an approximation bound for a schedule constructed by Algorithm 2.

Algorithm 2 $O(\ln^2 K)$- approximation algorithm for MINMAX $P||C_{\max}$

for $l \leftarrow 1$ to t **do**
  {repeat for t packing stages:}
  for $i \leftarrow 1$ to m **do**
    Find an (α,β)-approximate solution $\hat{J}$ to the largest volume packing problem with the current set of jobs - call an algorithm for PIP as a subroutine.
    Assign the jobs in $\hat{J}$ to machine l and remove them from the current set of jobs.
  end for
end for
Find a partial schedule for the remaining jobs using the list scheduling algorithm.
Combine the two partial schedules to obtain a complete schedule $\hat{\pi} \in \Pi$.
return $\hat{\pi}$.

Lemma 3. *Algorithm 2 yields a schedule* $\hat{\pi} \in \Pi$ *such that* $\max_{S\in\Gamma} C_{\max}(\hat{\pi},S) \leq t\cdot\alpha + K/e^{t\cdot\beta} + 1$.

Proof. Let us focus on the machine whose load in $\hat{\pi}$ is equal to $\max_{S\in\Gamma} C_{\max}(\hat{\pi},S)$. Let J_1 and J_2 denote the sets of jobs assigned to the machine in the packing stages and by the list scheduling based algorithm, respectively. Since J_1 is composed of (α,β)-approximations to the largest volume packing problems, $\max_{S\in\Gamma}\sum_{j\in J_1} p_j(S) \leq t\cdot\alpha$. Let J' be the set of jobs that remains after t packing stages scheduled by the list scheduling based algorithm. The set J' fulfills the equality (see [10, Corllary 2.10]):

$$\sum_{j\in J'}\sum_{S\in\Gamma} p_j(S) \leq (\sum_{j\in J}\sum_{S\in\Gamma} p_j(S))/e^{t\cdot\beta}. \tag{59}$$

An increase in the load of the machine while assigning jobs from J_2 by the list scheduling based algorithm to the considered machine, $\max_{S\in\Gamma}\sum_{j\in J_2} p_j(S)$, is upper bounded by:

$$\sum_{j\in J'}\sum_{S\in\Gamma} p_j(S)/m + p_{\max} \leq \frac{K}{e^{t\cdot\beta}}(\sum_{j\in J}\sum_{S\in\Gamma} p_j(S))/(Km) + 1 \leq \frac{K}{e^{t\cdot\beta}} + 1.$$

The above inequalities follow from a standard analysis of the list scheduling algorithm and (58) and (59). Combining two upper bounds on $\max_{S\in\Gamma}\sum_{j\in J_1} p_j(S)$ and $\max_{S\in\Gamma}\sum_{j\in J_2} p_j(S)$ gives $t\cdot\alpha + K/e^{t\cdot\beta} + 1$. □

The number of packing stages t can be chosen as a function of α and β to ensure the best approximation guarantee of Algorithm 2. If an $(O(\ln K), 1/2)$-approximation algorithm, proposed in [58], is invoked in the packing stages for solving the largest volume packing problems, then Algorithm 2 becomes an $O(\ln^2 K)$-approximation algorithm. We have thus proved:

Theorem 34. *There is an* $O(\ln^2 K)$*-approximation algorithm for the unbounded version of* MINMAX $P||C_{\max}$.

Using Proposition 11, we can conclude that the unbounded version of MINMAX $P||C_{\max}$ is approximable within $\min\{m,\gamma\}$, where γ is one of the approximation ratios given in Theorems 32, 33 or 34. Thus, in particular, it is approximable within 2 for two machines.

There is still and open gap, because we know (see Theorem 28) that the two-machine case is not approximable within $3/2-\varepsilon$ for any $\varepsilon > 0$.

3.5.2. Notes and references

The MINMAX $P||C_{\max}$ problem under discrete uncertainty representation is equivalent to the VECTOR SCHEDULING problem discussed by Chekuri and Khanna in [10]. Hence some results obtained in [10] such as Theorems 27, 31, and 34, can be directly applied to MINMAX $P||C_{\max}$. Theorem 27 was originally stated for the weaker hypotheses P≠ZPP. However, if we use the results presented in [62], then the hardness approximation result in Theorem 27 can be stated assuming only P≠NP. The case with 2 machines was discussed by Kasperski et al. in [29], where the proof of Theorem 28 can be found. Also the pseudopolynomial algorithm and the FPTAS shown in Theorems 29, and 30 were presented in [29]. Perhaps the most interesting open problem is to close the approximability gap for two machines, where a trivial algorithm gives a 2-approximate solution and the problem is hard to approximate within a ratio smaller than $3/2$.

