Operations Research Letters 44 (2016) 354–358

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Complexity of interval minmax regret scheduling on parallel identical machines with total completion time criterion

Maciej Drwal[a,*], Roman Rischke[b]

[a] Department of Computer Science, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland

[b] Center for Mathematics, Technische Universität München, Boltzmannstr. 3, 85747 Garching bei München, Germany

ARTICLE INFO

Article history:
Received 13 December 2014
Received in revised form
9 December 2015
Accepted 8 March 2016
Available online 16 March 2016

Keywords:
Robust optimization
Scheduling
Uncertainty
Computational complexity

ABSTRACT

We consider the problem of scheduling jobs on parallel identical machines, where only interval bounds of processing times of jobs are known. The optimality criterion of a schedule is the total completion time. In order to cope with the uncertainty, we consider the maximum regret objective and seek a schedule that performs well under all possible instantiations of processing times. We show how to compute the maximum regret, and prove that its minimization is strongly NP-hard.

1. Introduction

Robust optimization 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. Moreover, very often no good assumptions can be made even regarding their probability distributions. In such circumstances we would like to find a solution that is the best in the worst possible scenario of events. Such solutions can be characterized in terms of the maximum regret criterion [7,2,5]. Solutions that minimize the maximum regret are often much more reliable than the ones obtained by ignoring parameter uncertainty. However, in many cases, finding a robust solution for uncertain data is 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 the total completion time (sum of the completion times of all jobs) with interval uncertainty in the job processing times. The problem under consideration is denoted INTERVAL MINMAX REGRET $P \parallel \sum C_i$, using the notation from standard scheduling theory. The deterministic version of this problem can be solved in polynomial time by applying the *shortest processing time first* rule [10]. However, its minmax regret version becomes NP-hard even for a single machine, i.e., INTERVAL MINMAX REGRET $1 \parallel \sum C_i$. In [8] it is shown that even when the midpoints of all intervals are equal and the number of jobs is odd, finding an optimal robust sequence on a single machine is weakly NP-hard. Surprisingly, for an even number of jobs this problem is polynomially solvable. Thus the case in which the number of machines is given as part of the input can be no easier. Recently, Conde [3] indicated a simple reduction from the minmax regret assignment problem [1] of m jobs to m machines, which implies that in case of m parallel unrelated machines (INTERVAL MINMAX REGRET $R \parallel \sum C_i$) the problem is strongly NP-hard. However, this reduction does not suffice to prove hardness for the case of parallel identical machines. In this paper, we extend the aforementioned complexity results by showing that the problem is strongly NP-hard even on identical parallel machines.

2. Problem formulation

In the scheduling problem $P \parallel \sum C_i$ we are given m identical parallel machines (processors) for processing n jobs, where each job i has an integer processing time p_i, $i = 1, \dots, n$. If not stated otherwise, we assume that all $p_i \geq 0$. Each job has to be assigned to exactly one machine. Let π_j denote a vector of integers, where $\pi_j(k)$ is the index of the job scheduled on the jth machine as the kth to the last (jobs on each machine are scheduled starting from

* Corresponding author.
E-mail addresses: maciej.drwal@pwr.edu.pl (M. Drwal), rischke@ma.tum.de (R. Rischke).

http://dx.doi.org/10.1016/j.orl.2016.03.005
0167-6377/© 2016 Elsevier B.V. All rights reserved.

time zero and without idle times). Let n_j denote the number of elements in π_j, i.e., the number of jobs assigned to machine j. The *completion time* of the job scheduled as kth to the last on machine j is $C_{j,k} = \sum_{i=k}^{n_j} p_{\pi_j(i)}$ ($C_{j,k} = 0$ if there is no such job). The objective is to minimize the total sum of completion times (also called the *total flow time*), expressed as:

$$F(\pi) = \sum_{j=1}^{m}\sum_{k=1}^{n_j} C_{j,k} = \sum_{j=1}^{m}\sum_{k=1}^{n_j} kp_{\pi_j(k)}, \tag{1}$$

