Online Prediction under Submodular Constraints

Suehiro, Daiki
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

出版情報：2012-10. Springer
バージョン：
権利関係：

Online Prediction under Submodular Constraints

Daiki Suehiro[1], Kohei Hatano[1], Shuji Kijima[1], Eiji Takimoto[1], and Kiyohito Nagano[2]

[1] Department of Informatics, Kyushu University
[2] University of Tokyo
{daiki.suehiro, hatano, kijima, eiji}@inf.kyushu-u.ac.jp[1]
nagano@sat.t.u-tokyo.ac.jp[2]

Abstract. We consider an online combinatorial prediction problem where each combinatorial concept is represented as a vertex of a polyhedron described by a submodular function (base polyhedron). In general, there are exponentially many vertices in the base polyhedron. We propose polynomial time algorithms with regret bounds. In particular, for cardinality-based submodular functions, we give $O(n^2)$-time algorithms.

1 Introduction

Online learning of combinatorial or structured concepts have gained much attention these days [8, 2, 11, 15]. Such combinatorial concepts includes shortest paths, k-sets, spanning trees, permutations, and so on. In typical settings, we assume a finite set $\mathcal{C}$ of combinatorial concepts where each concept can be represented as a vector in $\mathbb{R}^n$ for some fixed n, i.e., $\mathcal{C} \subseteq \mathbb{R}^n$. Then we consider the following protocol: For each trial $t = 1, \ldots, T$, (i) the player predicts $\boldsymbol{c}_t \in \mathcal{C}$, (ii) the adversary returns a loss vector $\boldsymbol{\ell}_t \in [0, 1]^n$, and (iii) the player incurs loss $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$. The goal of the player is to minimize the regret: $\sum_{t=1}^{T} \boldsymbol{c}_t \cdot \boldsymbol{\ell}_t - \min_{\boldsymbol{c} \in \mathcal{C}} \sum_{t=1}^{T} \boldsymbol{c} \cdot \boldsymbol{\ell}_t$.

There are some approaches to attack this type of problems. A naive approach to minimize the regret in the above problem is to apply Hedge algorithm [6]. Hedge algorithm combines experts predictions, where each expert corresponds to each concept in $\mathcal{C}$. In general, however, the size of $\mathcal{C}$ is exponentially large w.r.t. n. Therefore, a straightforward implementation of Hedge algorithm is inefficient. There are some efficient online prediction methods for combinatorial concepts, for example, PermELearn [8] and Component Hedge [11] and Comband in bandit setting [2]. These methods consist of abstract subroutines.

Among subroutines, projection and decomposition are important and used in many online learning algorithms (see, e.g., [17, 8, 11]). Here, the projection routine, given a point, outputs its projection onto the convex hull of combinatorial concepts and the decomposition routine, given finds a liner combination of combinatorial concepts given a point of the convex hull. So far, for particular combinatorial concepts, we need to design projection and decomposition subroutines individually.

In this paper, we investigate a unified and efficient projection and decomposition algorithms for a wide class of combinatorial concepts. The class we consider is the set of vertices (extreme points) of a polyhedron described by a submodular function f. In submodular function literature, the polyhedron is called (submodular) *base polyhedron* and denoted as $B(f)$ (we will give the definition later). That is, we consider the situation where $\mathcal{C}$ is the set of extreme points in $B(f)$. The base polyhedron $B(f)$ is defined using 2^n linear constraints and it is known that there are at most $n!$ vertices [7]. Examples of our problems include experts, k-sets [14], permutahedron [15], spanning trees [2, 11], truncated permutahedron, and k-forest. To the best of our knowledge, the last two problems are new for the online learning literature.

We propose projection and decomposition algorithms for the base polyhedron $B(f)$. The running times of the algorithms are both $O(n^6 + n^5 EO))$, where EO denotes the unit time to evaluate the submodular function. Furthermore, for cardinality-based submodular functions, we derive $O(n^2)$-time projection and decomposition algorithms. Such examples include k-sets and (truncated) permutahedron.

Our projection algorithms are designed for Euclidean distance and unnormalized relative entropy. So, we can combine them with Online Gradient Descent(OGD) [17] or Hedge [6], respectively. Combined with our projection and decomposition algorithms for $B(f)$, their regret bounds become $O(D_{euc}\sqrt{nT})$ and $O(\sqrt{L^* f([n]) \ln n} + f([n]) \ln n)$, respectively, where $D_{euc} = \max_{\boldsymbol{c},\boldsymbol{c}' \in B(f)} \|\boldsymbol{c} - \boldsymbol{c}'\|_2$, and $L^* = \min_{\boldsymbol{c} \in B(f)} \sum_{t=1}^{T} \boldsymbol{c} \cdot \ell_t$.

Our contribution is to provide a unified view and efficient prediction strategies for an online prediction problem with exponentially many candidates by using rich theory of submodular function. Further, our $O(n^2)$-time algorithms for cardinality-based submodular functions are non-trivial for submodular optimization as well.

We discuss the relationship between previous and our results. First, we compare Follow the perturbed leader (FPL, [10]) with our algorithms. FPL uses an algorithm which solves "offline" linear optimization. It is well known that linear optimization over the base polyhedron is tractable and solved in $O(n \log n)$ time [4, 7]. So, the running time of FPL for our problem is $O(n \log n)$ at each trial. On the other hand, the regret of FPL is $O(D_1 \sqrt{nT})$, where $D_1 = \max_{\boldsymbol{c},\boldsymbol{c}' \in \mathcal{C}} \|\boldsymbol{c} - \boldsymbol{c}'\|_1$, which is worse than ours.

Next, we consider an algorithm proposed by [9] which converts an offline linear approximate optimization algorithm into the online one. This algorithm has an approximate projection subroutine. But the running time of the projection subroutine is $O(Tn \log n)$, which depends on T.

Component Hedge (CH, [11]) is also an efficient algorithm for predicting among exponentially many combinatorial concepts. CH represents a combinatorial concept as a matrix and solves an entropy minimization problem with linear constraints at each trial. CH has more known applications such as directed spanning trees, paths and so on. However, the class of concepts for which CH can deal with seems incomparable with ours. In an algorithmic sense, our algorithm

has advantages for some concepts. For example, for permutahedron and its truncated one, it can be shown that CH requires $O(n^2)$ memory whereas ours uses $O(n)$ memory (see [15] for related discussion).

2 Preliminaries

For any fixed positive integer n, we denote by $[n]$ the set $\{1, \ldots, n\}$. A function $f : 2^{[n]} \to \mathbb{R}$ is *submodular* if for any $A, B \subset [n]$, $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$. For simplicity, we assume that $f(\emptyset) = 0$. Given a submodular function f, *the base polyhedron* is defined as

$$B(f) = \left\{ \boldsymbol{x} \in \mathbb{R}^n \mid \sum_{i \in S} x_i \leq f(S), \text{ for any } S \subset [n], \text{and } \sum_{i=1}^{n} x_i = f([n]) \right\}.$$

A point in $B(f)$ is an *extreme point* if it is not represented as a convex combination of other two points in $B(f)$. Let $\mathcal{C}$ be the set of extreme points in $B(f)$. In general, there can be exponentially many extreme points in $B(f)$. In this paper, for any submodular function f, we assume an oracle that returns the value $f(S)$ for any input S.

