Efficient Bandit Combinatorial Optimization Algorithm with Zero-suppressed Binary Decision Diagrams

Shinsaku Sakaue
NTT

Masakazu Ishihata
Hokkaido University

Shin-ichi Minato
Hokkaido University

Abstract

We consider bandit combinatorial optimization (BCO) problems. A BCO instance generally has a huge set of all feasible solutions, which we call the *action set*. To avoid dealing with such huge action sets directly, we propose an algorithm that takes advantage of *zero-suppressed binary decision diagrams*, which encode action sets as compact graphs. The proposed algorithm achieves either $O(T^{2/3})$ regret with high probability or $O(\sqrt{T})$ expected regret at any T-th round. Typically, our algorithm works efficiently for BCO problems defined on networks. Experiments show that our algorithm is applicable to various large BCO instances including adaptive routing problems on real-world networks.

1 INTRODUCTION

Given finite set E and *action set* $\mathcal{S} \subseteq 2^E$, which is the collection of all subsets that satisfy certain constraints, we consider the online combinatorial optimization (OCO) problem [2], which is a sequential decision problem repeated for $t = 1, \dots, n$. In the t-th round, a *player* chooses *action* $X_t \in \mathcal{S}$. Then the player incurs a cost and obtains feedback according to the selected action X_t. The aim of the player is to minimize the cost accumulated up to time horizon n by adaptively choosing an action in each round.

Examples of the OCO include various important problems on networks: the *online shortest path* (OSP) problem [3, 15], the *dynamic Steiner tree* (DST) problem [16], and the *congestion game* (CG) [32, 34]. For instance, in an OSP on a communication network, E is the edge set of the network, an action is an s-r path that connects sender s to receiver r, and action set $\mathcal{S}$ is the set of all s-r paths. The cost of an edge is the transmission time taken to send a message along the edge, and it is expected to dynamically change due to the time-varying congestion or accidents (e.g., cyber attacks). Thus an s-r path must be chosen adaptively each time a message is sent from s to r.

The main difficulty with OCO is that the size of an action set is generally exponential in $|E|$. To handle huge action sets, existing methods for OCO problems assume that an action set has certain properties. For instance, some methods assume that an action set is given by m-sets [21], permutations [1], or s-r paths on a directed acyclic graph (DAG) [15, 35]. An OCO algorithm that operates efficiently over the convex hull of an action set are studied in [10, 27]. However, many important OCO instances, including OSP, DST, and CG on undirected networks with limited feedback, are still missing efficient algorithms.

1.1 Our Contribution

In this paper, we develop an efficient and theoretically guaranteed algorithm for OCO with *bandit* feedback, which we call *bandit combinatorial optimization* (BCO); the details of the feedback are described later. To obtain strong theoretical guarantees, we use the strategy of ComBand [9], which achieves $O(\sqrt{n})$ *regret* in expectation for BCO problems. We slightly modify ComBand to obtain *any-time guarantees* [6]; we call the modified algorithm CombD (ComBand with decreasing parameters) to distinguish it from the original ComBand. CombD achieves either $O(T^{2/3})$ regret with high probability or $O(\sqrt{T})$ expected regret at any T-th round ($T \in \{1, \dots, n\}$) without knowing n, and we can select which regret value is to be achieved using a hyper parameter. The naive implementation of CombD, however, is impractical given the huge size of action sets. More precisely, as we will see later, CombD has two costly steps: sampling an action from the action set and computing unbiased estimators of loss values for all $i \in E$. We address these problems by taking advantage of *zero-suppressed binary decision*

Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

diagrams (ZDDs), which encode action sets as compact DAGs. We show that the sampling step can be completed by applying an existing dynamic programming (DP) method [35] to ZDDs, and we provide a novel DP method on ZDDs to compute the unbiased estimators. These methods enable CombD to be performed on ZDDs efficiently, and thus we obtain a novel practical BCO algorithm, which we call CombD3 (CombD with decision diagrams); its time and space complexities in each round are proportional to ZDD size. While ZDD size can grow exponentially in general, it is bounded in some cases (e.g., a knapsack constraint and an s-r path constraint on a network with bounded *pathwidth*) as in Section 5.1; in such cases the complexities of CombD3 are also bounded. In practice, ZDDs tend to be orders of magnitude smaller than action sets, which is experimentally confirmed in [29] and our experiments (Section 6); in particular, ZDDs that encode network-based action sets (e.g., those arise in the OSP, DST, and CG) are often small. Experimental results on OSP and DST instances show that CombD3 is far more scalable than ComBand, which deals with action sets directly. We also apply CombD3 to CGs on real-world networks and show that it is useful for practical adaptive routing problems.

1.2 Problem Statements of BCO

Let $[m] := \{1, \ldots, m\}$ for any $m \in \mathbb{N}$. We define $E := [d]$, and we let $\mathcal{S} \subseteq 2^E$ be an action set; each $X \in \mathcal{S}$ is called an action. At each t-th round ($t \in [n]$), an *adversary* secretly chooses *loss vector* $\boldsymbol{\ell}_t := (\ell_{t,1}, \ldots, \ell_{t,d})^\top \in \mathbb{R}^d$ and the player chooses action $X_t \in \mathcal{S}$. The player then incurs, and observes, the cost $c_t = \boldsymbol{\ell}_t^\top \mathbf{1}_{X_t}$, where $\mathbf{1}_{X_t} \in \{0,1\}^d$ is an indicator vector such that its i-th entry is 1 if $i \in X_t$ and 0 otherwise. This observation setting is called the bandit feedback setting; note that the player *cannot* observe $\boldsymbol{\ell}_t$. The aim of the player is to minimize regret R_T defined as follows for any $T \in [n]$:

$$\mathrm{R}_T := \sum_{t=1}^{T} \boldsymbol{\ell}_t^\top \mathbf{1}_{X_t} - \min_{X \in \mathcal{S}} \sum_{t=1}^{T} \boldsymbol{\ell}_t^\top \mathbf{1}_X.$$

The first term is the cumulative cost and the second term is the total cost of the best single action selected with hindsight. Namely, R_T expresses the extra cost that the player incurs against the best single action. As in [9], we assume $\max_{t \in [n], X \in \mathcal{S}} |\boldsymbol{\ell}_t^\top \mathbf{1}_X| \leq 1$.