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

The definition of the minmax regret version of this problem with interval uncertainty, denoted as INTERVAL MINMAX REGRET $P \parallel \sum C_i$, differs as follows. Instead of having exact processing times p_i, we are now given only intervals $[p_i^-, p_i^+]$, $i = 1, \dots, n$, to which the actual processing times belong. We denote by $S = [p_1^S, \dots, p_n^S]$ any vector that satisfies $p_i^- \le p_i^S \le p_i^+$ for all $i = 1, \dots, n$. Such a vector will be called a *scenario*. For any schedule π and scenario S we define the value of *regret* as $Z(\pi, S) := F(\pi, S) - F^*(S)$, where $F(\pi, S)$ is the objective function (1) from the deterministic version of the problem $P \parallel \sum C_i$ with input data S, and $F^*(S)$ is the value of an optimal solution of this problem. The objective of INTERVAL MINMAX REGRET $P \parallel \sum C_i$ is to minimize over schedules the maximum of regret over scenarios:

$$Z^* = \min_{\pi}\max_{S}\left(F(\pi, S) - F^*(S)\right). \tag{2}$$

The above minmax regret formulation is a *robust optimization* formulation of the scheduling problem. A schedule minimizing the maximum regret will be called *robust optimal*.

3. Computation of maximum regret

The deterministic version of the considered problem is solvable in polynomial time (see [10, Theorem 5.3.1]). An optimal schedule can be obtained by first sorting all n jobs in order of non-decreasing processing times, and then we assign the first unassigned job in the list to the least loaded machine, i.e., to the machine with the smallest current makespan. Repeating this procedure until all jobs are assigned gives the desired schedule.

We show that for a fixed schedule π 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 [3] for parallel unrelated machines, with the main difference that in the identical machines case the input data contains a single interval $[p_i^-, p_i^+]$ instead of m intervals given in the case of unrelated machines. Thus we omit the simple proof of correctness of Formulas (3)–(5).

Let us encode a feasible solution π of the considered problem in terms of binary variable $\mathbf{x}$ as follows: let $x_{ijk} = 1$ iff the ith job is processed on the jth machine as the kth to the last.

For any feasible schedule $\mathbf{x}$ the maximum regret can be computed as:

$$\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{k=1}^{n} kp_i^+ x_{ijk} - \min_{\mathbf{y}} \sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{k=1}^{n} c_{ijk}(\mathbf{x}) y_{ijk}, \tag{3}$$

where

$$c_{ijk}(\mathbf{x}) = kp_i^- + (p_i^+ - p_i^-)\sum_{l=1}^{m}\sum_{r=1}^{n}\min\{r, k\} x_{ilr}. \tag{4}$$

The minimization in (3) is equivalent to the minimum assignment problem and thus can be solved in polynomial time, using e.g. the Hungarian method [9]. Variable $\mathbf{y}$ is an $n \times (mn)$ permutation matrix.

Let $x_{ij} = k$ iff $x_{ijk} = 1$, and $y_{ij} = k$ iff $y_{ijk} = 1$. For a fixed $\mathbf{x}$, given a solution $\mathbf{y}$ of the minimization in (3), the worst-case scenario can be obtained as:

$$p_i = \begin{cases} p_i^+ & \text{if } \sum_{j=1}^{m}(x_{ij} - y_{ij}) \ge 0, \\ p_i^- & \text{otherwise.} \end{cases} \tag{5}$$

4. Properties of optimal solutions

Denote by $Z(\pi)$ the solution value of the INTERVAL MINMAX REGRET $P \parallel \sum C_i$ problem for a schedule $\pi = [\pi_1, \dots, \pi_m]$. The maximum regret can be written as:

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

Here $F_j(\pi_j, S) = \sum_{k=1}^{n_j} kp_{\pi_j(k)}^S$ is the sum of completion times of the 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 the jobs on machine j in an 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 a scenario S that maximizes the total regret. Let π^* denote an optimal robust solution, i.e., a schedule that minimizes Z.

Consider a worst-case scenario for any schedule π (see Eq. (5)). This scenario defines an instance of the deterministic version of the problem. An optimal solution of this deterministic problem is called a *worst-case alternative* for π.