2.1 Examples

We illustrate some examples of our problems. In particular, the last two problems are new applications which are not previously studied.

Experts problem The classical expert problem [6] is an example of our problem. In the expert problem, we are given n experts and the player would like to predict as well as the best expert in hindsight. Here each expert i is represented as n-dimensional unit vector $\boldsymbol{e}_i$ whose i-th component is 1 and other components are 0. Then, the corresponding submodular function f is the constant function $f(S) = 1$, which is submodular.

k-sets The problem of k-sets is a generalization of Experts problem, where each combinatorial concept corresponds to a set of k experts among n experts. This problem was first considered by [14]. Each k-set is represented as a sum of k different unit vectors. Then, $\mathcal{C} = \{\boldsymbol{x} \in \{0,1\}^n \mid \sum_i^n x_i = k\}$. Let $f : 2^{[n]} \to \mathbb{R}$ such that $f(S) = g(|S|)$, where $g(i) = i$, if $i \leq k$ and $g(i) = k$, if $i > k$. This function is submodular since any concave function of $|S|$ is submodular (see, e.g., [7]).

Spanning trees Online prediction problems of undirected or directed spanning trees are studied in [2] and [11]. In this paper, we consider undirected spanning trees. Let $G = (V, E)$ be an undirected graph. Let $f : 2^E \to R$ such that $f(A) = |V(A)| - s(A)$, where $V(A)$ is the set of vertices of the subgraph induced by the set A of edges, and $s(A)$ is the number of the connected components of the subgraph [5, 3]. Especially, the base polyhedron is called spanning tree polyhedron and $\mathcal{C} = \{\boldsymbol{x} \in \{0,1\}^{|E|} \mid \text{the set of edges} \{e \mid x_e = 1\} \text{ forms a spanning tree of } G\}$.

Permutahedron Let $\mathcal{C} = \{(i_1, \ldots, i_n) \mid (i_1, \ldots, i_n)$ is a permutation of $\{1, \ldots, n\}\}$. Each permutation corresponds to an element of S_n. The corresponding submodular function is $f(S) = \sum_{i=1}^{|S|}(n+1-i)$. The base polyhedron $B(f)$ is called *Permutahedron* (see, e.g., [16, 7]). This concept class relates to an online scheduling problem of n jobs with a single processor where the sum of flow time of each job is to be minimized [15] . A different representation of permutations and the related problem was also considered by [8].

Truncated permutahedron For $k < n$, let $\mathcal{C} = \{(i_1, \ldots, i_n) \mid (i_1, \ldots, i_n)$ is a permutation of $1, 2, \ldots, n-k$,and k $(n-k)$s$\}$. For example, $(2, 2, 2, 1)$ is a member of $\mathcal{C}$ for $n = 4$ and $k = 2$. The corresponding function is $f(S) = (n-k)|S|$ if $|S| \leq k$ and $f(S) = (n-k)k + \sum_{j=k+1}^{|S|}(n+1-j)$, otherwise. This concept class is also related to a generalized version of the online scheduling problem [15] where the flow times of the first k jobs are neglected.

k-forest Let $\mathcal{C}$ denote the set of k-forests in a graph $G = (V, E)$, where $F \subset E$ is a k-forest if $|F| = k$ and F does not contain a cycle in G. It is known that $\mathcal{C}$ is a bases family of a truncation of a graphic matroid, that is known to be another matroid. The corresponding function is $f(X) = \min\{k, \max\{|F| \mid F \subseteq X$ is a forest$\}\}$.

2.2 Extreme points of the base polyhedron

In this subsection, we will see the correspondence between the permutations of $[n]$ and the extreme points of the base polyhedron $B(f)$.

Given a permutation $\boldsymbol{\sigma} = (i_1, \ldots, i_n)$ of $[n] = \{1, \ldots, n\}$, the greedy algorithm of [4] generates a point $\boldsymbol{c}^{\boldsymbol{\sigma}} \in \mathbb{R}^n$ determined by

$$c^{\boldsymbol{\sigma}}_j = f(\{j' \in [n] : i_{j'} \leq i_j\}) - f(\{j' \in [n] : i_{j'} < i_j\}) \text{ for each } j \in [n].$$

Then $\boldsymbol{c}^{\boldsymbol{\sigma}}$ is an extreme point of $B(f)$. We will say that $\boldsymbol{c}^{\boldsymbol{\sigma}}$ is an extreme point of $B(f)$ generated by $\boldsymbol{\sigma}$. Conversely, for each extreme point $\boldsymbol{c}$ of $B(f)$, there is a permutation that generates $\boldsymbol{c}$.

2.3 Bregman divergence

Let $\Phi : \Gamma \to \mathbb{R}$ be a strictly convex function defined on a closed convex set $\Gamma \subseteq \mathbb{R}^n$. The Bregman divergence Δ_Φ with respect to Φ is defined as $\Delta_\Phi(\boldsymbol{p}, \boldsymbol{q}) = \Phi(\boldsymbol{p}) - \Phi(\boldsymbol{q}) - \nabla\Phi(\boldsymbol{q}) \cdot (\boldsymbol{p} - \boldsymbol{q})$. The function Φ is *separable* if there exists functions $\phi_i : \Gamma_i \to \mathbb{R}$ for $i = 1, 2., \ldots, n$ such that $\Gamma = \Gamma_1 \times \Gamma_2 \times \cdots \times \Gamma_n$ and for any $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \Gamma$, $\Phi(\boldsymbol{x}) = \sum_{i=1}^n \phi_i(x_i)$. In particular, if all ϕ_i's are the same, then the function Φ is said to be *uniformly separable*. In this paper, we will sometimes consider two particular uniformly separable convex functions, the *2-norm function*: $\Phi_{\mathrm{EUC}}(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$ defined on $\mathbb{R}^n$, and the *unnormalized negative entropy function*: $\Phi_{\mathrm{URE}}(\boldsymbol{x}) = \sum_{i=1}^n x_i \ln x_i - \sum_{i=1}^n x_i$ defined on $\mathbb{R}^n_{>0}$.

It is well known that these functions define the Euclidean distance and the unnormalized relative entropy, respectively. That is, $\Delta_{\mathrm{EUC}}(\boldsymbol{x},\boldsymbol{z}) \stackrel{\mathrm{def}}{=} \Delta_{\Phi_{\mathrm{EUC}}}(\boldsymbol{x},\boldsymbol{z}) = \frac{1}{2}\sum_{i=1}^{n}(x_i - z_i)^2$, and $\Delta_{\mathrm{URE}}(\boldsymbol{x},\boldsymbol{z}) \stackrel{\mathrm{def}}{=} \Delta_{\Phi_{\mathrm{URE}}}(\boldsymbol{x},\boldsymbol{z}) = \sum_{i=1}^{n} x_i \ln \frac{x_i}{z_i} + \sum_{i=1}^{n} z_i - \sum_{i=1}^{n} x_i$.

3 Algorithm

In this section, we propose an algorithm that predicts extreme points of the base polyhedron $B(f)$ and prove its regret bounds.

3.1 Main Structure

The main structure of the algorithm we use is shown in Algorithm 1. The algorithm is Regularized Follow the Leader (RFTL) [**?**], which is a generalization of Hedge or Online Gradient Decent(OGD, [17]) using Bregman divergence, combined with our subroutines $\mathrm{Projection}_{\Phi}$ and Decomposition.

