Combinatorial Online Prediction

Hatano, Kohei
Kyushu University

Combinatorial Online Prediction

Kohei Hatano
Kyushu Univeristy /RIKEN AIP
hatano@inf.kyushu-u.ac.jp

***Abstract*—We present a short survey on recent results on combinatorial online prediction in the adversarial setting.**

***Index Terms*—online learning, combinatorial online prediction, online convex optimization, combinatorial optimization**

I. Introduction

Combinatorial online prediction problems arise in many situations such as scheduling [13], ranking [20], [43], routing [5], [39], network optimization [26], and so on. Recent advances in online convex optimization, offline convex and discrete optimization, and discrete data structures enable us to investigate broader problems than ever. In this paper, we survey results in combinatorial online prediction and related areas, as well as their applications.

The combinatorial online decision-making problem is defined as the following protocol between the player and the adversary. Let $\mathcal{C} \subset \mathbb{R}_+^n$ be the set of decisions of interest.

For each trial $t = 1, \dots, T$,

1) the player chooses a decision $\boldsymbol{c}_t \in \mathcal{C}$,
2) the adversary chooses a loss vector $\boldsymbol{\ell}_t \in [0,1]^n$, and
3) the adversary provides the player a feedback:
   - The feedback is the loss vector $\boldsymbol{\ell}_t$ itself (full-information setting).
   - The feedback is the inner product $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$ only (bandit setting).

There are options for feedbacks from the adversary. In the *full-information setting*, the loss vector $\boldsymbol{\ell}_t$ is always revealed to the player. In the *bandit setting*, the loss, which is the inner product between the decision $\boldsymbol{c}_t$ and the loss vector $\boldsymbol{\ell}_t$ is provided to the learner, while the information of the loss vector is not given. There is an intermediate setting between these, called semi-bandit setting. In the *semi-bandit setting*, where typically $\mathcal{C} \subseteq \{0,1\}^n$, not only the loss, but also components of the loss vector $\boldsymbol{\ell}_{t,i}$ s.t. $\boldsymbol{c}_{t,i} = 1$ are shown to the learner. This setting is natural in some applications, e.g., online routing where cost of each edge in the chosen route is available.

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

$$\alpha\text{-regret} = \mathrm{E}\left[\sum_{t=1}^{T} \boldsymbol{c}_t \cdot \boldsymbol{\ell}_t\right] - \alpha \min_{\boldsymbol{c}^* \in \mathcal{C}} \sum_{t=1}^{T} \boldsymbol{c}^* \cdot \boldsymbol{\ell}_t,$$

where the expectation is taken w.r.t. the randomness of the player. In typical settings, we consider the cases with $\alpha = 1$ and 1-regret. For simplicity, we abbreviate 1-regret as just *regret*.

When the set $\mathcal{C}$ is finite and the size is small or $\mathcal{C}$ is a closed convex set, the problem is called *the expert setting*, or *the online linear optimization*, for which there are algorithms whose regret bounds are $O(\sqrt{T})$, which are optimal w.r.t. T and running times at each trial are polynomial in $|C|$ for the expert setting and polynomial in n (other parameters are omitted).

However, when the size of $\mathcal{C}$ is huge, e.g., exponentially large w.r.t. n or other relevant parameters, those optimal algorithms are inefficient. We will review three approaches for combinatorial online prediction problems.

Note that we will focus on the setting where the adversary is literally "adversarial". More precisely, we assume that the adversary is *oblivious*, meaning that it determines loss vectors without seeing the player's decisions before the game starts. This setting is not weak as it seems, since if the player is a deterministic algorithm, the adversary could know its behavior in advance. But, we assume that the player is a randomized algorithm. Further, the adversary is assume to not see results of player's random choices. There are more relaxed setting where the adversary is *adaptive*, in which it can see the random choices of the player. Some approaches work in the adaptive setting, but some do not.

Another relevant assumption would be that the adversary is random and the loss vectors follows some probability distribution. We omit this case and we recommend interested readers to see, e.g., the survey of Cesa-Bianchi and Bubeck [6].

For other related topics on online prediction and online convex optimization, see textbooks of Cesa-Bianchi and Lugosi [7] and Hazan [19], respectively.

