Online Linear Optimization for Job Scheduling Under Precedence Constraints

Fujita, Takahiro
Department of Informatics, Kyushu University

Hatano, Kohei
九州大学附属図書館

Kijima, Shuji
Department of Informatics, Kyushu University

Takimoto, Eiji
Department of Informatics, Kyushu University

出版情報：Algorithmic Learning Theory : 26th International Conference, ALT 2015, proceedings, pp.332-346, 2015. Springer
バージョン：
権利関係：

Online Linear Optimization for Job Scheduling under Precedence Constraints

Takahiro Fujita[1], Kohei Hatano[1], Shuji Kijima[1], and Eiji Takimoto[1]
{takahiro.fujita,hatano,kijima,eiji}@inf.kyushu-u.ac.jp[1]

Department of Informatics, Kyushu University

Abstract. We consider an online job scheduling problem on a single machine with precedence constraints under uncertainty. In this problem, for each trial $t = 1, \ldots, T$, the player chooses a total order (permutation) of n fixed jobs satisfying some prefixed precedence constraints. Then, the adversary determines the processing time for each job, 9 and the player incurs as loss the sum of the processing time and the waiting time. The goal of the player is to perform as well as the best fixed total order of jobs in hindsight. We formulate the problem as an online linear optimization problem over the permutahedron (the convex hull of permutation vectors) with specific linear constraints, in which the underlying decision space is written with exponentially many linear constraints. We propose a polynomial time online linear optimization algorithm; it predicts almost as well as the state-of-the-art offline approximation algorithms do in hindsight. In preliminary experiments, our algorithm runs considerably faster than the alternatives while performing competitively.

1 Introduction

Job scheduling is a fundamental problem in the field of computer science and operations research, and it has been studied extensively for decades. It has broad applications in operating systems, assignments of tasks to workers, manufacturing systems, and many other areas.

Studies of how to schedule jobs that use a single machine under precedence constraints is well studied in the mathematical programming literature. More precisely, assume that there are n fixed jobs and a single processor. Let $[n] = \{1, \ldots, n\}$ be the set of jobs. Each job i needs *processing time* ℓ_i to be completed by the processor. A schedule is a permutation over n jobs, and the processor does the jobs sequentially according to the schedule. For a given schedule, the *completion time* of job i is the sum of the waiting time (the sum of the processing times of the jobs finished before completing job i) and the processing time of job i. There are precedence constraints over n jobs. For example, job 1 needs to be completed before job 3, and job 2 needs to be completed before job 5. The constraints are represented as a set of binary relations $\mathcal{A} \subset [n] \times [n]$, e.g., $\mathcal{A} = \{(1, 3), (2, 5)\}$. Given the processing time of n jobs and the precedence constraints $\mathcal{A}$, the typical goal is to find a schedule that minimizes the sum of the (weighted) completion times of the n jobs, subject to the constraints $\mathcal{A}$. This

problem is categorized as $1|prec|\sum_j C_j$ in the literature[1]. It is known that this problem is NP-hard [18,19].

For further developments, see, e.g., [2,9]. Several 2-approximation algorithms have been proposed for the offline setting [6, 7, 14, 22, 25] and a stochastic setting [24, 26]. In this paper, we consider a different scenario for the classical problem. What if the processing time of each job is unknown when we determine the schedule? This question is quite natural in many application areas that cope with uncertainty and in which the processing time of each job is uncertain or varies in time. It is impossible to solve this problem directly without knowing the processing time, so we consider an iterative scenario. Each day $t = 1, \ldots, T$, we determine a total order of n fixed jobs satisfying some prefixed precedence constraints. Then, after processing all n jobs according to the schedule, the processing time of each job is revealed. The goal is to minimize *the sum of the completion times* over all jobs and all T days under fixed precedence constraints, where the completion time of job i at day t is the sum of processing times of all jobs prior to i and the processing time of job i.

Now, let us formulate the problem in a formal way. A permutation σ is a bijection from $[n]$ to $[n]$. Another representation of a permutation $\boldsymbol{\sigma}$ over the set $[n]$ is a vector in $[n]^n$, defined as $\boldsymbol{\sigma} = (\sigma(1), \ldots, \sigma(n))$, which corresponds to σ. For example, $(3, 2, 1, 4)$ is a representation of a permutation for $n = 4$. The vector representation of permutations is convenient, since the sum of the completion times of the jobs according to some permutation $\boldsymbol{\sigma}$ is expressed as the inner product $\boldsymbol{\sigma} \cdot \boldsymbol{\ell}$, where $\boldsymbol{\ell}$ is the vector consisting of the processing times of the jobs. For example, there are 4 jobs to be processed according to the order $4, 1, 2, 3$. Each processing time is given as $\boldsymbol{\ell} = (\ell_1, \ell_2, \ell_3, \ell_4)$. The completion times of jobs $i = 4, 1, 2, 3$ are ℓ_4, $\ell_4 + \ell_1$, $\ell_4 + \ell_1 + \ell_2$, and $\ell_4 + \ell_1 + \ell_2 + \ell_3$, respectively, and the sum of completion times is $4\ell_4 + 3\ell_1 + 2\ell_2 + \ell_3$. Note that the completion time exactly corresponds to $\boldsymbol{\sigma} \cdot \boldsymbol{\ell}$, the inner product of $\boldsymbol{\ell}$ and the permutation vector $\boldsymbol{\sigma}_{=}(3, 2, 1, 4)$. Here, component σ_i of the permutation $\boldsymbol{\sigma}$ represents the priority of job i.

Let S_n be the set of all permutations over $[n]$, i.e., $S_n = \{\boldsymbol{\sigma} \in [n]^n | \ \boldsymbol{\sigma}$ is a permutation over $[n]\}$. In particular, the convex hull of all permutations is called the permutahedron, denoted as P_n. The set $\mathcal{A}$ of precedence constraints is given as $\mathcal{A} = \{(i_k, j_k) \in [n] \times [n] \mid i_k \neq j_k, k = 1, \ldots, m\}$, meaning that object i_k is preferred to object j_k. The set $\mathcal{A}$ induces the set defined by the linear constraints $\mathrm{Precons}(\mathcal{A}) = \{\boldsymbol{p} \in \mathbb{R}^n_+ \mid p_i \geq p_j \ \text{ for } (i, j) \in \mathcal{A}\}$. We further assume that there exists a linear ordering consistent with $\mathcal{A}$. In other words, we assume there to exist a permutation $\sigma \in S_n \cap \mathrm{Precons}(\mathcal{A})$.