We consider that the player (and the adversary) is allowed to use a randomized strategy to choose X_t (and $\boldsymbol{\ell}_t$). In such cases, R_T is a random variable of a joint distribution $p(\boldsymbol{\ell}_{1:T}, X_{1:T})$, where $X_{1:T} = \{X_1, \ldots, X_T\}$ and $\boldsymbol{\ell}_{1:T} = \{\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T\}$. Here p is assumed to satisfy the following conditional independence: $p(\boldsymbol{\ell}_{1:T}, X_{1:T}) = \prod_{t \in [T]} p(X_t \mid \boldsymbol{\ell}_{1:t-1}, X_{1:t-1}) p(\boldsymbol{\ell}_t \mid \boldsymbol{\ell}_{1:t-1}, X_{1:t-1})$, where $X_{1:0} = \boldsymbol{\ell}_{1:0} = \{\}$. $p(\boldsymbol{\ell}_t \mid X_{1:t-1}, \boldsymbol{\ell}_{1:t-1})$ corresponds to the adversary's strategy and $p(X_t \mid X_{1:t-1}, \boldsymbol{\ell}_{1:t-1})$ corresponds to the player's strategy. Since the player cannot directly observe $\boldsymbol{\ell}_{1:t}$, the player's strategy must satisfy $p(X_t \mid X_{1:t-1}, \boldsymbol{\ell}_{1:t-1}) = p(X_t \mid X_{1:t-1}, c_{1:t-1})$. Using the joint distribution p, expected regret $\overline{\mathrm{R}}_T$ is defined as

$$\overline{\mathrm{R}}_T := \max_{X \in \mathcal{S}} \mathbb{E}_{\boldsymbol{\ell}_{1:T}, X_{1:T} \sim p} \left[\sum_{t=1}^{T} \boldsymbol{\ell}_t^\top \mathbf{1}_{X_t} - \sum_{t=1}^{T} \boldsymbol{\ell}_t^\top \mathbf{1}_X \right].$$

The objective of BCO is to design the player's strategy $p(X_t \mid X_{1:t-1}, c_{1:t-1})$ so that it minimizes R_T or $\overline{\mathrm{R}}_T$. We abbreviate $p(X_t \mid X_{1:t-1}, c_{1:t-1})$ by $p_t(X_t)$.

2 BACKGROUND

There are three well-studied feedback settings for OCO: *full-information* ($\boldsymbol{\ell}_t$ is observable), *semi-bandit* ($\{\ell_{t,i} : i \in X_t\}$ is observable), and bandit ($c_t = \boldsymbol{\ell}_t^\top \mathbf{1}_{X_t}$ is observable). BCO is very challenging since its feedback is the least informative, and so it continues to be an attractive research subject [1, 3, 4, 7, 9, 10, 12, 21, 28, 36]; for further previous works on OCO we refer the readers to [2]. We here show some relevant existing works focusing on the two features of our algorithm: efficiency and any-time guarantees.

Efficiency: The multiplicative weight update (MWU) strategy with exponential factors (or *Hedge* algorithm [14]) is often employed to design efficient OCO algorithms. For the full-information OCO, various efficient Hedge-based algorithms have been developed [27, 33, 35]. Our algorithm is related to the Hedge-based algorithm for OSPs on DAGs [35]; both their algorithm and ours use the *weight pushing* technique [30] as their building block. Full-information OCO problems whose constraint is characterized by a polytope are studied in [27], and other full-information OCO problems called *dynamic programming games* are considered in [33]; we are interested in a different class of constraints that are represented compactly with ZDDs. Applying the Hedge-based algorithms to BCO problems is not straightforward since those algorithms require $\boldsymbol{\ell}_t$ in each t-th round, which cannot be observed in the bandit feedback setting. A common approach to overcome this difficulty is to use an unbiased estimator $\hat{\boldsymbol{\ell}}_t$ of $\boldsymbol{\ell}_t$, which can be obtained by computing a *co-occurrence probability matrix* (CPM); see, for example, [4, 9, 12]. ComBand [9] is a general BCO algorithm that employs such an approach and enjoys strong theoretical results. The drawback of ComBand is that its computation cost in each round is generally exponential in d; in particular computing a CPM requires $O(d^2|\mathcal{S}|)$ time. To avoid such expensive computation, CombEXP [10] was proposed, which is basically a bandit counterpart of [27]. CombEXP scales up ComBand by employing a projection onto the convex hull of an action set via KL-divergence. For some action sets for which the projection can be done efficiently (e.g., m-sets or a set of matchings), CombEXP runs faster than ComBand, while achieving almost the same regrets. However, it is difficult to perform the projection for other action sets (e.g., s-r paths or Steiner trees); actually it is NP-hard to do the projection in the case of OSP and DST on undirected networks. On the other hand, thanks to recent advances in constructing *decision diagrams* (DDs), optimization techniques using DDs are attracting much attention [5, 11, 31]. Those techniques are advantageous in that DDs can efficiently store all solutions satisfying some complex constraints; for example, constraints that are hard to represent as a set of inequalities. The ZDD [29], which we use in our algorithm, is a kind of DD that is known to be suitable for storing specific network substructures (e.g., s-r paths or Steiner trees). Thus our algorithm with ZDDs, called CombD3, runs fast in many BCO instances defined on networks, including OSP, DST, and CG. To the best of our knowledge, this is the first OCO algorithm that exploits the compactness of DDs.

Any-time guarantees: When using online optimization algorithms in practice, we rarely know how long the algorithms will continue to be executed in advance; in other words the value of time horizon n is seldom available. Given this, the regret values of online algorithms should be bounded at any T-th round ($T \in [n]$) without using n; such regret bounds are called *any-time guarantees* [6]. How to obtain any-time guarantees has been studied for ordinary online learning problems [8], OCO problems with a variant of semi-bandit feedback [26], and the BCO [6]. However any-time guarantees of ComBand [9], which we use to construct our efficient BCO algorithm, have not been established;[1] in [9] it is proved to achieve $O(\sqrt{n})$ expected regret at the n-th round. We prove that ComBand with slight modification, called CombD, achieves any-time guarantees by using the techniques shown in [6, 8].

3 OVERVIEW OF OUR METHOD

We provide a high-level sketch of our algorithm. Given action set $\mathcal{S}$, we first construct a ZDD that stores all $X \in \mathcal{S}$ compactly; all procedures of our algorithm are performed on the ZDD, and thus we avoid dealing with action set $\mathcal{S}$ explicitly. As shown later, a ZDD is a DAG-shaped data structure, and each action is represented as a path that connects two specified vertices. Therefore the original BCO instance reduces to an OSP on the ZDD with bandit feedback. The BCO algorithm, CombD, for this problem can be performed efficiently on the ZDD by using DP methods. This ZDD-based algorithm is called CombD3, and it runs in $O(d|V|)$ time in each round, where V is the vertex set of the ZDD. Empirically $|V|$ is much smaller than $|\mathcal{S}|$, and thus CombD3 runs dramatically faster than existing BCO algorithms whose complexity depends on $|\mathcal{S}|$.

In Section 4 we detail CombD, a BCO algorithm that takes $O(d^2|\mathcal{S}|)$ time in general. In Section 5 we explain CombD3; we first present the details of ZDDs, and then introduce the DP methods, with which we can perform CombD on ZDDs efficiently.

Remark 1. At first glance the original BCO instance may appear to be solved just by applying existing OSP algorithms [15, 27, 35] to the aforementioned OSP on ZDDs, however it is not true. The problems they considered are OSPs with full-information or semi-bandit feedback, and thus how to obtain the unbiased estimator $\hat{\boldsymbol{\ell}}_t$ of $\boldsymbol{\ell}_t$, which is computationally the most difficult part, is unclear in our bandit feedback setting. We overcome this problem by developing novel DP methods that enable us to compute a CPM efficiently on ZDDs; with the CPM we can compute $\hat{\boldsymbol{\ell}}_t$.

