Complexity Results for Common Due Date Scheduling Problems with Interval Data and Minmax Regret Criterion

PGMO2020, Online Event, FRANCE

Imed KACEM and Hans KELLERER

Contact: imed.kacem@univ-lorraine.fr

November 28, 2020

UNIVERSITÉ DE LORRAINE

Table of Contents

Problem description

- We are given a single machine and n independent jobs j with processing times p_j, $j = 1, \ldots, n$ and common date d. The objective is to maximize the number of early jobs. The corresponding objective function is $\sum_{j=1}^{n} \bar{U}_j$ where C_j is the completion time of job j and $\bar{U}_j$ is defined as

$$\bar{U}_j = \begin{cases} 1 & : \quad C_j \leq d, \\ 0 & : \quad C_j > d. \end{cases}$$

- In the classical three field notation the problem is denoted as $1|d_j = d|\sum \bar{U}_j$. It can also be considered as a knapsack problem with capacity d, weights p_j and unit profits 1. For fixed processing times the problem can be easily solved in $O(n \log n)$ time by sorting the jobs according to increasing processing times.

Problem description

- In this paper we assume that the processing times p_j are uncertain and take any value from a certain job dependent interval $[a_j, b_j]$. W.l.o.g., we assume that all input data, i.e., a_j, b_j, d, take only integer values. For any set of jobs J we denote by $p(J)$ the total processing time of the jobs in J.
- Note that we consider the problem with objective function of maximizing the number of early jobs $\sum_{j=1}^{n} \bar{U}_j$ instead of the equivalent problem with objective function of minimizing the number of late jobs $\sum_{j=1}^{n} U_j$ because the former is closely related to the knapsack problem.
- For measuring the quality of an algorithm for that problem with imprecise data we use the concept of *minimizing the maximum regret*. An assignment of specific values p_j from the intervals $[a_j, b_j]$ is called a *(processing time) scenario*.

Problem description

- Let Γ denote the set of all possible scenarios and Π the set of all permutations of the jobs $1, \ldots, n$. For any scenario $P = (p_1, \ldots, p_n)$ and any sequence X of the jobs the number of early jobs is denoted as $F(X, P)$.
- Our problems are illustrated by the two agents Alice and Bob. Alice selects a sequence X of jobs. The problem of Bob is defined for every feasible sequence X by Alice and it consists in selecting a sequence $Y = Y(X)$ and a scenario $P = P(X)$ such that the *regret* (of Alice) $R(X, Y, P) = F(Y, P) - F(X, P)$ is maximized.
- The value $Z(X) = \max_{Y \in \Pi, P \in \Gamma} R(X, Y, P)$ denotes the *maximum regret* for X.
- Alice has to select a sequence X which minimizes her maximum regret, i.e.,

$$\min_{X \in \Pi} Z(X). \tag{1}$$

Polynomial approximation: definitions

- Let a sequence X be given by Alice and $P^*(X)$ and $Y^*(X)$ be an optimal scenario and an optimal sequence for Bob's problem, respectively. A polynomial-time algorithm H by Bob that chooses a scenario $P^H(X)$ and creates a schedule $Y^H(X)$ such that

$$\frac{R(X, Y^H(X), P^H(X))}{R(X, Y^*(X), P^*(X))} = \frac{F(Y^H(X), P^H(X)) - F(X, P^H(X))}{F(Y^*(X), P^*(X)) - F(X, P^*(X))} \geq 1 - \rho,$$

  is called a $(1-\rho)-$*approximation algorithm* for Bob's problem.
- A family of $(1-\rho)$-approximation algorithms for Bob's problem is called a *fully polynomial-time approximation scheme,* or an *FPTAS* if for a given $\rho > 0$ it contains an algorithm that has a running time which is polynomial both in the length of the problem input and in $1/\rho$. When this running time is only polynomial in the length of the problem input for a fixed value of ρ, the scheme is called a PTAS.

