

Online Linear Optimization over Permutations

Yasutake, Shota

Department of Informatics, Kyushu University

Hatano, Kohei

Department of Informatics, Kyushu University

Kijima, Shuji

Department of Informatics, Kyushu University

Takimoto, Eiji

Department of Informatics, Kyushu University

他

<https://hdl.handle.net/2324/1524320>

出版情報 : Algorithms and Computation (Lecture Notes in Computer Science). 7074, pp.534-543, 2011-11-21. Springer Berlin Heidelberg

バージョン :

権利関係 :



Online Linear Optimization over Permutations

Shota Yasutake, Kohei Hatano, Shuji Kijima, Eiji Takimoto, and
Masayuki Takeda

Department of Informatics, Kyushu University
{shouta.yasutake, hatano, kijima, eiji, takeda}@inf.kyushu-u.ac.jp

Abstract. This paper proposes an algorithm for *online linear optimization problem over permutations*; the objective of the online algorithm is to find a permutation of $\{1, \dots, n\}$ at each trial so as to minimize the “regret” for T trials. The regret of our algorithm is $O(n^2 \sqrt{T \ln n})$ in expectation for any input sequence. A naive implementation requires more than exponential time. On the other hand, our algorithm uses only $O(n)$ space and runs in $O(n^2)$ time in each trial. To achieve this complexity, we devise two efficient algorithms as subroutines: One is for minimization of an entropy function over the *permutahedron* P_n , and the other is for randomized rounding over P_n .

1 Introduction

Permutation is one of fundamental concepts in discrete mathematics and computer science. Permutations can naturally represent ranking or allocation of fixed objects. So, they have been applied to ranking in machine learning, information retrieval, recommendation tasks, and scheduling tasks.

More formally, a permutation σ over the set $[n] = \{1, \dots, n\}$ of n fixed objects is a bijective function from $[n]$ to $[n]$. Another popular way of representing a permutation σ over the set $[n]$ is to describe it as the n -dimensional vector in $[n]^n$, defined as $\boldsymbol{\sigma} = (\sigma(1), \dots, \sigma(n))$. For example, $(3, 4, 2, 1)$ is a representation of a permutation for $n = 4$. Let S_n be the set of all permutations over $[n]$, i.e., $S_n = \{\boldsymbol{\sigma} \in [n]^n \mid \boldsymbol{\sigma} \text{ is a permutation over } [n]\}$. In particular, all permutations in S_n form a convex hull P_n , which is called *permutahedron*. Permutahedron has been studied extensively in submodular optimization [12, 4] or scheduling problems [10, 9, 11].

We consider the following online linear optimization problem over S_n . For each trial $t = 1, \dots, T$, (i) the player predicts a permutation $\boldsymbol{\sigma}_t \in S_n$, (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 regret: $\sum_{t=1}^T \boldsymbol{\sigma}_t \cdot \boldsymbol{\ell}_t - \min_{\boldsymbol{\sigma} \in S_n} \sum_{t=1}^T \boldsymbol{\sigma} \cdot \boldsymbol{\ell}_t$.

To illustrate an application of our linear optimization problem, we consider the following online job scheduling problem with a single processor. Suppose that we have a single processor and n fixed jobs to be processed sequentially. Every day, we determine a schedule represented by a permutation in S_n a priori. Then, after processing all n jobs, we are given the processing time $\ell_i \in [0, 1]$ of each job i (assume that each processing time is normalized up to 1). As a goal, we

might want to minimize *the sum of the completion time* over all jobs and T days, where the completion time of job i is the sum of processing time of jobs prior to i plus the processing time of job i . For example, suppose that we process jobs according to a permutation $\sigma = (3, 2, 1, 4)$ and each processing time is given as $\ell = (\ell_1, \ell_2, \ell_3, \ell_4)$. Here, we interpret σ so that job i is processed with priority $\sigma(i)$. In other words, jobs with higher priority are processed earlier. So, we process jobs 4, 1, 2, and 3 sequentially. The completion time of jobs $i = 4, 1, 2, 3$ are $\ell_4, \ell_4 + \ell_1, \ell_4 + \ell_1 + \ell_2$, and $\ell_4 + \ell_1 + \ell_2 + \ell_3$, respectively. So, loss $\sigma \cdot \ell$ is exactly the sum of the completion time.

In this paper, we propose a randomized prediction algorithm whose expected regret is at most $O(n^2 \sqrt{\log n} \sqrt{T})$, which is the best so far. For each trial, our algorithm runs in time $O(n^2)$ using $O(n)$ space. Note that, the competitive ratio converges to 1 as T increases under the natural assumption that the cumulative losses of the best permutation is $\omega(\sqrt{T})$.