4 CombD: ComBand WITH DECREASING PARAMETERS

Algorithm 1 CombD($\alpha, \mathcal{S}$)

```
1: $\hat{L}_{0,i} \leftarrow 0$, $w_{1,i} \leftarrow 1$ $(i \in E)$
2: for $t = 1, \dots, n$ do
3:     $\gamma_t \leftarrow \frac{t^{-1/\alpha}}{2}$, $\eta_{t+1} \leftarrow \frac{\lambda(t+1)^{-1/\alpha}}{2D^2}$
4:     $X_t \sim p_t$                      $\triangleright$ output for $t$-th round
5:     $c_t \leftarrow \boldsymbol{\ell}_t^\top \mathbf{1}_{X_t}$                  $\triangleright$ $\boldsymbol{\ell}_t$ is unobservable
6:     $P_t(i,j) \leftarrow \sum_{X \in \mathcal{S}: i,j \in X} p_t(X)$ $(i,j \in E)$
7:     $\hat{\boldsymbol{\ell}}_t \leftarrow c_t P_t^+ \mathbf{1}_{X_t}$
8:     $\hat{L}_{t,i} \leftarrow \hat{L}_{t-1,i} + \hat{\ell}_{t,i}$ $(i \in E)$
9:     $w_{t+1,i} \leftarrow \exp\left(-\eta_{t+1}\hat{L}_{t,i}\right)$ $(i \in E)$
```

We here describe CombD, which is obtained by introducing the decreasing parameters (see, e.g., [8]) to ComBand [9].

With CombD, the player's strategy $p_t(X_t)$ is designed as in Algorithm 1. In what follows, we define $D := \max_{X \in \mathcal{S}} \|\mathbf{1}_X\|$ for any given $\mathcal{S} \subseteq 2^E$, where $\|\cdot\|$ is the Euclidian norm. We also define λ as the smallest non-zero eigenvalue of $\mathbb{E}_{X \sim u}[\mathbf{1}_X \mathbf{1}_X^\top]$, where u is the uniform distribution over $\mathcal{S}$. For some cases, the value of λ is proved to be bounded from below [9, 10].

[1] An any-time guarantee of ComBand is stated in [6], but the proof seems to include some mistakes. However, their proof techniques are still useful, and so we prove the $O(T^{2/3})$ high-probability regret bound of CombD by modifying their proof and using the techniques in [8].

Given non-negative vector $\boldsymbol{w} = (w_1, \ldots, w_d)^\top \in \mathbb{R}^d$ and action set $\mathcal{S} \subseteq 2^E$, we define the *action distribution* as $p(X; \boldsymbol{w}, \mathcal{S}) := \frac{w(X)}{Z(\boldsymbol{w}, \mathcal{S})}$, where $w(X) := \prod_{i \in X} w_i$ and $Z(\boldsymbol{w}, \mathcal{S}) := \sum_{X \in \mathcal{S}} w(X)$. Using the above, we define the player's strategy $p_t(X_t)$, which appears in Step 4, as follows:

$$(1)\quad p_t(X_t) := (1 - \gamma_t) p(X_t; \boldsymbol{w}_t, \mathcal{S}) + \gamma_t p(X_t; \mathbf{1}_E, \mathcal{S}),$$

where $\boldsymbol{w}_t = (w_{t,1}, \ldots, w_{t,d})^\top$ is the weight vector defined in Step 9, and γ_t is the parameter defined in Step 3; we note that $p(X_t; \mathbf{1}_E, \mathcal{S})$ is the uniform distribution over $\mathcal{S}$. Thus p_t is a mixture of two action distributions with the mixture rate γ_t.

For any distribution p over $\mathcal{S}$, a matrix P is called a co-occurrence probability matrix (CPM) if its (i, j) entry $P(i, j)$ is given by the *co-occurrence probability* $p(i \in X, j \in X) := \sum_{X \in \mathcal{S}: i,j \in X} p(X)$. Matrix P_t computed in Step 6 is the CPM of p_t, and P_t^+ used in Step 7 is the pseudo-inverse of P_t. From Eq. (1), we have

$$(2)\quad \begin{aligned} P_t(i, j) = (1 - \gamma_t) p(i \in X, j \in X; \boldsymbol{w}_t, \mathcal{S}) + \\ \gamma_t p(i \in X, j \in X; \mathbf{1}_E, \mathcal{S}). \end{aligned}$$

With CPM P_t, the unbiased estimator, $\hat{\boldsymbol{\ell}}_t$, of $\boldsymbol{\ell}_t$ is obtained as in Step 7; strictly speaking, we have $\mathbb{E}_t[\hat{\boldsymbol{\ell}}_t^\top \mathbf{1}_X] = \boldsymbol{\ell}_t^\top \mathbf{1}_X$ for all $X \in \mathcal{S}$, where $\mathbb{E}_t[\cdot]$ is the conditional expectation in the t-th round (see Section S2.2 in the supplementary material for details).

The above CombD is based on ComBand [9]; CombD with fixed parameters $\gamma_t = \frac{D}{\lambda}\sqrt{\frac{\ln |\mathcal{S}|}{n(d/D^2 + 2/\lambda)}}$ and $\eta_t = \frac{1}{D}\sqrt{\frac{\ln |\mathcal{S}|}{n(d/D^2 + 2/\lambda)}}$ completely corresponds to the original ComBand and achieves $O(\sqrt{n})$ expected regret [9]. Whereas the regret of ComBand is bounded only at the n-th round, CombD achieves the following any-time guarantees:

Theorem 1. *Given any $\mathcal{S}$, CombD($\alpha = 3, \mathcal{S}$) achieves $\mathrm{R}_T \le O\Big(\Big(\frac{d\lambda}{D^2} + \sqrt{\frac{D^2}{\lambda} \ln \frac{|\mathcal{S}|+2}{\delta}}\Big) T^{2/3}\Big)$ with probability at least $1 - \delta$ for any fixed $T \in [n]$.*

Theorem 2. *Given any $\mathcal{S}$, CombD($\alpha = 2, \mathcal{S}$) achieves $\overline{\mathrm{R}}_T \le O\Big(\Big(\frac{d\lambda}{D^2} + \frac{D^2 \ln |\mathcal{S}|}{\lambda}\Big)\sqrt{T}\Big)$ for all $T \in [n]$.*

In other words, CombD achieves either $O(T^{2/3})$ regret with high probability or $O(\sqrt{T})$ expected regret as an any-time guarantee by setting the value of hyper parameter α appropriately. The technique of using decreasing parameters for online optimization problems is studied in [8], and similar results for BCO problems are obtained in [6]. However, we note that the proofs of Theorem 1 and Theorem 2 cannot be obtained simply by combining those existing results. For details see the supplementary material.