Illustration of the studied problem

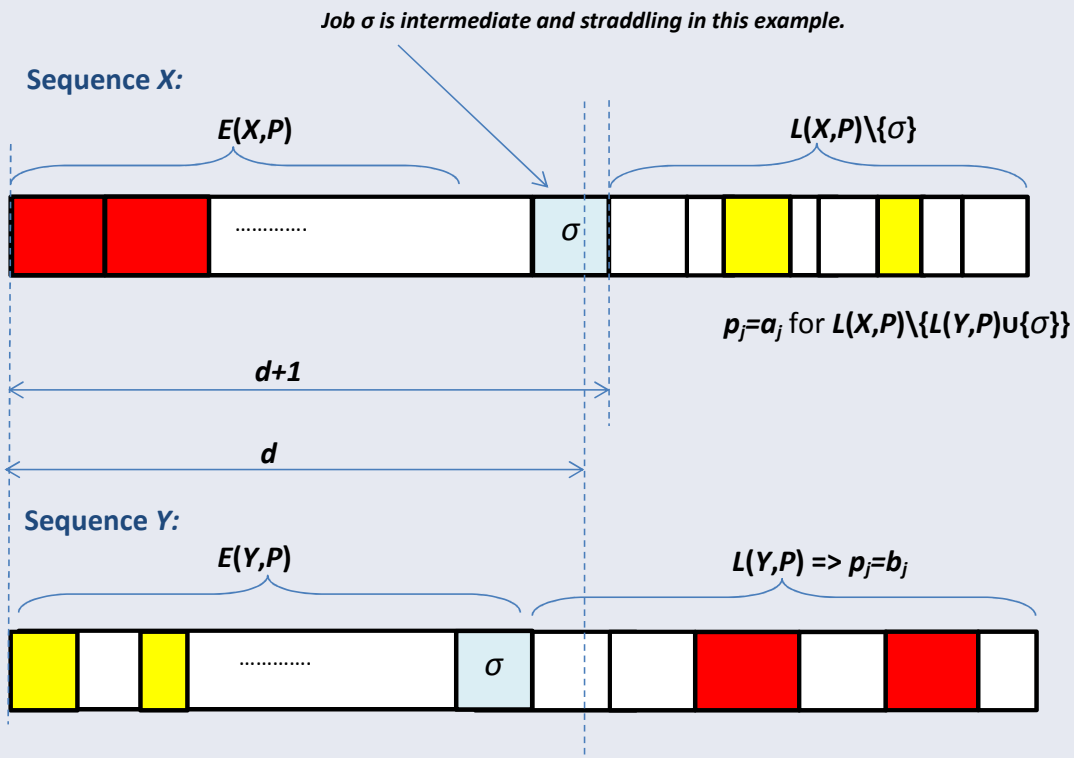


Figure: Illustration of the studied problem

Some key related works

- Furini, Iori, Martello and Yagiura: the behavior of classical combinatorial optimization approaches when adapted to the solution of the interval minmax regret knapsack problem.
- The first paper on the approximation of minmax regret on combinatorial optimization problems by Aissi, Bazgan and Vanderpooten. They have also written a survey on minmax regret of combinatorial optimization problems.
- Minmax regret and scheduling: minimizing the weighted sum completion times, by Lebedev and Averbakh for complexity results for this problem. A recent survey on minmax regret and scheduling by Kasperski and Zieli.
- Several minmax regret versions of minimizing the weighted number of late jobs: Most results were developed under the more restrictive assumption that the uncertainty can be modeled by a discrete set of parameter values. In the discrete uncertainty scenario the scenario set is finite (explicitly listed scenarios).