There is a related previous algorithm, PermELearn [6] proposed by Helmbold and Warmuth, that is applicable to our problem though the algorithm was developed for generalized settings. It can be shown that PermELearn has the same regret bound of ours. However, the constant factor of the regret bound of PermELearn is roughly 4/3 worse than that of ours. Further, PermELearn needs $O(n^2)$ space and $\tilde{O}(n^6)$ running time. Our preliminary experimental results show that our algorithm indeed performs better than PermELearn for some artificial data.

There are other related researches. Online convex optimization (including online linear optimization) has been extensively studied in Machine Learning these days (see, e.g., [5]). Originally, the job scheduling problem we illustrated was considered in the offline setting [7, 8].

2 Preliminaries

For any fixed positive integer n , we denote $[n]$ by the set $\{1, \dots, n\}$. *Permutahedron* P_n is the convex hull of the set of permutations S_n . It is known that P_n coincides with the set of points $\mathbf{p} \in \mathbb{R}_+^n$ 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 references of the permutahedron, see, e.g., [12, 4].

The *unnormalized relative entropy* $\Delta(\mathbf{p}, \mathbf{q})$ from $\mathbf{q} \in \mathbb{R}_+^n$ to $\mathbf{p} \in \mathbb{R}_+^n$ is defined as $\Delta(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^n p_i \ln \frac{p_i}{q_i} + \sum_{i=1}^n q_i - \sum_{i=1}^n p_i$. It is known that $\Delta(\mathbf{p}, \mathbf{q}) \geq 0$ and $\Delta(\mathbf{p}, \mathbf{q}) = 0$ if and only if $\mathbf{p} = \mathbf{q}$. Unnormalized relative entropy is not symmetric in general, i.e., $\Delta(\mathbf{p}, \mathbf{q}) \neq \Delta(\mathbf{q}, \mathbf{p})$ for some $\mathbf{p}, \mathbf{q} \in \mathbb{R}_+^n$. Also, unnormalized relative entropy is a special case of Bregman divergence [2, 3], which generalizes Euclid distance or natural distance measures.

In this paper, for simplicity, we assume that calculation of any elementary functions (addition, multiplication, exponential functions, and etc.) can be done in a unit time.

3 Algorithm

In this section, we propose our algorithm PermutahedLearn and prove its regret bound.

3.1 Main Structure

The main structure of PermutahedLearn is shown in Algorithm 1. The algorithm maintains a weight vector \mathbf{p}_t in \mathbb{R}_+^n , which represents a mixture of permutations in S_n . At each trial t , it decomposes \mathbf{p}_t into permutations, chooses a permutation σ_t randomly according to its coefficient, and predicts the permutation σ_t . After the loss ℓ_t is assigned, PermutahedLearn updates the weight vector \mathbf{p}_t in a multiplicative way and projects it onto the permutahedron P_n .

The main structure of our algorithm itself is built on a standard technique in online learning literature (see, e.g., [6]). Our technical contribution is to develop efficient projection and decomposition techniques specifically designed for the permutahedron.

Algorithm 1 PermutahedLearn

1. Let $\mathbf{p}_1 = ((n+1)/2, \dots, (n+1)/2) \in [0, n]^n$.
 2. For $t = 1, \dots, T$
 - (a) Run **Decomposition**(\mathbf{p}_t) and get \mathbf{p}_t as $\mathbf{p}_t = \sum_{i=1}^k \lambda_i \sigma^{(i)}$, where $k \leq n$, each $\lambda_i \geq 0$, $\sum_i \lambda_i = 1$, and each $\sigma^{(i)}$ is in S_n .
 - (b) Choose σ_t randomly from $\{\sigma^{(1)}, \dots, \sigma^{(k)}\}$ according to the distribution λ .
 - (c) Incur a loss $\sigma_t \cdot \ell_t$.
 - (d) Update $\mathbf{p}_{t+\frac{1}{2}}$ as $p_{t+\frac{1}{2},i} = p_{t,i} e^{-\eta \ell_{t,i}} / Z$, where Z is the normalization constant such that $\sum_{i=1}^n p_{t+\frac{1}{2},i} = n(n+1)/2$.
 - (e) Run **Projection**($\mathbf{p}_{t+\frac{1}{2}}$) and get \mathbf{p}_{t+1} , the projection of $\mathbf{p}_{t+\frac{1}{2}}$ onto the permutahedron P_n . That is, $\mathbf{p}_{t+1} = \arg \inf_{\mathbf{p} \in P_n} \Delta(\mathbf{p}, \mathbf{p}_{t+\frac{1}{2}})$.
-

First of all, we prove a cumulative regret bound of weight vectors \mathbf{p}_t .

