Approximating a two-machine flow shop scheduling under discrete scenario uncertainty[☆]

Adam Kasperski[a], Adam Kurpisz[b], Paweł Zieliński[b,*]

[a]*Institute of Industrial Engineering and Management, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland*
[b]*Institute of Mathematics and Computer Science, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland*

Abstract

This paper deals with the two machine permutation flow shop problem with uncertain data, whose deterministic counterpart is known to be polynomially solvable. In this paper, it is assumed that job processing times are uncertain and they are specified as a discrete scenario set. For this uncertainty representation, the min-max and min-max regret criteria are adopted. The min-max regret version of the problem is known to be weakly NP-hard even for two processing time scenarios. In this paper, it is shown that the min-max and min-max regret versions of the problem are strongly NP-hard even for two scenarios. Furthermore, the min-max version admits a polynomial time approximation scheme if the number of scenarios is constant and it is approximable with performance ratio of 2 and not $(4/3-\epsilon)$-approximable for any $\epsilon > 0$ unless P=NP if the number of scenarios is a part of the input. On the other hand, the min-max regret version is not at all approximable even for two scenarios.

Keywords: combinatorial optimization, scheduling, approximation, robust optimization, flow shop

---

[☆]The first and the third author of the paper were partially supported by Polish Committee for Scientific Research, grant N N206 492938.

[*]Corresponding author.

Email addresses: adam.kasperski@pwr.wroc.pl (Adam Kasperski), adam.kurpisz@pwr.wroc.pl (Adam Kurpisz), pawel.zielinski@pwr.wroc.pl (Paweł Zieliński)

1. Introduction

The permutation flow shop is one of the most important and most extensively studied scheduling problems. The classical result by Johnson [1] shows that the problem with two machines is polynomially solvable. However, the three-machine case becomes strongly NP-hard [2]. The parameters in the problem are job processing times, which in the classical deterministic case are assumed to be precisely known.

In the existing literature several methods of dealing with imprecise parameters have been proposed. One of the most extensively studied is a *robust approach*, where a *scenario set* containing all possible realizations of the parameters, called *scenarios*, is specified. No probability distribution in the scenario set is given. In order to choose a solution several *robust criteria* can be applied (see [3] for a survey). The simplest one is the min-max criterion, under which we choose a solution minimizing the largest cost over all scenarios. A less conservative criterion is the min-max regret, under which we choose a solution minimizing the largest deviation from optimum over all scenarios. Both criteria have a long tradition in decision making under uncertainty (see, e.g., [4]). In the last two decades they were applied to discrete optimization problems. An extensive description of the robust approach and some applications of this framework to several discrete optimization problems can be found in [5].

The scenario set can be specified in several ways. In the *discrete scenario case*, which is discussed in this paper, we simply list all the possible scenarios. We then distinguish the *bounded case*, where the number of scenarios is bounded by a constant and the *unbounded case*, where the number of scenarios is a part of the input. In the *interval case*, for each parameter a closed interval containing all its possible values is given. Then the scenario set is the Cartesian product of all these uncertainty intervals. Note that in this case the scenario set contains infinite number of scenarios. It turns out that both uncertainty representations may lead to problems having quite different computational properties (see [6, 7, 8, 9, 10] and a survey [11]).

In this paper we consider the two machine permutation flow shop problem with uncertain job processing times modeled by a discrete scenario set. This problem was investigated in [5] and [12], where it was shown that its min-max regret version with only two scenarios is weakly NP-hard. Then a branch and bound algorithm and simple heuristics for computing a solution were proposed. No additional results on this problem have been known up to now. The min-max regret permutation flow shop problem with interval job processing times was discussed in [5, 12, 13]. In [13] an $O(m)$-algorithm for the min-max regret permutation flow shop problem with 2 jobs and m machines was proposed. In [5, 12], a branch and bound method and some heuristics for the two-machine case were given. Let us point out that the computational complexity of the problem with interval processing times is still an unsolved open problem. Other examples of the min-max (regret) versions of the classical scheduling problems, with both interval and discrete scenario uncertainty representation, can be found in [14, 15, 16, 17, 18, 19, 20, 21].

Our results. In this paper we first discuss the case in which the scenario set contains only two scenarios. We show that even such a restricted min-max (regret) problem is strongly NP-hard. This result excludes the existence of a fully polynomial-time approximation scheme (FPTAS) for the bounded case, which is in contrast to the other min-max (regret) combinatorial optimization problems such as the shortest path or the minimum spanning tree, whose min-max (regret) versions for bounded scenario set admit FPTAS's [22]. Furthermore, we prove that the min-max regret version of the problem is not at all approximable while the min-max version admits a polynomial-time approximation scheme (PTAS) when the number of scenarios is constant. We then consider the min-max version of the problem with unbounded scenario set. We show that in this case the problem is approximable within 2, but not approximable within $(4/3-\epsilon)$ for any $\epsilon > 0$ unless P=NP. Hence, it does not admit a PTAS.

2. The problem formulation

We are given a set of jobs $\mathcal{J} = \{1, \ldots, n\}$, which must be processed on each of the two machines M_1 and M_2. Each job $i \in \mathcal{J}$ is processed first on machine M_1 for time p_{i1} and then on machine M_2 for time p_{i2}. Each job completes its processing on machine M_1 before it starts processing on M_2. Moreover, each machine can execute at most one job at a time. A *schedule* is a permutation $\pi = (\pi(1), \ldots, \pi(n))$ of jobs and it represents an order in which the jobs are processed on both machines. We use Π to denote the set of all the schedules. We denote by $\sigma_{ij}(\pi)$ and $C_{ij}(\pi)$ the starting time and the completion time of job i on machine M_j in schedule π, respectively. The goal is to find a schedule $\pi \in \Pi$ having the shortest *makespan*, i.e.

$C_{\max}(\pi) = \max_{i\in\mathcal{J}} C_{i2}(\pi)$. This problem is denoted as $F2||C_{\max}$ in Graham's notation (see, e.g., [23]) and is polynomially solvable by well known Johnson's algorithm [1].