The online job scheduling problem can be formulated as the following online linear optimization problem over $S_n \cap \mathrm{Precons}(\mathcal{A})$. For each trial $t = 1, \ldots, T$, (i) the player predicts a permutation $\boldsymbol{\sigma}_t \in S_n \cap \mathrm{Precons}(\mathcal{A})$, (ii) the adversary returns a loss vector $\boldsymbol{\ell}_t \in [0, 1]^n$, and (iii) the player incurs loss $\boldsymbol{\sigma}_t \cdot \boldsymbol{\ell}_t$. The goal

[1] The weighted version is known as $1|prec|\sum_j w_j C_j$.

of the player is to minimize the α-regret for some small $\alpha \geq 1$:

$$\alpha\text{-Regret} = \sum_{t=1}^{T} \boldsymbol{\sigma}_t \cdot \boldsymbol{\ell}_t - \alpha \min_{\boldsymbol{\sigma} \in S_n \cap \mathrm{Precons}(\mathcal{A})} \sum_{t=1}^{T} \boldsymbol{\sigma} \cdot \boldsymbol{\ell}_t.$$

In this paper, we propose an online linear optimization algorithm over $P_n \cap \mathrm{Precons}(\mathcal{A})$ whose the α-regret is $O(n^2\sqrt{T})$ for $\alpha = 2 - 2/(n+1)$. For each trial, our algorithm runs in polynomial time in n and m. More precisely, the running time at each trial is $O(n^4)$. Further, we show that a lower bound of the 1-regret is $\Omega(n^2\sqrt{T})$.

In addition, we prove that there is no polynomial time algorithm with α-regret $poly(n, m)\sqrt{T}$ with $\alpha < 2 - 2/(n+1)$ unless there exists a randomized approximation algorithm with approximation $\alpha < 2 - 2/(n+1)$ for the corresponding offline problem. Thus far, the state-of-the-art approximation algorithms have an approximation ratio $2 - 2/(n+1)$, and it is a longstanding open problem to find an approximation algorithm with a better ratio [30]. It has been determined that there is no polynomial-time $(1+\varepsilon)$-approximation algorithm (PTAS) for any constant $\varepsilon > 0$ under some standard assumption of the complexity theory [3]. Therefore, the regret bound is optimal among any polynomial algorithms unless there exists a better approximation algorithm for the offline problem.

Note that our online algorithm is deterministic, so some reader might worry that the algorithm works without any randomization. We show that our problem can be reduced to online problem over continuous space and rounding problem. For the online algorithm, some deterministic algorithms are known and can achieve a good regret bound. Therefore, there is no reason that our algorithm is stochastic.

2 Related Research

There has been related research on the online prediction of permutations. The earliest approach was to directly design online prediction algorithms for permutations. Helmbold and Warmuth [15] were the first to do this, and in their setting, a permutation is given as a permutation matrix, which is a more generic expression than a permutation vector (i.e., permutation matrices can encode permutation vectors). Thus, their algorithm can be used for our problem without precedence constraints. Yasutake et al. [31] proposed an online linear optimization algorithm over the permutahedron when there are no precedence constraints. Ailon proposed another online optimization algorithm with improved regret bound and time complexity [1]. Suehiro et al. [28] extended the result of Yasutake et al. [31] to the submodular base polyhedron; this can be used not only for permutations, but also for other combinatorial objects, such as spanning trees. These algorithms, however, are not designed for precedence-constrained problems.

The second approach is to transform an offline algorithm to an online optimization algorithm. By using the conversion method of Kakade et al. [16] or

that of Fujita et al. [13], we can construct online optimization algorithms with α-regret that are similar to ours.

However, with the method of Kakade et al. [16], the resulting time complexity per trial is linear in T, which is not desirable. For the method of Fujita et al., the α-regret is proved to be $\alpha = 2 - 2/(n-1) + \varepsilon$, which is slightly inferior to that of our method. The running time per trial is $poly(n, 1/\varepsilon)$, which is independent of T but depends on $1/\varepsilon$. Also, Fujita et al. showed that if an offline algorithm use an LP-relaxation and a metarounding, then FPL with the algorithm can achieve good bounds on the α-regret. But, the LP-relaxation based algorithm of previous work cannot be directly applied to our problem, because its rounding algorithm is not metarounding.

Our approach is completely different from previous offline algorithms. The known offline approximation algorithms rely on formulations that use completion-time variables or linear-ordering variables. In the first formulation, n variables indicate the time at which the job is completed. In this case, the relaxed problem is formulated as a linear program with exponentially many constraints (see, e.g., [14, 25]). The problem can be approximately solved in polynomial time by using the ellipsoid method. In the second case, there are $\binom{n}{2}$ variables, which represent relative comparisons between pairs of jobs. The relaxed problem is also formulated as a linear program with $O(n^2)$ variables and $O(n^3)$ linear constraints (e.g., [7]). In both formulations, the set of linear constraints and associated rounding methods require knowledge of the processing times of the jobs, and these are not available in the online setting. Our approach uses some geometric properties of the permutahedron, and thus it is totally different from the previous approaches. As a result, our rounding algorithm does not require knowledge of the processing times of the jobs. Thus, our approach is suitable for the online problem. On the other hand, a shortcoming of our approach is that it is only able to deal with the online problem of minimizing the unweighted total sum of the completion times.

Online learning approaches for job scheduling problems are not new. In particular, Even-Dar et al. [10] considered an online optimization problem with global functions, and its applications schedule jobs for several machines in order to minimize the makespan (the time at which the final job is completed). Both the objectives and the techniques of Even-Dar et al. [10] are different from ours. For online multi-task learning, Lugosi et al. considered some class of constraints [20]. For some natural class of constraints, they showed that the online task can be reduced to the shortest path problem, which is efficiently solvable. However, the constraints in our setting are much complicated and their method does not seem to be applicable to our problem.

3 Online Linear Optimization Algorithm over the Permutations

In this section, we propose our algorithm PermLearnPrec and prove its α-regret bound. We will use the notion of the permutahedron. The permutahedron P_n

is the convex hull of the set of permutations S_n. It is known that P_n can be represented as the set of points $\boldsymbol{p} \in \mathbb{R}^n_+$ satisfying $\sum_{i\in S} p_i \leq \sum_{i=1}^{|S|}(n+1-i)$ for any $S \subset [n]$, and $\sum_{i=1}^{n} p_i = n(n+1)/2$. For further discussion of the permutahedron, see, e.g., [12, 32].