There are two difficulties when performing CombD in practice; the first is sampling from the player's strategy $p_t(X_t)$ (Step 4), and the second is computing CPM P_t (Step 6). Naive methods for the two steps require $O(d|\mathcal{S}|)$ and $O(d^2|\mathcal{S}|)$ times, respectively, where $|\mathcal{S}|$ is generally exponential in d. In the next section, we show efficient ZDD-based methods for sampling from any given action distribution $p(X; \boldsymbol{w}, \mathcal{S})$ and for computing the CPM of $p(X; \boldsymbol{w}, \mathcal{S})$. Because p_t is a mixture of two action distributions, we can efficiently sample from p_t and compute the CPM of p_t using those methods.

5 CombD3: CombD WITH DECISION DIAGRAMS

As shown above, CombD requires sampling from action distributions and computing CPMs as its building blocks, which generally require $O(d|\mathcal{S}|)$ and $O(d^2|\mathcal{S}|)$ computation times, respectively. Moreover, computing D can also require $O(d|\mathcal{S}|)$ time. Those computation costs can be prohibitive since $|\mathcal{S}|$ is generally exponential in d. As the building blocks, we present efficient DP algorithms performed on a ZDD that represents $\mathcal{S}$. We first detail the ZDDs considered here and then present the DP methods for sampling an action and computing CPMs; the sampling method is based on [30, 35], and the CPM computation is our proposal. D can also be computed in a DP manner on a ZDD.

5.1 Zero-suppressed Binary Decision Diagrams (ZDDs)

A ZDD [29] is a compact graph representation of a family of sets. Given $\mathcal{S} \subseteq 2^E$, a ZDD for $\mathcal{S}$ is a DAG denoted by $\mathbf{G}_\mathcal{S} = (V, A)$, where $V = \{0, 1, \ldots, |V| - 1\}$ is a set of vertices and $A \subseteq V \times V$ is a set of arcs (directed edges). $\mathbf{G}_\mathcal{S}$ contains one root vertex $r \in V$ and two terminal vertices: 0-terminal and 1-terminal. Without loss of generality, we assume that $V = \{0, \ldots, |V| - 1\}$ is arranged in a topological order of $\mathbf{G}_\mathcal{S}$; $r = |V| - 1$ holds and the b-terminal ($b \in \{0, 1\}$) is denoted simply by $b \in V$. Each non-terminal vertex $v \in V \backslash \{0, 1\}$ is labeled by an integer in $E = [d]$ and has exactly two outgoing arcs: 0-arc and 1-arc. A vertex pointed by the b-arc of v is called the b-child of v. We use l_v, a_v^b, and c_v^b to denote v's label, b-arc, and b-child, respectively; consequently $a_v^b = (v, \mathrm{c}_v^b)$ holds. We use $\mathcal{R}_{v,u}$ ($v, u \in V$) to denote a set of routes (directed paths) from v to u on $\mathbf{G}_\mathcal{S}$, where route $R \in \mathcal{R}_{v,u}$ is a set of arcs: $R \subseteq A$. Given route R, we define $X(R) \subseteq E$ as $X(R) := \{l_v \in E \mid (v, \mathrm{c}_v^1) \in R\}$; intuitively, when proceeding along R, we add l_v to $X(R)$ if $(v, v') \in R$ is a 1-arc. Then, $\mathbf{G}_\mathcal{S}$ satisfies

$$(3)\qquad \mathcal{S} = \{X(R) \mid R \in \mathcal{R}_{r,1}\}.$$

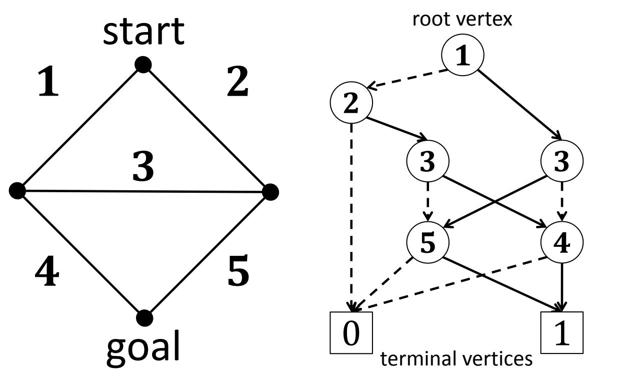


Figure 1: The left figure is an example network with edge set $E = \{1, \ldots, 5\}$. The right figure is a ZDD that stores all paths from the start to the goal; each non-terminal vertex v is labeled $l_v \in E$, and 0-arcs and 1-arcs are indicated by dashed and solid lines, respectively. Note that we have $\mathcal{S} = \{X(R) \mid R \in \mathcal{R}_{r,1}\} = \{\{1,4\},\{2,5\},\{1,3,5\},\{2,3,4\}\}$.

Thus, $\mathbf{G}_\mathcal{S}$ represents action set $\mathcal{S}$ as a set of all routes from its root r to the 1-terminal. Note that, once $\mathbf{G}_\mathcal{S}$ is obtained, $D = \max_{X \in \mathcal{S}} \sqrt{|X|}$ is easily computed by a DP method to find $R \in \mathcal{R}_{r,1}$ that maximizes $|X(R)|$.

In general, a ZDD is assumed to be *ordered* and *reduced*. $\mathbf{G}_\mathcal{S}$ is *ordered* if $v > u \Rightarrow l_v < l_u$ holds for all $v, u \in V \backslash \{0, 1\}$. A non-terminal vertex v is *redundant* if $\mathrm{c}_v^1 = 0$: its 1-arc directly points to the 0-terminal. Any redundant vertex v can be removed by replacing all $(u, v) \in A$ with (u, c_v^0) without loss of property (3). A non-terminal vertex v is said to be *sharable* if there exists another vertex v' such that $l_v = l_{v'}$ and $\mathrm{c}_v^b = \mathrm{c}_{v'}^b$ ($b \in \{0, 1\}$): v and v' have the same label and children. Any sharable vertex v can be removed by replacing $(u, v) \in A$ with (u, v'). $\mathbf{G}_\mathcal{S}$ is said to be *reduced* if no vertex is redundant or sharable. In this paper, we assume $\mathbf{G}_\mathcal{S}$ is ordered and reduced. Figure 1 shows an example of such a ZDD. We also provide a summary of notations and symbols for ZDDs in Table 1, which will be helpful to understand the following discussion.

ZDDs are known to be able to store various families of sets compactly in many applications. As we will see later, the time and space complexities of CombD3 in each round are both $O(d|V|)$, and thus it runs fast if the ZDD is small. Although $|V|$ is generally exponential in d, we can bound $|V|$ theoretically in some cases. We here present two examples of such cases:

Knapsack constraints: Suppose that each $i \in E$ is associated with a positive integer b_i and that a budget value $B > 0$ is given. We here consider an action set that arises from the *knapsack constraint*: $\mathcal{S} = \{X \subseteq E \mid \sum_{i \in X} b_i \leq B\}$. In this setting, the ZDD maintains at most B kinds of states for each $i \in E$

Table 1: Notations and symbols for ZDDs

$\mathbf{G}_\mathcal{S} = (V, A)$	ZDD storing $\mathcal{S}$
$r = \lvert V\rvert - 1 \in V$	root vertex
$0, 1 \in V$	0- and 1-terminals
$l_v \in E$	label of $v \in V$
$a_v^b \in A$	b-arc outgoing from v ($b \in \{0,1\}$)
$\mathrm{c}_v^b \in V$	b-child of v ($b \in \{0,1\}$)
$\mathcal{R}_{v,u} \subseteq 2^A$	set of all routes connecting v to u
$X(R) \subseteq E$	subset of E represented by route R

(see, [5] for details) and thus the resulting ZDD size is at most $O(dB)$. Consequently, our CombD3 runs in $O(d^2B)$ time in each round. This complexity result matches that of the state-of-the-art full-information OCO algorithm [33], even though our bandit setting is harder than the full-information setting.