In this paper we consider the case in which the processing times of jobs are uncertain. We model the uncertainty by specifying a *scenario set* $\Gamma = \{S_1, \dots, S_K\}$, containing all possible realizations of the processing times. A vector $S = (p^S_{11}, \dots, p^S_{n1}, p^S_{12}, \dots, p^S_{n2}) \in \Gamma$ is called a *scenario* and p^S_{ij} is the processing time of job i on machine M_j under scenario S. Now $\sigma^S_{ij}(\pi)$ and $C^S_{ij}(\pi)$ are the starting time and the completion time of job i on machine M_j in schedule π under a given scenario $S \in \Gamma$ and $C_{\max}(\pi, S) = \max_{i\in\mathcal{J}} C^S_{i2}(\pi)$ is the makespan of π under S. Let $C^*_{\max}(S) = \min_{\pi\in\Pi} C_{\max}(\pi, S)$, so $C^*_{\max}(S)$ is the makespan of an optimal schedule under scenario S.

In order to choose a solution, two optimization criteria, called the *min-max* and the *min-max regret*, can be adopted. In the MIN-MAX $F2||C_{\max}$ problem, we seek a schedule minimizing the largest makespan over all scenarios. Thus, we would like to solve:

$$\min_{\pi\in\Pi} \max_{S\in\Gamma} C_{\max}(\pi, S).$$

Let us define $Z(\pi) = \max_{S\in\Gamma}\{C_{\max}(\pi, S) - C^*_{\max}(S)\}$. The value of $Z(\pi)$ is called the *maximal regret* of a schedule π, i.e. the largest deviation from the optimum over all scenarios. In the MIN-MAX REGRET $F2||C_{\max}$ problem, we wish to find a schedule that minimizes the maximal regret, that is

$$\min_{\pi\in\Pi} Z(\pi).$$

The optimal solutions to the min-max and min-max regret versions of $F2||C_{\max}$ are called an *optimal min-max schedule* and an *optimal min-max regret schedule*, respectively.

The aim of this paper is to investigate the complexity and approximability of both robust problems. Up to now, we only know that the min-max regret version of the problem with two scenarios is NP-hard [12]. The reduction shown in [12] proceeds from the partition problem, which is not strongly NP-complete. So, it is possible that the min-max (regret) problem with 2 scenarios (or any fixed number of scenarios) is solvable in pseudopolynomial time and even admits an FPTAS. In Section 3 we will show that this is not possible unless P=NP.

Γ	Machine	Times	Jobs $i = 1, \dots, 4q+1$							
			1	$\dots$	$3q$	$3q+1$	$3q+2$	$\dots$	$4q$	$4q+1$
S_1	M_1	$p_{i1}^{S_1}$	0	$\dots$	0	0	$2B$	$\dots$	$2B$	$2B$
	M_2	$p_{i2}^{S_1}$	a_1	$\dots$	a_{3q}	B	B	$\dots$	B	B
S_2	M_1	$p_{i1}^{S_2}$	a_1	$\dots$	a_{3q}	B	B	$\dots$	B	B
	M_2	$p_{i2}^{S_2}$	0	$\dots$	0	$2B$	$2B$	$\dots$	$2B$	0

Figure 1: An instance of Min-Max $F2||C_{\max}$.

3. Hardness results

In this section, we study the complexity and hardness of approximation of Min-Max (Regret) $F2||C_{\max}$. We first consider the restricted bounded case in which the number of scenarios equals 2. We then investigate the unbounded case.

3.1. The bounded case

We prove the following result:

Theorem 1. Min-Max $F2||C_{\max}$ *is strongly NP-hard if* Γ *contains only two scenarios.*

Proof. We show a reduction from the following 3-Partition problem, which is known to be strongly NP-complete [24]:

3-Partition: *Input:* Positive integers q, B and a set of integers $A = \{a_1, \dots, a_{3q}\}$ such that $\sum_{k=1}^{3q} a_k = qB$ and $B/4 < a_k < B/2$ for $k = 1, \dots, 3q$.

Question: Is there a partition of A into sets $A_1, \dots, A_q$, such that $\sum_{a \in A_k} a = B$ for each $k = 1, \dots, q$?

Given an instance of 3-Partition, we construct an instance of Min-Max $F2||C_{\max}$ with $4q+1$ jobs and two scenarios $\Gamma = \{S_1, S_2\}$. The job processing times under both scenarios are shown in Figure 1.

We prove that the answer to 3-partition is "yes" if and only if there exists a schedule π such that $\max_{S \in \Gamma} C_{\max}(\pi, S) \leq (2q+1)B$.

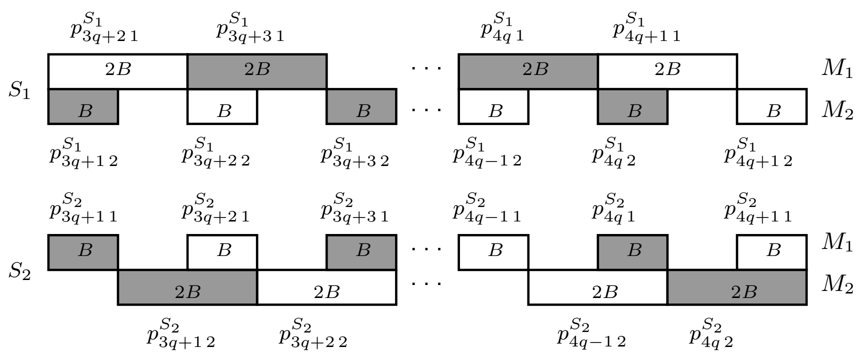


Figure 2: A partial schedule π with makespans of $(2q+1)B$ under scenarios S_1 and S_2.

Suppose that there is a partition $A_1,\dots,A_q$ of A such that $\sum_{a\in A_k} a = B$, for each $k=1,\dots q$. In Figure 2, a partial schedule $\pi' = (3q+1,\dots,4q+1)$ consisting of $q+1$ jobs is shown. The makespan of π' under both scenarios S_1, S_2 is equal to $(2q+1)B$. Observe that there are exactly q time intervals of length B on M_2 under S_1 and on M_1 under S_2. We form schedule π by inserting (in any order) 3 jobs corresponding to each set A_k between two consecutive jobs $3q+k$ and $3q+k+1$, $k=1,\dots,q$. Since the elements of each set A_k sum up to B, the processing times of the jobs corresponding to the sets A_k fill q time intervals on M_2 under S_1 and M_1 under S_2. Thus, the resulting schedule π has the makespan equal to $(2q+1)B$ under scenarios S_1, S_2 and $\max_{S\in\Gamma} C(\pi,S) \le (2q+1)B$.

