Complexity of scheduling on parallel identical machines to minimize total flow time with interval data and minmax regret criterion

Maciej Drwal
Institute of Computer Science, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland

Abstract

In this paper we consider the problem of scheduling jobs on parallel identical machines, where the processing times of jobs are uncertain: only interval bounds of processing times are known. The optimality criterion of a schedule is the total flow time. In order to cope with uncertainty we make use of the robust optimization framework: we seek a schedule which performs well under all possible instantiations of processing times. Consequently, the goal is to minimize the maximum regret value of solution. Although the deterministic version of this problem is solvable in polynomial time, the minmax regret version is known to be weakly NP-hard even for a single machine, and strongly NP-hard for parallel unrelated machines. In this paper we show that the problem is strongly NP-hard also in the case of parallel identical machines.

Keywords: robust optimization; scheduling; uncertainty; computational complexity

1. Introduction

Robust optimization framework has been applied to many combinatorial optimization problems, since in practical applications input data to most problems can be rarely given precisely. This is true in the context of scheduling, as in many actual execution environments (e.g., computer systems, transportation, manufacturing) processing times of tasks are not known exactly, but their values can fluctuate within certain bounds. When no good assumptions can be made even regarding their probability distributions, the best decision may be to use a solution that is good enough even in the worst possible scenario of events. Such solutions can be characterized in terms of maximum regret criterion \[1, 4, 3\]. Solutions that minimize the maximum regret are often much more reliable than
ones obtained by solving only deterministic formulation of the problem, ignoring parameters uncertainty. However, in many cases finding robust solution is much more difficult and may require more computational resources.

We apply the minmax regret approach to the problem of scheduling on parallel identical machines to minimize total flow time (sum of completion times of all jobs) with interval uncertainty in job processing times. The problem under consideration is denoted INTERVAL \( P_{m||\sum C_i} \), using standard scheduling theory notation [4]. Deterministic version of this problem can be solved in polynomial time, e.g., by applying shortest processing time first rule, simply sorting all jobs by processing times in nondecreasing order. However, its minmax regret version becomes NP-hard even for a single machine, i.e., INTERVAL 1||\( \sum C_i \). In [5] it is shown that even when midpoints of all intervals are equal and the number of jobs is odd, finding optimal robust sequence on a single machine is weakly NP-hard (surprisingly, for even number of jobs this problem is polynomially solvable). Thus the case in which the number of machines is given as a part of the input can be no easier. Recently, Conde [6] indicated a simple reduction from minmax regret assignment problem [7] of \( m \) jobs to \( m \) machines, which implies that in case of \( m \) parallel unrelated machines (INTERVAL \( R_{m||\sum C_i} \)) the problem is strongly NP-hard. Unfortunately, this reduction no longer applies to the case of parallel identical machines, since in that settings the assignment becomes trivial. In this paper we extend the aforementioned complexity results, showing that strong NP-hardness occurs even for identical machines.

2. Problem formulation

An instance of the considered scheduling problem \( P_{m||\sum C_i} \) is given by the integer \( n \), denoting the number of jobs, the integer \( m \), denoting the number of machines (processors), and the set of processing times of each job: integers \( p_i \) for \( i = 1, \ldots, n \). Each job has to be assigned to exactly one machine. Let \( \pi_j \) denote a vector of \( n_j \) integers, where \( \pi_j(k) \) is the index of job scheduled on \( j \)th machine as \( k \)th to the last (jobs on each machine are scheduled starting from time zero and without idle times). The completion time of job: \( C_{j,k} = \sum_{i=1}^{k} p_{\pi_j(i)} \) (\( C_{j,k} = 0 \) if there is no such job). The objective is to minimize the sum of completion times (also called total flow time), expressed as:

\[
F(\pi) = \sum_{j=1}^{m} \sum_{k=1}^{n_j} C_{j,k} = \sum_{j=1}^{m} \sum_{k=1}^{n_j} k p_{\pi_j(k)},
\]

where \( \pi = [\pi_1, \ldots, \pi_m] \) is called a schedule. We will sometimes refer to this problem formulation as a deterministic version of the scheduling problem.