At each trial t, our version of RFTL runs Decomposition and get $\boldsymbol{c}_t \in \mathcal{C}$ randomly so that $\mathbf{E}[\boldsymbol{c}_t] = \boldsymbol{x}_t$. Then, RFTL updates $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+\frac{1}{2}}$. Finally, RFTL computes the projection $\boldsymbol{x}_{t+1}$ of $\boldsymbol{x}_{t+\frac{1}{2}}$ onto the base polyhedron $B(f)$ using the procedure $\mathrm{Projection}_{\Phi}$. Using RFTL itself is standard, but we need to design efficient procedures for projection and decomposition.

Algorithm 1 RFTL with Projection and Decomposition

1. Let $\boldsymbol{x}_1$ be any point in $B(f)$.
2. For $t = 1, \ldots, T$
   (a) Run **Decomposition**$(\boldsymbol{x}_t)$ and get $\boldsymbol{c}_t \in \mathcal{C}$ randomly so that $\mathbf{E}[\boldsymbol{c}_t] = \boldsymbol{x}_t$.
   (b) Predict $\boldsymbol{c}_t$ and incur a loss $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$.
   (c) Update $\boldsymbol{x}_{t+\frac{1}{2}}$ as $\boldsymbol{x}_{t+\frac{1}{2}} = \nabla\Phi^{-1}(\nabla\Phi(\boldsymbol{x}_t) - \eta\boldsymbol{\ell}_t)$.
   (d) Run **Projection**$_{\Phi}(\boldsymbol{x}_{t+\frac{1}{2}})$ and get $\boldsymbol{x}_{t+1}$, the projection of $\boldsymbol{x}_{t+\frac{1}{2}}$ onto the base polyhedron $B(f)$. That is, $\boldsymbol{x}_{t+1} = \arg\inf_{\boldsymbol{x}\in B(f)} \Delta_{\Phi}(\boldsymbol{x}, \boldsymbol{x}_{t+\frac{1}{2}})$.

The following theorem is known.

Theorem 1 ([6, 17, ?]) *Let $\mathcal{C}$ be the set of extreme points in $B(f)$.*

1. *For $\Phi = \Phi_{EUC}$, the expected regret of RFTL is $O(D_{euc}\sqrt{nT})$ for some η, where $D_{euc} = \max_{\boldsymbol{c},\boldsymbol{c}'\in\mathcal{C}} \|\boldsymbol{c} - \boldsymbol{c}'\|_2$.*
2. *For $\Phi = \Phi_{URE}$, the expected regret of RFTL is $O(\sqrt{L^* D_{ure}} + D_{ure})$ for some η and $\boldsymbol{x}_1 \in B(f)$, where $D_{ure} = \max_{\boldsymbol{c}\in\mathcal{C}} \Delta_{\Phi_{URE}}(\boldsymbol{c}, \boldsymbol{x}_1)$ and $L^* = \min_{\boldsymbol{c}\in\mathcal{C}} \sum_{t=1}^{T} \boldsymbol{c}\cdot\boldsymbol{\ell}_t$.*

problem	Hedge	OGD
Experts	$O(\sqrt{L^* \ln n})$	$O(\sqrt{nT})$
k-sets	$O(\sqrt{L^* k \ln(n/k)} + k \ln(n/k))$	$O(\sqrt{knT})$
Spanning Trees	$O(\sqrt{L^* n \ln n} + n \ln n)$	$O(n\sqrt{T})$
Permutahedron	$O(n\sqrt{L^* \ln n} + n^2 \ln n)$	$O(n^2\sqrt{T})$
Truncated Perm.	$O(\sqrt{L^*(n^2-k^2)\ln n} + (n^2-k^2)\ln n)$	$O((n-k)\sqrt{n(n+k)T})$
k-forest	$O(\sqrt{L^* k \ln(n/k)} + k \ln(n/k))$	$O(\sqrt{knT})$

Table 1. The regrets of combinatorial concepts obtained using our projection and decomposition algorithms.

For particular combinatorial concepts, we summarize their regret bounds in Table 1.

To complete our analysis, we specify the procedures Projection$_\Phi$ for separable strictly convex function Φ and Decomposition, respectively, in the following subsections. We will see that both of the two procedures are no harder than the submodular function minimization problem. For a submodular function $f : 2^{[n]} \to \mathbb{R}$ with $f(\emptyset) = 0$, the submodular function minimization (SFM) is a problem of finding a subset $S \subseteq [n]$ with $f(S)$ minimum. Many combinatorial SFM algorithms are known (see [**?**]), and the fastest known strongly polynomial algorithm of [13] runs in $O(n^6 + n^5 EO)$ time, where EO is the unit time to evaluate the value of the submodular function. We will show that both of the procedures Projection$_\Phi$ and Decomposition can be implemented to run in $O(n^6 + n^5 EO)$ time.

3.2 Projection

For any given point $\boldsymbol{z} \in \mathbb{R}^n$, the procedure Projection$_\Phi$ in Algorithm 1 computes the projection of $\boldsymbol{z}$ onto the base polyhedron $B(f)$. We propose an efficient construction of this procedure. Formally, the projection problem is stated as follows:

$$\text{Projection}_\Phi(\boldsymbol{z}) = \arg \inf_{\boldsymbol{x} \in B(f)} \Delta_\Phi(\boldsymbol{x}, \boldsymbol{z}) \quad \text{sub. to: } \sum_{j \in S} x_j \leq f(S),\ \forall S \subset [n], \text{ and } \sum_{j=1}^{n} x_j = f([n]), \quad (1)$$

where $\Phi(\boldsymbol{x})$ is separable. This convex optimization problem with exponentially many constraints can be solved efficiently using the parametric submodular algorithm of [12], which is a parametric extension of the SFM algorithm of [13].

Theorem 2 ([12]) *There is an algorithm that solves problem (1) for separable strictly convex functions Φ in time $O(n^6 + n^5 EO)$.*

3.3 Decomposition

For any given point $\boldsymbol{x}$ in the base polyhedron $B(f) \subseteq \mathbb{R}^n$, the procedure Decomposition in Algorithm 1 finds extreme points $\boldsymbol{c}^{\boldsymbol{\sigma}^1}, \ldots, \boldsymbol{c}^{\boldsymbol{\sigma}^K}$ in $B(f)$ and $\lambda_1, \ldots, \lambda_K \in \mathbb{R}_{>0}$ such that $\sum_{i=1}^K \lambda_i \boldsymbol{c}^{\boldsymbol{\sigma}^i} = \boldsymbol{x}$ and $\lambda_1 + \cdots + \lambda_K = 1$, where each $\boldsymbol{c}^{\boldsymbol{\sigma}^i}$ is an extreme point of $B(f)$ generated by a permutation $\boldsymbol{\sigma}^i$ of $[n]$ via the greedy algorithm of [4]. In other words, this procedure represents $\boldsymbol{x}$ as a convex combination of extreme points of $B(f)$. Carathéodory's Theorem guarantees that $\boldsymbol{x} \in B(f)$ can be represented as a convex combination of at most n extreme points of $B(f)$.