Networks with bounded pathwidth: Given network $G = (N, E)$ and action set $\mathcal{S}$ that consists of certain subnetworks (e.g., s-r paths, Steiner trees, matching, and cliques), we consider a ZDD $\mathbf{G}_\mathcal{S} = (V, A)$ that stores all $X \in \mathcal{S}$. When constructing such a ZDD, the *frontier-based search* (FBS) [22] is often used, which is a fast construction algorithm based on Knuth's *Simpath* algorithm [25]. FBS constructs an ordered and reduced ZDD in a top-down manner using the *mate* arrays, which maintain current search states as some positive integers. If ω_G is the *pathwidth* of G and δ is the largest integer stored in the mate arrays, then the size of ZDDs constructed by FBS can be bounded as $|V| \leq d\delta^{\omega_G}$; in practice we often have $|V| \lll d\delta^{\omega_G}$ (see [18, 22] for the details). In the worst case we have $\omega_G = |N|$, and thus the resulting ZDD size, $|V|$, can be exponential in $|N|$. However, if ω_G is bounded by a small constant, $|V|$ is also bounded, and so are the time and space complexities of our algorithm. The time and space complexities of FBS are bounded by $O(d\delta^{\omega_G})$. In practice ZDDs for storing various subnetworks are obtained easily via existing software [17].

5.2 Sampling from Action Distributions

We show an efficient algorithm performed on ZDD $\mathbf{G}_\mathcal{S}$ for sampling from action distribution $p(X; \boldsymbol{w}, \mathcal{S})$, which is based on the weight pushing [30, 35]; similar methods on DDs have been studied in the field of logic-based probabilistic modeling [19, 20].

We first introduce the following *forward weight* (FW) F_v and *backward weight* (BW) B_v ($v \in V$):

$$\mathrm{F}_v := \sum_{R \in \mathcal{R}_{r,v}} w(R), \qquad \mathrm{B}_v := \sum_{R \in \mathcal{R}_{v,1}} w(R),$$

where $w(R)$ is an abbreviation of $w(X(R)) = \prod_{i \in X(R)} w_i$. Note that we have $Z(\boldsymbol{w}, \mathcal{S}) = \mathrm{B}_r = \mathrm{F}_1$.

```
1: Algorithm FW($\mathbf{G}_\mathcal{S}, \boldsymbol{w}$)
2: $\mathrm{F}_r \leftarrow 1$
3: $\mathrm{F}_v \leftarrow 0$ $(\forall v \in V \backslash \{r\})$
4: for $v = r, \dots, 2$ do
5:     $\mathrm{F}_{\mathrm{c}_v^0} \mathrel{+}= \mathrm{F}_v$
6:     $\mathrm{F}_{\mathrm{c}_v^1} \mathrel{+}= w_{l_v} \mathrm{F}_v$
7: return $\mathrm{F} := \{\mathrm{F}_0, \dots, \mathrm{F}_r\}$
```

```
1: Algorithm BW($\mathbf{G}_\mathcal{S}, \boldsymbol{w}$)
2: $\mathrm{B}_1 \leftarrow 1$
3: $\mathrm{B}_v \leftarrow 0$ $(\forall v \in V \backslash \{1\})$
4: for $v = 2, \dots, r$ do
5:     $\mathrm{B}_v \leftarrow \mathrm{B}_{\mathrm{c}_v^0} + w_{l_v} \mathrm{B}_{\mathrm{c}_v^1}$
6: return $\mathrm{B} := \{\mathrm{B}_0, \dots, \mathrm{B}_r\}$
```

```
1: Algorithm Draw($\mathbf{G}_\mathcal{S}, \boldsymbol{w}, \mathrm{B}$)
2: $X \leftarrow \{\}$, $v \leftarrow r$
3: while $v > 1$ do
4:     $\theta \leftarrow w_{l_v} \mathrm{B}_{\mathrm{c}_v^1} / \mathrm{B}_v$
5:     $b \sim \mathrm{Ber}(\theta)$
6:     $X \leftarrow X \cup \{l_v\}$ if $b = 1$
7:     $v \leftarrow \mathrm{c}_v^b$
8: return $X$
```

```
1: Algorithm BWC($\mathbf{G}_\mathcal{S}, \boldsymbol{w}, \mathrm{B}$)
2: $\mathrm{C}_{v,j} \leftarrow 0$ $(\forall v \in V, \forall j \in E)$
3: for $v = 2, \dots, r$ do
4:     $\mathrm{C}_{v,l_v} \leftarrow w_{l_v} \mathrm{B}_{\mathrm{c}_v^1}$
5:     for $j = l_v + 1, \dots, d$ do
6:         $\mathrm{C}_{v,j} \leftarrow \mathrm{C}_{\mathrm{c}_v^0, j} + w_{l_v} \mathrm{C}_{\mathrm{c}_v^1, j}$
7: return $\mathrm{C} := \{\mathrm{C}_{v,j} \mid v \in V, j \geq l_v\}$
```

```
1: Algorithm CPM($\mathbf{G}_\mathcal{S}, \boldsymbol{w}, \mathrm{B}, \mathrm{F}, \mathrm{C}$)
2: $\mathrm{P}_{i,j} \leftarrow 0$ $(\forall i, j \in E)$
3: for $v = 2, \dots, r$ do
4:     $i \leftarrow l_v$
5:     $\mathrm{P}_{i,i} \mathrel{+}= \mathrm{F}_v w_i \mathrm{B}_{\mathrm{c}_v^1} / \mathrm{B}_r$
6:     for $j = i + 1, \dots, d$ do
7:         $\mathrm{P}_{i,j} \mathrel{+}= \mathrm{F}_v w_i \mathrm{C}_{\mathrm{c}_v^1, j} / \mathrm{B}_r$
8: return $\mathrm{P} := \{\mathrm{P}_{i,j} \mid i, j \in [d], i \leq j\}$
```

$\mathrm{B} := \{\mathrm{B}_0, \dots, \mathrm{B}_r\}$ and $\mathrm{F} := \{\mathrm{F}_0, \dots, \mathrm{F}_r\}$ can be efficiently computed in a DP manner on $\mathbf{G}_\mathcal{S}$ as shown in Algorithm FW($\mathbf{G}_\mathcal{S}, \boldsymbol{w}$) and BW($\mathbf{G}_\mathcal{S}, \boldsymbol{w}$). Once we obtain B, we can draw a sample from $p(X; \boldsymbol{w}, \mathcal{S})$ by top-down sampling on $\mathbf{G}_\mathcal{S}$ without rejection as shown in Algorithm Draw($\mathbf{G}_\mathcal{S}, \boldsymbol{w}, \mathrm{B}$), where $\mathrm{Ber}(\theta)$ is the Bernoulli distribution with parameter $\theta \in [0, 1]$. The total space and time complexities are both $O(|V|)$.