Assume that there exists a schedule π such that $\max_{S\in\Gamma} C_{\max}(\pi,S) \le (2q+1)B$. It is easily seen that the makespans of π under the two scenarios are equal to $(2q+1)B$. Furthermore, the jobs on M_2 under S_1 must be processed without any idle time, which follows from equality $\sum_{i\in\mathcal{J}} p^{S_1}_{i2} = (2q+1)B$. Notice that we have exactly qB free time on machine M_2 under S_1 and on machine M_1 under S_2 for scheduling the jobs $1,\dots,3q$. From the definition of scenarios S_1 and S_2 it follows that a free time interval on M_2 under S_1 must correspond to exactly the same free interval on M_1 under S_2 and vice versa (in particular, job $3q+1$ must be processed first and job $4q+1$ must be processed last). Otherwise we could not fill the free time intervals without increasing the makespan $(2q+1)B$ under S_1 or S_2. We thus conclude that the partial schedule π' must have the form presented in Figure 2. Of course, the order of jobs $3q+2,\dots,4q$ can be arbitrary because they have the same processing times under both scenarios. Hence, the jobs placed in q free time intervals of length B define a partition of A into subsets $A_1, \dots, A_q$ whose elements sum up to B. $\square$

From Theorem 1, it follows that the Min-Max $F2||C_{\max}$ problem does not admit an FPTAS even for two scenarios, unless P=NP. The following theorem strengthens the previous result obtained in [12], where Min-Max Regret $F2||C_{\max}$ has been proved to be weakly NP-hard:

Theorem 2. Min-Max Regret $F2||C_{\max}$ *is strongly NP-hard and not at all approximable, even for two scenarios, unless P=NP.*

Proof. The reduction from 3-Partition is the same as in the proof of Theorem 1. It is easy to verify that each schedule has a makespan with the length at least $(2q+1)B$ under both scenarios. Furthermore, schedule $\pi' = (1, \dots, 3q, 3q+1, \dots, 4q+1)$ is such that $C_{\max}(\pi', S_1) = C^*_{\max}(S_1) = (2q+1)B$ and schedule $\pi'' = (3q+1, \dots, 4q+1, 1, \dots, 3q)$ is such that $C_{\max}(\pi'', S_2) = C^*_{\max}(S_2) = (2q+1)B$ and thus they are optimal under S_1 and S_2, respectively.

Assume that the answer for 3-partition is “yes”. Then there exists a schedule π such that $\max_{S\in\Gamma} C_{\max}(\pi, S) = (2q+1)B$ (see the proof of Theorem 1). According to the above remark, we have $Z(\pi) = \max\{C_{\max}(\pi, S_1) - C^*_{\max}(S_1), C_{\max}(\pi, S_2) - C^*_{\max}(S_2)\} = 0$. If the answer for 3-Partition is “no”, then every schedule π is such that $\max_{S\in\Gamma} C_{\max}(\pi, S) > (2q+1)B$ and, consequently, its maximal regret $Z(\pi)$ is positive. So, the problem of asserting whether there exists a schedule π such that $Z(\pi) = \max_{S\in\Gamma}\{C_{\max}(\pi, S) - C^*_{\max}(S)\} \leq 0$ is strongly NP-complete and thus Min-Max Regret $F2||C_{\max}$ is strongly NP-hard. Also, Min-Max Regret $F2||C_{\max}$ is not at all approximable, unless P=NP. Otherwise, any polynomial time approximation algorithm applied to the constructed instance would solve in polynomial time the 3-Partition problem. $\square$

3.2. The unbounded case

We now consider the unbounded version of Min-Max $F2||C_{\max}$. We prove the following result:

Theorem 3. *For the unbounded case* Min-Max $F2||C_{\max}$ *is not* $(4/3-\epsilon)$-*approximable for any* $\epsilon > 0$, *unless P=NP.*

	Clause scenarios												Contradictory literal scenarios																			
	C_1				C_2				C_3				$l_1^1 \frown l_2^1$				$l_1^2 \frown l_2^2$				$l_1^3 \frown l_3^1$				$l_2^3 \frown l_3^1$				$l_2^1 \frown l_3^3$			
	S_1		S_2		S_3		S_4		S_5		S_6		S_7		S_8		S_9		S_{10}		S_{11}		S_{12}		S_{13}		S_{14}		S_{15}		S_{16}	
	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
J_1^1	**2**	0	0	**2**	0	0	0	0	0	0	0	0	**3**	0	0	**3**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J_1^2	**2**	0	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0	0	**3**	0	0	0	0	0	0	0	0	0	0	0	0
J_1^3	**2**	0	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0	0	**3**	0	0	0	0	0	0	0	0
J_2^1	0	0	0	0	**2**	0	0	**2**	0	0	0	0	**3**	0	0	**3**	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0	0	**3**
J_2^2	0	0	0	0	**2**	0	0	**2**	0	0	0	0	0	0	0	0	**3**	0	0	**3**	0	0	0	0	0	0	0	0	0	0	0	0
J_2^3	0	0	0	0	**2**	0	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0	0	**3**	0	0	0	0
J_3^1	0	0	0	0	0	0	0	0	**2**	0	0	**2**	0	0	0	0	0	0	0	0	**3**	0	0	**3**	**3**	0	0	**3**	0	0	0	0
J_3^2	0	0	0	0	0	0	0	0	**2**	0	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J_3^3	0	0	0	0	0	0	0	0	**2**	0	0	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**3**	0	0	**3**
J_4	0	**2**	**2**	0	0	**2**	**2**	0	0	**2**	**2**	0	0	**2**	**2**	0	0	**2**	**2**	0	0	**2**	**2**	0	0	**2**	**2**	0	0	**2**	**2**	0

Figure 3: A scenario set for three clauses $C_1 = (x_1 \vee \overline{x}_2 \vee \overline{x}_3)$, $C_2 = (\overline{x}_1 \vee x_2 \vee \overline{x}_3)$ and $C_3 = (x_3 \vee \overline{x}_3 \vee x_1)$.

PROOF. We show a gap-introducing reduction from a special case of NAE 3-SAT. The NAE 3-SAT problem is known to be strongly NP-complete [24] and is defined as follows:

NAE 3-SAT: *Input:* A set $\mathscr{U} = \{x_1, \ldots, x_n\}$ of Boolean variables and a collection $\mathscr{C} = \{C_1, \ldots, C_m\}$ of clauses, where each clause in $\mathscr{C}$ has exactly three distinct literals.