To describe the procedure Decomposition, let us briefly review a common framework of algorithms for SFM. For a submodular function $f' : 2^{[n]} \to \mathbb{R}$ with $f'(\emptyset) = 0$, the result of [4] implies

$$\min_S \{f'(S) : S \subseteq [n]\} = \max_{\boldsymbol{z}} \{\sum_{j=1}^n \min\{0, z_j\} : \boldsymbol{z} \in B(f')\}. \tag{2}$$

In many combinatorial SFM algorithms, including Orlin's algorithm ([13]), we finally obtain a minimizer $S^* \subseteq [n]$ and a maximizer $\boldsymbol{z}^* \in B(f')$ of (2). Moreover, we obtain $\boldsymbol{z}^* \in B(f')$ as a convex combination of at most n extreme points of $B(f')$. By the use of this fact, we can give an efficient construction of the procedure Decomposition.

For a given point $\boldsymbol{x} \in B(f)$, the function $f_{\boldsymbol{x}} : 2^{[n]} \to \mathbb{R}$ defined by $f_{\boldsymbol{x}}(S) = f(S) - \sum_{j \in S} x_j$ $(S \subseteq [n])$ is submodular and satisfies $f_{\boldsymbol{x}}(\emptyset) = 0$. For each permutation $\boldsymbol{\sigma}$ of $[n]$, let $\boldsymbol{c}^{\boldsymbol{\sigma}}$ be extreme points in $B(f)$ generated by $\boldsymbol{\sigma}$, and let $\boldsymbol{c}^{\boldsymbol{\sigma}}_{\boldsymbol{x}}$ be extreme points in $B(f_{\boldsymbol{x}})$ generated by $\boldsymbol{\sigma}$. Then it holds that $\boldsymbol{c}^{\boldsymbol{\sigma}}_{\boldsymbol{x}} = \boldsymbol{c}^{\boldsymbol{\sigma}} - \boldsymbol{x}$. In view of the definition of the base polyhedron, we have that $\min_{S \subseteq [n]} f_{\boldsymbol{x}}(S) = 0$ and the n-dimensional zero vector $\boldsymbol{0}_n$ is in $B(f_{\boldsymbol{x}})$. Therefore, $\boldsymbol{z} = \boldsymbol{0}_n$ is the unique optimal solution to the right hand side of (2) with $f' = f_{\boldsymbol{x}}$.

Now we describe the procedure Decomposition. Initially, we apply some combinatorial SFM algorithm, e. g. Orlin's algorithm ([13]), to the submodular function $f_{\boldsymbol{x}}$. Then we obtain permutations $\boldsymbol{\sigma}^1, \ldots, \boldsymbol{\sigma}^K$ of $[n]$ and $\lambda_1, \ldots, \lambda_K \in \mathbb{R}_{>0}$ such that $\sum_{i=1}^K \lambda_i \boldsymbol{c}^{\boldsymbol{\sigma}^i}_{\boldsymbol{x}} = \boldsymbol{0}$, $\lambda_1 + \cdots + \lambda_K = 1$, and $K \leq n$. As for the function f, these permutations $\boldsymbol{\sigma}^1, \ldots, \boldsymbol{\sigma}^K$ and positive coefficients $\lambda_1, \ldots, \lambda_K$ generate another point $\sum_{i=1}^K \lambda_i \boldsymbol{c}^{\boldsymbol{\sigma}^i}$. For this point, we obtain

$$\sum_{i=1}^K \lambda_i \boldsymbol{c}^{\boldsymbol{\sigma}^i} = \sum_{i=1}^K \lambda_i (\boldsymbol{c}^{\boldsymbol{\sigma}^i}_{\boldsymbol{x}} + \boldsymbol{x}) = \sum_{i=1}^K \lambda_i \boldsymbol{c}^{\boldsymbol{\sigma}^i}_{\boldsymbol{x}} + \sum_{i=1}^K \lambda_i \boldsymbol{x} = \boldsymbol{x}.$$

Thus we have a required representation of $\boldsymbol{x}$. This gives the following.

Theorem 3 *For any $\boldsymbol{x} \in B(f)$, there is an algorithm that gives a convex combination representation of $\boldsymbol{x}$ using at most n extreme points of $B(f)$ in $O(n^6 + n^5 EO)$ time.*

4 Algorithm for cardinality-based submodular functions

In this section, we propose more efficient projection and decomposition algorithms when the underlying submodular function f is cardinality-based, i.e., $f(S) = g(|S|)$ for some $g : \mathbb{N} \to \mathbb{R}$. For projection, however, we only consider the Euclidean distance and the unnormalized relative entropy, rather than any Bregman divergence ∇_Φ for a separable function Φ as in the previous section.

A cardinality-based submodular function f has the following nice property: For any point $\boldsymbol{x} \in B(f)$ and any $i, j \in [n]$, the vector $\boldsymbol{x}'$ obtained by exchanging x_i and x_j in $\boldsymbol{x}$ is also contained in $B(f)$. A submodular function having this property is said to be *exchangeable*.

The following lemma says that for any exchangeable submodular function f, the projection onto $B(f)$ preserves the order of indices of vector with respect to the inequality relation.

Lemma 1 *Let $\boldsymbol{x}^*$ be the projection of $\boldsymbol{z}$ in (1) under the Bregman divergence ∇_Φ for a strictly convex and uniformly separable function Φ. Assume that the submodular function f is exchangeable and $z_1 \geq \cdots \geq z_n$. Then, it holds that $x_1^* \geq x_2^* \geq \cdots \geq x_n^*$.*

Proof. Suppose on the contrary that $x_i^* < x_j^*$ for some $i < j$. Let $\widehat{\boldsymbol{x}}$ be the point obtained by exchanging x_i^* and x_j^* in $\boldsymbol{x}^*$. Then, by definition, we have $\widehat{\boldsymbol{x}} \in B(f)$. Furthermore, observe that

$$\begin{aligned}\Delta_\Phi(\boldsymbol{x}^*, \boldsymbol{z}) - \Delta_\Phi(\hat{\boldsymbol{x}}, \boldsymbol{z}) =& \Phi(\boldsymbol{x}^*) - \Phi(\hat{\boldsymbol{x}}) - \nabla\Phi(\boldsymbol{z}) \cdot (\boldsymbol{x}^* - \hat{\boldsymbol{x}}) \\ =& \phi(x_i^*) + \phi(x_j^*) - \phi(x_i^*) - \phi(x_j^*) \\ & - (\phi'(z_i)(x_i^* - x_j^*) - \phi'(z_j)(x_j^* - x_i^*)) \\ =& (x_j^* - x_i^*) \cdot (\phi'(z_i) - \phi'(z_j)) \geq 0,\end{aligned}$$

which contradicts the assumption that $\boldsymbol{x}^*$ is the projection. □

In the following, we assume that $z_1 \geq \cdots \geq z_n$ without loss of generality (this can be achieved by sorting). Lemma 1 implies that for any $S \subseteq [n]$, $\sum_{i \in S} x_i^* \leq \sum_{j=1}^{|S|} x_j^*$, which means that, if the right hand side is bounded by $f(S) = g(|S|)$, the left hand side is also bounded by $g(|S|)$. Therefore, the projection problem (1) is equivalent to the following problem with only n constraints:

$$\begin{aligned}&\min_{\boldsymbol{x}} \Delta_\Phi(\boldsymbol{x}, \boldsymbol{z}) \\ \text{sub.to: } &\sum_{i=1}^{j} x_i \leq g(j), \text{ (for } j = 1, \ldots, n-1), \text{ and } \sum_{i=1}^{n} x_i = g(n). \end{aligned} \tag{3}$$

Now we propose an efficient implementation of Projection$_\Phi$ that solves the problem (3).