II. Preliminaries

We begin with some notations. For $n \geq 1$, let $[n] = \{1, \dots, n\}$. For $n \geq 2$, let Δ_n be the probability simplex over $1, \dots, n$, i.e., $\Delta_n = \{\boldsymbol{p} \in [0,1]^n \mid \sum_{i=1}^n p_i = 1\}$. For a finite set $\mathcal{C} \subset \mathbb{R}^n$, the convex hull $\mathrm{conv}(\mathcal{C})$ is defined as $\mathrm{conv}(\mathcal{C}) = \{\sum_{\boldsymbol{c} \in \mathcal{C}} \alpha_{\boldsymbol{c}} \boldsymbol{c} \mid \sum_{\boldsymbol{c} \in \mathcal{C}} \alpha_{\boldsymbol{c}} = 1, \boldsymbol{\alpha} \geq \boldsymbol{0}\}$.

A. Examples of combinatorial sets

We explain several combinatorial objects often treated in the literature.

a) k-sets: A k-set is a vector in $\{0,1\}^n$ whose 1-components are exactly k ($1 \leq k \leq n$). k-sets represent k multiple choices from n fixed alternatives. The set $\mathcal{C}$ of k-sets is represented as $\mathcal{C} = \{\boldsymbol{x} \in \{0,1\}^n \mid \sum_{i \in [n]} x_i = k\}$.

b) Paths in a Directed Acyclic Graph: Given a directed acyclic graph (DAG) with the source and sink nodes $G = (V, E)$ with $|V| = n$ and $|E| = m$, $\mathcal{C} = \{\boldsymbol{c} \in \{0,1\}^m \mid \boldsymbol{c}$ represents a path from the source node to the sink node in $G\}$.

c) Permutations: A permutation π over $[n]$ is a bijection from $[n]$ to $[n]$. Let S_n be the set of permutations over $[n]$. Given a permutation $\pi \in S_n$, the associated *permutation matrix* $\Pi \in [0,1]^{n\times n}$ is defined as $\Pi_{ij} = 1$ if $\pi(i) = j$ and $\Pi_{ij} = 0$, otherwise. The convex hull of all permutation matrices is called the *Birkhoff polytope* and represented as $\{X \in [0,1]^{n\times n} \mid \sum_{i\in[n]} X_{ij} = 1 \text{for } j \in [n] \text{and } \sum_{j\in[n]} X_{ij} = 1 \text{for } j \in [n]\}$. Similarly, a *permutation vector* $\boldsymbol{\pi}$ associated with a permutation π is defined as $\boldsymbol{\pi} = (\pi(1), \ldots, \pi(n))$. The convex hull of all permutation vectors is called the *permutahedron*.

d) Spanning Trees: Given an undirected graph G, a spanning tree of the graph G is a subgraph without cycles that contains all vertices in G.

B. Hedge Algorithm in the Experts Setting

Consider the simple case where $\mathcal{C} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$, where $\boldsymbol{e}_i \in \{0,1\}^n$ $(i = 1, \ldots, n)$ is the unit vector whose the i-th component is 1 and other components are 0s. This case is called the experts setting. In the experts setting, there are n experts and at each trial, the player chooses one of the experts and follows its prediction. This setting naturally generalizes the case where the player has several fixed choices. The *Hedge* algorithm was developed by Freund and Schapire [12]. In the experts setting, given a parameter $\eta > 0$ and the initial distriution $\boldsymbol{p}_1 \in \Delta_n$ over experts, at each trial t, the Hedge algorithm chooses $\boldsymbol{c}_t = \boldsymbol{e}_{i_t}$ randomly according to the distribution $\boldsymbol{p}_t$ s.t.

$$p_{t,i} = e^{-\eta \sum_{\tau=1}^{t-1} \ell_{\tau,i}} / Z_{t-1} \quad \text{for } i = 1, \ldots, n, \tag{1}$$

where $Z_{t-1} = \sum_{i=1}^{n} e^{-\eta \sum_{\tau=1}^{t-1} \ell_{\tau,i}}$.