Question: Is there a truth assignment for $\mathscr{U}$ such that each clause in $\mathscr{C}$ has at least one true literal and at least one false literal?

We can assume that for each variable $x_k \in \mathscr{U}$ both x_k and $\overline{x}_k$ appear in $\mathscr{C}$. If not, then we add two new clauses $(x_k \vee \overline{x}_k \vee x'_k)$ and $(x_k \vee \overline{x}_k \vee \overline{x}'_k)$, where x'_k is a new Boolean variable. Clearly, this modification does not change the complexity of the problem.

Given an instance of NAE 3-SAT, we construct the corresponding instance of MIN-MAX $F2||C_{\max}$ as follows. We associate with each clause $C_k = (l_k^1 \vee l_k^2 \vee l_k^3)$ three *clause jobs* J_k^1, J_k^2, J_k^3 corresponding to three literals in C_k, $k = 1, \ldots m$. We also add an additional job J_{m+1}. So, the number of jobs is $3m + 1$. Next, we form scenario set Γ.

We first create the *clause scenarios.* Namely, we associate with each clause C_k, $k = 1, \ldots, m$, two scenarios. Under the first scenario the jobs J_k^1, J_k^2, J_k^3 have processing times equal to 2 on M_1 and all the remaining processing times of clause jobs are set to 0. The job J_{m+1} has zero processing time on M_1 and its processing time on M_2 equals 2. The second scenario is symmetric, that is the jobs J_k^1, J_k^2, J_k^3 have processing times equal to 2 on M_2 and all the remaining processing times of the clause jobs are set to 0. The job J_{m+1} has processing time equal to 2 on M_1 and zero processing time on M_2 (see Figure 3). In the next step we add the *contradictory literal scenarios* as follows. For each pair of jobs J_v^u and J_q^r which correspond to contradictory literals l_v^u and l_q^r from different clauses, i.e. $l_v^u = \overline{l}_q^r$, $v \neq q$, we create two scenarios. Under the first scenario the processing times of J_v^u and J_q^r on M_1 are set to 3 and all the remaining processing times of the clause jobs are set to 0. The job J_{m+1} has zero processing time on M_1 and its processing time equals 2 on M_2. The second scenario is symmetric, that is the processing times of J_v^u and J_q^r on M_2 are set to 3 and all the remaining processing times of the clause jobs are set to 0. The job J_{m+1} has processing time equal to 2 on M_1 and zero processing time on M_2 (see Figure 3). Note that in the constructed instance each schedule has makespan of length at least 6 under each scenario, so $\max_{S\in\Gamma} C_{\max}(\pi, S) \geq 6$. The cardinality of Γ is bounded by a polynomial in the size of NAE 3-SAT and thus the instance of MIN-MAX $F2||C_{\max}$ can be constructed in a time bounded by a polynomial in the size of NAE 3-SAT.

If the answer to NAE 3-SAT is "yes", then basing on a satisfying truth assignment, one can construct a schedule π, in which all the jobs corresponding to true literals are executed before J_{m+1} and all the jobs corresponding to false literals are executed after J_{m+1} in π. Notice that the three jobs corresponding to the literals of the same clause are placed in π as follows: either one of these jobs precedes J_{m+1} and the remaining two jobs follow it or two of them precede J_{m+1} and the remaining one job follows it. This is due to the fact that the assignment is true and each clause in $\mathscr{C}$ under this assignment has at least one true literal and at least one false literal. An easy verification shows that $\max_{S\in\Gamma} C_{\max}(\pi, S) = 6$ (see Figure 4). In the example shown in Figure 3, a satisfying truth assignment is $x_1 = 1$, $x_2 = 1$, $x_3 = 1$ and a corresponding optimal schedule is $\pi = (J_1^1, J_2^2, J_3^1, J_3^3, J_4, J_1^2, J_1^3, J_2^1, J_2^3, J_3^2)$ with makespans of length 6 under each scenario.

On the other hand, if the answer is "no", then for all schedules π at least two jobs corresponding to contradictory literals either precede the distinguished job J_{m+1} or follow it, or for at least one clause all three jobs corresponding to this clause either precede J_{m+1} or follow it (because all the

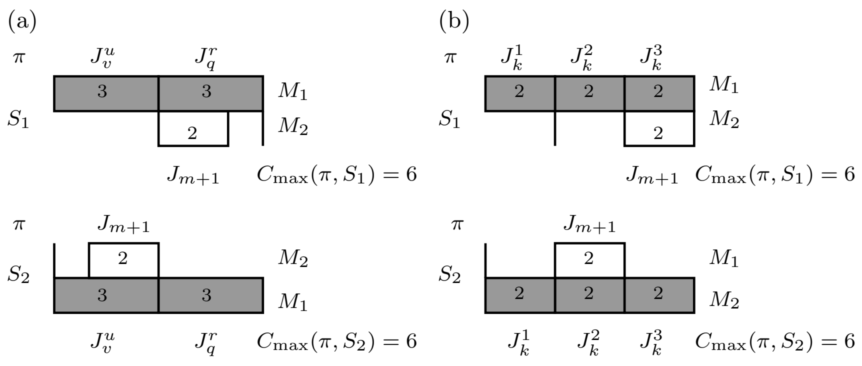


Figure 4: (a) The makespans of schedule $\pi = (\ldots, J_u^v, \ldots, J_{m+1}, \ldots, J_q^r, \ldots)$ under two contradictory literal scenarios S_1, S_2. (b) The makespans of schedule $\pi = (\ldots, J_k^1, \ldots, J_k^2, \ldots, J_{m+1}, \ldots, J_k^3, \ldots)$ under two clause scenarios S_1, S_2. Schedule $\pi = (\ldots, J_k^1, \ldots, J_{m+1}, \ldots, J_k^2, \ldots, J_k^3, \ldots)$ has the same makespans.

literals in this clause have the same value). So, $\max_{S\in\Gamma} F(\pi, S) \geq 8$ (see Figure 5).

This yields a gap of 4/3 and Min-Max $F2||C_{\max}$ for the unbounded case is not approximable within $(4/3 - \epsilon)$ for any $\epsilon > 0$ unless P=NP. □

Theorem 3 implies that Min-Max $F2||C_{\max}$ for the unbounded case does not admit a PTAS, unless P=NP.