3.1 Main Structure

Our algorithm PermLearnPrec is shown as Algorithm 1. The algorithm maintains a weight vector $\boldsymbol{p}_t$ in $\mathbb{R}^n_+$, which represents a mixture of the permutations in S_n. At each trial t, it "rounds" (see next section for description) a vector $\boldsymbol{p}_t$ into a permutation $\boldsymbol{\sigma}_t$ so that $\boldsymbol{\sigma}_t \leq \alpha \boldsymbol{p}_t$ for some $\alpha \geq 1$. After the loss vector $\boldsymbol{\ell}_t$ is given, PermLearnPrec updates the weight vector $\boldsymbol{p}_t$ in an additive way and successively projects it onto the set of linear constraints representing the precedence constraints Precons($\mathcal{A}$) and the intersection of the permutahedron P_n and Precons($\mathcal{A}$).

The main structure of our algorithm is built on a standard online convex optimization algorithm known as online gradient descent (OGD) [33]. OGD consists of an additive update of the weight vectors, followed by projection to some convex set of interest. In our case, the convex set is $P_n \cap \mathrm{Precons}(\mathcal{A})$. Using these procedures, the regret bound of OGD can be proved to be $O(n^2\sqrt{T})$. Thus, the successive projections are apparently redundant, and only one projection to $P_n \cap \mathrm{Precons}(\mathcal{A})$ would suffice. However, the projection onto $P_n \cap \mathrm{Precons}(\mathcal{A})$ is not known to be tractable, and it contains exponentially many linear constraints. Thus, we take a different approach. Instead of performing the projection directly, we use successive projections onto Precons($\mathcal{A}$) and $P_n \cap \mathrm{Precons}(\mathcal{A})$. Below, we will show that these successive projections are the key to an efficient implementation of our algorithm. First, we will prove an α-regret bound of the proposed algorithm, and then we will show that our algorithm can be efficiently realized.

We begin our analysis of PermLearnPrec with the following lemma. The lemma guarantees the progression of $\boldsymbol{p}_t$ towards any vector in $P_n \cap \mathrm{Precons}(\mathcal{A})$, as measured by the Euclidean norm squared.

Lemma 1 *For any $\boldsymbol{q} \in P_n \cap \mathrm{Precons}(\mathcal{A})$ and for any $t \geq 1$,*

$$\|\boldsymbol{q} - \boldsymbol{p}_t\|_2^2 - \|\boldsymbol{q} - \boldsymbol{p}_{t+1}\|_2^2 \geq 2\eta(\boldsymbol{q} - \boldsymbol{p}_t) \cdot \boldsymbol{\ell}_t - \eta^2 \|\boldsymbol{\ell}_t\|_2^2.$$

Proof. By using the generalized Pythagorean theorem (e.g., [5]),

$$\|\boldsymbol{q} - \boldsymbol{p}_{t+\frac{2}{3}}\|_2^2 \geq \|\boldsymbol{q} - \boldsymbol{p}_{t+1}\|_2^2 + \|\boldsymbol{p}_{t+1} - \boldsymbol{p}_{t+\frac{2}{3}}\|_2^2$$

and

$$\|\boldsymbol{q} - \boldsymbol{p}_{t+\frac{1}{3}}\|_2^2 \geq \|\boldsymbol{q} - \boldsymbol{p}_{t+\frac{2}{3}}\|_2^2 + \|\boldsymbol{p}_{t+\frac{2}{3}} - \boldsymbol{p}_{t+\frac{1}{3}}\|_2^2.$$

By combining these, we obtain

$$\begin{aligned}
&\|\boldsymbol{q} - \boldsymbol{p}_t\|_2^2 - \|\boldsymbol{q} - \boldsymbol{p}_{t+1}\|_2^2 \\
&\geq \|\boldsymbol{q} - \boldsymbol{p}_t\|_2^2 - \|\boldsymbol{q} - \boldsymbol{p}_{t+\frac{1}{3}}\|_2^2 + \|\boldsymbol{p}_{t+1} - \boldsymbol{p}_{t+\frac{2}{3}}\|_2^2 + \|\boldsymbol{p}_{t+\frac{2}{3}} - \boldsymbol{p}_{t+\frac{1}{3}}\|_2^2 \\
&\geq \|\boldsymbol{q} - \boldsymbol{p}_t\|_2^2 - \|\boldsymbol{q} - \boldsymbol{p}_{t+\frac{1}{3}}\|_2^2, \qquad (1)
\end{aligned}$$

Algorithm 1 PermLearnPrec

Input: parameter $\eta > 0$.

1. Let $\boldsymbol{p}_1 = ((n+1)/2, \ldots, (n+1)/2) \in [0,n]^n$.
2. For $t = 1, \ldots, T$
   (a) (Rounding) Run **Rounding**$(\boldsymbol{p}_t)$ and get $\boldsymbol{\sigma}_t \in S_n$ such that $\boldsymbol{\sigma}_t \leq (2 - 2/(n+1))\boldsymbol{p}_t$.
   (b) Incur a loss $\boldsymbol{\sigma}_t \cdot \boldsymbol{\ell}_t$.
   (c) Update $\boldsymbol{p}_{t+\frac{1}{3}}$ as $\boldsymbol{p}_{t+\frac{1}{3}} = \boldsymbol{p}_t - \eta\boldsymbol{\ell}_t$.
   (d) (1st projection) Let $\boldsymbol{p}_{t+\frac{2}{3}}$ be the Euclidean projection onto the set Precons$(\mathcal{A})$, .i.e.,
   $$\boldsymbol{p}_{t+\frac{2}{3}} = \arg\min_{\boldsymbol{p} \in \mathrm{Precons}(\mathcal{A})} \|\boldsymbol{p} - \boldsymbol{p}_{t+\frac{1}{3}}\|_2^2.$$
   (e) (2nd projection) Let $\boldsymbol{p}_{t+1}$ be the projection of $\boldsymbol{p}_{t+\frac{2}{3}}$ onto the set $P_n \cap$ Precons$(\mathcal{A})$, that is,
   $$\boldsymbol{p}_{t+1} = \arg\min_{\boldsymbol{p} \in P_n \cap \mathrm{Precons}(\mathcal{A})} \|\boldsymbol{p} - \boldsymbol{p}_{t+\frac{2}{3}}\|_2^2.$$

where the last inequality follows since Euclidean distance is nonnegative.