Theorem 1 (Freund &Schapire [12])**.** *The regret of Hedge algorithm is $O(\sqrt{T \ln n})$ and $O(\sqrt{L^* \ln n})$ with $\eta = \sqrt{\ln n / T}$ and $\eta = \ln(1 + \sqrt{(2\ln n)/L^*})$, respectively, where $L^* = \min_{i\in[n]} \sum_{t=1}^{T} \ell_{t,i}$.*

Note that the regret bound $O(\sqrt{L^* \ln n})$ has a matching lower bound and thus the bound is optimal (for proofs, see, e.g., [7], [20], [34]). prior knowledge, including An annoying issue is that, to obtain the optimal regret bound, we need to know the horizon T or the cumulative loss L^* of the best expert a priori. There are several variants of the Hedge, including the work of Koolen and Van Erven [25], which adaptively tune the time-dependent parameter η_t instead of the fixed η and attain the same optimal bound in the worst case and better bounds in some easy cases.

III. Simulation of Hedge over Large Decision Space

The first approach to deal with large decision spaces is to simulate Hedge efficiently. The computational bottleneck of Hedge is to sample decisions $\boldsymbol{c}_t \in \mathcal{C}$, where $\mathcal{C}$ is possibly exponentially large, according to the exponential weights $\boldsymbol{p}_t$

$$p_{t,\boldsymbol{c}} \propto e^{-\eta \sum_{\tau=1}^{t-1} \boldsymbol{c}_\tau \cdot \boldsymbol{\ell}_\tau}. \tag{2}$$

There is no known general efficient method to simulate the sampling and it would be computationally hard for certain classes of $\mathcal{C}$. This is because such algorithms implies randomized offline linear optimization algorithms over $\mathcal{C}$ (possibly NP-hard) via the standard online-to-batch conversion technique.

For some particular classes of $\mathcal{C}$, there exists efficient sampling algorithms.

a) Paths and k-Sets: For the set of paths, Takimoto and Warmuth developed the efficient sampling technique [39] using the weight pushing algorithm of Mohri [31]. The computation time of the method is $O(m+n)$. For k-sets, the set of $\{0,1\}$-vectors with k ones are encoded as paths in a DAG of size $O(nk)$ and the sampling problem over k-sets is reduced to that of paths in a DAG with $O(nk)$ nodes and edges.

b) Permutation Matrices and Vectors: For perfect matchings (permutation matrices), Cesa-Bianch and Lugosi pointed out that computing weights of all permutation matrices can be reduced to computing the permanent [8] and thus can be (approximately) computed by the polynomial time approximating algorithms of Jerrum et al. [21], which takes $\tilde{O}(n^{10})$ time. For permutation vectors, Ailon proposed an improved sampling scheme whose running time is $O(n \log n)$ [2].

c) Spanning Trees: For spanning trees, to the best of our knowledge, there is no known polynomial time algorithm (in terms of m and n). Cesa-Bianchi and Lugosi also mentioned that the Markov chain-based method of Propp and Wilson [35] could be used to sample spanning trees according to exponential weights efficiently [8].

Other examples of combinatorial objects include initial segments of lists, for which Warmuth et al. proposed an efficient sampling scheme [41].

For the bandit cases, Cesa-Bianchi and Lugosi proposed COMBAND, a variant of Hedge using the sampling scheme over $|C|$. The COMBAND employs the sampling scheme over the decision set $\mathcal{C}$ w.r.t. exponential weights and thus can be applied to combinatorial objects listed above. For paths in the bandit cases, Awerbuch and Kleinberg developed a technique using the barycentric spanner and obtained non-trivial regret bounds [5] (similar bounds are obtained by Mcmahan and Blum [28]). György et al. considered paths in the semi-bandit cases [18]. For the bandit setting, Dani et al. developed a method with$O(\sqrt{T})$ regret bounds using the barycentric spanner and the sampling scheme [10]. For permutation vectors, Ailon et al. proposed a generalization of Ailon's sampling scheme for the bandit setting [3].

As a summary, an advantage of this approach is that it is often easier to analyze the regret in the full-information, semi-bandit and bandit settings, once we assume that there is an efficient sampling algorithm according to the rule (2). However, the sampling scheme needs careful design for specific classes.

IV. Online Mirror Descent/FTRL with Projection and Decomposition

The second approach for combinatorial online prediction is to relax the problems from discrete to continuous convex ones.

More precisely, the strategy is as follows:

1) (Convex relaxation) Relax the decision set $\mathcal{C}$ to $Relax(\mathcal{C})$ (which includes $\mathrm{conv}(\mathcal{C})$, typically $Relax(\mathcal{C}) = \mathrm{conv}(\mathcal{C})$) and use an online linear optimization algorithm over $Relax(\mathcal{C})$.
2) (Rounding, or Decomposition) "Round" predictions $\boldsymbol{x}_t \in Relax(\mathcal{C})$ of the online linear optimization algorithm over $Relax(\mathcal{C})$ to discrete ones $\boldsymbol{c}_t \in \mathcal{C}$.