Algorithm 2 Projection under Euclidean distance

Input: $\boldsymbol{z} \in \mathbb{R}^n$ satisfying that $z_1 \geq z_2 \geq \cdots \geq z_n$.
Output: projection $\boldsymbol{x}$ of $\boldsymbol{z}$ onto $B(f)$.

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

4.1 Projection under Euclidean distance

First we give an algorithm which computes $\text{Projection}_{\Phi_{\text{EUC}}}$ under Euclidean distance. We show the algorithm in Algorithm 2. Then we prove the following.

Theorem 4 *(i) Given $\boldsymbol{z}$, Algorithm 2 outputs the projection of $\boldsymbol{x}$ onto the base polyhedron $B(f)$. (ii) The time complexity of Algorithm 2 is $O(n^2)$.*

Proof. By KKT condition(see, e.g., [1]), $\boldsymbol{x}^*$ is the solution of the problem (3) if and only if there exists $\alpha_1, \ldots, \alpha_{n-1}$ and η such that

$$
\begin{aligned}
& x_i^* = z_i - \sum_{j=1}^{i} \alpha_j - \eta, \ (\text{for } i = 1, \ldots, n-1), \ \text{and } x_n^* = z_n - \eta, \\
& \sum_{i=1}^{n} x_i^* = g(n), \\
& \alpha_i (\sum_{j=1}^{i} x_j^* - g(i)) = 0, \ \alpha_i \geq 0, \ \sum_{j=1}^{i} x_j^* \leq g(i) \ (\text{for } i = 1, \ldots, n-1). \qquad (4)
\end{aligned}
$$

Now we show that there indeed exists $\alpha_1, \ldots, \alpha_{n-1}$ such that the output $\boldsymbol{x}$ of $\text{Projection}_{\Phi_{EUC}}(\boldsymbol{z})$ satisfies the optimality conditions (4), which suffices to prove the first statement. To do so, first we show that $C^{t-1}(i_{t-1}) \leq C^t(i_t)$ for each iteration t. By the definition of $C^{t-1}(i_{t-1})$, we have $C^{t-1}(i_{t-1}) \leq C^{t-1}(i_t)$. Observe that

$$
\begin{aligned}
C^{t-1}(i_t) &= \frac{g(i_t) - g(i_{t-2}) - \sum_{j=i_{t-2}+1}^{i_t} z_j}{i_t - i_{t-2}} \\
&= \frac{g(i_t) - g(i_{t-1}) - \sum_{j=i_{t-1}+1}^{i_t} z_j + g(i_{t-1}) - g(i_{t-2}) - \sum_{j=i_{t-2}+1}^{i_{t-1}} z_j}{(i_t - i_{t-1}) + (i_{t-1} - i_{t-2})} \\
&= \frac{(i_t - i_{t-1})(C^t(i_t)) + (i_{t-1} - i_{t-2})(C^{t-1}(i_{t-1}))}{(i_t - i_{t-1}) + (i_{t-1} - i_{t-2})}.
\end{aligned}
$$

Since $C^{t-1}(i_{t-1}) \leq C^{t-1}(i_t)$,

$$\frac{(i_t - i_{t-1})(C^{t-1}(i_{t-1}))}{(i_t - i_{t-1}) + (i_{t-1} - i_{t-2})} \leq \frac{(i_t - i_{t-1})(C^{t}(i_{t}))}{(i_t - i_{t-1}) + (i_{t-1} - i_{t-2})}.$$

By simplifying this, we get $C^{t-1}(i_{t-1}) \leq C^t(i_t)$, as desired.

Then we determine each α_{i_t} so that $-\alpha_{i_t} + C^{t+1}(i_{t+1}) = C^t(i_t)$, i.e., $\alpha_{i_t} = C^{t+1}(i_{t+1}) - C^t(i_t)$ and fix η to be $C^T(n)$, where T satisfies $i_T = n$. Note that since $C^t(i_t) \leq C^{t+1}(i_{t+1})$, each α_{i_t} is strictly positive. For other $i \notin \{i_1, \ldots, i_T\}$, we set $\alpha_i = 0$. Then, each x_{i_t} can be expressed as

$$x_{i_t} = z_i + C^t(i_t) = z_i - (\alpha_{i_t} + \alpha_{i_{t+1}} + \cdots + \alpha_{i_T}) - \eta = z_i - (\alpha_{i_t} + \alpha_{i_t+1} + \cdots + \alpha_{i_{n-1}}) - \eta.$$

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

$$x_i = z_i + C^t(i_t) = z_i - (\alpha_{i_t} + \alpha_{i_t+1} + \cdots + \alpha_{n-1}) - \eta = z_i - (\alpha_i + \alpha_{i+1} + \cdots + \alpha_{n-1}) - \eta.$$

To check if the specified α_is and η satisfies the optimality conditions (4), observe that (i) for each i_t,

$$\sum_{j=1}^{i_t} x_j = \sum_{j=1}^{i_{t-1}} x_j + \sum_{j=i_{t-1}+1}^{i_t} (z_j + C^t(i_t)) = g(i_{t-1}) + (g(i_t) - g(i_{t-1})) = g(i_t)$$

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

$$\begin{aligned}\sum_{j=1}^{i} x_j = \sum_{j=1}^{i_{t-1}} x_j + \sum_{j=i_{t-1}+1}^{i} (z_j + C^t(i_t)) &\leq \sum_{j=1}^{i_{t-1}} x_j + \sum_{j=i_{t-1}+1}^{i} (z_j + C^t(i)) \\ &= g(i_{t-1}) + (g(i) - g(i_{t-1})) = g(i)\end{aligned}$$

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 proves the second statement of the lemma. □

4.2 Projection under unnormalized relative entropy

Next we propose an algorithm for Projection$_{\Phi_{URE}}$. We construct the projection algorithm by generalizing the one used for permutahedron [15] . Note that the algorithm is also a generalization of the capping algorithm in [14]. The algorithm shown in Algorithm 3 outputs the solution which satisfies the optimality conditions, and following theorem holds.

Theorem 5 *(i) Given $\boldsymbol{z}$, the Algorithm 3 outputs the projection of $\boldsymbol{x}$ onto the base polyhedron $B(f)$. (ii) The time complexity of Algorithm 3 is $O(n^2)$.*

The proof is also a generalization of the proof in [15] and omitted due to the space constraints.

Algorithm 3 Projection under unnormalized relative entropy

Input: $\boldsymbol{z} \in \mathbb{R}^n$ satisfying that $z_1 \geq z_2 \geq \cdots \geq z_n$.
Output: projection $\boldsymbol{x}$ of $\boldsymbol{z}$ onto $B(f)$.