From now on, we consider only instances satisfying the following assumption.

Assumption 1. The number of jobs n is divisible by the number of machines m, i.e., there exists an integer $n_0 > 0$ such that $n = m \cdot n_0$.

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

Lemma 1. *If $m|n$, then in an optimal robust schedule every machine is assigned the same number of jobs.*

Proof. Let π_2 be any schedule with different number of jobs on at least two machines. Under a fixed scenario S, there exists a schedule π_1 with an equal number of jobs on each machine, such that $F(\pi_1, S) < F(\pi_2, S)$. This follows from the fact that we can construct π_1 from π_2 by performing a sequence of the following job displacements: from the machine with the longest schedule remove the job from the first position and insert it at the first position on the machine with the least number of jobs (the multipliers k in (1) of the remaining jobs are unchanged, but the multiplier of the moved job may decrease; the last such displacement must be performed between two machines that differ in the number of jobs by 2, thus the multiplier of that job decreases by 1, and the overall cost of the schedule decreases).

Let us denote by S^π the worst-case scenario for π. Then we get:

$$\begin{aligned} Z(\pi_1) &= F(\pi_1, S^{\pi_1}) - F^*(S^{\pi_1}) < F(\pi_2, S^{\pi_1}) - F^*(S^{\pi_1}) \\ &\le F(\pi_2, S^{\pi_2}) - F^*(S^{\pi_2}) = Z(\pi_2). \end{aligned}$$

The last inequality follows from the fact that S^{π_1} is not necessarily the worst-case scenario for π_2, thus by definition the value of regret $Z(\pi_2, S^{\pi_1})$ is no greater than that of the maximum regret, computed for the worst-case scenario S^{π_2}. Lemma 1 follows from the above reasoning, which indicates that for any schedule with a different number of jobs on machines, there exists a strictly better solution with an equal number of jobs on each machine, so only such a solution could be optimal. □

Justified by Lemma 1, we restrict in the rest of the paper to schedules with an equal number of jobs on all machines, i.e., we consider $m \times n_0$ matrices, with each row representing a sequence of jobs on a machine. In order to prove the main result, we need the following lemmas.

Lemma 2. *Given a schedule π, where π is an $m \times n_0$ matrix, consider a schedule π' obtained by switching a pair of elements in any column of π. Both schedules have the same maximum regret.*

Proof. Let $k^{\pi(i)}$ be the position of the ith job on its machine, counting from the last position on the machine (i.e., if i happens to be scheduled on machine j, then the last position is n_j, and $1 \le k^{\pi(i)} \le n_j$). Since every job has to be assigned to exactly one position on one machine, then clearly the cost of the schedule π under the scenario S is $F(\pi, S) = \sum_{i=1}^{n} k^{\pi(i)} p_i^S$. Let σ be the worst-case alternative for π, and let $k^{\sigma(i)}$ be the position to the last of the ith job on its machine in the worst-case alternative. Let S^π be the worst-case scenario. Then the maximum regret can be expressed as:

$$Z(\pi) = F(\pi, S^\pi) - F(\sigma, S^\pi) = \sum_{i=1}^{n} \left(k^{\pi(i)} - k^{\sigma(i)}\right) p_i^{S^\pi}. \tag{6}$$

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

Observe that in Eq. (6) there are no machine indices associated with jobs. The value of the 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 π obtaining schedules with the same maximum regret and the same worst-case alternative. □

Lemma 3. *There exists an optimal robust schedule of* INTERVAL MINMAX REGRET $P \parallel \sum C_i$ *such that for any machine $j = 1, \dots, m$ the schedule on machine j is the same as the optimal robust schedule in the* INTERVAL MINMAX REGRET $1 \parallel \sum C_i$ *problem.*

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

Given a schedule π and its worst-case alternative σ, we construct a schedule π' and its worst-case alternative σ', such that:

(i) solution π' has the same maximum regret as π and
(ii) for each machine $j = 1, \dots, m$, each job processed in π' on the jth machine is also processed on the jth machine in its worst-case alternative σ'.