The MINMAX REGRET $P2||C_{\max}$ problem with interval processing times was discussed by Siepak and Józefczyk in [56]. Since this problem is not at all approximable and the computation of the maximum regret of a given schedule is NP-hard, only some heuristic algorithms should be taken into account. In [56] some attempts were made to find such heuristic solutions. The following heuristic was proposed: the scenario S in which the job processing times are the midpoints of their corresponding time intervals is first determined; then the classical $(4/3-1/3m)$-approximation algorithm based on list scheduling according to LPT rule [16] with determined S is applied to find an approximation schedule. The second algorithm is based on the scatter search metaheuristic. To check the computational performance of the heuristic algorithms some computational experiments were carried out. For details we refer the reader to [56].

4. Conclusion

The aim of this chapter was to present the state of the art in the field of minmax (regret) scheduling problems. In fact, we described only several very basic scheduling models in which a solution (schedule) has a simple structure. Namely, it is a permutation of jobs or an assignment of jobs to machines. There are a lot of other scheduling models whose minmax (regret) versions have never been explored. Most of the deterministic counterparts of the problems discussed in this chapter are polynomially solvable. Unfortunately, their minmax (regret) versions (with a few exceptions) become NP-hard and often hard to approximate. An analysis of minmax (regret) scheduling problems is very challenging. There are only a few general properties valid for all the problems and each problem has its own structure which need to be explored. For each problem some exact and approximation algorithms should be constructed. A promising method of solving large scale problems is local search.

Among the two uncertainty representations discussed in this chapter, the discrete uncertainty representation seems to be easier to analyze. In most cases it is not difficult to present a reduction which shows the hardness of a problem. It is also often possible to construct a mixed integer programming formulation which allows us to obtain optimal

solutions for smaller instances. Furthermore, computing the maximum and maximum regret criteria for a given schedule reduces to solving a small number of the deterministic problems. On the other hand, in the interval uncertainty representation the computation of the maximum regret may be a nontrivial task. For some problems, such as MINMAX REGRET $1|prec|\max w_jT_j$, no method of computing the maximum regret is known. Also, computing the maximum regret of a given schedule may be much more time consuming than solving a deterministic problem. This is the case for MINMAX REGRET $1||\sum C_j$ with interval processing times, where computing the value of $Z(\pi)$ requires $O(n^3)$ time while solving the deterministic problem requires only $O(n\log n)$ time. There are several interesting open problems for the interval uncertainty representation. Perhaps the most interesting one is to characterize the complexity of MINMAX REGRET $F2||C_{\max}$ with interval processing times, which is open to date.

The minmax (regret) framework presented in this chapter can be extended. For example, some other methods of defining scenario set, which better reflect various types of uncertainty, can be considered. However, the complexity results described in this chapter may be useful when more complex scenario set are analyzed. For the discrete uncertainty representation, the maximum criterion can be generalized by using the Ordered Weighted Averaging aggregation operator (shortly OWA) proposed by Yager [60]. The minmax approach is often regarded as too conservative (see [42] for some examples where applying the minmax criterion is controversial). Using the OWA operator one can take all scenarios into account while computing a solution and also take into account the attitude of decision makers towards the risk. Some attempts in this area have been recently made in [29]. In many cases it is possible to provide an additional information with scenario set. Even if the probability distribution is not available, decision makers often feel, or have some knowledge, which scenarios are more likely to occur. In this case the possibilistic approach seems to be appropriate (see [35]). The criteria used in the possibilistic approach generalize the minmax and minmax regret criteria and allows us to take more information into account while computing a solution.

References

[1] Hassene Aissi, Mohamed Ali Aloulou, and Mikhail Y. Kovalyov, *Minimizing the number of late jobs on a single machine under due date uncertainty*, Journal of Scheduling **14** (2011), 351–360.

[2] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten, *Min–max and min–max regret versions of combinatorial optimization problems: A survey*, European Journal of Operational Research **197** (2009), 427–438.

[3] Noga Alon, Yossi Azar, Gerhard J. Woeginger, and Tal Yadid, *Approximation Schemes for Scheduling on Parallel Machines*, Journal of Scheduling **1** (1998), 55–66.

[4] Mohamed Ali Aloulou and Federico Della Croce, *Complexity of single machine scheduling problems under scenario-based uncertainty*, Operations Research Letters **36** (2008), 338–342.

[5] Igor Averbakh, *Minmax regret solutions for minimax optimization problems with uncertainty*, Operations Research Letters **27** (2000), 57–65.

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

[7] ———, *The minmax regret permutation flow-shop problem with two jobs*, European Journal of Operational Research **169** (2006), 761–766.

[8] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski, *Robust optimization*, Princeton Series in Applied Mathematics, Princeton University Press, Princeton, NJ, 2009.

