Efficient Learning in Large-Scale Combinatorial Semi-Bandits

Zheng Wen ZHENGWEN@YAHOO-INC.COM
Yahoo Labs, Sunnyvale, CA

Branislav Kveton KVETON@ADOBE.COM
Adobe Research, San Jose, CA

Azin Ashkan AZIN.ASHKAN@TECHNICOLOR.COM
Technicolor Research, Los Altos, CA

Abstract

A stochastic combinatorial semi-bandit is an online learning problem where at each step a learning agent chooses a subset of ground items subject to combinatorial constraints, and then observes stochastic weights of these items and receives their sum as a payoff. In this paper, we consider efficient learning in large-scale combinatorial semi-bandits with linear generalization, and as a solution, propose two learning algorithms called *Combinatorial Linear Thompson Sampling (*`CombLinTS`*)* and *Combinatorial Linear UCB (*`CombLinUCB`*)*. Both algorithms are computationally efficient as long as the offline version of the combinatorial problem can be solved efficiently. We establish that `CombLinTS` and `CombLinUCB` are also provably statistically efficient under reasonable assumptions, by developing regret bounds that are independent of the problem scale (number of items) and sublinear in time. We also evaluate `CombLinTS` on a variety of problems with thousands of items. Our experiment results demonstrate that `CombLinTS` is scalable, robust to the choice of algorithm parameters, and significantly outperforms the best of our baselines.

1. Introduction

Combinatorial optimization is a mature field (Papadimitriou & Steiglitz, 1998), which has countless practical applications. One of the most studied problems in combinatorial optimization is maximization of a modular function subject to combinatorial constraints. Many important problems, such as minimum spanning tree (MST), shortest path, and maximum-weight bipartite matching, can be viewed as instances of this problem.

In practice, the optimized modular function is often unknown and needs to be learned while repeatedly solving the problem. This class of learning problems was recently formulated as a combinatorial bandit/semi-bandit, depending on the feedback model (Audibert et al., 2014). Since then, many combinatorial bandit/semi-bandit algorithms have been proposed: for the stochastic setting (Gai et al., 2012; Chen et al., 2013; Russo & Van Roy, 2014; Kveton et al., 2015b); for the adversarial setting (Cesa-Bianchi & Lugosi, 2012; Audibert et al., 2014; Neu & Bartók, 2013); and for subclasses of combinatorial problems, matroid and polymatroid bandits (Kveton et al., 2014a;b;c), submodular maximization (Wen et al., 2013; Gabillon et al., 2013), and cascading bandits (Kveton et al., 2015a). Many regret bounds have been established for the combinatorial semi-bandit algorithms. To achieve an $O(\sqrt{n})$ dependence on time n, all of the regret bounds are $\Omega(\sqrt{L})$, where L is the number of items. The dependence on L is intrinsic because the algorithms estimate the weight of each item separately, and matching lower bounds have been established (Section 3.2).

However, in many real-world problems, the number of items L is intractably large. For instance, online advertising in a mainstream commercial website can be viewed as a bipartite matching problem with millions of users and products; routing in the Internet can be formulated as a shortest path problem with billions of edges. Thus, learning algorithms with $\Omega(\sqrt{L})$ regret are impractical in such problems. On the other hand, in many problems, items have features and their weights are similar when the features are similar. In movie recommendation, for instance, the expected ratings of movies that are close in the latent space are also similar. In this work, we show how to leverage this structure to learn to make good decisions more efficiently. More specifically, we assume a *linear generalization* across the items: conditioned on the features of an item, the expected

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37. Copyright 2015 by the author(s).

weight of that item can be estimated using a linear model. Our goal is to develop more efficient learning algorithms for combinatorial semi-bandits with linear generalization.

