Online Linear Optimization over Permutations with Precedence Constraints

Takahiro Fujita, Kohei Hatano, Shuji Kijima and Eiji Takimoto
Department of Informatics, Kyushu University
{takahiro.fujita, hatano, kijima, eiji}@inf.kyushu-u.ac.jp

Abstract

We consider an online linear optimization problem over the set of permutations under some precedence constraints. In this problem, the player is supposed to predict a permutation of n fixed objects at each trial, under the constraints that some objects have higher priority than other objects in each permutation. This problem is naturally motivated by a scheduling problem whose objective is to minimize the sum of completion times of n sequential tasks under precedence constraints. We propose an online linear optimization algorithm which predicts almost as well as the best known offline approximation algorithms in hindsight. Furthermore, our algorithm runs in $O(n^4)$ time for each trial.

1 Introduction

Problems of learning and predicting of permutations appear in many contexts such as ranking, recommendation, and scheduling tasks. More precisely, a permutation $\boldsymbol{\sigma}$ over the set $[n] = \{1, \ldots, n\}$ of n fixed objects is a bijective function from $[n]$ to $[n]$. Another representation of a permutation $\boldsymbol{\sigma}$ over the set $[n]$ is to describe it as a n-dimensional vector in $[n]^n$, defined as $\boldsymbol{\sigma} = (\sigma(1), \ldots, \sigma(n))$. E.g., $(3, 4, 2, 1)$ is a representation of a permutation for $n = 4$. 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 permutahedron, denoted as P_n.

We assume a set of precedence constraints in permutations. 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 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 exits a linear ordering consistent with $\mathcal{A}$. In other words, we assume there exists a permutation $\sigma \in S_n \cap \mathrm{Precons}(\mathcal{A})$.

In this paper, we consider 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 of the player is to minimize the α-regret for some small $\alpha \geq 1$:

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

This problem is motivated by an online version of job scheduling with a single processor under some precedence constraints. Assume that there is a single processor and n fixed jobs. Every day t, we determine a schedule represented by a permutation $\boldsymbol{\sigma}_t$ in $S_n \cap \mathrm{Precons}(\mathcal{A})$. Then, after processing all n jobs according to the schedule, the processing time $\ell_{t,i} \in [0, 1]$ of each job i is revealed. The goal is to to minimize *the sum of the completion time* over all jobs and T days, under the fixed precedence constraints, where the completion time of job i at day t is the sum of processing times of jobs prior to i and the processing time of job i. For example, at day t, we process 4 jobs according to a permutation $\boldsymbol{\sigma}_t = (3, 2, 1, 4) \in S_n \cap \mathrm{Precons}(\mathcal{A})$ and each processing time is given as $\boldsymbol{\ell}_t = (\ell_{t,1}, \ell_{t,2}, \ell_{t,3}, \ell_{t,4})$. Note that the component $\sigma_{t,i}$ of each permutation $\boldsymbol{\sigma}_t$ represents the priority of each job i. That is, jobs with higher priority are processed earlier. Therefore, jobs $4, 1, 2$, and 3 are processed sequentially. The completion time of jobs $i = 4, 1, 2, 3$ are $\ell_{t,4}$, $\ell_{t,4} + \ell_{t,1}$, $\ell_{t,4} + \ell_{t,1} + \ell_{t,2}$, and $\ell_{t,4} + \ell_{t,1} + \ell_{t,2} + \ell_{t,3}$, respectively. So, loss $\boldsymbol{\sigma}_t \cdot \boldsymbol{\ell}_t$ exactly corresponds to the sum of the completion time at day t.

In this paper, we propose an online linear optimization algorithm over $P_n \cap \mathrm{Precons}(\mathcal{A})$ whose 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. Further, we show that the 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 (which we discuss later). So far, the state-of-the-art approximation algorithms have approximation ratio $2 - 2/(n+1)$ and it is an open problem to find an approximation algorithm with better ratio [18]. Therefore, our algorithm is optimal among any polynomial algorithms unless the open problem is positively solved.

The corresponding offline problem has been extensively investigated in the literature. The problem is, given a loss vector $\boldsymbol{\ell} \in [0, 1]^n$ and the set of precedence constraints $\mathcal{A}$ as inputs, to output a permutation $\boldsymbol{\sigma} \in S_n \cap \mathrm{Precons}(\mathcal{A})$ which minimizes the inner product $\boldsymbol{\sigma} \cdot \boldsymbol{\ell}$, i.e., the sum of completion times. More generally, the problem of minimizing the weighted sum of completion times are typically considered. It is known that the problem is NP-hard [10, 11]. Several $2 - O(1/n)$-approximation algorithms are proposed (Schulz [15], Hall et al. [8], Chudak and Hochbaum [4], Margot et al. [13], and Chekuri and Motwani [3]). For further developments, see, e.g., [6, 2].