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

4.3 Decomposition

In this subsection, we describe how to represent a point $\boldsymbol{x} \in B(f)$ by a convex combination of extreme points of $B(f)$. More precisely, we are concerned with the following randomized rounding problem; given a point $\boldsymbol{x} \in B(f)$, output an extreme point X of $B(f)$ with a probability such that $E[X] \stackrel{\text{def}}{=} \sum_{j=1}^{k} \Pr\left[X = \boldsymbol{c}^j\right] \cdot \boldsymbol{c}^j = \boldsymbol{x}$ for an appropriate $k > 0$.

As a preliminary step, we explain the following Propositions 6, 7, and 8, which are well-known facts (see e.g., [7]). Let $a \in \mathbb{R}_{>0}$ be a constant satisfying $a > g(n-1) - g(n)$, and we define $\tilde{f}\colon 2^{[n]} \to \mathbb{R}$ by $\tilde{f}(S) \stackrel{\text{def}}{=} f(S) + a|S|$ for any $S \subseteq [n]$. Notice that $\tilde{f}$ is clearly a cardinality based function; let $\tilde{g}(z) \stackrel{\text{def}}{=} g(z) + a \cdot z$ then $\tilde{f}(S) = \tilde{g}(|S|)$ holds.

Proposition 6 *The function $\tilde{f}$ is cardinality based* submodular*, as well as* monotone increasing*, i.e., $\tilde{g}(i) < \tilde{g}(i+1)$ for each $i \in [n-1]$.*

Note that $\tilde{f}(\emptyset) = 0$, and $\tilde{f}(S) > 0$ hold for any S ($\emptyset \subset S \subseteq [n]$).

Proposition 7 *A point $\boldsymbol{x}$ is in $B(f)$ if and only if $\tilde{\boldsymbol{x}} \stackrel{\text{def}}{=} \boldsymbol{x} + a\mathbf{1}$ is in $B(\tilde{f})$. A point $\boldsymbol{c}$ is an extreme point of $B(f)$ if and only if $\tilde{\boldsymbol{c}} \stackrel{\text{def}}{=} \boldsymbol{c} + a\mathbf{1}$ is an extreme point of $B(\tilde{f})$.*

Proposition 8 *Suppose $\boldsymbol{x} \in B(f)$ satisfies $\boldsymbol{x} = \sum_{j=1}^{k} \lambda_j \boldsymbol{c}^j$ for $\lambda_j > 0$ ($j \in [k]$) satisfying $\sum_{j=1}^{k} \lambda_j = 1$ and $\boldsymbol{c}^j \in B(f)$ ($j \in [k]$). Then, $\tilde{\boldsymbol{x}} \stackrel{\text{def}}{=} \boldsymbol{x} + a\mathbf{1} \in B(\tilde{f})$ satisfies $\tilde{\boldsymbol{x}} = \sum_{j=1}^{k} \lambda_j \tilde{\boldsymbol{c}}^j$ where $\tilde{\boldsymbol{c}}^j \stackrel{\text{def}}{=} \boldsymbol{c}^j + a\mathbf{1} \in B(\tilde{f})$.*

Now, let $\tilde{f}\colon 2^{[n]} \to \mathbb{R}_{\geq 0}$ be a cardinality based submodular function which is *monotone increasing*, then we consider the randomized rounding problem; given a point $\tilde{\boldsymbol{x}} \in B(\tilde{f})$, output an extreme point X of $B(\tilde{f})$ with a probability such that $E[X] \stackrel{\text{def}}{=} \sum_{j=1}^{k} \Pr\left[X = \tilde{\boldsymbol{c}}^j\right] \cdot \tilde{\boldsymbol{c}}^j = \tilde{\boldsymbol{x}}$ for an appropriate $k > 0$. By Proposition 8, it is easily transformed into the case from a general cardinality

based submodular function. Without loss of generality, we may assume that $\tilde{x}_1 \geq \cdots \geq \tilde{x}_n$ in the following. We remark that our randomized rounding algorithm is a generalization of [15] for the *permutahedron*, in a sense.

To begin with, we define special points in $B(\tilde{f})$, which we call *partially averaged points*. Suppose $\tilde{\boldsymbol{q}} \in B(\tilde{f})$ satisfies that $\tilde{q}_1 \geq \tilde{q}_2 \geq \cdots \geq \tilde{q}_n$, then, $\tilde{\boldsymbol{q}}$ is a partially averaged point if $\sum_{j=1}^{i} \tilde{q}_j = \tilde{g}(i)$ holds for each $i \in [n]$ satisfying $\tilde{q}_i > \tilde{q}_{i+1}$. Notice that if $\tilde{q}_i > \tilde{q}_{i+1} = \cdots = \tilde{q}_j > \tilde{q}_{j+1}$ hold for $i, j \in [n]$ then $q_{i+1} = \cdots = q_j = (\tilde{g}(j) - \tilde{g}(i))/(j-i)$. This means that the partially averaged point is uniquely determined only by a sequence of equalities(=)/inequalities(>). We simply say "a partially averaged point of $\tilde{\boldsymbol{x}}$" ($\tilde{\boldsymbol{x}} \in B(\tilde{f})$) as a partially averaged point determined by a sequence of equalities/inequalities derived from $\tilde{x}_1 \geq \tilde{x}_2 \geq \cdots \geq \tilde{x}_n$ of $\tilde{\boldsymbol{x}}$.

Proposition 9 *Suppose $\tilde{\boldsymbol{q}} \in B(\tilde{f})$ is a partially averaged point satisfying $\tilde{q}_1 \geq \tilde{q}_2 \geq \cdots \geq \tilde{q}_n$. Let $\mathit{\Pi} \stackrel{\text{def}}{=} \{\boldsymbol{\sigma} \in \mathrm{Sym}(n) \mid \tilde{q}_{\sigma(1)} \geq \tilde{q}_{\sigma(2)} \geq \cdots \geq \tilde{q}_{\sigma(n)}\}$, and let $\tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}} = (\tilde{c}_1^{\boldsymbol{\sigma}}, \ldots, \tilde{c}_n^{\boldsymbol{\sigma}})$ for $\boldsymbol{\sigma} \in \mathit{\Pi}$ denote the extreme point defined by hyperplanes $\sum_{j=1}^{i} \tilde{c}_{\sigma(j)}^{\boldsymbol{\sigma}} = \tilde{g}(i)$ for all $i \in [n]$. Note that $\boldsymbol{\sigma} \neq \boldsymbol{\sigma}'$ does not imply $\tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}} \neq \tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}'}$ in general. Then, $\tilde{\boldsymbol{q}} = \frac{1}{|\mathit{\Pi}|} \sum_{\boldsymbol{\sigma} \in \mathit{\Pi}} \tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}}$.*

Proof. Suppose $i \in [n-1]$ satisfies $\tilde{q}_i > \tilde{q}_{i+1}$. Since any $\boldsymbol{\sigma} \in \mathit{\Pi}$ satisfies $\tilde{q}_{\sigma(1)} \geq \tilde{q}_{\sigma(2)} \geq \cdots \geq \tilde{q}_{\sigma(n)}$, we see that $\{\sigma(1), \ldots, \sigma(i)\} = [i]$ holds for any $\boldsymbol{\sigma} \in \mathit{\Pi}$. This implies that $\sum_{j=1}^{i} \tilde{c}_j^{\boldsymbol{\sigma}} = \sum_{j=1}^{i} \tilde{c}_{\sigma(j)}^{\boldsymbol{\sigma}} = \tilde{g}(i)$. Since $\tilde{\boldsymbol{q}}$ is a partially averaged point, remember that $\sum_{j=1}^{i} \tilde{q}_j = \tilde{g}(i)$ holds, too.