Then, because $\boldsymbol{p}_{t+\frac{1}{3}} = \boldsymbol{p}_t - \eta\boldsymbol{\ell}_t$, the right-hand side of inequality (1) is

$$\|\boldsymbol{q} - \boldsymbol{p}_t\|_2^2 - \|\boldsymbol{q} - \boldsymbol{p}_{t+\frac{1}{3}}\|_2^2 = 2\eta(\boldsymbol{q} - \boldsymbol{p}_t) \cdot \boldsymbol{\ell}_t - \eta^2\|\boldsymbol{\ell}_t\|_2^2. \tag{2}$$

By combining (1) and (2), we complete the proof.

Lemma 2 (Cf. Zinkevich [33]) *For any $T \geq 1$ and $\eta = (n+1)/(2\sqrt{T})$,*

$$\sum_{t=1}^{T} \boldsymbol{p}_t \cdot \boldsymbol{\ell}_t \leq \min_{\boldsymbol{p} \in P_n \cap \mathrm{Precons}(\mathcal{A})} \sum_{t=1}^{T} \boldsymbol{p} \cdot \boldsymbol{\ell}_t + \frac{n(n+1)}{2}\sqrt{T}.$$

Proof. By Lemma 1, summing the inequality from $t = 1$ to T and rearranging, we obtain that for any $\boldsymbol{q} \in P_n \cap$ Precons$(\mathcal{A})$,

$$\begin{aligned}
\sum_{t=1}^{T} (\boldsymbol{p}_t - \boldsymbol{q}) \cdot \boldsymbol{\ell}_t &\leq \frac{1}{2\eta} \sum_{t=1}^{T} (\|\boldsymbol{q} - \boldsymbol{p}_t\|_2^2 - \|\boldsymbol{q} - \boldsymbol{p}_{t+1}\|_2^2) + \frac{\eta}{2} \sum_{t=1}^{T} \|\boldsymbol{\ell}_t\|_2^2 \\
&= \frac{1}{2\eta}(\|\boldsymbol{q} - \boldsymbol{p}_1\|_2^2 - \|\boldsymbol{q} - \boldsymbol{p}_T\|_2^2) + \frac{\eta}{2} \sum_{t=1}^{T} \|\boldsymbol{\ell}_t\|_2^2 \\
&\leq \frac{1}{2\eta} n (\frac{n+1}{2})^2 + \frac{\eta}{2} nT,
\end{aligned}$$

where the last inequality holds since for any $i \in [n]$, $(q_i - p_{i,1})^2$ is at most $p_{1,i}^2 = (\frac{n+1}{2})^2$, and $\boldsymbol{\ell}_t \in [0,1]^n$. By setting $\eta = (n+1)/(2\sqrt{T})$, we have the cumulative loss bound as desired.

4 Efficient Implementations of Projection and Rounding

In this section, we propose efficient algorithms for successive projections onto Precons($\mathcal{A}$) and $P_n \cap$ Precons($\mathcal{A}$). We then show an implementation of the procedure Rounding.

4.1 Projection onto the Set Precons($\mathcal{A}$) of the Precedence Constraints

The problem of projection onto Precons($\mathcal{A}$) is described as follows:

$$\begin{aligned} &\min_{\boldsymbol{p}\in\mathbb{R}^n} \|\boldsymbol{p}-\boldsymbol{q}\|_2^2 \\ &\text{sub.to: } p_i \geq p_j, \quad \text{for } (i,j)\in\mathcal{A}. \end{aligned}$$

This problem is known as the isotonic regression problem [21, 23, 27]. Previously known algorithms for the isotonic regression run in $O(mn^2 \log n)$ or $O(n^4)$ time (see [21] for details), where $m = |\mathcal{A}|$.

4.2 Projection of a point in Precons($\mathcal{A}$) onto $P_n \cap$ Precons($\mathcal{A}$)

In this subsection, we show an efficient algorithm, which we will call Projection, for computing the projection a point in Precons($\mathcal{A}$) onto the intersection of the permutahedron P_n and the set Precons($\mathcal{A}$) of the precedence constraints. In fact, we will show that the problem can be reduced to projection onto P_n only, and thus we can use the algorithm of Suehiro et al. [28] for finding the projection onto P_n. This is shown as Algorithm 2.

Formally, the problem is stated as follows:

$$\begin{aligned} &\min_{\boldsymbol{p}\in\mathbb{R}^n} \|\boldsymbol{p}-\boldsymbol{q}\|_2^2 \\ &\text{sub. to:} \sum_{j\in S} p_j \leq \sum_{j=1}^{|S|} (n+1-j), \text{ for any } S\subset[n], \\ &\sum_{j=1}^{n} p_j = \frac{n(n+1)}{2}, \\ &p_i \geq p_j, \quad \text{for } (i,j)\in\mathcal{A}. \end{aligned}$$

Without loss of generality, we may assume that elements in $\boldsymbol{q}$ are sorted in descending order, i.e., $q_1 \geq q_2 \geq \cdots \geq q_n$. This can be achieved in time $O(n \log n)$ by sorting $\boldsymbol{q}$. First, we show that this projection preserves the order in $\boldsymbol{q}$.

Lemma 3 (Order Preserving Lemma [28]) *Let $\boldsymbol{p}^*$ be the projection of $\boldsymbol{q}$ onto P_n s.t. $q_1 \geq q_2 \geq \cdots \geq q_n$. Then, the projection $\boldsymbol{p}^*$ also satisfies $p_1^* \geq p_2^* \geq \ldots \geq p_n^*$.*

Furthermore, we need to show that the projection onto P_n preserves equality, and this is guaranteed by the following lemma.