5.3 Computing Co-occurrence Probabilities

Given action distribution $p(X; \boldsymbol{w}, \mathcal{S})$, we define $\mathrm{P}_{i,j} := p(i \in X, j \in X; \boldsymbol{w}, \mathcal{S})$ as the co-occurrence probability of i and j $(i, j \in E)$. We here propose an efficient algorithm for computing $\mathrm{P}_{i,j}$ $(i \leq j)$, which suffices for obtaining $\mathrm{P}_{i,j}$ for all $i, j \in E$ since $\mathrm{P}_{i,j} = \mathrm{P}_{j,i}$. If we compute $P_{i,j}$ $(i \leq j)$ naively by traversing a ZDD, we need $O(d^2|V|)$ computation time, which is too expensive if d is large. Fortunately, they can be computed in $O(d|V|)$ time by using FW, BW, and *backward weighted co-occurrence* (BWC) as follows.

From Eq. (3), $\mathrm{P}_{i,j}$ can be written as follows:

$$\mathrm{P}_{i,j} = \sum_{R \in \mathcal{R}_{r,1} : i,j \in X(R)} \frac{w(R)}{Z(\boldsymbol{w}, \mathcal{S})}. \tag{4}$$

We first consider $\mathrm{P}_{i,i}$ as a special case of $\mathrm{P}_{i,j}$, which can be expressed as follows by using FW and BW (see the supplementary material for details):

$$\mathrm{P}_{i,i} = \sum_{v \in V : l_v = i} \frac{\mathrm{F}_v w_i \mathrm{B}_{\mathrm{c}_v^1}}{\mathrm{B}_r}.$$

Next, to compute $\mathrm{P}_{i,j}$ $(i < j)$, we rewrite the right hand side of Eq. (4) using BWC $\mathrm{C}_{v,j}$ $(j > l_v)$ as follows (details are provided in the supplementary material):

$$\mathrm{P}_{i,j} = \sum_{v \in V : l_v = i} \frac{\mathrm{F}_v w_i \mathrm{C}_{\mathrm{c}_v^1, j}}{\mathrm{B}_r},$$

where $\mathrm{C}_{v,j} := \sum_{R \in \mathcal{R}_{v,1} : j \in X(R)} w(R)$.

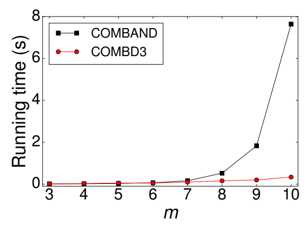
Because $\mathrm{C}_{v,j}$ is a variant of B_v, $\mathrm{C} := \{\mathrm{C}_{v,j} \mid v \in V, j \geq l_v\}$ can be computed in a similar manner to B as shown in Algorithm BWC($\mathbf{G}_\mathcal{S}, \boldsymbol{w}, \mathrm{B}$), whose time and space complexities are both $O(d|V|)$. To conclude, $\mathrm{P}_{i,j}$ can be computed by Algorithm CPM($\mathbf{G}_\mathcal{S}, \boldsymbol{w}, \mathrm{F}, \mathrm{B}, \mathrm{C}$). The total space and time complexities of computing $\mathrm{P} := \{\mathrm{P}_{i,j} \mid i \leq j\}$ are both $O(d|V|)$.

6 EXPERIMENTS

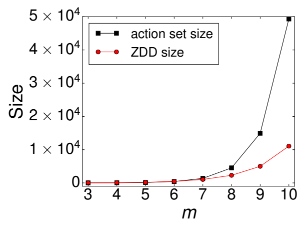
We consider three network-based BCO problems: OSP, DST, and CG. For OSP and DST (Section 6.1), we use artificial networks to observe the scalability of algorithms. For CG (Section 6.2), we use two real-world networks to show the practical utility of CombD3. All algorithms were implemented in the C programming language, and we used Graphillion [17] to obtain the ZDDs. We note that constructing ZDDs with the software is not a drawback; in all of the following instances ZDD construction took only a few seconds at most.

6.1 OSP and DST on Artificial Networks

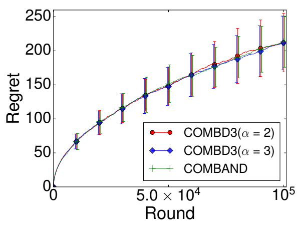
Experimental setting: We consider OSP and DST instances on artificial networks, which are undirected



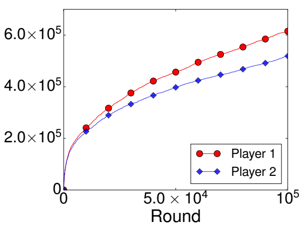
(a) OSP



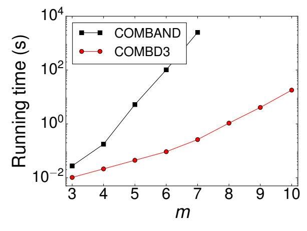
(c) OSP



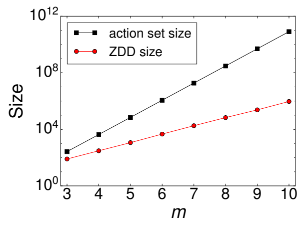
(e) OSP



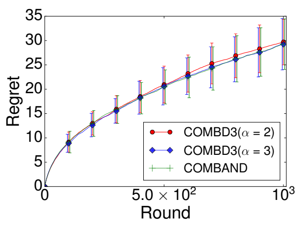
(g) CG on MCI



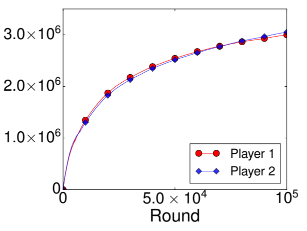
(b) DST (semi-log)



(d) DST (semi-log)



(f) DST



(h) CG on ATT

Figure 2: (a),(b) Running times that CombD3 and ComBand required for 100 iterations (averaged over 100 trials). (c),(d) Action set sizes, $|\mathcal{S}|$, and ZDD sizes, $|V|$. (e),(f) Achieved regret values of each algorithm on the 3×10 grid; the regret values are averaged over 100 trials and the error bars indicate the standard deviations. (g),(h) Regret values of each player for CG on MCI and ATT, respectively.