More precisely, *rounding* is defined as, given $\boldsymbol{x} \in Relax(\mathcal{C})$, to choose $\boldsymbol{c} \in \mathcal{C}$ randomly so that $\mathrm{E}[\boldsymbol{c}] \leq \alpha \boldsymbol{x}$ for some $\alpha \geq 1$, where the expectation is defined in terms of the algorithm's randomness. In particular, the *decomposition*, which is a special case of rounding, is to find a convex combination of $\boldsymbol{c} \in \mathcal{C}$ satisfying $\boldsymbol{x} = \sum_{\boldsymbol{c} \in \mathcal{C}} \alpha_{\boldsymbol{c}} \boldsymbol{c}$ with $\boldsymbol{\alpha} \in [0,1]^{\mathcal{C}}$ and $\sum_{\boldsymbol{c} \in \mathcal{C}} \alpha_{\boldsymbol{c}} = 1$. A decomposition method implies a rounding method with $\mathrm{E}[\boldsymbol{c}] = \boldsymbol{x}$ with $\alpha = 1$. A detailed description of the scheme is shown in Algorithm 1. With these components, it is easy

Algorithm 1 OLO over $Relax(\mathcal{C})$ with Rounding (full information setting)

```
Given: an online linear optimization algorithm $A$ with decision set $Relax(\mathcal{C})$
  1) For $t = 1, \dots, T$
       a) Run an algorithm $A$ over $Relax(\mathcal{C})$ and get a decision $\boldsymbol{x}_t \in Relax(\mathcal{C})$.
       b) "Round" $\boldsymbol{x}_t$ and predict $\boldsymbol{c}_t = Round(\boldsymbol{x}_t) \in \mathcal{C}$ s.t. $E(\boldsymbol{c}_t) \leq \alpha \boldsymbol{x}_t$.
       c) Receive the loss vector $\boldsymbol{\ell}_t$ and incur loss $\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t$. Feed the loss vector $\boldsymbol{\ell}_t$ to the algorithm $A$.
```

to construct an combinatorial online prediction algorithm with regret $O(\sqrt{T})$ in the full-information setting.

Theorem 2. *Suppose that an online linear optimization algorithm A with a decision set $Relax(\mathcal{C})$ has a regret bound $O(\sqrt{T})$. Then, Algorithm 1 has α-regret $O(\sqrt{T})$.*

Proof.

$$\begin{aligned}(\alpha\text{-Regret}) =& \sum_{t=1}^{T} \mathrm{E}[\boldsymbol{c}_t \cdot \boldsymbol{\ell}_t] - \alpha \min_{\boldsymbol{c}^* \in \mathcal{C}} \sum_{t=1}^{T} \boldsymbol{c}^* \cdot \boldsymbol{\ell}_t \\ \leq& \alpha \sum_{t=1}^{T} \boldsymbol{x}_t \cdot \boldsymbol{\ell}_t - \alpha \min_{\boldsymbol{c}^* \in Relax(\mathcal{C})} \sum_{t=1}^{T} \boldsymbol{c}^* \cdot \boldsymbol{\ell}_t \\ & \text{(by definition of rounding and } \mathcal{C} \subset Relax(\mathcal{C})) \\ =& O(\alpha\sqrt{T}) \quad \text{(by definition of } A).\end{aligned}$$

□

A. FTRL / OMD and Projection

For constructing an online linear optimization algorithm, we can use the *Follow the Regularized Leader* (FTRL) or the *Online Mirror Descent* (OMD) over $Relax(\mathcal{C})$. Let $R : \mathcal{X} \to \mathbb{R}$ (where $\mathcal{X} \subset \mathbb{R}^n$) be a twice-differentiable strictly convex function. The FTRL over $Relax(\mathcal{C})$ with the regularizer R is defined as, for $t = 1, \dots, T$,

$$\boldsymbol{x}_t = \arg \max_{\boldsymbol{x} \in Relax(\mathcal{C})} \eta \sum_{\tau=1}^{t-1} \boldsymbol{x} \cdot \boldsymbol{\ell}_\tau + R(\boldsymbol{x}).$$