Lemma 4 (Equality Preserving Lemma) *Let $\boldsymbol{p}^*$ be the projection of $\boldsymbol{q}$ onto P_n. Then, the projection $\boldsymbol{p}^*$ satisfies $p_i = p_j$ if $q_i = q_j$.*

Proof. Assume that the lemma is false. Then, there exists a pair i and j such that $q_i = q_j$ and $p_i^* < p_j^*$. Let $\boldsymbol{p}'$ be the vector obtained by letting $p'_i = p'_j = (p_i^* + p_j^*)/2$ and $p'_k = p_k^*$ for $k \neq i, j$. It can be easily verified that $\boldsymbol{p}'s \in P_n$. Now, observe that

$$\begin{aligned}
\|\boldsymbol{p}^* - \boldsymbol{q}\|_2^2 - \|\boldsymbol{p}' - \boldsymbol{q}\|_2^2 =& p_i^{*2} + p_j^{*2} - p_i^{'2} - p_j^{'2} + 2\boldsymbol{p}' \cdot \boldsymbol{q} - 2\boldsymbol{p}^* \cdot \boldsymbol{q} \\
=& p_i^{*2} + p_j^{*2} - (p_i^* + p_j^*)^2/2 + 2(p'_i - p_i^*)q_i + 2(p'_j - p_j^*)q_j \\
=& \frac{1}{2}(p_i^* - p_j^*)^2 + 2(p'_i + p'_j - p_i^* - p_j^*)q_i \\
=& \frac{1}{2}(p_i^* - p_j^*)^2 > 0,
\end{aligned}$$

which contradicts the fact that $\boldsymbol{p}^*$ is the projection.

Now we are ready to show one of our main technical lemmas.

Lemma 5 *For any $\boldsymbol{q} \in \mathrm{Precons}(\mathcal{A})$,*

$$\arg\min_{\boldsymbol{p} \in P_n} \|\boldsymbol{p} - \boldsymbol{q}\| = \arg\min_{\boldsymbol{p} \in P_n \cap \mathrm{Precons}(\mathcal{A})} \|\boldsymbol{p} - \boldsymbol{q}\|.$$

Proof. Let $\boldsymbol{p}^* = \arg\min_{\boldsymbol{p} \in P_n} \|\boldsymbol{p} - \boldsymbol{q}\|$. By definition of the projection, for any $\boldsymbol{p} \in P_n \cap \mathrm{Precons}(\mathcal{A}) \subseteq P_n$, $\|\boldsymbol{p} - \boldsymbol{q}\| \geq \|\boldsymbol{p}^* - \boldsymbol{q}\|$. Further, by Lemmas 3 and 4, $\boldsymbol{p}^*$ preserves the order and equality in $\boldsymbol{q}$. That is, $\boldsymbol{p}^*$ also satisfies the constraints defined by $\mathrm{Precons}(\mathcal{A})$. Therefore, we have $\boldsymbol{p}^* \in \mathrm{Precons}(\mathcal{A})$. These facts imply that $\boldsymbol{p}^*$ is indeed the projection of $\boldsymbol{q}$ onto $P_n \cap \mathrm{Precons}(\mathcal{A})$.

So, by Lemma 2, when a vector $\boldsymbol{q} \in \mathrm{Precons}(\mathcal{A})$ is given, we can compute the projection of $\boldsymbol{q}$ onto $P_n \cap \mathrm{Precons}(\mathcal{A})$ by computing the projection of $\boldsymbol{q}$ onto P_n only. By applying the projection algorithm of Suehiro et al. [28] for the base polyhedron (which generalizes the permutahedron), we obtain the following result.

Theorem 1 *There exists an algorithm with input $\boldsymbol{q} \in \mathrm{Precons}(\mathcal{A})$ that outputs the projection of $\boldsymbol{q}$ onto $P_n \cap \mathrm{Precons}(\mathcal{A})$ in time $O(n^2)$ and space $O(n)$.*

4.3 Rounding

Algorithm 3 is Rounding. The algorithm is simple. Roughly speaking, if the input $\boldsymbol{p} \in P_n \cap \mathrm{Precons}(\mathcal{A})$ is sorted as $p_1 \geq \cdots \geq p_n$, the algorithm outputs $\boldsymbol{\sigma}$ such that $\sigma_1 \geq \cdots \geq \sigma_n$, i.e., $\boldsymbol{\sigma} = (n, n-1, \ldots, 1)$. Note that we need to break

Algorithm 2 Projection onto $P_n \cap \mathrm{Precons}(\mathcal{A})$

Input: $\boldsymbol{q} \in \mathrm{Precons}(\mathcal{A})$ s.t. $q_1 \geq q_2 \geq \cdots \geq q_n$.
Output: projection $\boldsymbol{x}$ of $\boldsymbol{q}$ onto P_n.

1. Let $i_0 = 0$.
2. **For** $k = 1, \ldots,$
   (a) Let $C^k(i) = \frac{g(i) - g(i_{k-1}) - \sum_{j=i_{k-1}+1}^{i} q_j}{i - i_{k-1}}$,
   where $g(i) = \sum_{j=1}^{i}(n+1-j)$,
   and $i_k = \arg\min_{i: i_{k-1}+1 \leq i \leq n} C^k(i)$;
   if there are multiple minimizers, choose the largest one as i_k.
   (b) Set $x_i = q_i + C^t(i_k)$ (for $i_{k-1} + 1 \leq i \leq i_k$).
   (c) **If** $i_k = n$, **then** break.
3. **Output** $\boldsymbol{x}$.

Algorithm 3 Rounding

Input: $\boldsymbol{p} \in P_n \cap \mathrm{Precons}(\mathcal{A})$ satisfying $p_1 \geq p_2 \geq \cdots \geq p_n$ and the transitive closure $\mathcal{A}^*$ of $\mathcal{A}$
Output: Permutation $\boldsymbol{\sigma} \in S_n \cap \mathrm{Precons}(\mathcal{A})$

1. Sort elements of $\boldsymbol{p}$ in the descending order, where for elements i, j such that $p_i = p_j$, i is larger than j if $(i, j) \in \mathcal{A}^*$, otherwise beak the tie arbitrarily.
2. Output the permutation $\boldsymbol{\sigma}$ s.t. $\sigma_i = (n+1) - r_i$, where r_i is the ordinal of i in the above order.

ties in $\boldsymbol{p}$ to construct $\boldsymbol{\sigma}$. Let $\mathcal{A}^*$ be the transitive closure of $\mathcal{A}$. Then, given an equivalence set $\{j \mid p_i = p_j\}$, we break ties so that if $(i, j) \in \mathcal{A}^*$, $\sigma_i \geq \sigma_j$. This can be done by, e.g., quicksort. First, we will show that Rounding guarantees that for each $i \in [n]$, $\sigma_i \leq (2 - 2/(n+1))p_i$, and then discuss its time complexity.

We prove the following lemma for Rounding.