Lemma 1 *For any $T \geq 1$ it holds that*

$$\sum_{t=1}^T \mathbf{p}_t \cdot \ell_t \leq \frac{\eta \inf_{\mathbf{p} \in P_n} \sum_{t=1}^T \mathbf{p} \cdot \ell_t + \frac{n(n+1)}{2} \ln n}{1 - e^{-\eta}}.$$

The proof is standard in the online learning literature (see, e.g., [6]) and is omitted.

To complete our analysis for our algorithm, we specify the subroutines Projection and Decomposition, respectively, in the following subsections.

3.2 Projection

We propose an efficient algorithm Projection for computing the projection onto the permutahedron P_n . Formally, the problem is stated as follows: $\inf_{\mathbf{p}} \Delta(\mathbf{p}, \mathbf{q})$ subject to the constraints that

$$\sum_{j \in S} p_j \leq \sum_{j=1}^{|S|} (n+1-j), \text{ for any } S \subset [n], \text{ and } \sum_{j=1}^n p_j = \frac{n(n+1)}{2}. \quad (1)$$

Here we omit the positivity constraints $\mathbf{p} \geq \mathbf{0}$ since relative entropy projection always preserves positivity.

Apparently, this problem does not seem to be tractable as it has exponentially many constraints. But, we show that relevant constraints are only linearly many.

For simplicity, we assume that elements in \mathbf{q} are sorted in descending order, i.e., $q_1 \geq q_2 \geq \dots \geq q_n$. This can be achieved in time $O(n \log n)$ by sorting \mathbf{q} . First, we show that this projection preserves the order in \mathbf{q} .

Lemma 2 *Let \mathbf{p}^* be the projection of \mathbf{q} . Then we have $p_1^* \geq p_2^* \geq \dots \geq p_n^*$.*

Proof. Assume that the claim is false. Then, there exists $i \leq j$ such that $p_i^* < p_j^*$ and $q_i \geq q_j$. Let \mathbf{r} be the vector obtained by exchanging p_i^* and p_j^* in \mathbf{p}^* . Then,

$$\Delta(\mathbf{p}^*, \mathbf{q}) - \Delta(\mathbf{r}, \mathbf{q}) = p_i^* \ln \frac{q_j}{q_i} + p_j^* \ln \frac{q_i}{q_j} = (p_j^* - p_i^*) \ln \frac{q_i}{q_j} \geq 0,$$

where, the last inequality holds since $p_j^* \geq p_i^*$ and $q_i \geq q_j$. This contradicts the assumption that \mathbf{p}^* is the projection. \square

By Lemma 2, observe that if the conditions $\sum_{j=1}^i p_j \leq \sum_{j=1}^i (n+1-j)$, for $i = 1, \dots, n-1$, are satisfied, then other inequality constraints are satisfied as well since for any $S \subset [n]$ such that $|S| = i$, $\sum_{j \in S} p_j \leq \sum_{j=1}^i p_j$.

Therefore, the problem (1) is reduced to the same minimization problem whose constraints are linearly many:

$$\sum_{j=1}^i p_j \leq \sum_{j=1}^i (n+1-j), \text{ for } i = 1, \dots, n-1, \text{ and } \sum_{j=1}^n p_j = \frac{n(n+1)}{2}. \quad (2)$$

The KKT conditions (e.g., [1]) imply that \mathbf{p}^* is the projection if and only if \mathbf{p}^* satisfies the following conditions: There exist non-negative real numbers $\alpha_1, \dots, \alpha_{n-1}$ such that

$$\begin{aligned} p_i^* &= q_i e^{-\sum_{j=i}^{n-1} \alpha_j} / Z \text{ (for } i = 1, \dots, n-1), \quad p_n^* = q_n / Z, \\ \sum_{j=1}^i p_j^* &\leq \sum_{j=1}^i (n+1-j) \text{ (for } i = 1, \dots, n-1), \quad \sum_{i=1}^n p_i^* = \frac{n(n+1)}{2}, \\ \alpha_i \left(\sum_{j=1}^i p_j^* - \sum_{j=1}^i (n+1-j) \right) &= 0 \text{ (for } i = 1, \dots, n-1), \end{aligned} \quad (3)$$

Algorithm 2 Projection

Input: $\mathbf{q} \in \mathbb{R}_+^n$ satisfying that $q_1 \geq q_2 \geq \dots \geq q_n$.