See Fig. 1 for an example. If we can prove that such a construction always exists, then we can argue that we may restrict to the single machine problem.

To accomplish this, we construct an $n_0 \times n_0$ matrix M with the following properties. Let x_i and y_i be the position (column) of job i

$$\pi = \begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix} \qquad \sigma = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

$$\pi' = \begin{bmatrix} 1 & 6 & 11 & 16 \\ 2 & 5 & 12 & 15 \\ 3 & 8 & 9 & 14 \\ 4 & 7 & 10 & 13 \end{bmatrix} \qquad \sigma' = \begin{bmatrix} 1 & 6 & 11 & 16 \\ 5 & 2 & 15 & 12 \\ 9 & 14 & 3 & 8 \\ 13 & 10 & 7 & 4 \end{bmatrix}$$

Fig. 1. Example of a pair of schedule π and its worst-case alternative σ, along with a corresponding pair π', σ' satisfying the condition of the Lemma 3; $m = 4$ machines, $n_0 = 4$ jobs on each machine (numbers denote the indices of jobs).

in π and σ, respectively. Then job i is inserted into M at (x_i, y_i). Note that several jobs can be at the same position in M, but in every row and in every column of M we have exactly m jobs. Empty positions in M are marked with an empty sign. Matrix M tells us that if two jobs i_1 and i_2 are in the same row or column of M, then they are in conflict, i.e., i_1 and i_2 cannot run on the same machine in π' and σ'. The question is now if we can color the jobs in M with m colors such that the jobs of each row and of each column have different colors. If two jobs have the same color, then they run on the same machine in π' and σ', the position remains the same as in π and σ.

We give a positive answer and show it by induction on m. Clearly, if we have only one machine, then we need only one color (machine). Suppose the statement holds for $m - 1$ machines. Assume for the moment that we can choose a set of jobs from M so that we have exactly one job in each row and each column. Removing the chosen jobs from M gives the $m - 1$ case, which by induction can be $(m-1)$-colored. The chosen jobs can be colored with a new color, yielding an m-coloring.

Now we need to show that it is possible to select a set of jobs so that we have exactly one job in each row and each column of M. This can be shown by Hall's marriage theorem [4]. We associate with our matrix M the following bipartite graph. We create a vertex for each row and each column of M and connect the vertex for row i with the vertex for column j if the matrix M has at least one element at position (i, j). If this graph has a perfect matching, then we can select a set of jobs so that we have exactly one job in each row and each column. Assume, by contradiction, that this graph has no perfect matching. Then, by Hall's marriage theorem, there is a set R of row vertices such that the neighboring set $N(R)$ of column vertices is strictly smaller, i.e., $|N(R)| < |R|$. Consider the total number τ of jobs being in the rows and columns which are represented by R and $N(R)$, i.e., we consider the submatrix formed by rows R and columns $N(R)$. In every row from R there are m jobs and, by the definition of the graph, they are all in the columns associated with $N(R)$. That is, $\tau = m|R|$. In every column of $N(R)$ there are at most m jobs when restricting to rows in R. Therefore, we have that $\tau \le m|N(R)|$, which contradicts our assumption that $|N(R)| < |R|$, since $m > 1$.

Let π^* be an optimal robust solution. We have shown that it is possible to obtain a schedule π' which has the same total processing time as π^* and the property that all jobs from the jth machine in π' also appear on the jth machine in the worst-case alternative for π'. Thus π' is also optimal robust and we may consider independently each schedule π'_j on each jth machine. For any such machine the minimum of $Z_j(\pi_j) \ge 0$ is obtained if π_j is the optimal robust solution of the single machine scheduling problem INTERVAL MINMAX REGRET $1 \parallel \sum C_i$. □

Let us take a closer look at the INTERVAL MINMAX REGRET $1 \parallel \sum C_i$ problem. Observe that its formulation remains valid when the 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 the maximum regret [8].