Lemma 6 *For any $\boldsymbol{p} \in P_n \cap \mathrm{Precons}(\mathcal{A})$ s.t. $p_1 \geq \cdots \geq p_n$, given $\boldsymbol{p}$, the output $\boldsymbol{\sigma}$ of Rounding satisfies that for each $i \in [n]$, $\sigma_i \leq (2 - 2/(n+1))p_i$.*

Proof. For each $i \in [n]$, by definition of the permutahedron, we have

$$\sum_{j=1}^{i} p_j \leq \sum_{j=1}^{i-1} j = \frac{i(i-1)}{2}. \tag{3}$$

By the assumption that $p_1 \geq \cdots \geq p_n$, the average of $p_i + p_{i+1} + \cdots + p_n$ is not larger than p_i. Thus, we have

$$p_i \geq \frac{\sum_{j=i}^{n} p_j}{n+1-i} = \frac{\sum_{j=1}^{n} p_j - \sum_{j=1}^{i-1} p_j}{n+1-i} \geq \frac{(n+i)(n+1-i)}{2(n+1-i)} = \frac{n+i}{2},$$

where the second inequality follows from (3). Thus, for each $i \in [n]$,

$$\frac{\sigma_i}{p_i} \leq \frac{n-i+1}{\frac{1}{2}(n+i)} = \frac{2(n+i-1)}{n+i} = 2 - \frac{4i-2}{n+i}.$$

Here, the second term $\frac{4i-2}{n+i}$ is minimized when $i = 1$. Therefore, $\sigma_i/p_i \leq 2 - 2/(n+1)$, as claimed.

For computing Rounding, we need to construct the transitive closure $\mathcal{A}^*$ of $\mathcal{A}$ before the protocol begins. It is well known that a transitive closure can be computed by using algorithms for all-pairs shortest paths. For this problem, the Floyd-Warshall algorithm can be used; it runs in time $O(n^3)$ and space $O(n^2)$ (see, e.g., [8]). When $\mathcal{A}$ is small, for example, $m << n^2$, we can use Johnson's algorithm, which runs in time $O(n^2 \log n + nm)$ and space $O(m^2)$.

The time complexity of Rounding is $O(n^2)$, which is due to the sorting. The space complexity is $O(n^2)$, if we use the Floyd-Warshall algorithm with an adjacency matrix. The space complexity can be reduced to $O(m^2)$ if we employ Johnson's algorithm, which uses an adjacency list. On the other hand, we need an extra $O(\log m)$ factor in the time complexity since we need $O(\log m)$ time to check if $(i,j) \in \mathcal{A}^*$ when $\mathcal{A}^*$ is given as an adjacency list.

4.4 Main Result

We are now ready to prove the main result. From Lemma 2, Lemma 6, Theorem 1 and the fact that for any $\boldsymbol{x} \in \mathbb{R}^n_+$, $\min_{\boldsymbol{p} \in P_n \cap \mathrm{Precons}(\mathcal{A})} \boldsymbol{p} \cdot \boldsymbol{x} \leq \min_{\boldsymbol{\sigma} \in S_n \cap \mathrm{Precons}(\mathcal{A})} \boldsymbol{\sigma} \cdot \boldsymbol{x}$, we immediately get the following theorem.

Theorem 2 *There exists an online linear optimization algorithm over* $P_n \cap \mathrm{Precons}(\mathcal{A})$ *such that*

1. *its* $(2 - 2/(n+1))$*-regret is* $O(n^2\sqrt{T})$*, and*
2. *its per-trial running time is* $O(n^4)$*.*

5 Lower Bound

In this section, we derive a lower bound for the regret for our online prediction problem over the permutahedron P_n. Here, we consider the special case of no precedence constraint being given.

Theorem 3 *For our prediction problem over the permutahedron* P_n*, the* 1*-regret is* $\Omega(n^2\sqrt{T})$*.*

Proof. We consider an adversary who makes random choices. More precisely, at each trial t, the adversary randomly chooses a loss vector $\boldsymbol{\ell}_t$ from $\boldsymbol{\ell}^0, \boldsymbol{\ell}^1$, where $\boldsymbol{\ell}^0$ ($\boldsymbol{\ell}^1$) is the loss vector in which the first $\frac{n}{2}$ elements are 0s (1s) and the remaining

elements are 1s (0s). Then, for any online optimization algorithm that outputs $\boldsymbol{\sigma}_t \in S_t$ at trial t,

$$E[\sum_{t=1}^{T} \boldsymbol{\sigma}_t \cdot \boldsymbol{\ell}_t] = \frac{n(n+1)T}{4}.$$

Now, let us consider the best fixed permutation. Let $\boldsymbol{\sigma}^0 = (n, n-1, n-2, \ldots, 1)$ and $\boldsymbol{\sigma}^1 = (1, 2, 3, 4, \ldots, n)$. Suppose that $\boldsymbol{\ell}^0$ appears more frequently than $\boldsymbol{\ell}^1$ by k. The best permutation is $\boldsymbol{\sigma}^0$, and its cumulative loss is

$$\begin{aligned}
&\sum_{i=1}^{\frac{n}{2}} i\left(\frac{T}{2}+\frac{k}{2}\right) + \sum_{i=\frac{n}{2}+1}^{n} i\left(\frac{T}{2}-\frac{k}{2}\right) \\
=&\frac{n(n+1)T}{4} + \frac{k}{2}\left(2\frac{\frac{n}{2}(\frac{n}{2}+1)}{2} - \frac{n(n+1)}{2}\right)\frac{k}{2} \\
=&\frac{n(n+1)T}{4} - \frac{k}{2}n\left(\frac{n+1}{2} - \frac{\frac{n}{2}+1}{2}\right) \\
=&\frac{n(n+1)T}{4} - \frac{k}{2}\frac{n^2}{4}.
\end{aligned}$$

The same argument follows for the opposite case, where $\boldsymbol{\ell}^1$ is more frequent by k. In fact, k can be expressed as $k = \sum_{t=1}^{T} \delta_t$, where each δ_t is a discrete uniform random variable that takes values of ± 1. Then, the expected regret of any online optimization algorithm is at least $\frac{n^2}{8} E\left[\left|\sum_{t=1}^{T} \delta_t\right|\right]$. By the central limit theorem, the distribution of $\sum_{t=1}^{T} \delta_t$ converges to a Gaussian distribution with mean 0 and variance $\sqrt{T}$. Thus, for sufficiently large T, $\Pr[|\sum_{t=1}^{T} \delta_t| \geq \sqrt{T}]$ is a constant: c $(0 < c < 1)$. Therefore, the expected regret bound has a lower bound of $\frac{n^2}{8} c\sqrt{T}$. This implies that there exists a sequence of loss vectors that enforces any online optimization algorithm to incur regret that is at least $\Omega(n^2\sqrt{T})$.