The definition of minmax regret version of this problem with interval uncertainty, denoted INTERVAL \( P_{m||\sum C_i} \), differs in that, instead of exact processing times, we are given only intervals \([p_i^-, p_i^+]\), \( i = 1, \ldots, n \), to which the actual processing times belong. Denote by \( S = [p_1^S, \ldots, p_n^S] \) any vector that satisfies \( p_i^- \leq p_i^S \leq p_i^+ \) for all \( i = 1, \ldots, n \). Such a vector will be called
a scenario. For any schedule and scenario we define the value of regret as: 
\[ Z(\pi, S) = F(\pi, S) - F^*(S), \]
where \( F(\pi, S) \) is the objective function from the deterministic version of problem \( P_m || \sum C_i \) with input data \( S \), and \( F^*(S) \) is the value of optimal solution of this problem. The objective of INTERVAL \( P_m || \sum C_i \) is to minimize over schedules the maximum of regret over scenarios:
\[ Z^* = \min_{\pi} \max_S (F(\pi, S) - F^*(S)). \] (2)
The above minmax regret formulation is an example of robust optimization formulation of the scheduling problem. A schedule minimizing the maximum regret will be called robust optimal.

3. Computation of maximum regret

Deterministic version of the considered problem is easily solvable in polynomial time by sorting all jobs in the order of nondecreasing processing times, and then by filling subsequently positions on machines in that order, so that the difference between the numbers of jobs assigned to any two machines at any time is never greater than 1.

We show that it is possible to compute the value of maximum regret \( Z(\pi) = \max_S Z(\pi, S) \) in polynomial time. The method is analogous to the one presented in [6] for parallel unrelated machines, with the main difference that in the identical machines case the input data contains a single interval \([p_i^-, p_i^+]\) in place of \( m \) intervals given in unrelated machines case. Thus we omit the simple proof of correctness of formulas (3)–(5).

Let us encode a feasible solution \( \pi \) of the considered problem in terms of binary variables \( x \) as follows: let \( x_{ijk} = 1 \) iff \( i \)th job is processed on \( j \)th machine as \( k \)th to the last.

For any feasible schedule \( x \) the maximum regret can be computed as:
\[ Z(x) = \sum_{j=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{n} kp_i^+ x_{ijk} - \min_y \sum_{j=1}^{m} \sum_{i=1}^{n} \sum_{k=1}^{k} c_{ijk}(x)y_{ijk}, \] (3)
where:
\[ c_{ijk}(x) = kp_i^- + (p_i^+ - p_i^-) \sum_{i=1}^{m} \sum_{r=1}^{n} \min\{r, k\} x_{ir}. \] (4)
The minimization in (3) is equivalent to minimum assignment problem and thus can be solved in polynomial time, using, e.g., the Hungarian method. Variable \( y \) is \( n \times (mn) \) permutation matrix.

Let \( x_{ij} = k \) iff \( x_{ijk} = 1 \), and \( y_{ij} = k \) iff \( y_{ijk} = 1 \). The worst-case scenario for fixed \( x \) can be obtained, given \( y \) solving minimization in (3):
\[ p_i = \begin{cases} p_i^+ & \text{if } \sum_{j=1}^{m} x_{ij} - y_{ij} \geq 0, \\ p_i^- & \text{otherwise}. \end{cases} \] (5)
4. Problem complexity

Denote by $Z(\pi)$ the value of solution of problem \( P_m||\sum C_i \) with schedule \( \pi = [\pi_1, \ldots, \pi_m] \). Maximum regret can be written as:

$$Z(\pi) = \max_S (F(\pi, S) - F^*(S)) = \max_S \sum_{j=1}^{m} (F_j(\pi_j, S) - F_j^*(S)) = \sum_{j=1}^{m} Z_j(\pi_j).$$

Here $F_j(\pi_j, S) = \sum_{i=1}^{n_j} (n_j - i + 1)p_{\pi_j(i)}^S$ is the sum of completion times of jobs on machine \( j \) under scenario \( S \), \( n_j = |\pi_j| \) is the number of jobs assigned to machine \( j \), $F_j^*(S)$ is the sum of completion times of jobs on machine \( j \) in optimal solution under scenario \( S \), and finally, $Z_j(\pi_j)$ is the difference between $F_j(\pi_j, S)$ and $F_j^*(S)$ for such \( S \) that maximizes total regret. Let $\pi^*$ denote the optimal robust solution, i.e., the one minimizing $Z$.

Consider the worst-case scenario for any schedule $\pi$. The optimal solution of deterministic version of the problem obtained by taking such worst-case scenario as an input data is called the worst-case alternative for $\pi$.