4. Approximating Min-Max $F2||C_{\max}$

In this section, we investigate the approximation of Min-Max $F2||C_{\max}$. We start by introducing some additional notation:

- $C^{\text{rob}}_{\max} = \min_{\pi\in\Pi} \max_{S\in\Gamma} C_{\max}(\pi, S)$ is the value of an optimal min-max schedule,
- $L_j^S = \sum_{i\in\mathcal{J}} p_{ij}^S$ is the total load of machine M_j, $j = 1, 2$, under scenario $S \in \Gamma$,
- $L^S_{\max} = \max\{L_1^S, L_2^S\}$ stands for the maximum machine load in S,
- $L_{\max} = \max_{S\in\Gamma} L^S_{\max}$ denotes the maximum machine load over the set of scenarios,

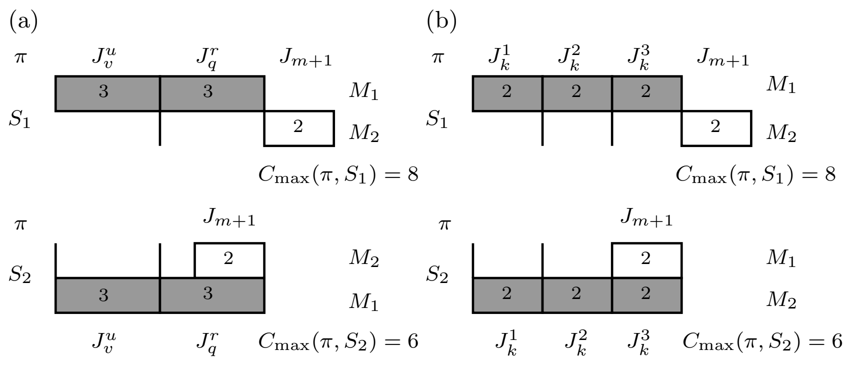


Figure 5: (a) The makespans of schedule $\pi = (\ldots, J_u^v, \ldots, J_q^r, \ldots, J_{m+1}, \ldots)$ under two contradictory literal scenarios S_1, S_2. Schedule $\pi = (\ldots, J_{m+1}, \ldots, J_u^v, \ldots, J_q^r, \ldots)$ has makespans $C_{\max}(\pi, S_1) = 6$ and $C_{\max}(\pi, S_2) = 8$. (b) The makespans of schedule $\pi = (\ldots, J_k^1, \ldots, J_k^2, \ldots, J_k^3, \ldots, J_{m+1}, \ldots)$ under two clause scenarios S_1, S_2. Schedule $\pi = (\ldots, J_{m+1}, \ldots, J_k^1, \ldots, J_k^2, \ldots, J_k^3, \ldots)$ has makespans $C_{\max}(\pi, S_1) = 6$ and $C_{\max}(\pi, S_2) = 8$.

– $l_i^S = p_{i1}^S + p_{i2}^S$ is the length of job $i \in \mathcal{J}$ in scenario S,

– $l_i^{\max} = \max_{S \in \Gamma} l_i^S$ denotes the maximum job length $i \in \mathcal{J}$ over the set of scenarios,

– $p_{\max} = \max_{S \in \Gamma, i \in \mathcal{J}, j \in \{1,2\}} p_{ij}^S$ is the maximal job processing time in an input instance.

We prove the following simple observation for the unbounded version of the problem:

Observation 1. *The unbounded version of* MIN-MAX $F2||C_{\max}$ *is approximable within 2.*

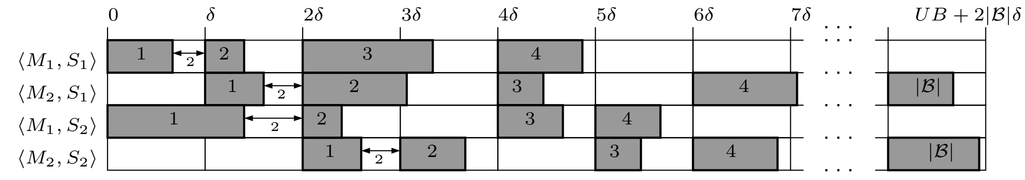
PROOF. It is clear that $C_{\max}^{\text{rob}} \geq L_{\max}$. On the other hand, for any schedule π and scenario S, $C_{\max}(\pi, S) \leq 2L_{\max}^S$. Therefore, $\max_{S \in \Gamma} C_{\max}(\pi, S) \leq 2L_{\max} \leq 2C_{\max}^{\text{rob}}$. Hence a trivial algorithm that outputs any schedule yields the approximation bound of 2 for MIN-MAX $F2||C_{\max}$. $\square$

We now construct a polynomial time approximation scheme (PTAS) for the bounded version of the problem, i.e. when the number of scenarios K is a fixed constant. Our PTAS will be based on the idea of the PTAS for the deterministic job shop problem with a fixed number of machines proposed in [25] (we also use some ideas from [26] and [27]). However, the details of our algorithm will be different. Let us fix $\epsilon \in (0,1)$ and let $\alpha > 0$ be a fixed number fulfilling the following inequality

$$\epsilon^{\lceil 2/\epsilon \rceil} \leq \alpha \leq \epsilon. \tag{1}$$

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

Step 1. Let $UB = 2L_{\max}$ be the upper bound on $C_{\max}^{\text{rob}}$. Define $\delta = \alpha\epsilon L_{\max}$ and let us assign the time interval $[0, UB + 2\delta|\mathcal{B}|]$ to machines M_1 and M_2 under each scenario $S \in \Gamma$. We will refer to the interval on M_j under S as $\langle M_j, S\rangle$. Each interval $\langle M_j, S\rangle$ 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 $i \in \mathcal{B}$ in each $\langle M_j, S\rangle$ so that i starts at the beginning of some interval of the first type and the order of the big jobs in all $\langle M_j, S\rangle$ is the same as in π. Following the notation of [27], the resulting partial schedule will be called an *outline* and we will denote it by $\mathcal{O}$ (our outline is, however, different than that in [27]). An outline is feasible if under any scenario: the jobs do not overlap on any machine, the precedence constraints between the jobs on machines M_1 and M_2 are not violated, and no job is completed after $UB + 2\delta|\mathcal{B}|$. A sample feasible outline is shown in Figure 6. 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 $2/\alpha\epsilon + 4K/\alpha$. 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.