Consider instances with equal midpoints c of all job processing time intervals, that is $[p_i^-, p_i^+] = [c-p_i, c+p_i]$, for all $i = 1, \dots, n$. Due to [8] we know that an optimal solution for a single machine can be obtained as:

(a) for an even number of jobs $n = 2k$ on a machine, $Z(\pi^*) = k\sum_i p_i$,

(b) for an odd number of jobs $n = 2k+1$ on a machine we have:

$$Z(\pi^*) = k\sum_{i=1}^{n} p_i + \max\{P_1, P_2\}, \tag{7}$$

where (P_1, P_2) is a solution of the optimization version of the BALANCED-PARTITION problem, i.e., P_1 and P_2 are the sums of two disjoint k-element subsets of the set of $2k$ smallest values p_i, and the value $|P_1 - P_2|$ is minimal among all such 2-partitions. The job with the widest interval 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 the job is to the middle of the permutation (in [8] authors call such schedules *uniform*).

5. Problem complexity

The main result in this paper, stated as Theorem 2, is based on the reduction from a variant of a set partitioning problem that is strongly NP-complete.

Consider an instance of the 3-PARTITION problem: given is a set of $3m$ positive integers a_i, $i = 1, \dots, 3m$, 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 sums up to exactly B?

We define the 4-PARTITION-INTO-PAIRS problem (the 4-PP problem for short) as follows: given is a set of $4m$ positive integers a_i, $i = 1, \dots, 4m$. The question is whether it is possible to partition the given set of integers into m disjoint quadruplets of integers $A_1, \dots, A_m$, such that there exists a bijective function $f : \{1, \dots, m\} \to \{1, \dots, m\}$, such that:

$$\forall_{i\in\{1,\dots,m\}}\ s(A_i) = s(A_{f(i)}) \quad \text{and} \quad f(i) \neq i \text{ and } f(f(i)) = i,$$

where $s(A)$ is the sum of elements in A. In other words, we want to partition the set of integers into m 4-sets in such a way that all the 4-sets can be arranged in distinct pairs of equal sums.

Theorem 1. *The 4-PP problem is strongly NP-complete.*

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

(a) all integers from the instance of 3-PARTITION, denoted $a_1, a_2, \dots, a_{3m}$,

(b) for $j = 1, 2, \dots, m$, an integer $5^{j+1}B$,

(c) for $j = 1, 2, \dots, m$, a group of four integers $(5^{j+1}+1)B/4$.

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): $A_1, A_2, \dots, A_m$. Then clearly the obtained instance of 4-PP is positive, since we take each triplet of integers A_j and add one integer $5^{j+1}B$ from the set (b) to obtain a 4-set with the sum $s(A_j \cup \{5^{j+1}B\}) = (5^{j+1}+1)B$, for $j = 1, \dots, m$. The corresponding 4-set with an equal sum is just the four integers $(5^{j+1}+1)B/4$ from the set (c).

Suppose the instance of 3-PARTITION is negative. Then in the set (a) in every possible combination of triplets there would always be at least one triplet with a sum different than B, strictly between $3B/4$ and $3B/2$. We argue that it is not possible to match every 4-set with another one, so that these matches are all distinct, and both 4-sets in every pair have equal sums.

Given a 4-set S, if there exists a 4-set S' such that $s(S) = s(S')$, we say that S' is a *match* for S.

We first show that no match is possible for any 4-set containing more than one (b)-element. To see this, consider the representations of the given numbers in numeral system with the base 5, after dividing all the numbers by B. Each element from (b) is represented by a distinct digit 1. Numbers $\frac{1}{4}5^k$ are represented by the first k digits 1 in the integral part, numbers $\frac{1}{2}5^k$ are represented by the first k digits 2 in the integral part, and numbers $\frac{3}{4}5^k$ are represented by the first k digits 3 in the integral part, where $k \geq 2$ (all these numbers also have digits in their fractional parts that continue indefinitely). For example, $(5^k+1)/4$ is represented by $(11\dots1.\bar{2})_5$, where there are k digits 1 in the integral part.

Sum of any number of (b)-elements with up to 3 (c)-elements cannot result in the carry in integral digits, except at the first integral digit and in the fractional digits. Since all $a_i < B/2$, adding elements from the set (a) may affect only the first integral digit and the fractional digits.