There are related researches on online optimization over the permutahedron. The first result without precedence constraints is proposed by Yasutake et al. [19]. Ailon proposed another online optimization algorithm with an improved regret bound[1]. Suehiro et al. [17] extended Yasutake et al.'s result to the submodular base polyhedron which can be used for not only permutations, but also other combinatorial objects such as spanning trees.

It is possible to obtain online optimization algorithms using "offline-to-online" conversion techniques. By using conversion method of Kakade et al. [9] or Fujita et al. [7], we can construct online optimization algorithms with α-regret close to ours. But, with the method of Kakade et al. [9], the resulting algorithm takes time linear in T, which is not desirable. With the method of Fujita et al. [7], the running time per trial is $poly(n, 1/\varepsilon)$ which is independent of T but depends on $1/\varepsilon$ and its α-regret is proved for $\alpha = 2 - 2/(n+1) + \varepsilon$, which is inferior to ours.

2 Online Linear Optimization Algorithm over the Permutations

In this section, we propose our algorithm PermLearnPrec and prove its α-regret bound.

2.1 Main Structure

The description of PermLearnPrec is shown in Algorithm 1. The algorithm maintains a weight vector $\boldsymbol{p}_t$ in $\mathbb{R}^n_+$, which represents a "mixture" of permutations in S_n. At each trial t, it "rounds" a vector $\boldsymbol{p}_t$ into a permutation $\boldsymbol{\sigma}_t$ so that $\boldsymbol{\sigma}_t \leq \alpha \boldsymbol{p}_t$ for some $\alpha > 0$. This procedure is done by Rounding, which we will show the details in the next section. After the loss vector $\boldsymbol{\ell}_t$ is given, PermLearnPrec updates the weight vector $\boldsymbol{p}_t$ in an additive way and projects it onto the set of linear constraints representing precedence constraints $\mathrm{Precons}(\mathcal{A})$ and the intersection of the permutahedron P_n and $\mathrm{Precons}(\mathcal{A})$ successively.

The main structure of our algorithm itself is built on a standard online convex optimization algorithm Online Gradient Descent (OGD) [20] in online learning literature. OGD consists of the additive update of weight vectors and the 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 $O(n^2\sqrt{T})$. So, apparently, our successive projections seem redundant and only one projection to $P_n \cap \mathrm{Precons}(\mathcal{A})$ would suffice. The problem of the standard approach is that the projection onto $P_n \cap \mathrm{Precons}(\mathcal{A})$ looks not tractable since it deals exponentially many linear constraints. Later, we will show that the successive projections are the keys to an efficient implementation of our algorithm.

Algorithm 1 PermLearnPrec

1. Let $\boldsymbol{p}_1 = ((n+1)/2, \ldots, (n+1)/2) \in [0,n]^n$.
2. For $t = 1, \ldots, T$
   (a) 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) Let $\boldsymbol{p}_{t+\frac{2}{3}}$ be the Euclidean projection onto the set $\mathrm{Precons}(\mathcal{A})$, .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) Let $\boldsymbol{p}_{t+1}$ be the projection of $\boldsymbol{p}_{t+\frac{2}{3}}$ onto the set $P_n \cap \mathrm{Precons}(\mathcal{A})$, that is, $\boldsymbol{p}_{t+1} = \arg\inf_{\boldsymbol{p}\in P_n\cap\mathrm{Precons}(\mathcal{A})} \|\boldsymbol{p} - \boldsymbol{p}_{t+\frac{2}{3}}\|_2^2$.

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

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

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

Lemma 2 (Cf. Zinkevich [20])**.** *For any* $T \geq 1$ *and* $\eta = (n+1)/(2\sqrt{T})$,

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

3 Efficient Implementations of Projection and Rounding

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

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

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

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

This problem is known as the isotonic regression problem [14, 16, 12]. Previously known algorithms for the isotonic regression run in $O(mn^2 \log n)$ or $O(n^4)$ time see [14, 16, 12] for details.

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

Now we show an efficient algorithm Projection for computing the projection onto the intersection of the permutahedron P_n and the set $\mathrm{Precons}(\mathcal{A})$ of precedence constraints. In fact, we will show that the problem can be reduced to projection onto P_n only. So, we will just use the algorithm of Suehiro et al. [17] for finding the projection onto P_n.

Algorithm 2 Rounding

Input: $\boldsymbol{p} \in P_n \cap \mathrm{Precons}(\mathcal{A})$ satisfying that $p_1 \geq p_2 \geq \cdots \geq p_n$ 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.

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 (Suehiro et al.[17])). *Let $\mathbf{p}^*$ be the projection of $\boldsymbol{q}$ s.t. $q_1 \geq q_2 \geq \ldots \geq q_n$ and $\mathcal{A}'$ is the set of violating constraints w.r.t. $\boldsymbol{q}$. Then the projection $\mathbf{p}^*$ satisfies that $p_1^* \geq p_2^* \geq \ldots \geq p_n^*$.*

Further, we show that the projection onto P_n preserves equality as well.