The equivalent algorithm with the FTRL is the OMD (sometimes called lazy version of OMD) is defined as

$$\begin{aligned}\boldsymbol{y}_t &= r^{-1}(r(\boldsymbol{y}_{t-1}) - \eta \boldsymbol{\ell}_{t-1}) \\ \boldsymbol{x}_t &= \arg \min_{\boldsymbol{x} \in Relax(\mathcal{C})} B_R(\boldsymbol{x}, \boldsymbol{y}_t), \quad \text{(projection)}\end{aligned}$$

where $r(\boldsymbol{x}) = \nabla R(\boldsymbol{x})$, B_R is the Bregman divergence associated with R and defined as $B_R(\boldsymbol{x}, \boldsymbol{y}) = R(\boldsymbol{x}), -R(\boldsymbol{y}) - \nabla R(\boldsymbol{y}) \cdot (\boldsymbol{x} - \boldsymbol{y})$, and $\boldsymbol{y}_0$ is s.t. $r(\boldsymbol{y}_0) = \mathbf{0}$. Note that r^{-1} exists since R is strictly convex. In particular, the second step of the OMD is called the *projection* step (onto $Relax(\mathcal{C})$). One of computational bottleneck in this approach lies in this step.

The OMD contains the Online Gradient Descent (OGD) [44] and the Hedge as special cases. It is known that the OMD achieves regret $O(\sqrt{T})$ (neglecting other factors) for online linear optimization. See, e.g., Hazan's textbook [19] for the details of the FTRL and OMD.

B. Projection and Rounding (Decomposition)

a) k-set: Warmuth and Kuzmin developed projection and decomposition methods onto the convex hull of the k-sets in order to solve the online optimization of PCAs [42]. The convex hull can be represented as $\{\boldsymbol{p} \in \Delta_n \mid p_i \leq 1/k, i \in [n]\}$. The projection onto the hull w.r.t. the relative entropy can be done in linear time. The decomposition can be done in $O(n^2)$ time.

b) Permutations: Helmbold and Warmuth proposed the PermELearn for the class $\mathcal{C}$ of permutation matrices [20]. The relaxed decision space is $Relaxed(\mathcal{C}) = \mathrm{conv}(\mathcal{C})$, which is the Birkhoff polytope. The Birkhoff polytope can be represented by $O(n)$ linear constraints. The OMD corresponds to a matrix version of Hedge and the Bregman projection is defined w.r.t. the von Neuman matrix divergence and can be approximately solved by the Sinkhorn balancing algorithm in time $O(n^6)$. The decomposition step can be solved by iteratively finding a perfect matching in a bipartite graph in time $O(n^4)$.

For permutation vectors, Yasutake et al. developed a different algorithm whose running time is $O(n^2)$ per trial [43] (the running time is improved to $O(n \log n)$ later [27]). The convex hull of the decision space is the permutahedron, described with $O(2^n)$ linear constraints. Apparently, the projection seems hard, but only $O(n)$ linear constraints are shown to be relevant. A simple projection algorithm w.r.t. the relative entropy can be done in $O(n^2)$ time. The decomposition step is solved in $O(n \log n)$ time. Lim and Wright proposed improved projection algorithms for a large class of Bregman divergences whose running time is $O(n \log n)$ as well [27].

c) Spanning trees: Koolen et al. proposed the Component Hedge which generalizes the framework of Helmbold and Warmuth [20] for wider classes of combinatorial objects including spanning trees [26] . The convex hull of the spanning trees is represented with $O(n^3)$ variables and linear constraints and the unnormalized relative entropy is the instance of the Bregman divergence. The decomposition step can be done by iteratively solving linear optimization of the spanning trees at most n times.

d) Paths: Koolen et al. also considered projection and decomposition steps for paths [26]. For paths in a DAG with the source and sink nodes, the convex hull of them corresponds to the flow polytope, which is described with $O(n+m)$ linear constraints. So, the Bregman projection onto the flow polytope is a convex optimization over $O(n+m)$ constraints. Rounding can be done in $O(n+m)$ time.

Suehiro et al. provided a unified framework of the projection and the decomposition steps for several classes of combinatorial objects characterized with submodular functions, which includes k-sets, permutation vectors, spanning trees and so on [38].