Some key related works

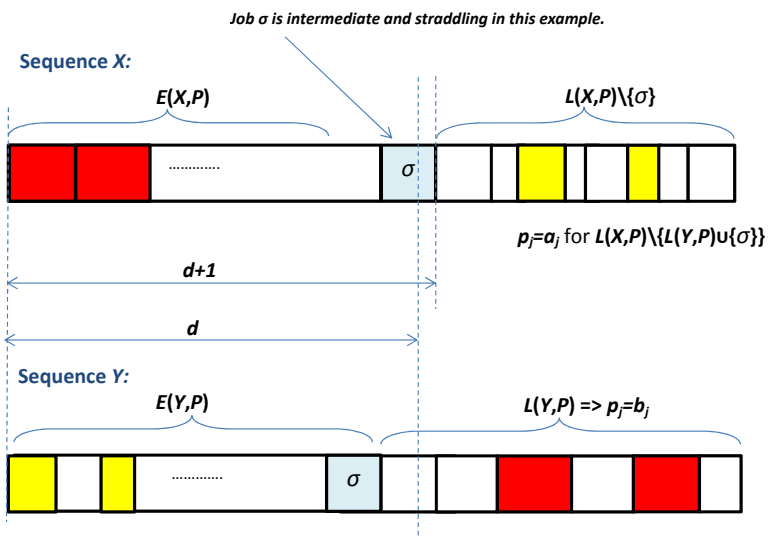
- Discrete scenario: Aloulou and Della Croce: $1|\cdot|\sum U_j$ is NP-hard if the job due dates are deterministic and there are only two processing time scenarios.
- Aissi, Aloulou and Kovalyov showed that $1|p_j = 1|\sum U_j$ is strongly NP-hard and not approximable with a ratio less than 2 if the number of due dates scenarios is unbounded. The minmax regret version of $1|p_j = 1, d_j = d|\sum w_j U_j$ is equivalent to the *Minmax Selecting Items Problem*.
- This problem was first discussed by Averbakh: the problem is even NP-hard for two weight scenarios.
- A few results have been obtained for the interval data scenario. Polynomial algorithms for the minmax regret selecting items with interval weights have been found by Averbakh and Conde.
- For $1|p_j = 1|\sum w_j U_j$ with interval weights Kasperski found a 2-approximation. Only one paper deals with interval processing times by Drwal (heuristics based on integer programming).

Lemma

(a) *There is an optimal scenario for Bob's problem for which the jobs in* $L(Y, P)$ *take maximal processing times.*

(b) *There is an optimal scenario for Bob's problem for which the jobs in* $L(X, P) \setminus \{\sigma\}$ *take minimal processing times.*

(c) *There is an optimal scenario for Bob's problem for which the jobs in* $L(Y, P)$ *take maximal processing times and the jobs in* $L(X, P) \setminus \{L(Y, P) \cup \{\sigma\}\}$ *take minimal processing times.*



Lemma

A job which does not take the minimal or the maximal processing time for a given scenario P is called intermediate. There is always an optimal scenario P for Bob's problem which has the following properties:

(a) *P contains at most one intermediate job* $i \in E(Y,P) \cap (E(X,P) \cup \{\sigma\})$.

(b) *This intermediate job can be chosen as*

$$i = \arg\max\{b_j - a_j | j \in E(Y,P) \cap (E(X,P) \cup \{\sigma\}\}.$$

(c) *The processing times* p_j *in P can always be chosen to be integers.*

(d) *If the intermediate job i is straddling in Alice's sequence, then we can choose* p_i *such that* $p_i + p(E(X,P)) = d + 1$.

Special cases

If $a_i \leq a_j$ and $b_i \leq b_j$ for a pair of jobs i, j, we say that job i *dominates* job j. Thus, for an arbitrary pair of jobs i, j exactly one of the following two conditions holds:

(i) One job dominates the other job.

(ii) One job interval is a proper subinterval of the other job interval.

If (i) holds for all job pairs, i.e., there is an ordering of the jobs such that $a_1 \leq a_2 \leq \ldots \leq a_n$ and $b_1 \leq b_2 \leq \ldots \leq b_n$, we say that the jobs are *totally ordered*.

If (ii) holds for all job pairs, i.e., there is an ordering of the jobs such that $a_1 \leq a_2 \leq \ldots \leq a_n \leq b_n \leq b_{n-1} \leq \ldots \leq b_1$, the jobs are called *nested*.