From now on we consider only instances with $n$ jobs and $m$ machines where $m|n$, that is, there exists integer $n_0$, such that $n = n_0m$.

In particular, if $m|n$ then the worst-case alternative for any schedule has always equal number of jobs on each machine. Moreover, the following is true.

**Lemma 1.** In optimal robust schedule every machine is assigned the same number of jobs.

**Proof.** Let $\pi_1$ be a schedule with equal number of jobs on each machine, and let $\pi_2$ be a schedule with different number of jobs on at least two machines. Observe that for any scenario $S$ we have $F(\pi_1, S) < F(\pi_2, S)$ Let us denote by $S^\pi$ the worst-case scenario for $\pi$. Then we get:

$$Z(\pi_1) = F(\pi_1, S^{\pi_1}) - F^*(S^{\pi_1}) < F(\pi_2, S^{\pi_1}) - F^*(S^{\pi_1})$$

$$\leq F(\pi_2, S^{\pi_2}) - F^*(S^{\pi_2}) = Z(\pi_2)$$

The second inequality follows from the fact that $S^{\pi_1}$ is not necessarily the worst-case scenario for $\pi_2$, thus by definition its value is no greater than that of solution $\pi_2$ under its worst-case scenario $S^{\pi_2}$. Lemma 1 follows from the above reasoning, which indicates that for any schedule with different numbers of jobs on machines, there exists a strictly better solution with equal number of jobs on each machine, so only such a solution could be optimal.

**Lemma 2.** Given a schedule $\pi$, where $\pi$ is a $m \times n_0$ matrix, consider a schedule $\pi'$ obtained by switching a pair of elements in any column of $\pi$. Both schedules have the same maximum regret.
Proof. Let $k^{\pi(i)}$ be the position of $i$th job on its machine, counting from the last position on machine (i.e., if $i$ happens to be scheduled on machine $j$, then the last position is $n_j$, and $1 \leq k^{\pi(i)} \leq n_j$). Since every job has to be assigned to exactly one position on one machine, then clearly the cost of schedule $\pi$ under scenario $S$ is $F(\pi, S) = \sum_{i=1}^{n} k^{\pi(i)} p_i^S$. Let $\sigma$ be the worst-case alternative for $\pi$, and let $k^{\sigma(i)}$ be the position to the last of $i$th job on its machine in the worst-case alternative. Let $S^{\pi}$ be the worst-case scenario. Then maximum regret can be expressed as:

$$Z(\pi) = F(\pi, S^{\pi}) - F(\sigma, S^{\pi}) = \sum_{i=1}^{n} \left( k^{\pi(i)} - k^{\sigma(i)} \right) p_i^{S^{\pi}}.$$  \hfill (7)

The worst-case scenario $S^{\pi}$ can be found by taking $p_i^{S^{\sigma}} = p_i^+$ if $k^{\pi(i)} - k^{\sigma(i)} \geq 0$, and $p_i^{S^{\pi}} = p_i^-$ otherwise. Values $k^{\pi(i)}$ and $k^{\sigma(i)}$ can be easily determined given $\pi$ and $\sigma$.

Observe that in the formula (7) there are no machine indices associated with jobs. The value of maximum regret can be computed knowing only positions of jobs on machines, while the assignment of jobs to machines is irrelevant. Thus we may arbitrarily permute jobs within columns of $\pi$ obtaining schedules with the same maximum regret and the same worst-case alternative. 

Observe that the formulation of the considered problem remains valid when bounds of intervals $p_i^-$ and $p_i^+$ are arbitrary (possibly negative) integers, and that adding the same constant to all bounds of intervals does not change the value of maximum regret.

Consider instances with equal midpoints, that is $[p_i^-, p_i^+] = [-p_i^+, p_i^+]$. Due to [5] we know that optimal solution for a single machine $j$ can be obtained as:

(R.1) for even number of jobs $n_j = 2k_j$ on machine $j$, $Z_j(\pi^*) = k_j \sum_i p_i^+$,

(R.2) for odd number of jobs $n_j = 2k_j + 1$ on machine $j$ we have:

$$Z_j(\pi^*) = k_j \sum_i p_i^+ + \max\{P_1, P_2\},$$  \hfill (8)

where $(P_1, P_2)$ is a solution to optimization version of PARTITION problem (i.e., $P_1$ and $P_2$ are sums of two disjoint subsets of $2k_j$ smallest jobs, and the value $|P_1 - P_2|$ is minimal among all such 2-partitions). The widest job is always inserted in the middle of the permutation and does not appear in $P_1$ or $P_2$. The remaining jobs are scheduled in such a way that the wider the interval, the closer it is to the middle of permutation (in [5] authors call such schedules uniform).