C. Bandit Settings

Uchiya et al. and Kale et al. independently considered a decomposition method for k-sets in the semi-bandit setting [24], [40]. Combes proposed a general framework for projection and decomposition steps for the bandit setting [9].

V. Offline-to-Online Conversion Methods

The third approach to online combinatorial prediction is to "convert" offline linear optimization algorithms over $\mathcal{C}$ to online ones. One of advantages of this approach is that, there are many offline linear optimization algorithms for many classes $\mathcal{C}$ and they are often efficient.

A. Offline Exact Optimization Algorithms to Online

For 1-regret minimization, we assume an offline linear optimization algorithm OL over $\mathcal{C}$. That is, given a loss vector $\boldsymbol{\ell} \in [0,1]^n$, OL outputs $\arg\min_{\boldsymbol{c}^* \in \mathcal{C}} \boldsymbol{c}^* \cdot \boldsymbol{\ell}$. A naive way to use the offline algorithm is to predict the best offline minimizer so far, given past loss functions $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_{t-1}$. The strategy is called the Follow the Leader (FTL). Unfortunately, the regret of FTL is $\Omega(T)$ in general (see, e.g., Shalev-Shwartz's survey for the proof [37]).

The Follow the Perturbed Leader (FPL), a slight modification of FTL achieves the regret $O(\sqrt{T})$ [23]. At each trial t, the FPL chooses $\boldsymbol{c}_t = \arg\min_{\boldsymbol{c} \in \mathcal{C}} \boldsymbol{c} \cdot \left(\eta \sum_{\tau=1}^{t-1} \boldsymbol{\ell}_\tau + \boldsymbol{Z}_t\right)$, where $\boldsymbol{Z}_t$ is a random vector drawn according to some distribution (e.g., uniform distribution over $[-1,1]^n$ or the component-wise exponential distributions). There are some variants of the FPL with better bounds in some case (see, Devroye et al. [11], Neu and Bartok [33] and Abernethy et al. [1]). For the semi-bandit setting, Neu developed an efficient variant of the FPL [32].

Since we only need an offline linear optimization algorithm for $\mathcal{C}$, many classes of combinatorial concepts can be dealt with the FPL, including all examples raised in previous sections. However, a drawback of the FPL is that the regret bounds are sometimes inferior to bounds of other types. But recent analyses improving the gaps in full information, semi-bandit cases [33]. Also, as far as we know, there is no non-trivial extension of the FPL for the bandit setting.

B. Offline Approximation Algorithms to Online

For many classes of $\mathcal{C}$, exact linear optimization of $\mathcal{C}$ is NP-hard and thus it is natural to assume that there is only an approximation algorithm for $\mathcal{C}$ available. More precisely, an α-approximation algorithm for $\mathcal{C}$ ($\alpha \geq 1$), given a loss vector $\boldsymbol{\ell} \in [0,1]^n$ as input, outputs $\boldsymbol{c}$ s.t. $\boldsymbol{c} \cdot \boldsymbol{\ell} \leq \alpha \min_{\boldsymbol{c}^* \in \mathcal{C}} \boldsymbol{c}^* \cdot \boldsymbol{\ell}$. The FPL does not work well with approximation algorithms in general. Its regret bound becomes $O(\alpha^T \sqrt{T})$ [23].

Kakade et al. developed a better online algorithm using an α-approximation algorithm for $\mathcal{C}$, which achieves $O(\alpha\sqrt{T})$ regret bounds [22]. Their method basically follows the second approach in the previous section. Roughly speaking, there method is a combination of the OGD over $\mathrm{conv}(\alpha\mathcal{C})$ with projection and decomposition steps. The key technique is to perform (approximate) projection and decomposition steps based on the Frank-Wolf method which iteratively solves linear optimization problems over $\mathrm{conv}(\alpha\mathcal{C})$. To implement the Frank-Wolf method, the α-approximation algorithm is used as a key component. The Kakade et al.' s method, however, takes $O(T)$ computation time per trial and thus takes $O(T^2)$ time after T trials. Garber proposed an improved conversion algorithm with running time $O(\sqrt{T}\log T)$ per trial, while preserving the same regret bounds [17].