If the job intervals are given as $[a_j, b_j] = [C - \delta_j, C + \delta_j], \quad j = 1, \ldots, n$, the jobs are called *centered with center* C.

Obviously, a centered job set is also nested.

Special cases

Lemma

Suppose job i dominates job j. There is always an optimal sequence for Alice's problem in which job i is processed before job j.

Assume j is processed before job i in an optimal sequence X^*. Consider sequence X which is obtained by interchanging jobs i and j and leaving the positions of the other jobs unchanged. Let $P = (p_1, \ldots, p_n)$ and Y be the optimal scenario and the optimal sequence for sequence X, respectively.
If $p_i < p_j$, choose scenario P and sequence Y also for sequence X^*. The jobs between j and i will finish later for sequence X^* than for sequence X. Thus, $F(X^*, P) \leq F(X, P)$. This implies

$$Z(X^*) \geq R(X^*, Y, P) \geq R(X, Y, P) = Z(X).$$

Special cases

Suppose now that $p_i \geq p_j$. Consider the scenario $P' = (p'_1, \ldots, p'_n)$ with $p'_i = p_j$, $p'_j = p_i$ and leaving the other processing times unchanged as in scenario P. This scenario is feasible since i dominates j. The sequence Y' is constructed from sequence Y by interchanging jobs i and j and leaving the positions of the other jobs unchanged. Choose scenario P' and sequence Y' for sequence X^*. Since jobs i, j have exchanged their positions and their processing times from sequence Y to Y', the completion times of the jobs in Y' for scenario P' are identical to the completion times of the jobs in Y for scenario P. The same holds for the completion times of the jobs in X with scenario P and the jobs in X^* with scenario P'. Consequently, $F(X^*, P') = F(X, P)$ and $F(Y', P') = F(Y, P)$. Thus, $Z(X^*) \geq Z(X)$. □

Theorem

If the jobs are totally ordered, the total ordering gives an optimal sequence for Alice's problem.

Equal Cardinality Partition

- Equal Cardinality Partition: Given n positive integers s_j with n even such that $\sum_{j=1}^{n} s_j = 2S$, does there exist a partition of the index set $N = \{1, 2, \ldots, n\}$ into two subsets N_1 and N_2 with equal cardinalities such that $\sum_{j \in N_1} s_j = \sum_{j \in N_2} s_j = S$?
- It is folklore to observe that Equal Cardinality Partition is equivalent to the standard 2-Partition problem, which means that the two problems are NP-complete in the ordinary sense.

Theorem

Bob's problem is NP-hard in the ordinary sense.