Output: projection \mathbf{p} of \mathbf{q} onto P_n .

1. Let $i_0 = 0$.
 2. **For** $t = 1, \dots$,
 - (a) Let $C_i^t = \frac{\sum_{j=1}^{i_t-1} (n+1-j) - \sum_{j=1}^{i_t-1} p_j}{\sum_{j=i_{t-1}+1}^i q_j}$ for $i = 1, \dots, n$ and $i_t = \arg \min_{i: i_{t-1} < i \leq n} C_i^t$.
If there are multiple minimizers, choose the largest one as i_t .
 - (b) Set $p_j = q_j C_{i_t}^t$ for $j = i_{t-1} + 1, \dots, i_t$.
 - (c) **If** $i_t = n$, **then** break.
 3. **Output** \mathbf{p} .
-

where Z is the normalization constant so that $\sum_i p_i^* = n(n+1)/2$.

Now we describe the detail of the projection algorithm in Algorithm 2.

Lemma 3 (i) Given \mathbf{q} , the algorithm Projection outputs the projection of \mathbf{q} onto the permutahedron P_n . (ii) The time complexity of Projection is $O(n^2)$.

Proof. We show that there exists $\alpha_1, \dots, \alpha_{n-1}$ and Z such that the output \mathbf{p} satisfies the optimality conditions (3), which completes the proof of the first statement.

First of all, we show that $C_{i_{t-1}}^{t-1} \leq C_{i_t}^t$ for each iteration t . Because of the definition of $C_{i_{t-1}}^{t-1}$, we have $C_{i_{t-1}}^{t-1} < C_{i_t}^{t-1}$. So, it suffices to prove that $C_{i_t}^{t-1} < C_{i_t}^t$. To see this, observe that

$$\sum_{j=1}^{i_{t-2}} p_j + C_{i_t}^{t-1} \sum_{j=i_{t-2}+1}^{i_t} q_j = \sum_{j=1}^{i_t} (n+1-j) = \sum_{j=1}^{i_{t-1}} p_j + C_{i_t}^t \sum_{j=i_{t-1}+1}^{i_t} q_j$$

and

$$\sum_{j=1}^{i_{t-1}} p_j + C_{i_t}^t \sum_{j=i_{t-1}+1}^{i_t} q_j < \sum_{j=1}^{i_{t-2}} p_j + C_{i_t}^{t-1} \sum_{j=i_{t-2}+1}^{i_{t-1}} q_j + C_{i_t}^t \sum_{j=i_{t-1}+1}^{i_t} q_j,$$

where the last inequality holds since $C_{i_{t-1}}^{t-1} < C_{i_t}^{t-1}$. By rearranging the inequalities above, we have $C_{i_t}^{t-1} < C_{i_t}^t$.

Now we fix each α_{i_t} so that $e^{-\alpha_{i_t}} C_{i_{t+1}}^{t+1} = C_{i_t}^t$, i.e., $\alpha_{i_t} = \ln(C_{i_{t+1}}^{t+1}/C_{i_t}^t)$ and fix Z to be $Z = C_n^T$, where T satisfies $i_T = n$. Note that since $C_{i_{t+1}}^{t+1} > C_{i_t}^t$, each α_{i_t} is strictly positive. For other $i \notin \{i_1, \dots, i_T\}$, we set $\alpha_i = 0$. Then, each p_{i_t} can be expressed as

$$p_{i_t} = q_{i_t} C_{i_t}^t = q_{i_t} e^{-\alpha_{i_t} - \alpha_{i_{t+1}} - \dots - \alpha_{i_T}} / Z = q_{i_t} e^{-\alpha_{i_t} - \alpha_{i_t+1} - \dots - \alpha_{n-1}} / Z.$$

Similarly, for other i such that $i_{t-1} < i < i_t$, we have

$$p_i = q_i C_{i_t}^t = q_i e^{-\alpha_{i_t} - \alpha_{i_t+1} - \dots - \alpha_{n-1}} / Z = q_i e^{-\alpha_i - \alpha_{i+1} - \dots - \alpha_{n-1}} / Z.$$

To see if the specified α_i s and Z satisfies the optimality conditions (3), observe that (i) for each i_t ,

$$\sum_{j=1}^{i_t} p_j = \sum_{j=1}^{i_{t-1}} p_j + \sum_{j=i_{t-1}+1}^{i_t} q_j C_{i_t}^t = \sum_j (n+1-j)$$

and $\alpha_{i_t} > 0$, and (ii) for each i such that $i_{t-1} < i < i_t$,

$$\sum_{j=1}^i p_j = \sum_{j=1}^{i_{t-1}} p_j + \sum_{j=i_{t-1}+1}^i q_j C_{i_t}^t \leq \sum_{j=1}^{i_{t-1}} p_j + \sum_{j=i_{t-1}+1}^{i_t} q_j C_i^t = \sum_j (n+1-j)$$

and $\alpha_i = 0$.