Next, suppose $\tilde{q}_i > \tilde{q}_{i+1} = \cdots = \tilde{q}_j > \tilde{q}_{j+1}$ hold for $i, j \in [n]$. From the above arguments, we see that $\sum_{k=i+1}^{j} \tilde{c}_k^{\boldsymbol{\sigma}} = \tilde{g}(j) - \tilde{g}(i)$ holds for any $\boldsymbol{\sigma} \in \mathit{\Pi}$. For an arbitrary $\boldsymbol{\sigma} \in \mathit{\Pi}$, let $\boldsymbol{\sigma}' \in \mathrm{Sym}(n)$ satisfy $\sigma'(k) = \sigma(k)$ for each k ($k \leq i$ or $k > j$), then $\boldsymbol{\sigma}'$ is also in $\mathit{\Pi}$. Thus, let $\tilde{\boldsymbol{r}} \stackrel{\text{def}}{=} \frac{1}{|\mathit{\Pi}|} \sum_{\boldsymbol{\sigma} \in \mathit{\Pi}} \tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}}$ for convenience, then we see that $\tilde{r}_{i+1} = \cdots = \tilde{r}_j = (\tilde{g}(j) - \tilde{g}(i))/(j-i)$ hold. Since $\tilde{\boldsymbol{q}}$ is a partially averaged point, remember that $\tilde{q}_{i+1} = \cdots = \tilde{q}_j = (\tilde{g}(j) - \tilde{g}(i))/(j-i)$ hold, too. □

Proposition 9 and its proof immediately suggest an algorithm for randomized rounding of a partially averaged point; generate $\boldsymbol{\sigma} \in \mathit{\Pi}$ uniformly at random, and output $\tilde{\boldsymbol{c}}^{\boldsymbol{\sigma}}$. It's running time is $O(n)$, clearly.

Now, we explain our Algorithm 4, which provides a convex combination of partially average points representing $\tilde{\boldsymbol{x}} \in B(\tilde{f})$, i.e., given $\tilde{\boldsymbol{x}} \in B(\tilde{f})$, find partially average points $\tilde{\boldsymbol{q}}^1, \ldots, \tilde{\boldsymbol{q}}^K$ and $\lambda_1, \ldots, \lambda_K \in \mathbb{R}_{>0}$ such that $\sum_{i=1}^{K} \lambda_i \tilde{\boldsymbol{q}}^i = \tilde{\boldsymbol{x}}$ and $\sum_{i=1}^{K} \lambda_i = 1$. Once we obtain such a convex combination, it is clear to obtain an algorithm for randomized rounding into partially average points. Combining the above arguments concerning Proposition 9, we obtain a desired algorithm. We will prove the following lemma on Algorithm 4.

Theorem 10 *Algorithm 4 provides a convex combination of at most n partially averaged points representing an arbitrarily given $\tilde{\boldsymbol{x}} \in B(\tilde{f})$. Its running time is $O(n^2)$.*

Algorithm 4 Decomposition by partially average points

Input: $\tilde{\boldsymbol{x}} \in B(\tilde{f})$ satisfying that $\tilde{x}_1 \geq \tilde{x}_2 \geq \cdots \geq \tilde{x}_n$.
Output: Partially average points $\tilde{\boldsymbol{q}}^1, \ldots, \tilde{\boldsymbol{q}}^K$ and $\lambda_1, \ldots, \lambda_K \in \mathbb{R}_{>0}$ s.t. $\sum_{i=1}^K \lambda_i \tilde{\boldsymbol{q}}^i = \tilde{\boldsymbol{x}}$, $\sum_{i=1}^K \lambda_i = 1$.

1. Let $\tilde{\boldsymbol{x}}^1 = \tilde{\boldsymbol{x}}$ and $\lambda = 1$.
2. **For** $t = 1, \ldots,$
   (a) Find a partially averaged point $\tilde{\boldsymbol{q}}^t$ for $\tilde{\boldsymbol{x}}^t$.
   (b) Let $\lambda_t = \min\left\{\lambda, \min_{i \in [n-1]} \left\{ \frac{\tilde{x}_i^t - \tilde{x}_{i+1}^t}{\tilde{q}_i^t - \tilde{q}_{i+1}^t} \mid \tilde{q}_i^t \neq \tilde{q}_{i+1}^t \right\} \right\}$.
   (c) Let $\tilde{\boldsymbol{x}}^{t+1} = \tilde{\boldsymbol{x}}^t - \lambda_t \tilde{\boldsymbol{q}}^t$ and let $\lambda = \lambda - \lambda_t$.
   (d) **If** $\lambda = 1$ **then** let $K = t$ and break.
3. **Output** $\tilde{\boldsymbol{q}}^1, \ldots \tilde{\boldsymbol{q}}^K$ and $\lambda_1, \ldots, \lambda_K$.

To show Theorem 10, we show the following lemmas.

Lemma 2 *At any iteration t in Algorithm 4, $\tilde{\boldsymbol{x}}^t$ satisfies that $\tilde{x}_i^t \geq \tilde{x}_{i+1}^t$ 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 $\tilde{x}_i^{t-1} \geq \tilde{x}_{i+1}^{t-1}$ holds for any $i \in [n-1]$. If $\tilde{x}_i^{t-1} = \tilde{x}_{i+1}^{t-1}$, then $\tilde{q}_i^{t-1} = \tilde{q}_{i+1}^{t-1}$ holds, from the definition of $\tilde{\boldsymbol{q}}^{t-1}$. Thus

$$\tilde{x}_i^t = \tilde{x}_i^{t-1} - \lambda_{t-1}\tilde{q}_i^{t-1} = \tilde{x}_{i+1}^{t-1} - \lambda_{t-1}\tilde{q}_{i+1}^{t-1} = \tilde{x}_{i+1}^t$$

and we obtain the claim. If $\tilde{x}_i^{t-1} > \tilde{x}_{i+1}^{t-1}$, then $\tilde{q}_i^{t-1} > \tilde{q}_{i+1}^{t-1}$ holds, and

$$\tilde{x}_i^{t+1} - \tilde{x}_{i+1}^{t+1} = \tilde{x}_i^t - \tilde{x}_{i+1}^t - \lambda_t(\tilde{q}_i^t - \tilde{q}_{i+1}^t) = (\tilde{q}_i^t - \tilde{q}_{i+1}^t)\left(\frac{\tilde{x}_i^t - \tilde{x}_{i+1}^t}{\tilde{q}_i^t - \tilde{q}_{i+1}^t} - \lambda_t\right) \geq 0$$

where the last inequality comes from the definition of λ_t, followed by $\lambda_t \leq \min_{i \in [n-1]} \left\{ \left(\tilde{x}_{i+1}^t - \tilde{x}_i^t\right) / \left(\tilde{q}_{i+1}^t - \tilde{q}_i^t\right) \mid \tilde{q}_{i+1}^t \neq \tilde{q}_i^t \right\}$. □

Lemma 3 *In Algorithm 4, $\tilde{\boldsymbol{x}}^{K+1}$ $(= \tilde{\boldsymbol{x}}^K - \lambda^K \tilde{\boldsymbol{q}}^K) = 0$ holds.*