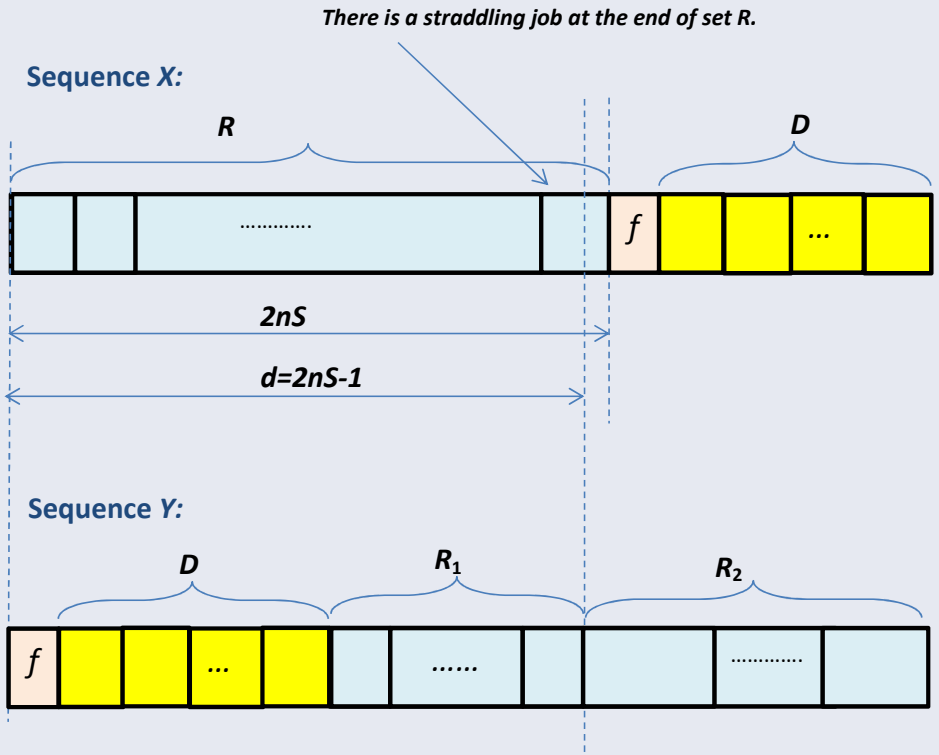
Bob's problem complexity

Given an instance of Equal Cardinality Partition, define the following instance of Bob's problem:

- a set R of n *regular* jobs j with $a_j = 2S - s_j$ and $b_j = 2S + s_j$, $j = 1, \ldots, n$,
- a set D of $\frac{n}{2}$ identical *dummy* jobs with $a = 2S - 1$ and $b = 2nS$,
- a small dummy job f with $a = S + n/2 - 1$ and $b = 2nS$,
- a common due date $d = 2nS - 1$,
- an arbitrary sequence X given by Alice which processes first the regular jobs and then the dummy jobs.

We claim that Equal Cardinality Partition has a solution if and only if Bob finds a sequence Y and a scenario P with $F(Y, P) - F(X, P) \geq 2$.

Complexity of Bob's problem



Theorem

Bob's problem inapproximability. Let $\rho > 1/2$. Finding a ρ-approximation algorithm for Bob's problem is not possible unless $P = NP$. In particular, there is no PTAS for Bob's problem.

Let us consider an instance of Equal Cardinality Partition (ECP) for which the answer is *YES*. Based on it, we define exactly the same instance of Bob's problem, denoted as I_B, as we did in the proof of the previous theorem. For this special instance I_B, we know that Bob has a sequence Y^* and a scenario P^* with the best possible value for him $R(X, Y^*, P^*) = F(Y^*, P^*) - F(X, P^*) = 2$. If there is a ρ-approximation algorithm H for Bob's problem such that $\rho > 1/2$, then this algorithm will return a sequence $Y^H(X)$ with a regret value $R(X, Y^H(X), P^H(X)) \geq \rho \cdot R(X, Y^*, P^*) > (1/2) \cdot 2 > 1$, which means that $R(X, Y^H(X), P^H(X)) \geq 2 = R(X, Y^*, P^*)$. Then, algorithm H finds an optimal solution for instance I_B, which means that it is able to identify every *YES*-instance of ECP in polynomial time.

Dynamic Programming exists $O(n^2 d)$

We assume in the special context of Bob's problem that Alice's sequence X is renumbered so that $X = (1, 2, \ldots, n)$. The algorithm will generate a set $\mathcal{X}_j$ of states at every iteration j, $j = 0, 1, \ldots, n$. Every state e in $\mathcal{X}_j$ is corresponding to a partial schedule of Bob's problem. This partial schedule $Y(e)$ is associated to the decisions taken for solving Bob's problem on the first j jobs $\{1, 2, \ldots, j\}$ given in Alice's sequence. We represent every state $e \in \mathcal{X}_j$ by four decision variables and parameters: $e = [t, t', \alpha, f]$, where

- t is the total processing time of early jobs in $Y(e)$.
- t' is the total processing time of tardy jobs in $Y(e)$.
- α is a binary variable indicating whether a job is intermediate or not, i.e., $\alpha = 1$ means that there is an intermediate job among the the first j jobs $\{1, 2, \ldots, j\}$ and $\alpha = 0$ otherwise.
- f is the objective function value associated to e, i.e., the number of early jobs in $Y(e)$ minus the number of early jobs among $\{1, 2, \ldots, j\}$ in Alice's sequence X. Here, f can also be negative.