It is clear that no match is possible for any 4-set consisting of only (b)-elements, since they are all distinct powers of 5.

Moreover, if there are three (b)-elements and one element from either (a) or (c) in S, then a match would require that S' contains the same four elements as S, but the (b)-elements are all unique.

Denote the elements from (b) by $b(j) = 5^{j+1}B$ and the elements from (c) by $c(j) = (5^{j+1}+1)B/4$. If there are two (b)-elements in S, then in order to find a match we need to have S' without any (b)-elements. The only way to accomplish this is to use a (c)-element in S' that can be expressed as the sum of a (b)-element and a (c)-element in S, that is, $c(j+1) = c(j) + b(j)$. But then we have $S = \{b(j), b(j+1), c(j), a_{i_1}\}$ and $S' = \{c(j+2), a_{i_2}, a_{i_3}, a_{i_4}\}$, and since $B/4 < a_i < B/2$, then no such match is possible. The same is true if we replace a_{i_1} in S by a (c)-element, as S' would then have to contain another (c)-element.

Consequently, only single (b)-elements in any 4-set S are possible. But again, examining a base-5 representation of $s(S)$ containing element $b(j)$, we can see that a match for S must have the same digit at the $(j+1)$-th position set. This can be obtained only in two ways:

(i) by using $c(j+1) = b(j) + c(j)$ and matching S with a set S' containing element $c(j+1)$,

(ii) by matching S with S' consisting of four elements $c(j)$.

Suppose we have one match of the former type for a 4-set S containing $b(j)$. Then element $c(j+1)$ cannot be used in a match of the second type, because all four elements $c(j+1)$ are needed for that and one has already been used. But then $b(j+1)$ needs to be matched to a set containing $c(j+2)$. Continuing this reasoning, we conclude that we would need to use the first type of match for a set with the element $b(m)$, but this is impossible, since element $c(m+1)$ does not exist.

Thus a set S containing $b(j)$ needs to be matched to S' consisting of four elements $c(j)$, for all $j = 1, \dots, m$. Then $s(S')$ has exactly two digits 1 in the base-5 representation: the first digit, and the $(j+1)$-th digit. But to have the first digit 1 in $s(S)$, there must be three (a)-elements in S that sum up to B.

Since there are m (b)-elements, then m 4-sets must contain a single (b)-element. But that requires that there are also m triplets of (a)-elements that sum up to B each. This contradicts the assumption that there is no solution for the considered instance of the 3-PARTITION problem. □

Theorem 2. *The* INTERVAL MINMAX REGRET $P \parallel \sum C_i$ *problem, in which the number of machines is given as a part of the input, is strongly NP-hard.*

Proof. Given an instance of 4-PP with a set of $4m'$ integers $a_1, \dots, a_{4m'}$, let us construct an instance of the INTERVAL MINMAX REGRET $P \parallel \sum C_i$ 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 a processing interval $[B-a_i, B+a_i]$ to the set of jobs, and additionally, we add m jobs with processing intervals $[0, 2B]$, where B is an integer greater than the sum of 4 largest integers in the given input data.

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

We use Lemma 3 and property (7), which imply that a single machine with 9 jobs yields the 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 values $a_i = p_i^+ - B = p_i^- + B$).

We show that in an optimal robust solution exactly one job $[0, 2B]$ must be assigned to each machine (such job is consequently always in the middle of each permutation). Then given two 4-job sets with an equal sum $B_j = s(A_{j1}) = s(A_{j2})$ on each machine j, with $B_j \le B$, we obtain the total maximum regret:

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

Suppose for the sake of contradiction that a schedule in which at least one machine is not assigned any $[0, 2B]$ jobs is optimal robust. By the pigeonhole principle, there must be a machine with more than one $[0, 2B]$ job. Denote by 1 the machine with more than one $[0, 2B]$ jobs, and by 2 the machine with no $[0, 2B]$ jobs.

Let us construct another schedule by exchanging one $[0, 2B]$ job from machine 1 with job $[B - a, B + a]$ from machine 2, $a < B$.