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

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

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

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

So, by Lemma 5, 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. [17] for the base polyhedron (which generalizes the permutahedron), we obtain the following result.

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

3.3 Rounding

We show an algorithm for Rounding in Algorithm 2. 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 ties in $\boldsymbol{p}$ to construct $\boldsymbol{\sigma}$. Let $\mathcal{A}^*$ be the transitive closure of $\mathcal{A}$. So, 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 show that the rounding guarantees that $\boldsymbol{\sigma} \leq (2 - 2/(n+1))$. Then we discuss time complexity of Rounding.

We prove the following lemma on Rounding.

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

For computing Rounding, we need to construct the transitive closure $\mathcal{A}^*$ of $\mathcal{A}$ before the protocol starts. It is well known that a transitive closure can be computed by using algorithms for all-pairs shortest pathes. For this problem, Floyd-Warshall algorithm can be used and it runs in time $O(n^3)$ and space $O(n^2)$ (see, e.g., [5]). When $\mathcal{A}$ is small, for example, $m << n^2$, we can use Johnson's algorithm running 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 Floyd-Warshall algorithm with a 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.

3.4 Main Result

Now we are ready to prove the main result. By Lemma 5, 6 and Theorem 1, we get the following theorem immediately.

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 running time is $O(n^4)$ time per trial.*

4 Lower Bound

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

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

In fact, this lower bound on 1-regret is tight in general, since there are online algorithms which achieve 1-regret $O(n^2\sqrt{T})$ ([17, 1]).

Now it is natural to ask if the $(2-2/(n+1))$-regret $O(n^2\sqrt{T})$ is tight under precedence constraints. So far, we have no lower bound for this case. But, we give an alternative argument that our algorithm is optimal unless there are an offline algorithm with approximation ratio $\alpha < 2$.

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

References

[1] N. Ailon. Improved bounds for online learning over the permutahedron and other ranking polytopes. In *Proceedings of 17th International Conference on Artificial Intelligence and Statistics (AISTAT2014)*, pages 29–37, 2014.

[2] C. Ambühl, M. Mastrolilli, N. Mutsanas, and O. Svensson. On the Approximability of Single-Machine Scheduling with Precedence Constraints Christoph Ambühl. *Mathematics of Operations Research*, 36(4):653–669, 2011.

[3] C. Chekuri and R. Motwani. Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. *Discrete Applied Mathematics*, 98(1-2):29–38, 1999.

[4] F. A. Chudak and D. S. Hochbaum. A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters*, 25:199–204, 1999.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and S. Clifford. *Introduction to Algorithms*. The MIT Press, third edit edition, 2009.

[6] J. R. Correa and A. S. Schulz. Single-machine scheduling with precedence constraints. *Mathematics of Operations Research*, 30(4):1005–1021, 2005.

[7] T. Fujita, K. Hatano, and E. Takimoto. Combinatorial Online Prediction via Metarounding. In *Proceedings of 24th Annual Conference on Algorithmic Learning Theory (ALT 2013)*, pages 68–82, 2013.

[8] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to Minimize Average Completion Time: Off-Line and On-Line Approximation Algorithms. *Mathematics of Operations Research*, 22(3):513–544, 1997.

[9] S. Kakade, A. T. Kalai, and L. Ligett. Playing games with approximation algorithms. *SIAM Journal on Computing*, 39(3):1018–1106, 2009.

[10] E. L. Lawler. On Sequencing jobs to minimize weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics 2*, 2:75–90, 1978.

[11] J. K. Lenstra and A. H. G. R. Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26:22–35, 1978.

[12] R. Luss, S. Rosset, and M. Shahar. Efficient regularized isotonic regression with application to gene-gene interaction search. *Annals of Applied Statistics*, 6(1), 2012.

[13] F. Margot, M. Queyranne, and Y. Wang. 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.

[14] W. Maxwell and J. Muckstadt. Establishing consistent and realistic reorder intervals in production-distribution systems. *Operations Research*, 33:1316–1341, 1985.

[15] A. S. Schulz. 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)*, pages 301–315, 1996.

[16] J. Spouge, H. Wan, and W. Wilbur. Least squares isotonic regression in two dimensions. *J. Optimization Theory and Apps.*, 117:585–605, 2003.

[17] D. Suehiro, K. Hatano, S. Kijima, E. Takimoto, and N. Kiyohito. Online prediction under submodular constraints. In *In Proceedings of 23th Annual Conference on Learning Theory (ALT 2012)*, 2012.

[18] G. J. Woeginger and P. Schuurman. Polynomial time approximation algorithms for machine scheduling: Ten open problems. *Journal of Scheduling*, 2:203–213, 1999.

[19] S. Yasutake, K. Hatano, S. Kijima, E. Takimoto, and M. Takeda. Online Linear Optimization over Permutations. In *Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC 2011)*, pages 534–543, 2011.

[20] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML '03)*, pages 928–936, 2003.