It is relatively easy to extend many linear bandit algorithms, such as Thompson sampling (Thompson, 1933; Agrawal & Goyal, 2012; Russo & Van Roy, 2013) and Linear UCB (LinUCB, see Auer (2002); Dani et al. (2008); Abbasi-Yadkori et al. (2011)) , to combinatorial semi-bandits with linear generalization. In this paper, we propose two learning algorithms, Combinatorial Linear Thompson Sampling (CombLinTS) and Combinatorial Linear UCB (CombLinUCB), based on Thompson sampling and LinUCB. Both CombLinTS and CombLinUCB are computationally efficient, as long as the offline version of the combinatorial problem can be solved efficiently. The first major contribution of the paper is that we establish a *Bayes regret bound* on CombLinTS and a *regret bound* on CombLinUCB, under reasonable assumptions. Both bounds are L-independent, and sublinear in time. The second major contribution of the paper is that we evaluate CombLinTS on a variety of problems with thousands of items, and two of these problems are based on real-world datasets. We only evaluate CombLinTS since recent literature (Chapelle & Li, 2011) suggests that Thompson sampling algorithms usually outperform UCB-like algorithms in practice. Our experimental results demonstrate that CombLinTS is scalable, robust to the choice of algorithm parameters, and significantly outperforms the best of our baselines. It is worth mentioning that our derived L-independent regret bounds also hold in cases with $L = \infty$. Moreover, as we will discuss in Section 7, our proposed algorithms and their analyses can be easily extended to the *contextual combinatorial semi-bandits*.

Finally, we briefly review some relevant papers. Gabillon et al. (2014) and Yue & Guestrin (2011) focus on submodular maximization with linear generalization. Our paper differs from these two papers in the following two aspects: (1) our paper allows general combinatorial constraints while they do not; (2) our paper focuses on maximization of modular functions while they focus on submodular maximization.

2. Combinatorial Optimization

We focus on a class of combinatorial optimization problems that aim to find a *maximum-weight* set from a given family of sets. Specifically, one such combinatorial optimization problem can be represented as a triple $(E, \mathcal{A}, \mathbf{w})$, where (1) $E = \{1, \dots, L\}$ is a set of L items, called the *ground set*, (2) $\mathcal{A} \subseteq \{A \subseteq E : |A| \leq K\}$ is a family of subsets of E with up to K items, where $K \leq L$, and (3) $\mathbf{w} : E \to \mathbb{R}$ is a *weight function* that assigns each item e in the ground set E a real number. The total weight of all items in a set $A \subseteq E$ is defined as:

$$f(A, \mathbf{w}) = \textstyle\sum_{e \in A} \mathbf{w}(e), \tag{1}$$

which is a linear functional of $\mathbf{w}$ and a modular function in A. A set A^{opt} is a maximum-weight set in $\mathcal{A}$ if:

$$A^{\text{opt}} \in \arg\max_{A \in \mathcal{A}} f(A, \mathbf{w}) = \arg\max_{A \in \mathcal{A}} \textstyle\sum_{e \in A} \mathbf{w}(e). \tag{2}$$

Many classical combinatorial optimization problems, such as finding an MST, bipartite matching, the shortest path problem and the traveling salesman problem (TSP), have form (2). Though some of these problems can be solved efficiently (e.g. bipartite matching), others (e.g. TSP) are known to be NP-hard. However, for many such NP-hard problems, there exist computationally efficient *approximation algorithms* and/or *randomized algorithms* that achieve near-optimal solutions with high probability. Similarly to Chen et al. (2013), in this paper, we allow the agent to use any approximation / randomized algorithm ORACLE to solve (2), and denote its solution as $A^* = \texttt{ORACLE}(E, \mathcal{A}, \mathbf{w})$. To distinguish from a learning algorithm, we refer to a combinatorial optimization algorithm as an *oracle* in this paper.

3. Combinatorial Semi-Bandits with Linear Generalization

Many real-world problems are combinatorial in nature. In recommender systems, for instance, the user is typically recommended K items out of L. The value of an item, such as the expected rating of a movie, is never known perfectly and has to be refined while repeatedly recommending to the pool of the users. Recommender problems are known to be highly structured. In particular, it is well known that the user-item matrix is typically low-rank (Koren et al., 2009) and that the value of an item can be written as a linear combination of its position in the latent space. In this work, we propose a learning algorithm for combinatorial optimization that leverages this structure. In particular, we assume that the weight of each item is a linear function of its features and then we learn the parameters of this model, jointly for all items.