In general, this lower bound on 1-regret is tight, since there are online algorithms that achieve a 1-regret with $O(n^2\sqrt{T})$ ([1,28]).

It is natural to ask if the $(2 - 2/(n+1))$-regret $O(n^2\sqrt{T})$ is tight under precedence constraints. We do not yet have a lower bound for this case, but we will show that our algorithm is optimal unless there is an offline algorithm with an approximation ratio $\alpha < 2$.

Theorem 4 *If there exists a polynomial-time online linear optimization algorithm with an α-regret of $poly(n, m)\sqrt{T}$, then there also exists a randomized polynomial-time algorithm for the offline problem with an approximation ratio α.*

Proof. The proof is based on standard online-to-offline conversion methods that can be found in the online learning literature (see, e.g., [11]). Let A be such an online linear optimization algorithm, and let its output at each trial t be denoted as $\boldsymbol{\sigma}_t$. Let $\boldsymbol{\ell} \in [0, 1]^n$ be the loss vector in the offline problem. We consider an

adversary who returns $\boldsymbol{\ell}_t = \boldsymbol{\ell}$ at each trial t. Then, the cumulative loss of A divided by T is bounded as follows:

$$\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\sigma}_t \cdot \boldsymbol{\ell} \leq \alpha \min_{\boldsymbol{\sigma} \in S_n \cap \mathrm{Precons}(\mathcal{A})} \boldsymbol{\sigma} \cdot \boldsymbol{\ell} + \frac{poly(n,m)}{T}.$$

Now, let $\widehat{\boldsymbol{\sigma}}$ be a uniformly and randomly chosen permutation from $\{\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_T\}$. Then,

$$E[\widehat{\boldsymbol{\sigma}} \cdot \boldsymbol{\ell}] \leq \alpha \min_{\boldsymbol{\sigma} \in S_n \cap \mathrm{Precons}(\mathcal{A})} \boldsymbol{\sigma} \cdot \boldsymbol{\ell} + \frac{poly(n,m)}{T}.$$

By setting $T = poly(n, m)$, the expected cumulative loss of $\widehat{\boldsymbol{\sigma}}$ is at most α times the cumulative loss of the best permutation (with a constant additive term), which completes the proof.

6 Experiments

In this section, we show preliminary experiments with artificial data sets in order to compare the performance of our algorithm with other methods. The experiments were performed on a server with four cores of Intel Xeon CPU X5560 2.80 GHz and a memory of 198 GB. We implemented the programs using Matlab with its Optimization Toolbox. To generate the loss vector at each trial t, we independently and randomly specified each element $\ell_{t,i}$ of the loss vector $\boldsymbol{\ell}_t$ as follows: Let $\ell_{t,i} = 1$ with probability r_i and $\ell_{t,i} = 0$, otherwise. We set $r_i = i/n$ so that $E[\boldsymbol{\ell}_t] = (1/n, 2/n, ..., 1)$. We constructed random acyclic precedence constraints on n jobs in the following way. First, we constructed a random total order over n jobs (vertices). Then, we constructed an acyclic directed graph over n vertices by adding $\binom{n}{2}$ directed edges according to the total order. Finally, we kept each edge (i, j) alive with probability $\pi = 0.2$, and otherwise, we removed the edge. The resulting directed graph represented the set of precedence constraints.

Using the above method, for each fixed n and T, we constructed three random sequences of loss vectors and three random sets of precedence constraints. The results (cumulative loss or computation time) were then averaged.

We compared our algorithm PermLearnPrec (PLP) to the following algorithms. We used the offline-to-online conversion techniques of Kakade et al. [16] (KKL) and the metarounding technique of Fujita et al. [13] combined with (FPL) ([17]; FPLM). For the metarounding of Fujita et al., we set $\epsilon = 0.01$ to guarantee $(\alpha + \varepsilon)$-regret when using an α-approximation offline algorithm. We used the linear programming (LP) relaxation-based scheduling algorithm of Chudak and Hochbaum [7] as the offline algorithm. This algorithm solves a minimum-cut problem on a network with $O(n^2)$ nodes and $O(n^3)$ arcs. We used the maxflow algorithm of Boykov and Kolmogorov [4] to solve the minimum-cut problem. In our PLP algorithm, we solved the isotonic regression by using the standard quadratic programming (QP) solver in Matlab.

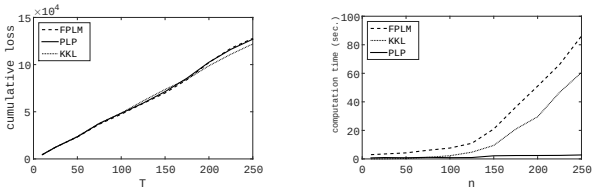

Fig. 1. Upper panel: cumulative losses of the three algorithms with the artificial data set for $n = 50$ and $T = 10, 20, 50, 100, 200, 250$; lower panel: total computation times of the three algorithms with $T = 50$ with $n = 10, 20, 50, 100, 200, 250$.

In Figure 1, we summarize the cumulative losses and total computation times, both averaged over three random data sets. As can be seen, the cumulative losses of all of the algorithms are quite similar. KKL performed slightly better than the other two algorithms, which is not surprising since they have almost identical α-regret bounds. On the other hand, if we consider the computation times of the algorithms, there is a very large difference. Our algorithm runs roughly 20 to 30 times faster than the other methods. The reason for this is that since the data are relatively "easy," the best permutation might not change frequently over time. Thus, in many trials, the projections onto the set of precedence constraints are already satisfied, and if this is the case, our algorithm can skip this step, whereas the other methods must compute the precedence constraints for every trial.

7 Conclusion

In this paper, we propose a polynomial-time online linear optimization algorithm over the permutahedron under precedence constraints. Our algorithm achieves a $(2 - 2/(n + 1))$-regret bound $O(n^2\sqrt{T})$, which means that it can predict as well as the state-of-the art offline approximation algorithms in hindsight. The approximation algorithm for which the approximation ratio is strictly less than $2 - 2/(n + 1)$.