0
δ
2δ
3δ
4δ
5δ
6δ
7δ
$UB + 2|\mathcal{B}|\delta$
$\langle M_1, S_1\rangle$
$\langle M_2, S_1\rangle$
$\langle M_1, S_2\rangle$
$\langle M_2, S_2\rangle$
1
2
3
4
$|\mathcal{B}|$

Figure 6: 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:

$$\sum_{i\in\mathcal{S}} l_i^{\max} \leq \epsilon K L_{\max}. \tag{2}$$

The reasoning is similar to that in [28, 25]. Consider a sequence of real numbers $(\alpha_1, \alpha_2, \ldots, \alpha_{\lceil 2/\epsilon\rceil})$, where $\alpha_k = \epsilon^k$. Each α_k defines a set of small jobs $\mathcal{S}_k = \{i \in \mathcal{J} \mid \alpha_k\epsilon L_{\max} < l_i^{\max} < \alpha_k L_{\max}\}$. Clearly $\mathcal{S}_j \cap \mathcal{S}_k = \emptyset$ for $j \neq k$. If the inequality (2) would be violated for all $\mathcal{S}_k$, $k = 1, \ldots, \lceil 2/\epsilon\rceil$, then

$$\sum_{i\in\mathcal{J}}\sum_{S\in\Gamma}(p_{i1}^S + p_{i2}^S) \geq \sum_{i\in\mathcal{J}} \max_S(p_{i1}^S + p_{i2}^S) = \sum_{i\in\mathcal{J}} l_i^{\max} \geq \sum_{k=1}^{\lceil 2/\epsilon\rceil}\sum_{i\in\mathcal{S}_k} l_i^{\max} > 2KL_{\max},$$

which is a contradiction. Hence, inequality (2) must hold for at least one $\mathcal{S}_k$ and we can fix $\alpha = \alpha_k$. Inequality (2) shows that the total length of the small jobs is upper bounded by $\epsilon K L_{\max}$ and the total contribution of them to the makespan is at most $\epsilon K L_{\max}$. 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 $\mathcal{O}$ all the tiny jobs from the set $\mathcal{T}$. Let us number the big jobs with respect to their positions in $\mathcal{O}$, i.e. the first big job is indexed by 1 and the last one by $|\mathcal{B}|$. We denote by $\sigma_{kj}^S(\mathcal{O})$ and $C_{kj}^S(\mathcal{O})$ the starting and completion time, respectively, of the big job k on machine M_j under scenario S. For ease of notation, we will also define $C_{0j}^S(\mathcal{O}) = 0$ and $\sigma_{|\mathcal{B}|+1\,j}^S(\mathcal{O}) = UB + 2|\mathcal{B}|\delta$ for each S and $j = 1, 2$. Making use of these starting and completion times we create in each $\langle M_j, S\rangle$ additional intervals $[C_{k-1\,j}^S(\mathcal{O}), \sigma_{kj}^S(\mathcal{O})]$ 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 6. The time contained in an interval of the second type in $\langle M_j, S\rangle$ will be used for processing tiny jobs in this interval on machine M_j 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 $\mathcal{O}$, we use a linear programming formulation. We define the following decision variables: x_{ik}, $i \in \mathcal{T}$, $k = 1, \ldots, |\mathcal{B}| + 1$ and C, where $x_{ik} = f$, $0 \leq f \leq 1$, means that the same fraction f of a tiny job i 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:

$$C_{\min}(\mathcal{O}) = \min C \tag{3}$$

$$\sum_{k=1}^{|\mathcal{B}|+1} x_{ik} = 1, \qquad i \in \mathcal{T}, \tag{4}$$

$$C^S_{k-1\,j}(\mathcal{O}) + \sum_{i\in\mathcal{T}} p^S_{ij} x_{ik} \leq \sigma^S_{kj}(\mathcal{O}), \qquad k = 1, \ldots, |\mathcal{B}|, \quad j = 1, 2, \;\; S \in \Gamma, \tag{5}$$

$$C^S_{|\mathcal{B}|\,j}(\mathcal{O}) + \sum_{i\in\mathcal{T}} p^S_{ij} x_{i\,|\mathcal{B}|+1} \leq C, \qquad j = 1, 2, \;\; S \in \Gamma, \tag{6}$$

$$C \leq UB + 2|\mathcal{B}|\delta, \tag{7}$$

$$C, x_{ik} \geq 0, \qquad i \in \mathcal{T}, \;\; k = 1, \ldots, |\mathcal{B}| + 1. \tag{8}$$

Constraints (4) assure that each tiny job i is fully assigned. Constraints (5) and (6) 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 (6) together with the objective function minimize the maximal length, over all scenarios, of the assignment computed. Constraint (7) assures that this length is not too large, i.e. it does not exceed $UB + 2|\mathcal{B}|\delta$. It is not clear that the linear program (3)-(8) is feasible. We will provide a feasible solution for (3)-(8) in the following observation.

Observation 2. *There is a feasible outline $\mathcal{O}^*$ such that*

$$C_{\min}(\mathcal{O}^*) \leq C^{rob}_{\max} + 2|\mathcal{B}|\delta. \tag{9}$$

Proof. Let π^* be an optimal min-max schedule with the value $C^{\mathrm{rob}}_{\max}$. We transform π^* into $\tilde{\pi}^*$ using the method which is illustrated in Figure 7. Consider intervals $\langle M_1, S\rangle$ and $\langle M_2, S\rangle$ for some scenario S. In Figure 7, the first

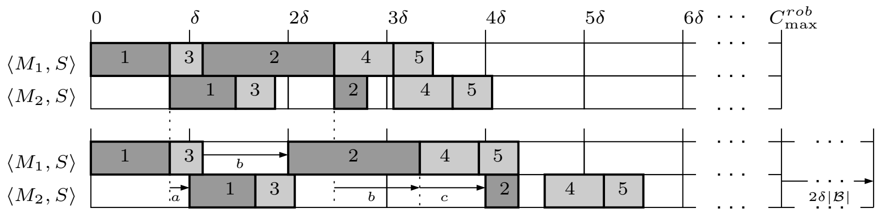


Figure 7: A sample transformation of the schedule π^* into $\tilde{\pi}^*$.