Dynamic Programming exists in $O(n^2 d)$

At every iteration j, Bob has to fix the processing time of job j and to assign it to the early part of his sequence Y or to its tardy part. By the dominance rules we can restrict ourselves to the following possibilities:

1. Job j is early in Bob's sequence and has minimal processing time, i.e., $j \in E(Y, P)$ and $p_j = a_j$.
2. Job j is early in Bob's sequence and has maximal processing time, i.e., $j \in E(Y, P)$ and $p_j = b_j$.
3. Job j is late in Bob's sequence and has maximal processing time, i.e., $j \in L(Y, P)$ and $p_j = b_j$.
4. At most one job j is intermediate. Then, job j is early in Bob's sequence and is early in Alice's sequence or $j = \sigma$.

When Bob decides that job j is intermediate, by the dominance properties, p_j can be taken from the set $\{a_j + 1, a_j + 2, ..., b_j - 1\}$ and α passes from 0 to 1. Moreover, if $j = \sigma$, we have $t + p_j + t' = d + 1$.

Refined Equal Cardinality Partition - RECP

- We are given n integers s_j with n even such that $\sum_{j=1}^{n} s_j = 2S$, and again we ask whether there exists a partition of the index set $N = \{1, 2, \ldots, n\}$ into two subsets N_1 and N_2 with equal cardinalities such that $\sum_{j \in N_1} s_j = \sum_{j \in N_2} s_j = S$. W.l.o.g. let the integers $s_1, \ldots, s_n$ be sorted in non-increasing order, i.e., $s_1 \geq s_2 \geq \ldots \geq s_n$ and S be an even number. For defining (RECP) the following two additional assumptions are made. $\sum_{j=n/2+1}^{n} s_j > \frac{S}{2}$ and $s_1 \leq \frac{S}{2}$.
- Note that it can be easily seen that (RECP) is NP-complete by using the instance of n integers $L + s_1, \ldots, L + s_n$ (with L being a sufficiently large integer) and reducing it from ECP.

Theorem

Alice's problem is NP-hard in the ordinary sense.