Lemma 3. There exists optimal robust schedule of INTERVAL $P_m||\sum C_i$, such that for any machine $j = 1, \ldots, m$, the schedule on machine $j$ is the same as the optimal robust schedule in problem INTERVAL $1||\sum C_i$.

Proof. According to Lemma[2] given any schedule $\pi$, it is possible to construct another schedule $\pi'$ by permuting columns of $\pi$ arbitrarily. The worst-case
scenario for $\pi'$ is identical to the worst case scenario of $\pi$ due to the Lemma 2 and formula (5). The worst-case alternative for $\pi'$ can be obtained by sorting all $mn_0$ processing times in ascending order, then grouping them into $n_0$ consecutive sets of $m$ numbers, so that jobs from the same group occupy $i$th position on machine, $i = 1, \ldots, n_0$ (again, jobs within a group may be assigned to machines arbitrarily without changing the value of solution). By $\sigma$ we denote the worst-case alternative for $\pi$.

Given a schedule $\pi$ and its worst-case alternative $\sigma$, we construct schedule $\pi'$ and its worst-case alternative $\sigma'$, such that:

1) solution $\pi'$ has the same maximum regret as $\pi$,
2) for each machine $j = 1, \ldots, m$, each job processed in $\pi'$ on $j$th machine is also processed on machine $j$ in its worst-case alternative $\sigma'$.

To accomplish this, let us consider the following algorithm. Initialize $\pi'$ and $\sigma'$ to be $m \times n_0$ matrices containing all zeros. Iterate over all jobs $i = 1, \ldots, n$, and let $k$ be the index of column of $\pi$ containing $i$, and let $l$ be the index of column in $\sigma$ containing $i$. We now try to insert $i$ at the first unoccupied position in column $k$ in matrix $\pi'$, and at the same position in column $l$ in $\sigma'$. If both positions contained zero, we insert $i$ there, and proceed to the next job. It may happen that these positions are already occupied by previously inserted jobs. In that case we backtrack to job $i - 1$, and remove it from both $\pi'$ and $\sigma'$. If this job was inserted in row $j$ then we try inserting it into first unoccupied positions in $\pi'$ and $\sigma'$, now starting from row $j + 1$.

To see the correctness of this algorithm, observe that the procedure never changes the column index of any job in $\pi$ and $\sigma$, and inserts every job $i$ in the same row in both $\pi'$ and $\sigma'$ (possibly in different columns). There is always the same number of unoccupied positions left in each row of both $\pi'$ and $\sigma'$, at any time during the execution of the algorithm. As the number of conflicting positions in $\pi$ and $\sigma$ can be never greater than $\min\{m, n_0\}$, it is always possible to insert all jobs. Consequently, the procedure always terminates. An example of the use of this algorithm is shown in Figure 1.

Let $\pi^*$ be an optimal robust solution. Then it is possible to obtain a schedule $\pi'$ which has the same total processing time as $\pi^*$, and the property that all jobs from $j$th machine in $\pi'$ also appear on $j$th machine in the worst-case alternative for $\pi'$. Thus $\pi'$ is also optimal robust and we may consider independently each schedule $\pi'_j$ on each $j$th machine. For any such machine the minimum of $Z_j(\pi_j)$ is obtained if $\pi_j$ is the optimal robust solution of single machine scheduling problem $\text{INTERVAL 1||}\sum C_i$.

The main result in this paper, stated as Theorem 2, is based on the reduction from a variant of a set partitioning problem, given as Theorem 1.

Consider an instance of 3-PARTITION problem: given is a set of $3m$ positive integers $a_i$, $i = 1, \ldots, 3m$, and an integer $B$, such that $\sum_{i=1}^{3m} a_i = mB$, and for all $i$, $B/4 < a_i < B/2$. The question is: can we partition the given set of integers into $m$ disjoint triplets of integers, such that each triplet sum up to exactly $B$?
We define problem 4-PARTITION-INTO-PAIRS (problem 4-PP for short) as follows: given is a set of $4m$ positive integers $a_i, i = 1, \ldots, 4m$. The question is whether it is possible to partition the given set of integers into $m$ disjoint quadruplets of integers $A_1, \ldots, A_m$, such that:

$$\forall A_i \exists A_j \neq A_i, \; s(A_i) = s(A_j),$$

where $s(A)$ is the sum of elements in $A$.