Proof. Without loss of generality, we may assume that $\tilde{x}_1 \geq \tilde{x}_2 \geq \cdots \geq \tilde{x}_n$, for simplicity of notations. First we show $\tilde{\boldsymbol{x}}^{K+1} \geq 0$. Since Lemma 2, if there exists $j \in [n]$ satisfying that $\tilde{x}_j^{K+1} < 0$, then $\tilde{x}_n^{K+1} < 0$ holds. Thus it is enough to show $\tilde{x}_n^{K+1} \geq 0$. Let $i^* = \min\{j \in [n] \mid \tilde{x}_j^K = \tilde{x}_n^K\}$. Then we have $\tilde{x}_{i^*}^K = \tilde{x}_{i^*+1}^K = \cdots = \tilde{x}_n^K$ and $\tilde{q}_{i^*}^K = \tilde{q}_{i^*+1}^K = \cdots = \tilde{q}_n^K$. Hence, we get $\tilde{x}_{i^*}^{K+1} = \tilde{x}_{i^*+1}^{K+1} = \cdots = \tilde{x}_n^{K+1}$. In case of $i^* \geq 2$, $\tilde{x}_{i^*-1}^t > \tilde{x}_{i^*}^t$ holds for any $t \in [K]$, meaning that $\tilde{q}_{i^*-1}^t > \tilde{q}_{i^*}^t$ holds for any $t \in [K]$. Thus we can see that $\sum_{j=i^*}^n \tilde{q}_j^t = \tilde{g}(n) - \tilde{g}(i^* - 1)$ holds for any $t \in [K]$, from the definition of $\tilde{\boldsymbol{q}}^t$. Then we obtain

$$\sum_{j=i^*}^{n} \sum_{t=1}^{K} \lambda_t \tilde{q}_j^t = \sum_{t=1}^{K} \lambda_t \left(\tilde{g}(n) - \tilde{g}(i^* - 1)\right) = \tilde{g}(n) - \tilde{g}(i^* - 1) \leq \sum_{j=i^*}^{n} \tilde{x}_j$$

where the last inequality is due to constraints of $B(\tilde{f})$ $\sum_{j=1}^{i^*-1} \tilde{x}_j \leq \tilde{g}(i^*-1)$ and $\sum_{j=1}^{n} \tilde{x}_j = \tilde{g}(n)$. Thus we obtain that $\sum_{j=i^*}^{n} \tilde{x}_j^{K+1} = \sum_{j=i^*}^{n} \left(\tilde{x}_j - \sum_{t=1}^{K} \lambda_t \tilde{q}_j^t\right) \geq 0$. As discussed above, $\tilde{x}_{i^*}^{K+1} = \tilde{x}_{i^*+1}^{K+1} = \cdots = \tilde{x}_n^{K+1}$ holds, and we obtain $\tilde{x}_n^{T+1} \geq 0$. In case of $i^* = 1$, the proof is done in a similar way.

Now we show $\tilde{\boldsymbol{x}}^{K+1} = 0$. Since $\tilde{\boldsymbol{x}} \in B(\tilde{f})$, $\sum_{j=1}^{n} \tilde{x}_j^{K+1} = \tilde{g}(n)$ holds. In a similar way as the proof of $\tilde{\boldsymbol{x}}^{K+1} \geq 0$,

$$\sum_{j=1}^{n} \sum_{t=1}^{K} \lambda_t \tilde{q}_j^t = \sum_{t=1}^{K} \lambda_t \sum_{j=1}^{n} \tilde{q}_j^t = \sum_{t=1}^{K} \lambda_t \tilde{g}(n) = \tilde{g}(n).$$

Since $\boldsymbol{x}^{K+1} \geq 0$, $\tilde{\boldsymbol{x}}^{K+1} = \tilde{\boldsymbol{x}} - \sum_{t=1}^{K} \lambda_t \tilde{\boldsymbol{q}}^t = 0$. □

Lemma 4 *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 $\tilde{x}_i^t > \tilde{x}_{i+1}^t$ and $\tilde{x}_i^{t+1} = \tilde{x}_{i+1}^{t+1}$. If $\tilde{x}_i^t = \tilde{x}_{i+1}^t$, then $\tilde{x}_i^{t+1} = \tilde{x}_{i+1}^{t+1}$ as discussed in the proof of Lemma 2. Now the claim is clear. □

Proof of Theorem 10. Since Lemma 3, it is clear that the output $\sum_{t=0}^{K} \lambda_t \tilde{\boldsymbol{q}}^t$ by Algorithm 4 is equal to an arbitrarily given $\tilde{\boldsymbol{x}} \in B(\tilde{f})$. It is not difficult to see that every lines in Algorithm 4 is done in $O(n)$. Hence, the running time of Algorithm 4 is $O(n^2)$ by Lemma 4. □

Note that, by modifying Algorithm 4, we can design an algorithm for randomized rounding of $\boldsymbol{x} \in B(f)$ using only $O(n)$ space, with the same time complexity of $O(n^2)$. We can also improve the algorithm with a time complexity of $O(n \log n)$ using a heap, with $O(n)$ space.

5 Conclusion

In this paper, we consider an prediction problem over the base polyhedron defined by a submodular function and propose efficient prediction algorithms. An open problem is to derive a tight lower bound of the regret of our problem.

Acknowledgements

We thank anonymous reviewers for their helpful comments. This work is supported in part by JSPS Grand-in-Aid for Young Scientists (B) 23700178, JSPS Grand-in-Aid for Scientific Research (B) 23300003, and Aihara Project, the FIRST program from JSPS.

References

1. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.
2. N. Cesa-Bianchi and G. Lugosi. Combinatorial Bandits. In *Proceedings of the 22nd Conference on Learning Theory*, 2009.
3. S. Chopra. On the spanning tree polyhedron. *Operations Research Letters*, 8(1):25–29, 1989.
4. J. Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Structures and Their Applications*, pages 69–87, 1970.
5. J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, 1971.
6. Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
7. S. Fujishige. *Submodular functions and optimization*. Elsevier Science, 2nd edition, 2005.
8. D. P. Helmbold and M. K. Warmuth. Learning Permutations with Exponential Weights. *Journal of Machine Learning Research*, 10:1705–1736, 2009.
9. S. Kakade, A. T. Kalai, and L. Ligett. Playing games with approximation algorithms. *SIAM Journal on Computing*, 39(3):1018–1106, 2009.
10. A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.
11. W. M. Koolen, M. K. Warmuth, and J. Kivinen. Hedging Structured Concepts. In *Proceedings of the 23rd Conference on Learning Theory*, pages 93–105, 2010.
12. K. Nagano. A faster parametric submodular function minimization algorithm and applications. Technical Report METR 2007–43, Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo, 2007.
13. J. B. Orlin. A Faster Strongly Polynomial Time Algorithm for Submodular Function Minimization. In *Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 240–251, 2007.
14. M. K. Warmuth and D. Kuzmin. Randomized Online PCA Algorithms with Regret Bounds that are Logarithmic in the Dimension. *Journal of Machine Learning Research*, 9:2287–2320, 2008.
15. 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.
16. G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics 152. Springer-Verlag, 1995.
17. 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.