[9] Peter Brucker, *Scheduling algorithms*, 5th ed., Springer Verlag, Heidelberg, 2007.

[10] Chandra Chekuri and Sanjeev Khanna, *On Multi-dimensional Packing Problems*, SIAM Journal on Computing **33** (2004), 837–851.

[11] Eduardo Conde, *An improved algorithm for selecting p items with uncertain returns according to the minmax-regret criterion*, Mathematical Programming **100** (2004), no. 2, 345–353.

[12] Thomas Cormen, Charles Leiserson, and Ronald Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[13] Richard L. Daniels and Panos Kouvelis, *Robust scheduling to hedge against processing time uncertainty in single-stage production*, Management Science **41** (1995), 363–376.

[14] Michael R. Garey and David S. Johnson, *"Strong" NP-Completeness Results: Motivation, Examples, and Implications*, Journal of the ACM **25** (1978), 499–508.

[15] Michael R. Garey, David S. Johnson, and Ravi Sethi, *The complexity of flowshop and jobshop scheduling*, Mathematics of Operations Research **1** (1976), 117–129.

[16] Ronald L. Graham, *Bounds on Multiprocessing Timing Anomalies*, SIAM Journal of Applied Mathematics **17** (1969), 416–429.

[17] Ronald L. Graham, Eugene L. Lawler, Jan Karel Lenstra, and Alexander H. G. Rinnooy Kan, *Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey*, Annals of Discrete Mathematics **5** (1979), 169–231.

[18] Leslie A. Hall, *Approximation Algorithms for Scheduling*, Approximation Algorithms for NP-Hard Problems (Dorit Hochbaum, ed.), PWS, 1995, pp. 1–43.

[19] ———, *Approximability of flow shop scheduling*, Mathematical Programming **82** (1998), 175–190.

[20] Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein, *Scheduling to minimize average completion time: off-line and on-line approximation problems*, Mathematics of Operations Research **22** (1997), 513–544.

[21] Dorit S. Hochbaum and David B. Shmoys, *Using dual approximation algorithms for scheduling problems: theoretical and practical results*, Journal of the ACM **34** (1987), 144–162.

[22] IBM ILOG, *CPLEX Optimizer*, http://www.ibm.com/software/commerce/optimization/cplex-optimizer/.

[23] Klaus Jansen, Roberto Solis-Oba, and Maxim Sviridenko, *Makespan Minimization in Job Shops: A Linear Time Approximation Scheme*, SIAM Journal on Discrete Mathematics **16** (2003), 288–300.

[24] Selmer Martin Johnson, *Optimal Two- and Three-Stage Production with Setup Times Included*, Naval Research Logistic Quarterly **1** (1954), 61–68.

[25] Richard M. Karp, *Reducibility Among Combinatorial Problems*, Complexity of Computer Computations, 1972, pp. 85–103.

[26] Adam Kasperski, *Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion*, Operations Research Letters **33** (2005), no. 4, 431–436.

[27] ———, *Discrete optimization with interval data - minmax regret and fuzzy approach*, Studies in Fuzziness and Soft Computing, vol. 228, Springer, 2008.

[28] Adam Kasperski, Adam Kurpisz, and Pawel Zielinski, *Approximating a two-machine flow shop scheduling under discrete scenario uncertainty*, European Journal of Operational Research **217** (2012), no. 1, 36–43.

[29] Adam Kasperski, Adam Kurpisz, and Paweł Zieliński, *Parallel machine scheduling under uncertainty*, Advances in Computational Intelligence - 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU 2012, Proceedings, Part IV (Salvatore Greco, Bernadette Bouchon-Meunier, Giulianella Coletti, Mario Fedrizzi, Benedetto Matarazzo, and Ronald R. Yager, eds.), Communications in Computer and Information Science, vol. 300, Springer, 2012, pp. 74–83.

[30] ———, *Approximating the min-max (regret) selecting items problem*, Information Processing Letters **113** (2013), 23–29.

[31] Adam Kasperski and Paweł Zieliński, *An approximation algorithm for interval data minmax regret combinatorial optimization problems*, Information Processing Letters **97** (2006), 177–180.

[32] ———, *On combinatorial optimization problems on matroids with uncertain weights*, European Journal of Operational Research **177** (2007), 851–864.

[33] ———, *A 2-approximation algorithm for interval data minmax regret sequencing problems with the total flow time criterion*, Operation Research Letters **36** (2008), 343–344.

[34] ———, *A randomized algorithm for the min-max selecting items problem with uncertain weights*, Annals of Operations Research **172** (2009), 221–230.

[35] Adam Kasperski and Paweł Zieliński, *Possibilistic minmax regret sequencing problems with fuzzy parameters*, IEEE Transactions on Fuzzy Systems **19** (2011), 1072–1082.