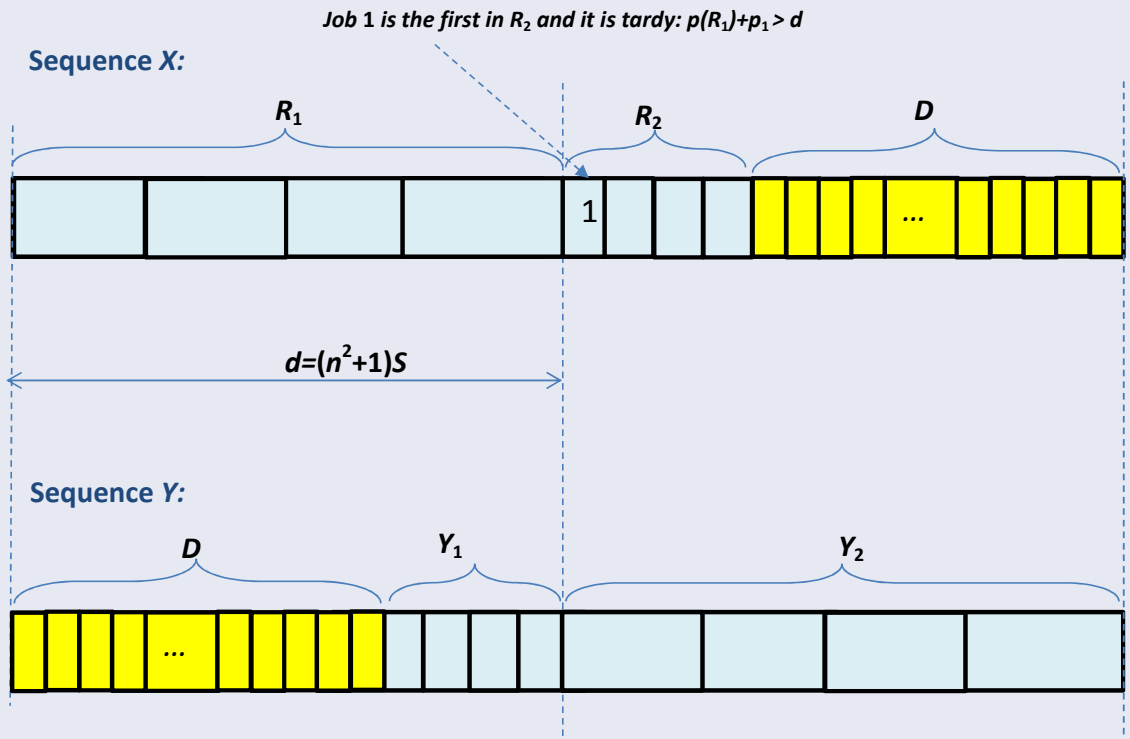
Proof principle

We are given an instance of (RECP) with positive integers $s_1, \ldots, s_n$ such that $\sum_{j=1}^{n} s_j = 2S$ and $s_1 \geq s_2 \geq \ldots \geq s_n$. Define the following instance of Alice's problem:

- a set R of n *regular* jobs j with $a_j = S - s_j$ and $b_j = 2nS + s_j$, $j = 1, \ldots, n$,
- a set D of $2n^2 - n + 4$ *dummy* jobs which consists of $2n^2 - n + 3$ identical jobs with $a = \frac{S}{2}$ and $b = 2n^2 S$ and one dummy job with $a = \frac{S}{2} + 1$ and $b = 2n^2 S$.
- a common due date $d = (n^2 + 1)S$.

It is sufficient to take S even to guarantee that all processing times are integer. We claim that (RECP) has a solution if and only if there is sequence X such that $Z(X) \leq 2n^2 - n + 3 \quad$ for all $X \in \Pi$.

Complexity of Alice's problem



Theorem

Alice's problem inapproximability. Finding an FPTAS for Alice's problem is not possible unless $P = NP$.

Proof principle

Assume that an FPTAS exists for Alice's problem. Then, we can ensure the existence of a $(1 + 1/2n)$-approximation in polynomial time in n. If F_A and OPT_A denote respectively the value given by the FPTAS and the value of an optimal solution of Alice's problem for a given instance, then $F_A \leq (1 + 1/2n)OPT_A = OPT_A + OPT_A/2n$. Since $OPT_A \leq n$, it can be deduced that $OPT_A/2n \leq 1/2$. Thus, we have that $F_A \leq OPT_A + 1/2 < OPT_A + 1$. Since the objective function value is integer, we get $F_A = OPT_A$, which is impossible unless $P = NP$ (since Alice's problem is NP-hard).

Main results

- Our problem is particularly difficult since a small change of the processing times can change the feasibility of the selected items.
- For totally ordered jobs, we could find an optimal sorting.
- We presented a pseudopolynomial algorithm for Bob's problem, and we proved that there is no possible approximation for it with a performance ratio better than $1/2$.
- Alice's problem is NP-hard and it has an FPTAS.

Perspectives

- Pseudopolynomial algorithm for Alice's problem?
- We conjecture that there is no constant approximation for Alice's problem, and that there is a heuristic H with asymptotic performance guarantee, with small constant values of ρ and β:

$$R(X^H, Y^*(X^H), P^*(X^H)) \leq (1+\rho)\, R(X^*, Y^*(X^*), P^*(X^*)) + \beta$$

THANK YOU FOR YOUR ATTENTION

Questions?

Reference: I. Kacem, H. Kellerer. Complexity Results for Common Due Date Scheduling Problems with Interval Data and Minmax Regret Criterion.
Published in: Discrete Applied Mathematics (ELSEVIER), July 2019, DOI: https://doi.org/10.1016/j.dam.2018.09.026
Presented at: ROADEF2020, Montpellier, February 2020.

Contact: imed.kacem@univ-lorraine.fr