Fujita et al. considered a mild but natural assumption for $\mathcal{C}$ [14]. Their assumption is that there exists a relaxation-based approximation algorithm for $\mathcal{C}$. That is, there exists an algorithm A such that, for some closed convex space $Relax(\mathcal{C})$ which includes $\mathcal{C}$, given a loss vector $\boldsymbol{\ell} \in [0,1]^n$, A outputs $\boldsymbol{c} \in \mathcal{C}$ satisfying $\boldsymbol{c} \cdot \boldsymbol{\ell} \leq \alpha \min_{\boldsymbol{x}^* \in Relax(\mathcal{C})} \boldsymbol{x}^* \cdot \boldsymbol{\ell}$. Under this assumption, their algorithm, again, a combination of the OGD and projection and decomposition steps achieves $((\alpha+\varepsilon)\sqrt{T})$ regret bounds with $O((1/\varepsilon^2))$ time complexity per trial.

VI. Conclusion

In this paper, we surveyed recent results in online combinatorial prediction in the adversarial setting. There are many open problems. For example, more efficient algorithms for particular classes are needed. Better regret bounds of projection-decomposition based methods and the FPL-based methods for the bandit setting are still open.

Another interesting direction is to application of combinatorial online prediction techniques to other areas, such as online scheduling [16]. boosting over compressed data [15]. In particular, for practical applications, data structures BDD [4], MDD [29] and ZDD [30] can be useful. These data structures succinctly represent combinatorial sets as DAGS in which each combinatorial object corresponds to a path. Therefore, by combining these structures, the combinatorial online prediction problem over any combinatorial decision set is reduced to that of paths. For example, Sakaue et al. considered a combination of ZDDs and online combinatorial prediction techniques [36]. Fujita et al. proposed an algorithm which simulates AdaBoost and its variants over compressed data by ZDDs [15].

Acknowledgments

This work is supported in part by JSPS KAKENHI Grant Number JP16K00305.

References

[1] J. Abernethy, C. Lee, A. Sinha, and A. Tewari. Online Linear Optimization via Smoothing. In *Proceedings of the 27th Conference on Learning Theory (COLT'14)*, volume 35 of *JMLR W&CP*, pages 807–823, 2014.

[2] 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.

[3] N. Ailon, K. Hatano, and E. Takimoto. Bandit optimization over the permutahedron. *Theoretical Computer Science*, 650:92–108, 2016.

[4] S. B. Akers. Binary Decision Diagrams. *IEEE Transactions on Computers*, C-27(6):509–516, 1978.

[5] B. Awerbuch and R. D. Kleinberg. Adaptive routing with end-to-end feedback. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (STOC2004)*, pages 45–53, 2004.

[6] S. Bubeck and N. Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[7] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

[8] N. Cesa-Bianchi and G. Lugosi. Combinatorial Bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

[9] R. Combes, M. S. Talebi Mazraeh Shahi, A. Proutiere, and M. Lelarge. Combinatorial Bandits Revisited. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 2116–2124. ., 2015.

[10] V. Dani, S. M. Kakade, and T. P. Hayes. The Price of Bandit Information for Online Optimization. In *Advances in Neural Information Processing Systems 20 (NIPS2008)*, pages 345–352. 2008.

[11] L. Devroye, G. Lugosi, and G. Neu. Random-Walk Perturbations for Online Combinatorial Optimization. *IEEE Transactions on Information Theory*, 61(7):4099–4106, 2015.

[12] 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.

[13] T. Fujita, K. Hatano, S. Kijima, and E. Takimoto. Online Linear Optimization for Job Scheduling under Precedence Concstraints. In *Proceedings of 26th International Conference on Algorithmic Learning Theory(ALT 2015)*, volume 6331 of *LNCS*, pages 345–359, 2015.

[14] T. Fujita, K. Hatano, and E. Takimoto. Combinatorial Online Prediction via Metarounding. In *Proceedings of 24th Annual Conference on Algorithmic Learning Theory (ALT 2013)*, volume 8139 of *LNCS*, pages 68–82, 2013.

[15] T. Fujita, K. Hatano, and E. Takimoto. Boosting over Non-deterministic ZDDs. In *Proceeding of the 12th international conference and workshops on algorithms and computation (WALCOM 2018)*, LNCS, vol. 10755, pages 195–206, 2018.

[16] T. Fujita, K. Hatano, and E. Takimoto. Online Combinatorial Optimization with Multiple Projections and Its Application to Scheduling Problem. *IEICE Transactions on Information and Systems*, (to appear), 2018.