Let Z_1 be the regret on machine 1 before the exchange, and Z_1' be the regret on machine 1 after the exchange:

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

After the exchange the regret on machine 1 has decreased by $5(B - a)$. Similarly, let Z_2 be the regret on machine 2 before the exchange, and Z_2' be the regret on machine 2 after the exchange:

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

where a' corresponds to the widest interval $[B - a', B + a']$ of a job on machine 2 before the exchange, $a' < B$. After the exchange the regret on machine 2 has increased by $4(B - a) - a + a'$. Consequently, the new schedule has less total regret by $B - a'$, thus the initial schedule could not be optimal. By repeating the above job exchange operation we conclude that an optimal schedule must have exactly one $[0, 2B]$ job on each machine.

Now assume that there is no solution of 4-PP. Then it is not possible to have two 4-job sets with an equal sum on each of m machines, and consequently, at least one machine gives the regret strictly greater than $4B + 9B_j$. Let this be the machine with index $j = 1$. To that machine there are assigned two 4-job sets, with the corresponding values a_i denoted A_{11} and A_{12} respectively, so that $s(A_{11}) > s(A_{12})$. Let us denote the total value of these 8 jobs $s(A_{11}) + s(A_{12}) = 2B_1$, for some $B_1 > 0$. The regret generated by this machine is:

$$Z_1 = 4(s(A_{11}) + s(A_{12}) + B) + s(A_{11}) = 4B + 8B_1 + s(A_{11}).$$

Since $s(A_{11}) > B_1$, and $\sum_{j=1}^{m} B_j = C/2$, the total regret is:

$$Z_1 + \sum_{j=2}^{m}(4B + 9B_j) = 4mB + 9\sum_{j=2}^{m} B_j + 8B_1 + s(A_{11})$$
$$> 4mB + 9\sum_{j=1}^{m} B_j = 4mB + 9C/2 = Z^*.$$

It follows that given a polynomial time algorithm for INTERVAL MINMAX REGRET $P \parallel \sum C_i$ we would be able to decide 4-PP in polynomial time. □

6. Conclusions

In this paper, we proved the strong NP-hardness of the INTERVAL MINMAX REGRET $P \parallel \sum C_i$ problem, the minmax regret version of one of the basic multiprocessor scheduling problems. It was shown how to compute the maximum regret of a schedule in polynomial time, and how the problem on parallel identical machines relates to its single machine variant. An interesting open problem is to settle the complexity status of the single machine version of the considered problem, when the input data is encoded in unary. Another future research direction is to design approximation algorithms for robust scheduling problems that guarantee approximation ratio below two [6], or prove that it is impossible (unless P = NP).

References

[1] H. Aissi, C. Bazgan, D. Vanderpooten, Complexity of the min–max and min–max regret assignment problems, Oper. Res. Lett. 33 (6) (2005) 634–640.
[2] H. Aissi, C. Bazgan, D. Vanderpooten, Min–max and min–max regret versions of combinatorial optimization problems: A survey, European J. Oper. Res. 197 (2) (2009) 427–438.
[3] E. Conde, A MIP formulation for the minmax regret total completion time in scheduling with unrelated parallel machines, Optim. Lett. 8 (4) (2014) 1577–1589.
[4] P. Hall, On representatives of subsets, J. Lond. Math. Soc. 10 (1) (1935) 26–30.
[5] A. Kasperski, Discrete Optimization with Interval Data: Minmax Regret and Fuzzy Approach, Springer, 2008.
[6] A. Kasperski, P. Zieliński, A 2-approximation algorithm for interval data minmax regret sequencing problems with the total flow time criterion, Oper. Res. Lett. 36 (3) (2008) 343–344.
[7] P. Kouvelis, G. Yu, Robust Discrete Optimization and its Applications, Springer, 1997.
[8] V. Lebedev, I. Averbakh, Complexity of minimizing the total flow time with interval data and minmax regret criterion, Discrete Appl. Math. 154 (15) (2006) 2167–2177.
[9] C. Papadimitriou, K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Courier Corporation, 1998.
[10] M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Springer, 2012.