An interesting open question is how our online framework can be extended to minimize the sum of weighted completion times. We note that Woeginger [29] showed that the offline problem of minimizing the sum of weighted completion time can be reduced to that of minimizing the unweighted sum. This reduction might be useful for designing an online version.

Acknowledgments

We thank anonymous reviewers for useful comments. Hatano is grateful to the supports from JSPS KAKENHI Grant Number 25330261. Takimoto is grateful to the supports from JSPS KAKENHI Grant Number 15H02667. In addition, the authors acknowledge the support from MEXT KAKENHI Grant Number 24106010 (the ELC project).

References

1. Ailon, N.: Improved bounds for online learning over the permutahedron and other ranking polytopes. In: Proceedings of 17th International Conference on Artificial Intelligence and Statistics (AISTAT2014). pp. 29–37 (2014)
2. Ambühl, C., Mastrolilli, M., Mutsanas, N., Svensson, O.: On the Approximability of Single-Machine Scheduling with Precedence Constraints Christoph Ambühl. Mathematics of Operations Research 36(4), 653–669 (2011)
3. Ambühl, C., Monaldo Mastrolilli, Swensson, O.: Inapproximability Results for Sparsest Cut, Optimal Linear Arrangement, and Precedence Constrained Scheduling. In: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007). pp. 329 – 337 (2007)
4. Boykov, Y., Kolmogorov, V.: An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Computer Vision. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(9), 1124–1137 (2004)
5. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press (2006)
6. Chekuri, C., Motwani, R.: Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. Discrete Applied Mathematics 98(1-2), 29–38 (1999)
7. Chudak, F.A., Hochbaum, D.S.: A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. Operations Research Letters 25, 199–204 (1999)
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Clifford, S.: Introduction to Algorithms. The MIT Press, third edit edn. (2009)
9. Correa, J.R., Schulz, A.S.: Single-machine scheduling with precedence constraints. Mathematics of Operations Research 30(4), 1005–1021 (2005)
10. Even-Dar, E., Kleinberg, R., Mannor, S., Mansour, Y.: Online Learning for Global Cost Functions. In: Proceedings of the 22nd Conference on Learning Theory (COLT 2009) (2009)
11. Freund, Y., Schapire, R.E.: Large Margin Classification Using the Perceptron Algorithm. Machine Learning 37(3), 277–299 (1999)
12. Fujishige, S.: Submodular functions and optimization. Elsevier Science, 2nd edn. (2005)
13. Fujita, T., Hatano, K., Takimoto, E.: Combinatorial Online Prediction via Metarounding. In: Proceedings of 24th Annual Conference on Algorithmic Learning Theory (ALT 2013). LNCS, vol. 8139, pp. 68–82 (2013)
14. Hall, L.A., Schulz, A.S., Shmoys, D.B., Wein, J.: Scheduling to Minimize Average Completion Time: Off-Line and On-Line Approximation Algorithms. Mathematics of Operations Research 22(3), 513–544 (1997)

15. Helmbold, D.P., Warmuth, M.K.: Learning Permutations with Exponential Weights. Journal of Machine Learning Research 10, 1705–1736 (2009)
16. Kakade, S., Kalai, A.T., Ligett, L.: Playing games with approximation algorithms. SIAM Journal on Computing 39(3), 1018–1106 (2009)
17. Kalai, A., Vempala, S.: Efficient algorithms for online decision problems. Journal of Computer and System Sciences 71(3), 291–307 (2005)
18. Lawler, E.L.: On Sequencing jobs to minimize weighted completion time subject to precedence constraints. Annals of Discrete Mathematics 2 2, 75–90 (1978)
19. Lenstra, J.K., Kan, A.H.G.R.: Complexity of scheduling under precedence constraints. Operations Research 26, 22–35 (1978)
20. Lugosi, G., Papaspiliopoulos, O., Stoltz, G.: Online Multi-task Learning with Hard Constraints. In: Proceedings of the 22nd Conference on Learning Theory (COLT 2009) (2009)
21. Luss, R., Rosset, S., Shahar, M.: Efficient regularized isotonic regression with application to gene-gene interaction search. Annals of Applied Statistics 6(1) (2012)
22. Margot, F., Queyranne, M., Wang, Y.: Decompositions, Network Flows, and a Precedence Constrained Single-Machine Scheduling ProblemDecompositions, Network Flows, and a Precedence Constrained Single-Machine Scheduling Problem. Operations Research 51(6), 981–992 (2003)
23. Maxwell, W., Muckstadt, J.: Establishing consistent and realistic reorder intervals in production-distribution systems. Operations Research 33, 1316–1341 (1985)
24. Mohring, H.R., Schulz, A.S., Uetz, M.: Approximation in Stochastic Scheduling: The Power of LP-based Priority Policies. Journal of the ACM 46(6), 924–942 (1999)
25. Schulz, A.S.: Scheduling to Minimize Total Weighted Completion Time: Performance Guarantees of LP-Based Heuristics and Lower Bounds. In: Proceedings of the 5th Conference on Integer Programming and Combinatorial Optimization (IPCO1996). pp. 301–315 (1996)
26. Skutella, M., Uetz, M.: Stochastic Machine Scheduling with Precedence Constraints. SIAM Journal on Computing 34(4), 788–802 (2005)
27. Spouge, J., Wan, H., Wilbur, W.: Least squares isotonic regression in two dimensions. J. Optimization Theory and Apps. 117, 585–605 (2003)
28. Suehiro, D., Hatano, K., Kijima, S., Takimoto, E., Nagano, K.: Online Prediction under Submodular Constraints. In: Proceedings of 23th Annual Conference on Algorithmic Learning Theory (ALT 2012). LNCS, vol. 7568, pp. 260–274 (2012)
29. Woeginger, G.J.: On the approximability of average completion time scheduling under precedence constraints. In: Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP 2001). pp. 862–874 (2001)
30. Woeginger, G.J., Schuurman, P.: Polynomial time approximation algorithms for machine scheduling: Ten open problems. Journal of Scheduling 2, 203–213 (1999)
31. Yasutake, S., Hatano, K., Kijima, S., Takimoto, E., Takeda, M.: Online Linear Optimization over Permutations. In: Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC 2011). LNCS, vol. 7074, pp. 534–543 (2011)
32. Ziegler, G.M.: Lectures on Polytopes. Graduate Texts in Mathematics 152, Springer-Verlag (1995)
33. Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003). pp. 928–936 (2003)