[17] D. Garber. Efficient Online Linear Optimization with Approximation Algorithms. In *Advances in Neural Information Processing Systems 30 (NIPS2017)*, pages 627–635, 2017.

[18] A. György, T. Linder, G. Lugosi, and G. Ottucsák. The On-Line Shortest Path Problem Under Partial Monitoring. *Journal of Machine Learning Research*, 8:2369–2403, 2007.

[19] E. Hazan. Introduction to Online Convex Optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.

[20] D. P. Helmbold and M. K. Warmuth. Learning Permutations with Exponential Weights. *Journal of Machine Learning Research*, 10:1705–1736, 2009.

[21] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51:671–697, 2004.

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

[23] A. Kalai and S. Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

[24] S. Kale, L. Reyzin, and R. E. Schapire. Non-Stochastic Bandit Slate Problems. In *Advances in Neural Information Processing Systems 23*, pages 1054–1062. 2010.

[25] W. M. Koolen and T. V. Erven. Second-order Quantile Methods for Experts and Combinatorial Games. In *Proceedings of the 28th Conference on Learning Theory (COLT2015)*, volume 40 of *PMLR*, pages 1155–1175, 2015.

[26] W. M. Koolen, M. K. Warmuth, and J. Kivinen. Hedging Structured Concepts. In *Proceedings of the 23rd Conference on Learning Theory (COLT 2010)*, pages 93–105, 2010.

[27] C. H. Lim and S. Wright. Beyond the Birkhoff Polytope: Convex Relaxations for Vector Permutation Problems. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 2168–2176, 2014.

[28] H. B. McMahan and A. Blum. Online Geometric Optimization in the Bandit Setting Against an Adaptive Adversary. In *Learning Theory*, volume 3120 of *LNCS*, pages 109–123. Springer Berlin Heidelberg, 2004.

[29] M. D. Miller. Multiple-Valued Logic Design Tools. In *Proceedings of the23rd IEEE International Symposium on Multiple-Valued Logic (ISMVL'93)*, pages 2–11, 1993.

[30] S.-i. Minato. Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems. In *Proceedings of the 30th international Design Automation Conference (DAC'93)*, pages 272–277, 1993.

[31] M. Mohri. General algebraic frameworks and algorithms for shortest distance problems, 1998.

[32] G. Neu. First-order regret bounds for combinatorial semi-bandits. In *Proceedings of the 28th Conference on Learning Theory (COLT2015)*, volume 40 of *PMLR*, pages 1360–1375, 2015.

[33] G. Neu and G. Bartók. Importance Weighting Without Importance Weights: An Efficient Algorithm for Combinatorial Semi-Bandits. *Journal of Machine Learning Research*, 17(154):1–21, 2016.

[34] F. Orabona and D. Pal. Optimal Non-Asymptotic Lower Bound on the Minimax Regret of Learning with Expert Advice, nov 2015.

[35] J. G. Propp and D. B. Wilson. How to Get a Perfectly Random Sample from a Generic Markov Chain and Generate a Random Spanning Tree of a Directed Graph. *Journal of Algorithms*, 27(2):170–217, may 1998.

[36] S. Sakaue, M. Ishihata, and S.-i. Minato. Efficient Bandit Combinatorial Optimization Algorithm with Zero-suppressed Binary Decision Diagrams. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics (AISTATS 2018)*, volume 84 of *PMLR*, pages 585–594, 2018.

[37] S. Shalev-Shwartz. Online Learning and Online Convex Optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

[38] D. Suehiro, K. Hatano, S. Kijima, E. Takimoto, and K. Nagano. Online Prediction under Submodular Constraints. In *Proceedings of 23th Annual Conference on Algorithmic Learning Theory (ALT 2012)*, volume 7568 of *LNCS*, pages 260–274, 2012.

[39] E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4(5):773–818, 2003.

[40] T. Uchiya, A. Nakamura, and M. Kudo. Algorithms for Adversarial Bandit Problems with Multiple Plays. In *Proceedings of the 21st International Conference on Algorithmic Learning Theory (ALT2010)*, volume LNAI 6331, pages 375–389, 2010.

[41] M. K. Warmuth, W. M. Koolen, and D. P. Helmbold. Combining initial segments of lists. *Theoretical Computer Science*, 519:29–45, jan 2014.

[42] 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.

[43] 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)*, volume 7074 of *LNCS*, pages 534–543, 2011.

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