Finally, the algorithm terminates in time $O(n^2)$ since the number of iteration is at most n and each iteration takes $O(n)$ time, which completes the second statement of the lemma. \square

3.3 Decomposition

In this subsection, we describe how to represent a point $\mathbf{p} \in P_n$ by a convex combination of permutations. For simplicity assume that $p_1 \geq \dots \geq p_n$.

To begin with, we define special points in the permutahedron, which we call *permutations with ties*. Suppose $\mathbf{q} \in P_n$ satisfies that $q_1 \geq q_2 \geq \dots \geq q_n$. A permutation with ties $\mathbf{q} \in P_n$ satisfies that if $q_i > q_{i+1}$ then $\sum_{j=1}^i q_j = \sum_{j=1}^i (n+1-i)$ hold for any $i \in [n]$. For example, if $\mathbf{q} \in P_5$ satisfies $q_1 = q_2 > q_3 = q_4 = q_5$, then \mathbf{q} is uniquely determined as $\mathbf{q} = (4.5, 4.5, 2, 2, 2)$. Note that every permutation with ties \mathbf{q} satisfying that $q_1 \geq q_2 \geq \dots \geq q_n$ is represented by a convex combination of (at most) two permutations, namely $(\boldsymbol{\sigma} + \boldsymbol{\sigma}')/2$ where $\boldsymbol{\sigma} = (n, n-1, \dots, 1)$ and $\boldsymbol{\sigma}'$ is a “partially reversed” permutation satisfying that $\sigma'(i) > \sigma'(j)$ if $q_i > q_j$ and $\sigma'(i) < \sigma'(i+1)$ if $q_i = q_{i+1}$. Note that $\boldsymbol{\sigma}'$ is uniquely determined by the permutation with ties $\mathbf{q} \in P_n$. For example, let $\mathbf{q} = (4.5, 4.5, 2, 2, 2)$ and $\boldsymbol{\sigma} = (5, 4, 3, 2, 1)$, then its partially reversed permutation $\boldsymbol{\sigma}'$ is $(4, 5, 1, 2, 3)$. Further, for any positive vector $\mathbf{p} \in \mathbb{R}_+^n$ such that $p_1 \geq \dots \geq p_n$, we say that $\boldsymbol{\sigma}'$ is the partially reversed permutation w.r.t. \mathbf{p} if there exists a permutation with ties $\mathbf{q} \in P_n$ such that $p_i = p_{i+1}$ if and only if $q_i = q_{i+1}$ for each $i = 1, \dots, n-1$, $\boldsymbol{\sigma}'$ is the partially reversed permutation w.r.t. \mathbf{q} . Note that such \mathbf{q} is unique.

Now we describe our algorithm to represent $\mathbf{p} \in P_n$ with a convex combination of permutations. Note that $\boldsymbol{\sigma}^1 = \boldsymbol{\sigma}^0$ holds if the input \mathbf{p} satisfies that $p_i > p_{i+1}$ for any $i \in [n-1]$.

We will prove the following lemma on Decomposition.

Lemma 4 *Decomposition provides a convex combination of at most $n+1$ permutations representing an arbitrarily given $\mathbf{p} \in P_n$. Its running time is $O(n^2)$.*

To show Lemma 4, we show the following Lemmas.

Algorithm 3 Decomposition

Input: $\mathbf{p} \in P_n$ satisfying that $p_1 \geq p_2 \geq \dots \geq p_n$.

Output: Permutations $\sigma^0, \dots, \sigma^K$ and $\lambda_0, \dots, \lambda_K \in \mathbb{R}_{>0}$ s.t. $\sum_{i=0}^K \lambda_i \sigma^i = \mathbf{p}$, $\sum_{i=0}^K \lambda_i = 1$.