[36] Panos Kouvelis, Richard L. Daniels, and George Vairaktarakis, *Robust scheduling of a two-machine flow shop with uncertain processing times*, IIE Transactions **32** (2000), 421–432.

[37] Panos Kouvelis and Gang Yu, *Robust discrete optimization and its applications*, Kluwer Academic Publishers, 1997.

[38] Eugene L. Lawler, *Optimal sequencing of a single machine subject to precedence constraints*, Management Science **19** (1973), 544–546.

[39] Vasilij Lebedev and Igor Averbakh, *Complexity of minimizing the total flow time with interval data and minmax regret criterion*, Discrete Applied Mathematics **154** (2006), 2167–2177.

[40] Jan Karel Lenstra and Alexander H. G. Rinnooy Kan, *Complexity of scheduling under precedence constraints*, Operations Research **26** (1978), 22–35.

[41] Jan Karel Lenstra, Alexander H. G. Rinnooy Kan, and Peter Brucker, *Complexity of Machine Scheduling Problems*, Studies in Integer Programming (P.L. Hammer, E. L. Johnson, B. H. Korte, and G. L. Nemhauser, eds.), Annals of Discrete Mathematics, vol. 1, North-Holland Publishing Company, 1977, pp. 343–363.

[42] R. Duncan Luce and Howard Raiffa, *Games and Decisions: Introduction and Critical Survey*, Dover Publications Inc., 1957.

[43] Monaldo Mastrolilli, Nikolaus Mutsanas, and Ola Svensson, *Single machine scheduling with scenarios*, Theoretical Computer Science **477** (2013), 57–66.

[44] Roberto Montemanni, *A mixed integer programming formulation for the total flow time single machine robust scheduling problem with interval data*, Journal of Mathematical Modelling and Algorithms **6** (2007), 287–296.

[45] J. Michael Moore, *An n job, one machine sequencing algorithm for minimizing the number of late jobs*, Management Science **15** (1968), 102–109.

[46] Michael Pinedo, *Scheduling: Theory, algorithms, and systems*, Prentice Hall, 2002.

[47] Chris N. Potts, *An algorithm for the single machine sequencing problem with precedence constraints*, Mathematical Programming Study **13** (1980), 78–87.

[48] ———, *Analysis of a linear-programming heuristic for scheduling unrelated parallel machines*, Discrete Applied Mathematics **10** (1985), 155–164.

[49] Ismael Regis de Farias, Hongxia Zhao, and Ming Zhao, *A family of inequalities valid for the robust single machine scheduling polyhedron*, Computers and Operations Research **37** (2010), 1610–1614.

[50] Sartaj K. Sahni, *Algorithms for Scheduling Independent Tasks*, Journal of the ACM **23** (1976), 116–127.

[51] Leonard J. Savage, *The Foundations of Statistics*, 2 ed., Dover, New York, 1972.

[52] Andreas S. Schulz, *Polytopes and Scheduling*, Ph.D. thesis, Technical University of Berlin, Germany, 1996.

[53] ———, *Scheduling to minimize total weighted completion time: Performance guarantees of LP-Based heuristics and lower bounds*, IPCO, 1996, pp. 301–315.

[54] Sergey V. Sevastianov, *Vector summation in Banach space and polynomial time algorithms for flow shops and open shops*, Mathematics of Operations Research **20** (1995), 90–103.

[55] Sergey V. Sevastianov and Gerhard J. Woeginger, *Makespan minimization in open shops: A polynomial time approximation scheme*, Mathematical Programming **82** (1998), 191–198.

[56] Marcin Siepak and Jerzy Józefczyk, *Minmax regret algorithms for uncertain PCmax problem with interval processing times*, 21st International Conference on Systems Engineering (ICSEng 2011) (Henry Selvaraj and Dawid Zydek, eds.), IEEE Computer Society, 2011, pp. 115–119.

[57] Wayne E. Smith, *Various optimizers for single-stage productions*, Naval Research Logistics Quarterly **3** (1956), 59–66.

[58] Aravind Srinivasan, *Improved Approximation Guarantees for Packing and Covering*, SIAM Journal on Computing **29** (1999), 648–670.

[59] A. Ton Volgenant and Cees W. Duin, *Improved polynomial algorithms for robust bottleneck problems with interval data*, Computers and Operations Research **37** (2010), 909–915.

[60] Ronald R. Yager, *On ordered weighted averaging aggregation operators in multicriteria decision making*, IEEE Transactions on Systems, Man and Cybernetics **18** (1988), 183–190.

[61] Jian Yang and Gang Yu, *On the robust single machine scheduling problem*, Journal of Combinatorial Optimization **6** (2002), 17–33.

[62] David Zuckerman, *Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number*, Theory of Computing **3** (2007), 103–128.