**Theorem 1.** Problem 4-PP is strongly NP-complete.

**Proof.** We reduce 3-PARTITION to 4-PP. Given an instance of 3-PARTITION with $3m$ elements and target sum $B$, let us consider an instance of 4-PP with the following set of integers:

(a) all integers from the instance of 3-PARTITION, denoted $a_1, a_2, \ldots, a_{3m}$,
(b) for $j = 1, 2, \ldots, m$, an integer $2j+1B$,
(c) for $j = 1, 2, \ldots, m$, a group of four integers $(2j+1+1)B/4$.

In the obtained instance we have $8m$ integers, and the solution consists of $2m$ disjoint quadruplets.

Suppose the instance of 3-PARTITION is positive. Denote its solution ($m$ triplets of integers): $A_1, A_2, \ldots, A_m$. Then clearly the obtained instance of 4-PP is positive, since we take each triplet of integers $A_j$ and add one integer $2j+1B$ from set (b) to obtain a quadruplet with sum $s(A_j \cup \{2j+1B\}) = (2j+1+1)B$, for $j = 1, \ldots, m$. The corresponding quadruplet with equal sum is just four integers $(2j+1+1)B/4$ from set (c).

Suppose the instance of 3-PARTITION is negative. Then in set (a) in every possible combination of triplets there would always be at least one triplet with different sum than $B$, strictly between $3B/4$ and $3B/2$. Now it can be seen that every choice of four integers from set (c) adds up to an integral multiplicity of $B$, as well as any choice of any number of integers from (b). Consequently, no match similar to the one from the positive case above is possible. It can be
seen that combining elements from set (a) and elements from set (c) will also result in quadruplet of sum being nonintegral multiplicity of $B$. Consequently, the instance of 4-PP is also negative.

\[\square\]

Theorem 2. Problem $\text{interval } P_m||\sum C_i$, in which the number of machines is given as a part of the input, is strongly NP-complete.

Proof. Given an instance of 4-PP with a set of $4m'$ integers $a_1, \ldots, a_{4m'}$, let us construct an instance of $\text{interval } P_m||\sum C_i$ problem with $m = m'/2$ machines and $n = 4m' + m'/2$ jobs (we assume w.l.o.g. that $m'$ is even): for each integer $a_i$ in the input data of 4-PP we add a job with processing interval $[-a_i, a_i]$ to the set of jobs, and additionally, we add $m$ jobs with processing intervals $[-B, B]$, where $B$ is greater or equal to the sum of 4 largest integers in the input data.

Each processing interval is a subset of $[-B, B]$, and midpoints of all processing intervals are the same and equal to 0, i.e., $(p_i^+ + p_i^-)/2 = 0$.

We use Lemma 3 and property (8), which imply that each machine with 9 jobs yields optimal maximum regret:

$$Z_j(\pi_j) = 4\left(\sum_{i=1}^{9} a_{\pi_j(i)}\right) + \max\{s(A_{j1}), s(A_{j2})\},$$

where $s(A_{j1}) = a_{\pi_j(1)} + a_{\pi_j(2)} + a_{\pi_j(3)} + a_{\pi_j(4)}$, and $s(A_{j2}) = a_{\pi_j(6)} + a_{\pi_j(7)} + a_{\pi_j(8)} + a_{\pi_j(9)}$.

Denote $C = \sum_{i=1}^{4m'} a_i$. We claim that the instance of 4-PP has a solution if and only if $Z(\pi^*) = 4mB + 9C/2$.

Assume that there exists a solution of 4-PP.

We know that, for each single machine, given 9 jobs on that machine, the minimal maximum regret is obtained when the widest job is in the middle of the schedule (position 5), and 4-element subsets on both sides of the widest job have equal sums (this is always possible if there exists a solution of 4-PP: we take any two quadruplets of jobs with matching sums of upper interval bound values).