1. Let $\sigma^0 = (n, n-1, \dots, 1)$, $\mathbf{p}^1 = \mathbf{p}$ and $\lambda = 1$.
 2. **For** $t = 1, \dots$,
 - (a) Find a partially reversed permutation σ^t with respect to \mathbf{p}^t and σ^0 . Let $\mathbf{q}^t = (\sigma^0 + \sigma^t)/2$.
 - (b) Let $\lambda_t = \min \left\{ \lambda, \min_{i \in [n-1]} \left\{ \frac{p_i^t - p_{i+1}^t}{q_i^t - q_{i+1}^t} \mid q_i^t \neq q_{i+1}^t \right\} \right\}$.
 - (c) Let $\mathbf{p}^{t+1} = \mathbf{p}^t - \lambda_t \mathbf{q}^t$ and let $\lambda = \lambda - \lambda_t$.
 - (d) **If** $\lambda=0$ **then** let $K = t$ and break.
 3. Set $\lambda_0 = 1/2$ and $\lambda_t = \lambda_t/2$ for $t \in [K]$.
Output permutations $\sigma^0, \dots, \sigma^K$ and $\lambda_0, \dots, \lambda_K$.
-

Lemma 5 *At any iteration t in Decomposition, \mathbf{p}^t satisfies that $p_i^t \geq p_{i+1}^t$ for any $i \in [n-1]$.*

Proof. We give an inductive proof with respect to t . In case of $t = 1$, it is clear. In case of $t > 1$, we assume $p_i^{t-1} \geq p_{i+1}^{t-1}$ holds for any $i \in [n-1]$. If $p_i^{t-1} = p_{i+1}^{t-1}$, then $q_i^{t-1} = q_{i+1}^{t-1}$ holds, from the definition of \mathbf{q}^{t-1} . Thus

$$p_{\sigma(i)}^t = p_i^{t-1} - \lambda_{t-1} q_i^{t-1} = p_{i+1}^{t-1} - \lambda_{t-1} q_{i+1}^{t-1} = p_{i+1}^t$$

and we obtain the claim. If $p_i^{t-1} > p_{i+1}^{t-1}$, then $q_i^{t-1} > q_{i+1}^{t-1}$ holds, and

$$p_i^{t+1} - p_{i+1}^{t+1} = p_i^t - p_{i+1}^t - \lambda_t (q_i^t - q_{i+1}^t) = (q_i^t - q_{i+1}^t) \left(\frac{p_i^t - p_{i+1}^t}{q_i^t - q_{i+1}^t} - \lambda_t \right) \geq 0$$

where the last inequality becomes from the definition of λ_t , followed by

$$\lambda_t \leq \min_{i \in [n-1]} \left\{ (p_{i+1}^t - p_i^t) / (q_{i+1}^t - q_i^t) \mid q_{i+1}^t \neq q_i^t \right\}.$$

□

Lemma 6 *In Decomposition, $\mathbf{p}^{K+1} (= \mathbf{p}^K - \lambda^K \mathbf{q}^K) = 0$ holds.*

Proof. Without loss of generality, we may assume that $p_1 \geq p_2 \geq \dots \geq p_n$, for simplicity of notations. First we show $\mathbf{p}^{K+1} \geq 0$. Since Lemma 5, if there exists $j \in [n]$ satisfying that $p_j^{K+1} < 0$, then $p_n^{K+1} < 0$ holds. Thus it is enough to show $p_n^{K+1} \geq 0$. Let $i^* = \min\{j \in [n] \mid p_j^K = p_n^K\}$. Then we have $p_{i^*}^K = p_{i^*+1}^K = \dots = p_n^K$ and $q_{i^*}^K = q_{i^*+1}^K = \dots = q_n^K$. Hence, we get $p_{i^*}^{K+1} = p_{i^*+1}^{K+1} = \dots = p_n^{K+1}$. In case of $i^* \geq 2$, $p_{i^*-1}^t > p_{i^*}^t$ holds for any $t \in [K]$, meaning that $q_{i^*-1}^t > q_{i^*}^t$ holds

for any $t \in [K]$. Thus we can see that $\sum_{j=i^*}^n q_j^t = \sum_{j=i^*}^n (n+1-j)$ holds for any $t \in [K]$, from the definition of \mathbf{q}^t . Then we obtain

$$\sum_{j=i^*}^n \sum_{t=1}^K \lambda_t q_j^t = \sum_{t=1}^K \lambda_t \sum_{j=i^*}^n q_j^t = \sum_{t=1}^K \lambda_t \sum_{j=i^*}^n (n+1-j) = \sum_{j=i^*}^n (n+1-j) \leq \sum_{j=i^*}^n p_j$$