big job 1 is placed properly (i.e. at the beginning of some interval of the first type) on M_1 but not properly on M_2. So, we delay the starting times of all the jobs on M_2 by a, so that job 1 starts processing at time δ on M_2. The next big job is 2. It is not placed properly on M_1, so we delay the starting times of 2 and all the jobs succeeding 2 on M_1 and M_2 by b. Then job 2 is still not placed properly on M_2, so we delay the starting times of 2 and all the jobs succeeding 2 by c on M_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 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 $\mathcal{O}^*$ 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 $\mathcal{O}^*$. Consider the set of constraints (4)-(8) built with respect to $\mathcal{O}^*$. Let us fix $C = C^{\rm rob}_{\max} + 2\delta|\mathcal{B}|$. Note that $C^{\rm rob}_{\max} \leq UB = 2L_{\max}$ 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 $\mathcal{O}^*$ and set $x_{ik} = 1$ if and only if job $i \in \mathcal{T}$ 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_{ik} satisfies all the constraints (4)-(8). In consequence, there is at least one feasible solution for $C = C^{\rm rob}_{\max} + 2\delta|\mathcal{B}|$, which implies (9). $\square$

We now make an observation on basic feasible solutions of the linear program (3)-(8).

Observation 3. *In an optimal solution to the linear program (3)-(8) at most* $2K(|\mathcal{B}|+1)+1$ *tiny jobs receive fractional assignment.*

Proof. 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 (4), we have:

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

Combining these inequalities yields $|\mathcal{T}_2| \leq 2K(|\mathcal{B}| + 1) + 1$. □

Observation 3 implies the tiny jobs $\mathcal{T}_2$, receiving fractional assignment, are such that:

$$\sum_{i\in\mathcal{T}_2} l_i^{\max} \leq (2K(|\mathcal{B}| + 1) + 1)\alpha\epsilon L_{\max}. \tag{10}$$

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 (3)-(8). 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 [29] for the *Compact Vector Summation* problem. We recall the following result on this problem:

Lemma 1 ([29, 30]). *Consider a set of vectors* $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \in \mathbb{R}^d$ *such that* $\sum_{k=1}^{n} \boldsymbol{v}_k = \mathbf{0}$. *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* $i = 1, \ldots, n$,

$$\left\| \sum_{k=1}^{i} \boldsymbol{v}_{\varrho(k)} \right\|_\infty \leq d \max_{1\leq k\leq n} \|\boldsymbol{v}_k\|_\infty.$$

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 maximal processing time among all the processing times of the tiny jobs.

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

$$C_{\max}(\varrho, S) \leq L^S_{\max} + (K+1)p_{\max} \quad \text{for all } S \in \Gamma. \tag{11}$$

Proof. The proof is adapted from [30, Sect. 1.5.2], where the use of Sevastianov's algorithm to produce an algorithm for $F||C_{\max}$ was presented. We first transform the instance of the problem so that the loads of the two machines under each scenario S are equal, i.e. $L^S_{\max} = L^S_1 = L^S_2$. 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 (11) for the modified instance, then ρ also satisfies (11) for the original one. The following equality holds for any permutation schedule π:

$$C_{\max}(\pi, S) = I^S_2 + \sum_{i=1}^{n} p^S_{\pi(i)2} = I^S_2 + L^S_2 \quad \text{for all } S \in \Gamma, \tag{12}$$

where I^S_2 is the total idle time on machine M_2 under scenario S in π. The amount of the idle time on machine M_2 under scenario S, before it starts processing job $\pi(i)$ can be determined by the following formula:

$$I^S_{\pi(i)2} = \begin{cases} p^S_{\pi(1)1} & \text{if } i = 1, \\ I^S_{\pi(i-1)2} & \text{if } C^S_{\pi(i)2}(\pi) = C^S_{\pi(i-1)2}(\pi) + p^S_{\pi(i)2}, \\ \sum_{k=1}^{i} p^S_{\pi(k)1} - \sum_{k=1}^{i-1} p^S_{\pi(k)2} & \text{if } C^S_{\pi(i)2}(\pi) = C^S_{\pi(i)1}(\pi) + p^S_{\pi(i)2}. \end{cases} \tag{13}$$

We can rewrite the third case of (13) as $p^S_{\pi(i)2} + \sum_{k=1}^{i}(p^S_{\pi(k)1} - p^S_{\pi(k)2})$. Now, if we can find a permutation ϱ such that

$$\sum_{k=1}^{i}(p^S_{\varrho(k)1} - p^S_{\varrho(k)2}) \leq Kp_{\max}, \; i = 1, \ldots, n, \text{ for all } S \in \Gamma \tag{14}$$

then we conclude from (13) that $I^S_{\varrho(i)2} \leq \max\{I^S_{\varrho(i-1)2}, (K+1)p_{\max}\}$ and, since $I^S_{\varrho(1)2} \leq p_{\max}$, $I^S_2 = I^S_{\varrho(n)2} \leq (K+1)p_{\max}$. Because $\sum_{i=1}^n p^S_{\varrho(i)2} = L^S_2 = L^S_{\max}$, equality (12) implies (11).

It remains to show that a permutation ρ fulfilling (14) 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 $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$, where $\boldsymbol{v}_k = (p^{S_1}_{k1} - p^{S_1}_{k2}, p^{S_2}_{k1} - p^{S_2}_{k2}, \dots, p^{S_K}_{k1} - p^{S_K}_{k2})$, $k = 1, \dots, n$. This set satisfies the assumptions of Lemma 1. Indeed, $\sum_{k=1}^n \boldsymbol{v}_k = (L^{S_1}_1 - L^{S_1}_2, L^{S_2}_1 - L^{S_2}_2, \dots, L^{S_K}_1 - L^{S_K}_2) = \mathbf{0}$ by equality $L^S_{\max} = L^S_1 = L^S_2$. 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 $\max\{\sigma^S_{k1}(\mathcal{O}) - C^S_{k-1\,1}(\mathcal{O}), \sigma^S_{k2}(\mathcal{O}) - C^S_{k-1\,2}(\mathcal{O})\} + (K+1)p^{\mathcal{T}}_{\max}$, see (11). Hence, in order to accommodate the tiny jobs according to the computed permutation schedule to the intervals $[C^S_{k-1\,1}(\mathcal{O}), \sigma^S_{k1}(\mathcal{O})]$ and $[C^S_{k-1\,2}(\mathcal{O}), \sigma^S_{k2}(\mathcal{O})]$, $C^S_{k-1\,1}(\mathcal{O}) \leq C^S_{k-1\,2}(\mathcal{O})$, $\sigma^S_{k1}(\mathcal{O}) \leq \sigma^S_{k2}(\mathcal{O})$, for all S without violating its feasibility, it suffices to increase the length of the kth interval of the second type on machine M_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 M_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. According to Observation 2, there must be a feasible outline $\mathcal{O}^*$ which satisfies inequality (9). The linear program applied to $\mathcal{O}^*$ 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 for all the jobs in $\mathcal{J}$ with the maximal makespan over all scenarios $\tilde{C}^{\mathrm{rob}}_{\max}$. According to (9) and the construction described this makespan can be upper bounded as follows:

$$\tilde{C}^{\mathrm{rob}}_{\max} \leq C^{\mathrm{rob}}_{\max} + 2\delta|\mathcal{B}| + \sum_{i \in \mathcal{T}_2} l^{\max}_i + \sum_{i \in \mathcal{S}} l^{\max}_i + (|\mathcal{B}|+1)(K+1)p^{\mathcal{T}}_{\max}.$$

Now using (1), (2), (10) and the conditions: $\delta = \alpha\epsilon L_{\max}$, $|\mathcal{B}| \leq 2K/\alpha$, $C_{\max}^{\text{rob}} \geq L_{\max}$ and $p_{\max}^{\mathcal{T}} \leq \alpha\epsilon L_{\max}$, we get after easy computations:

$$\tilde{C}_{\max}^{\text{rob}} \leq (1 + O(K^2)\epsilon)C_{\max}^{\text{rob}}.$$

For the constant values of K and ϵ, our algorithm is polynomial with respect to the number of jobs n. Thus it is a PTAS for the problem considered.

5. Conclusions

In this paper we have investigated the two-machine flow shop problem under the discrete scenario uncertainty representation with bounded and unbounded scenario sets. For the bounded case, we have proved that the min-max version of the problem is strongly NP-hard even for two scenarios. This result excludes the existence of an FPTAS for the problem but admits the existence of a PTAS. In this paper, we have constructed a PTAS for the bounded min-max version of the problem. For the bounded min-max regret version we have shown that the situation is much worse, namely, in this case the problem has turned out to be strongly NP-hard and not at all approximable even for two scenarios. For the unbounded scenario set we have proved that the min-max version is approximable within 2 and not $(4/3 - \epsilon)$-approximable for any $\epsilon > 0$. There is still an unresolved gap between the positive and negative approximation results for the unbounded case. The 2-approximation algorithm shown in this paper is trivial and further research should involve the development of efficient approximation algorithms with better than 2 worst case ratio for the unbounded min-max version of the problem.

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

[2] M. R. Garey, D. S. Johnson, R. Sethi, The complexity of flowshop and jobshop scheduling, Mathematics of Operations Research 1 (1976) 117–129.

[3] B. Roy, Robustness in operational research and decision aiding: A multi-faceted issue, European Journal of Operational Research 200 (2010) 629–638.

[4] R. D. Luce, H. Raiffa, Games and Decisions: Introduction and Critical Survey, Dover Publications Inc., 1957.

[5] P. Kouvelis, G. Yu, Robust Discrete Optimization and its applications, Kluwer Academic Publishers, 1997.

[6] G. Yu, J. Yang, On the robust shortest path problem, Computers and Operations Research 6 (1998) 457–468.

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

[8] I. Averbakh, V. Lebedev, Interval data minmax regret network optimization problems, Discrete Applied Mathematics 138 (2004) 289–301.

[9] A. Kasperski, P. Zieliński, On the approximability of minmax (regret) network optimization problems, Information Processing Letters 109 (2009) 262–266.

[10] A. Kasperski, P. Zieliński, On the approximability of robust spanning tree problems, Theoretical Computer Science 412 (2011) 365–374.

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

[12] P. Kouvelis, R. L. Daniels, G. Vairaktarakis, Robust scheduling of a two-machine flow shop with uncertain processing times, IIE Transactions 32 (2000) 421–432.

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

[14] M. A. Alouloua, F. D. Croce, Complexity of single machine scheduling problems under scenario-based uncertainty, Operation Research Letters 36 (2008) 338–342.

[15] R. L. Daniels, P. Kouvelis, Robust scheduling to hedge against processing time uncertainty in single stage production, Management Science 41 (1995) 363–376.

[16] A. Kasperski, Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion, Operation Research Letters 33 (4) (2005) 431–436.

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

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

[19] R. 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 (2) (2007) 287–296.

[20] M. Mastrolilli, N. Mutsanas, O. Svensson, Approximating Single Machine Scheduling with Scenarios, in: APPROX-RANDOM 2008, Vol. 5171 of Lecture Notes in Computer Science, Springer-Verlag, 2008, pp. 153–164.

[21] J. Yang, G. Yu, On the Robust Single Machine Scheduling Problem, Journal of Combinatorial Optimization 6 (2002) 17–33.

[22] H. Aissi, C. Bazgan, D. Vanderpooten, General approximation schemes for minmax (regret) versions of some (pseudo-)polynomial problems, Discrete Optimization 7 (2010) 136–148.

[23] P. Brucker, Scheduling Algorithms, 5th Edition, Springer Verlag, Heidelberg, 2007.

[24] M. R. Garey, D. S. Johnson, Computers and Intractability. A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979.

[25] K. Jansen, R. Solis-Oba, M. Sviridenko, Makespan Minimization in Job Shops: A Linear Time Approximation Scheme, SIAM Journal on Discrete Mathematics 16 (2003) 288–300.

[26] C. N. Potts, Analysis of a linear-programming heuristic for scheduling unrelated parallel machines, Discrete Applied Mathematics 10 (1985) 155–164.

[27] L. A. Hall, Approximability of flow shop scheduling, Mathematical Programming 82 (1998) 175–190.

[28] S. V. Sevastianov, G. J. Woeginger, Makespan minimization in open shops: A polynomial time approximation scheme, Mathematical Programming 82 (1998) 191–198.

[29] S. 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.

[30] L. A. Hall, Approximation Algorithms for Scheduling, in: D. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, PWS, 1995, pp. 1–43.