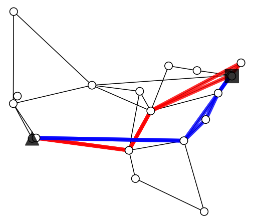
grid networks with $3 \times m$ nodes ($m = 3, \ldots, 10$). In both problems, E is the edge set of the given network. In OSP, action set $\mathcal{S}$ is the set of all s-r paths from starting node s to goal node r that are placed on diagonal corners of the given grid. In DST, $\mathcal{S}$ is the set of all Steiner trees that contain the four corners of the grid. The aim of the player is to minimize the cumulative cost of the selected subnetworks over some time horizon. In this experiment, we define loss vector $\boldsymbol{\ell}_t$ as follows: We first uniformly sample $\boldsymbol{\mu}_0$ from $[0,1]^d$. In the t-th round, we set $\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1}$ with probability 0.9 or draw a new $\boldsymbol{\mu}_t$ uniformly from $[0,1]^d$ with probability 0.1. Then, we set $\ell_{t,i} = 1/d$ with probability $\mu_{t,i}$ and $-1/d$ with probability $1 - \mu_{t,i}$ for each $i \in E$, where $\mu_{t,i}$ is the i-th entry of $\boldsymbol{\mu}_t$. In the short run, this setting is a stochastic OCO problem where each $\ell_{t,i}$ is drawn from $\mathrm{Ber}(\mu_{t,i})$, but the adversary secretly resets $\boldsymbol{\mu}_t$ with probability 0.1 in each round to foil the player.

Computation cost: We compared CombD3 with ComBand that uses explicit enumeration of possible actions. The running times of the two methods for OSPs and DSTs are shown in Figures 2 (a) and (b), respectively; running times of 100 iterations averaged over 100 trials are shown for each method. The results of ComBand for DSTs with $m \geq 8$ are omitted due to memory shortage. Figures 2 (c) and (d) plot the sizes of action sets, $|\mathcal{S}|$, and corresponding ZDD sizes, $|V|$, for OSPs and DSTs, respectively. We see that CombD3, which runs in $O(d|V|)$ time, is far more scalable than ComBand, which runs in $O(d^2|\mathcal{S}|)$ time, thanks to the compactness of ZDDs. In particular, for DST on the 3×10 grid, we see that $|V|$ is five orders of magnitude smaller than $|\mathcal{S}|$. In such cases, where existing methods suffer from the complex structure of $\mathcal{S}$ (e.g., no efficient optimization algorithm over $\mathcal{S}$ is available) or the enormous size of $\mathcal{S}$, our CombD3 is the only practical method.

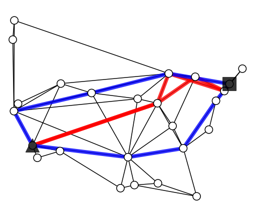
Empirical regret: Regret values achieved by CombD3 ($\alpha = 2, 3$) and ComBand for OSP and DST on the 3×10 grid are shown in Figures 2 (e) and (f), respectively. As shown previously, ComBand with explicit enumeration is too costly, and thus we here implemented ComBand using ZDDs. We see that the regret values of the three methods actually grow sublinearly. Note that the regret values of CombD3, which does not require n, is comparable to those of ComBand, which requires n as its input. We confirmed that CombD3 yielded lower regret values than those of the theoretical bounds stated in Theorems 1 and 2; the precise values of the bounds are shown in the supplementary material.

6.2 CG on Real-world Networks

Experimental setting: We applied CombD3 to the congestion game (CG), which is a multi-player version of the OSP, on two real-world networks. CG is described as follows: Given m players and an undirected network with starting node s and goal node r, the players concurrently send a message from s to r. The aim of each player is to minimize the cumulative time needed to send n messages. In this problem E is the edge set of the given network, an action is an s-r path,



(a) MCI



(b) ATT

Figure 3: (a) and (b) are the topologies of the two networks; the triangles (squares) are the starting (goal) nodes. The red (blue) paths indicate the top two paths most frequently chosen by player 1 (2).

and action set $\mathcal{S}$ is the set of all s-r paths. The loss value of an edge in E is the transmission time taken to send a message along the edge, and the cost of an action is the total transmission time needed to send a message along the selected s-r path. We assume that the loss of each edge increases with the number of players who use the same edge at the same time; formally a player regards the other players as adversaries. We use $X_t^k \in \mathcal{S}$ ($k \in [m]$) to denote the k-th player's choice in the t-th round and use $X_{t,i}^k \in \{0,1\}$ to denote the i-th entry of $\mathbf{1}_{X_t^k}$. We also use $\ell_{t,i}^k$ to denote the transmission time taken by the k-th player to send a message along the i-th edge in the t-th round. We here define $\ell_{t,i}^k := \beta_i \kappa^{N_{t,i}^{-k}}$, where $\beta_i \in \mathbb{R}$ is the length of the edge, κ is an overhead constant, and $N_{t,i}^{-k} := \sum_{k' \neq k} X_{t,i}^{k'}$ is the number of adversaries who also choose the i-th edge at the t-th round. Namely, we assume that the transmission time of each edge increases exponentially with the number of players using the same edge at the same time. Consequently, to reduce the total transmission time, the players should adaptively avoid contending with each other. In this experiments, we set $m = 2$ and $\kappa = 10$. This setting violates the assumption $|c_t| < 1$; however, in practice, this violation barely matters. We let both players use CombD3, and the objective of this problem is to let each player avoid contending with each other and reduce his/her transmission time.

We use two real-world communication networks in the Internet topology zoo [24]: the InternetMCI network (MCI), and the ATT North America network (ATT). Figures 3 (a) and (b) illustrate the topologies of MCI and ATT, respectively. Both networks correspond to the U.S. map and we choose Los Angeles as the starting point s and New York as the goal r. The statistics for each network are shown in Table 2.

Experimental results: Table 2 shows that the action sets of real-world networks can also be represented as compact ZDDs, and thus CombD3 is more efficient than BCO algorithms that deal with the action sets directly. Figures 2 (g) and (h) plot the regret values of each player for MCI and ATT, respectively. The figures show that each player attained sublinear regrets. Figures 3 (a) and (b) show the top two most frequently selected paths for each player. We see that each player successfully avoided congestion. For the case where the players can observe the costs of all s-r paths after choosing the current paths, it is known that the MWU-based algorithms can achieve the Nash equilibria on CG [23, 32]. In this experiment, even though we employed the bandit feedback setting where each player can observe only the cost of the selected path, the players successfully found almost optimal strategies on both networks by using CombD3. To conclude, the results suggest that CombD3 is useful for adaptive routing problems on real-world networks.

Table 2: Statistics for two real-world networks.