where the last inequality is due to constraints of the permutahedron $\sum_{j=1}^{i^*-1} p_j \leq \sum_{j=1}^{i^*-1} (n+1-j)$ and $\sum_{j=1}^n p_j = \sum_{j=1}^n (n+1-j)$. Thus we obtain that $\sum_{j=i^*}^n p_j^{K+1} = \sum_{j=i^*}^n (p_j - \sum_{t=1}^K \lambda_t q_j^t) \geq 0$. As discussed above, $p_{i^*}^{K+1} = p_{i^*+1}^{K+1} = \dots = p_n^{K+1}$ holds, and we obtain $p_n^{T+1} \geq 0$. In case of $i^* = 1$, the proof is done in a similar way.

Now we show $\mathbf{p}^{K+1} = 0$. Since $\mathbf{p} \in P_n$, $\sum_{j=1}^n p_j^{K+1} = \sum_{j=1}^n (n+1-j)$ holds. In a similar way as the proof of $\mathbf{p}^{K+1} \geq 0$,

$$\sum_{j=1}^n \sum_{t=1}^K \lambda_t q_j^t = \sum_{t=1}^K \lambda_t \sum_{j=1}^n q_j^t = \sum_{t=1}^K \lambda_t \sum_{j=1}^n (n+1-j) = \sum_{j=1}^n (n+1-j).$$

Since $\mathbf{p}^{K+1} \geq 0$, $\mathbf{p}^{K+1} = \mathbf{p} - \sum_{t=1}^K \lambda_t \mathbf{q}^t = 0$. \square

Lemma 7 *The number of iterations K is at most n .*

Proof. From the definition of λ_t , there is at least one $i \in [n]$ satisfying that $p_i^t > p_{i+1}^t$ and $p_i^{t+1} = p_{i+1}^{t+1}$. If $p_i^t = p_{i+1}^t$, then $p_i^{t+1} = p_{i+1}^{t+1}$ as discussed in the proof of Lemma 5. Now the claim is clear. \square

Proof of Lemma 4. Since Lemma 6, it is clear that the output $\sum_{t=0}^K \lambda_t \boldsymbol{\sigma}^t$ by Decomposition is equal to an arbitrarily given $\mathbf{p} \in P_n$.

It is not difficult to see that every lines in Decomposition is done in $O(n)$. Hence, we obtain that the running time is $O(n^2)$, since Lemma 7. Finally, we remark that $|\{\boldsymbol{\sigma}^0, \boldsymbol{\sigma}^1, \dots, \boldsymbol{\sigma}^K\}| \leq n+1$ holds, the existence of such representation is suggested by well-known Caratheodory's theorem. \square

Memory-Efficient Implementation of Decomposition Now we discuss an algorithm with $O(n)$ space and in $O(n^2)$ time to obtain a random permutation $\boldsymbol{\pi} \in \{\boldsymbol{\sigma}^0, \boldsymbol{\sigma}^1, \dots, \boldsymbol{\sigma}^K\}$ according to the probability λ_t , using a modified version of Decomposition. Firstly notice that we do not need to memorize $\boldsymbol{\sigma}^t$ in Decomposition to compute λ_s and $\boldsymbol{\sigma}^s$ for $s > t$. Thus two-paths algorithm is easily obtained; Thus generate a random number $\xi \in (0, 1]$, and find $t \in \{1, \dots, K\}$ satisfying that $1 - \sum_{i=1}^{t-1} \lambda_i > \xi \geq 1 - \sum_{i=1}^t \lambda_i$, where $1 - \sum_{i=1}^0 \lambda_i = 1$ for convenience, then $\boldsymbol{\sigma}^t$ is the desired sample in $\{\boldsymbol{\sigma}^0, \dots, \boldsymbol{\sigma}^K\}$ with probability λ_t .

In fact, we can reduce the time complexity of to $O(n \log n)$ using a heap with $O(n)$ space, though we omit the detail here.

3.4 Main Result

Now we are ready to prove the main result. Note that by using the algorithm Decomposition, $E[\sigma_t \cdot \ell_t] = p_t$. So, by Lemma 1, we get the following theorem immediately.