We show that in optimal robust solution exactly one job $[-B, B]$ must be assigned to each machine (such job is consequently always in the middle of each permutation). Then given two 4-job sets with equal sum $B_j$ on each machine $j$, with $B_j \leq B$, we obtain total maximum regret:

$$Z = \sum_{j=1}^{m} (4(2B_j + B_j)) = 4mB + 9 \sum_{j=1}^{m} B_j = 4mB + 9C/2.$$

Consider a schedule with job $[-B, B]$ on each machine, and construct another schedule, by exchanging job $[-B, B]$ from machine 1 with job $[-a, a]$ from machine 2, $a < B$. Regret on machine 2 has increased by $5(B - a)$ ($Z_2$ is the regret before the exchange, $Z'_2$ after the exchange):

$$Z_2 = 4(s(A_{21}) + B + s(A_{22})) + s(A_{21}),$$
$$Z_2' = 4(s(A_{21} \setminus \{a\} \cup \{B\}) + B + s(A_{22})) + s(A_{21} \setminus \{a\} \cup \{B\}).$$

Regret on machine 1 has decreased by $4(B - a) + a' - a$, where $a'$ is the upper endpoint of the widest interval of job on machine 1 ($Z_1$ is the regret before exchange, $Z_1'$ after the exchange):

$$Z_1 = 4(s(A_{11}) + B + s(A_{12})) + s(A_{11}),$$
$$Z_1' = 4(s(A_{11}) + a + s(A_{12})) + s(A_{11} \setminus \{a'\} \cup \{a\}).$$

Consequently, the new schedule is worse by at least $B - a'$.

Similarly, it can be shown for three machines (moving three $[-B, B]$ jobs to a single machine), and so on, up to 9 machines (8 machines with no $[-B, B]$ jobs and one with 9 $[-B, B]$ jobs). All other cases are covered, e.g., the case with four machines, where instead of moving four $[-B, B]$ jobs to a single machine, we move a pair of $[-B, B]$ jobs to two machines is the same as the case of two machines, etc. Using these 9 cases it can be shown by induction, that whenever at least one out of $m$ machines is not assigned $[-B, B]$ job, then the solution is worse than in the case in which every machine is assigned one $[-B, B]$ job.

Now assume that there is no solution of 4-PP. Then it is not possible to have two 4-job sets with equal sum on each of $m$ machines, and consequently, at least one machine gives regret strictly greater than $4B + 9B_j$, thus the total regret is greater than $4mB + 9C/2$.

\[ \square \]

5. Conclusions

In this paper we proved strong NP-hardness of problem \textsc{interval} $P_m||\sum C_i$, the minmax regret version of one of the basic scheduling problems. It was shown how to compute maximum regret of a schedule in polynomial time, and how the problem relates to the single machine case. In particular, for equal midpoint intervals case, there exists an optimal robust solution $\pi^*$ of problem \textsc{interval} $P_m||\sum C_i$ in which for each machine $j = 1, \ldots, m$ the schedule $\pi_j^*$ of jobs on that machine is identical to the one in optimal solution of problem \textsc{interval} $1||\sum C_i$, restricted to that jobs. An interesting open problem is to settle the complexity status of the latter problem when its input data is encoded in unary. Another future research direction is to design approximation algorithms for scheduling problems that guarantee approximation ratio below two \[8\] (or prove that it is impossible unless P=NP).

Acknowledgement

Author would like to thank Adam Kasperski and Jerzy Jozefczyk for their valuable comments which helped to improve the paper.
References

[1] P. Kouvelis, G. Yu, Robust discrete optimization and its applications, Vol. 14, Springer, 1997.

[2] 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 (2) (2009) 427–438.

[3] A. Kasperski, Discrete optimization with interval data: minmax regret and fuzzy approach, Vol. 228, Springer, 2008.

[4] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Springer, 2012.

[5] V. Lebedev, I. Averbakh, Complexity of minimizing the total flow time with interval data and minmax regret criterion, Discrete Applied Mathematics 154 (15) (2006) 2167–2177.

[6] E. Conde, A MIP formulation for the minmax regret total completion time in scheduling with unrelated parallel machines, Optimization Letters 8 (4) (2014) 1577–1589.

[7] H. Aissi, C. Bazgan, D. Vanderpooten, Complexity of the min–max and min–max regret assignment problems, Operations Research Letters 33 (6) (2005) 634–640.

[8] A. Kasperski, P. Zieliński, A 2-approximation algorithm for interval data minmax regret sequencing problems with the total flow time criterion, Operations Research Letters 36 (3) (2008) 343–344.