	# nodes	# edges	# s-r paths	ZDD size
MCI	19	33	1,444	756
ATT	25	56	213,971	37,776

7 CONCLUSION

We proposed CombD3, which is a practical and theoretically guaranteed algorithm for BCO. CombD3 is effective particularly for network-based BCO instances, including OSP, DST, and CG. The efficiency of CombD3 is thanks to its use of ZDDs, which offer compact representation of action sets. The time and space complexities of CombD3 are bounded by ZDD size; more precisely, they are $O(d|V|)$. ZDD size is bounded in some important cases (e.g., a knapsack constraint and some constraints defined on networks with bounded pathwidth), and the complexities of CombD3 are also bounded in such cases. When implementing the BCO algorithm with ZDDs, the most difficult part is designing how to compute the unbiased estimator $\hat{\boldsymbol{\ell}}_t$ of $\boldsymbol{\ell}_t$, which is obtained using CPMs. We overcame this difficulty by developing DP-based methods for computing CPMs on ZDDs. Experiments showed that CombD3 is far more scalable than the existing method that deals with action sets directly. CombD3 is theoretically guaranteed to achieve either $O(T^{2/3})$ regret with high probability or $O(\sqrt{T})$ expected regret as an anytime guarantee, and we experimentally confirmed that CombD3 attained sublinear regrets. The results on CG showed that CombD3 is useful for adaptive routing problems on real-world networks.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Numbers 15H05711 and 16K16011.

References

[1] N. Ailon, K. Hatano, and E. Takimoto. Bandit online optimization over the permutahedron. In *International Conference on Algorithmic Learning Theory*, pages 215–229. Springer, 2014.

[2] J.-Y. Audibert, S. Bubeck, and G. Lugosi. Regret in online combinatorial optimization. *Math. Oper. Res.*, 39(1):31–45, February 2014.

[3] B. Awerbuch and R. D. Kleinberg. Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches. In *the 36th Annual ACM Symposium on Theory of Computing*, pages 45–53. ACM, 2004.

[4] P. L. Bartlett, V. Dani, T. Hayes, S. Kakade, A. Rakhlin, and A. Tewari. High-probability regret bounds for bandit online linear optimization. In *the 21st Annual Conference on Learning Theory*, pages 335–342. Omnipress, 2008.

[5] D. Bergman, A. A. Cire, W.-J. van Hoeve, and J. Hooker. *Decision Diagrams for Optimization*. Springer, first edition, 2016.

[6] G. Braun and S. Pokutta. An efficient high-probability algorithm for linear bandits. *arXiv preprint arXiv:1610.02072*, 2016.

[7] S. Bubeck, N. Cesa-Bianchi, and S. M. Kakade. Towards minimax policies for online linear optimization with bandit feedback. In *Conference on Learning Theory*, pages 1–14, 2012.

[8] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

[9] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. *J. Comput. Syst. Sci.*, 78(5):1404 – 1422, 2012.

[10] R. Combes, M. Sadegh Talebi, A. Proutiere, and M. Lelarge. Combinatorial bandits revisited. In *Advances in Neural Information Processing Systems*, pages 2116–2124, 2015.

[11] O. Coudert. Solving graph optimization problems with ZBDDs. In *1997 European Conference on Design and Test*, page 224. IEEE Computer Society, 1997.

[12] V. Dani, S. M. Kakade, and T. P. Hayes. The price of bandit information for online optimization. In *Advances in Neural Information Processing Systems*, pages 345–352, 2008.

[13] X. Fan, I. Grama, and Q. Liu. Hoeffding's inequality for supermartingales. *Stoch. Proc. Appl.*, 122(10):3545–3559, 2012.

[14] Y. Freund and R. E. Schapire. Adaptive game playing using multiplicative weights. *Games Econom. Behav.*, 29(1):79 – 103, 1999.

[15] A. György, T. Linder, G. Lugosi, and G. Ottucsák. The on-line shortest path problem under partial monitoring. *J. Mach. Learn. Res.*, 8(Oct):2369–2403, 2007.

[16] M. Imase and B. M. Waxman. Dynamic Steiner tree problem. *SIAM J. Discrete. Math.*, 4(3):369–384, 1991.

[17] T. Inoue, H. Iwashita, J. Kawahara, and S. Minato. Graphillion: software library for very large sets of labeled graphs. *Int. J. Software Tool. Tech. Tran.*, 18(1):57–66, 2016.

[18] Y. Inoue and S. Minato. Acceleration of ZDD construction for subgraph enumeration via pathwidth optimization. Technical report, TCS-TR-A-16-80, Hokkaido University, 2016.

[19] M. Ishihata, Y. Kameya, T. Sato, and S. Minato. Propositionalizing the EM algorithm by BDDs. In *the 18th International Conference on Inductive Logic Programming*, pages 44–49, 2008.

[20] M. Ishihata and T. Sato. Bayesian inference for statistical abduction using Markov chain Monte Carlo. In *the 3rd Asian Conference on Machine Learning*, pages 81–96, 2011.

[21] S. Kale, L. Reyzin, and R. E. Schapire. Non-stochastic bandit slate problems. In *Advances in Neural Information Processing Systems*, pages 1054–1062, 2010.

[22] J. Kawahara, T. Inoue, H. Iwashita, and S. Minato. Frontier-based search for enumerating all constrained subgraphs with compressed representation. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E100.A(9):1773–1784, 2017.

[23] R. Kleinberg, G. Piliouras, and E. Tardos. Multiplicative updates outperform generic no-regret learning in congestion games: Extended abstract. In *the 41st Annual ACM Symposium on Theory of Computing*, pages 533–542. ACM, 2009.

[24] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The Internet topology zoo. *IEEE J. Sel. Areas Commum.*, 29(9):1765–1775, 2011.

[25] D. E. Knuth. *The Art of Computer Programming: Combinatorial Algorithms, Part 1*, volume 4A. Addison-Wesley Professional, 1st edition, 2011.

[26] T. Kocák, G. Neu, M. Valko, and R. Munos. Efficient learning by implicit exploration in bandit problems with side observations. In *Advances in Neural Information Processing Systems*, pages 613–621, 2014.

[27] W. M. Koolen, M. K. Warmuth, and J. Kivinen. Hedging structured concepts. In *the 23rd Annual Conference on Learning Theory*, pages 93–105, 2010.

[28] H. B. McMahan and A. Blum. Online geometric optimization in the bandit setting against an adaptive adversary. In *the 17th Annual Conference on Learning Theory*, pages 109–123, 2004.

[29] S. Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *the 30th International Design Automation Conference*, pages 272–277. ACM, 1993.

[30] M. Mohri. Weighted automata algorithms. *Handbook of weighted automata*, pages 213–254, 2009.

[31] D. R. Morrison, E. C. Sewell, and S. H. Jacobson. Solving the pricing problem in a branch-and-price algorithm for graph coloring using zero-suppressed binary decision diagrams. *INFORMS J. Comput.*, 28(1):67–82, 2016.

[32] G. Palaiopanos, I. Panageas, and G. Piliouras. Multiplicative weights update with constant step-size in congestion games: convergence, limit cycles and chaos. In *Advances in Neural Information Processing Systems*, 2017.

[33] H. Rahmanian, S. Vishwanathan, and M. K. Warmuth. Online dynamic programming. In *Advances in Neural Information Processing Systems*, 2017.

[34] R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *Internat. J. Game Theory*, 2(1):65–67, 1973.

[35] E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *J. Mach. Learn. Res.*, 4(Oct):773–818, 2003.

[36] T. Uchiya, A. Nakamura, and M. Kudo. Algorithms for adversarial bandit problems with multiple plays. In *International Conference on Algorithmic Learning Theory*, pages 375–389. Springer, 2010.