Theorem 1 $E \left[\sum_{t=1}^T \sigma_t \cdot \ell_t \right] \leq \frac{\eta \min_{\sigma \in S_n} \sum_{t=1}^T \sigma \cdot \ell_t + \frac{n(n+1)}{2} \ln n}{1 - e^{-\eta}}.$

In particular, if we set $\eta = 2 \ln(1 + \sqrt{\ln n / \sqrt{T}})$, since, $\eta \leq e^{\frac{\eta}{2}} - e^{-\frac{\eta}{2}}$ for $\eta \geq 0$, we have $\frac{\eta}{1 - e^{-\eta}} \leq e^{\frac{\eta}{2}} = (1 + \sqrt{\ln n / \sqrt{T}})$, and $\frac{1}{1 - e^{-\eta}} = \frac{(1 + 1/\sqrt{T})^2}{1/T + 2/\sqrt{T}} \leq 1 + \frac{1}{2} \sqrt{T / \ln n}$. Further, by using the fact that $\sigma \cdot \ell_t \leq n(n+1)/2$, we get the following corollary.

Corollary 2 For $\eta = 2 \ln(1 + \sqrt{\ln n / \sqrt{T}})$, the expected regret of PermutahedLearn is at most $\frac{3n(n+1)}{4} \sqrt{T \ln n}$.

It can be shown that the regret bound of PermELearn is at most $n(n+1/2)\sqrt{T \ln n}$, which is roughly 4/3 times worse than that of PermtahedLearn. The proof is omitted due to the page constraint.

4 Experimental Results

In this section, we show our initial experiments of our algorithms for artificial data. For our artificial data, we fix $n = 10$. To generate a loss vector at each trial t , we specify each i -the element $\ell_{t,i}$ of the loss vector ℓ_t independently randomly as follows: Let $\ell_{t,i} = 1$ with probability r_i and $\ell_{t,i} = 0$, otherwise. Here, we set $r_i = i/n$ so that $E[\ell_t] = (1/n, 2/n, \dots, 1)$. We generate $T = 600$ random loss vectors.

The algorithms we compare are PermtahedLearn, PermELearn and the best permutation in hindsight. As the parameter η , we consider $\eta \in \{0.025, 0.05, 0.1, 0.2\}$. For each setting of η , we run algorithms for 3 times and choose the one attaining the lowest average cumulative losses as the best parameters for each of them. As a result, we specify $\eta = 0.2$ for PermtahedLearn and $\eta = 0.1$ for PermELearn, respectively.

We plot the regrets of algorithms with their best parameters in Fig.1. As can be seen, the regret of PermutahedLearn is much smaller than that of PermELearn. This may be due to the fact that the regret bound of PermutahedLearn is constant times smaller than PermELearn.

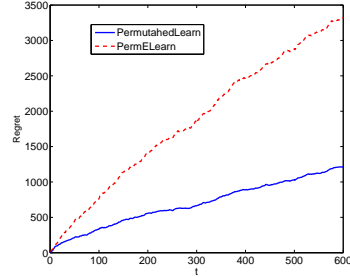


Fig. 1. Regrets of PermtahedLearn and PermELearn.

5 Conclusion

In this paper, we propose an efficient algorithm for an online linear optimization problem over permutations of n items. Our algorithm has the best regret bound so far, runs in time $O(n^2)$ and uses $O(n)$ space at each trial.

An interesting future work includes proving a matching lower bound of the regret and investigating the case where permutations to predict have to meet some partial-order constraints.

Acknowledgement

We thank anonymous reviewers for their helpful comments. This work is supported in part by MEXT Grand-in-Aid for Young Scientists (B) 21700171.

References

1. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
2. L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Physics*, 7:200–217, 1967.
3. N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
4. S. Fujishige. *Submodular functions and optimization*. Elsevier Science, 2nd edition, 2005.
5. E. Hazan. A survey: The convex optimization approach to regret minimization. <http://www.cs.princeton.edu/~ehazan/papers/OCO-survey.pdf>, 2009.
6. D. P. Helmbold and M. K. Warmuth. Learning permutations with exponential weights. *Journal of Machine Learning Research*, 10:1705–1736, 2009.
7. E. L. Lawler. On sequencing jobs to minimize weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics* 2, 2:75–90, 1978.
8. J. Lenstra and A. R. Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26:22–35, 1978.
9. M. Queyranne and Y. Wang. Single-machine scheduling polyhedra with precedence constraints. *Mathematics of Operations Research*, 16(1):1–20, 1991.
10. A. von Arnim, U. Faigle, and R. Schrader. The permutahedron of series-parallel posets. *Discrete Applied Mathematics*, 28(1):3–9, 1990.
11. A. von Arnim and A. S. Schulz. Facets of the generalized permutahedron of a poset. *Discrete Applied Mathematics*, 72:179–192, 1997.
12. G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics 152. Springer-Verlag, 1995.