3.1. Combinatorial Semi-Bandits

We formalize our learning problem as a combinatorial semi-bandit. A combinatorial semi-bandit is a triple $(E, \mathcal{A}, P)$, where E and $\mathcal{A}$ are defined in Section 2 and P is a probability distribution over the weights $\mathbf{w} \in \mathbb{R}^L$ of the items in the ground set E. We assume that the weights $\mathbf{w}$ are drawn i.i.d. from P. The mean weight is denoted by $\bar{\mathbf{w}} = \mathbb{E}[\mathbf{w}]$. Each item e is associated with an *arm* and we assume that *multiple arms* can be pulled. A subset of arms $A \subseteq E$ can be pulled if and only if $A \in \mathcal{A}$. The return of pulling arms A is $f(A, \mathbf{w})$ (Equation (1)), the sum of the weights of all items in A. After the arms A are pulled, we observe the individual return of each arm, $\{\mathbf{w}(e) : e \in A\}$. This feedback model is known as *semi-bandit* (Audibert et al., 2014).

We assume that the combinatorial structure $(E, \mathcal{A})$ is known and the distribution P is unknown. We would like to stress that we do not make any structural assumptions on P. The optimal solution to our problem is a maximum-weight set in expectation:

$$A^{\text{opt}} \in \arg\max_{A \in \mathcal{A}} \mathbb{E}_{\mathbf{w}}[f(A, \mathbf{w})] = \arg\max_{A \in \mathcal{A}} \sum_{e \in A} \bar{\mathbf{w}}(e). \quad (3)$$

This objective is equivalent to the one in Equation (2).

Our learning problem is episodic. In each episode t, the learning agent adaptively chooses $A^t \in \mathcal{A}$ based on its observations of the weights up to episode t, gains $f(A^t, \mathbf{w}_t)$, and observes the weights of all chosen items in episode t, $\{(e, \mathbf{w}_t(e)) : e \in A^t\}$. The learning agent interacts with the combinatorial semi-bandit for n times and its goal is to maximize the expected cumulative return in n-episodes $\mathbb{E}[\sum_{t=1}^n f(A^t, \mathbf{w}_t)]$, where the expectation is over (1) the random weights $\mathbf{w}_t$'s, (2) possible randomization in the learning algorithm, and (3) $\bar{\mathbf{w}}$ if it is randomly generated. Notice that the choice of A^t impacts both the return and observations in episode t. So we need to trade off *exploration* and *exploitation*, similarly to other bandit problems.

3.2. Linear Generalization

As we have discussed in Section 1, many provably efficient algorithms have been developed for various combinatorial semi-bandits of form (3) (Chen et al., 2013; Gai et al., 2012; Russo & Van Roy, 2014; Kveton et al., 2014b; 2015b). However, since there are L parameters to learn and these algorithms do not consider *generalization* across items, the derived upper bounds on the expected cumulative regret and/or the Bayes cumulative regret of these algorithms are at least $O(\sqrt{L})$. Furthermore, Audibert et al. (2014) has derived an $\Omega(\sqrt{LKn})$ lower bound on adversarial combinatorial semi-bandits, while Kveton et al. (2014b; 2015b) have derived asymptotic $\Omega(L \log(n)/\Delta)$ gap-dependent lower bounds on stochastic combinatorial semi-bandits, where Δ is an appropriate "gap".

However, in many modern combinatorial semi-bandit problems, L tends to be enormous. Thus, an $O(\sqrt{L})$ regret is unacceptably large in these problems. On the other hand, in many practical problems, there exists a *generalization model* based on which the weight of one item can be (approximately) inferred based on the weights of other items. By exploiting such generalization models, an $o(\sqrt{L})$ or even an L-independent cumulative regret might be achieved.

In this paper, we assume that there is a (possibly imperfect) linear generalization model across the items. Specifically, we assume that the agent knows a *generalization matrix* $\Phi \in \mathbb{R}^{L \times d}$ s.t. $\bar{\mathbf{w}}$ either lies in or is "close" to the subspace $\text{span}[\Phi]$. We use ϕ_e to denote the transpose of the e-th row of Φ, and refer to it as the *feature vector* of item e. Without loss of generality, we assume that $\text{rank}[\Phi] = d$.

Similar to some existing literature (Wen & Van Roy, 2013; Van Roy & Wen, 2014), we distinguish between the *coherent learning* cases, in which $\bar{\mathbf{w}} \in \text{span}[\Phi]$, and the *agnostic learning* cases, in which $\bar{\mathbf{w}} \notin \text{span}[\Phi]$. Like existing literature on linear bandits (Dani et al., 2008; Abbasi-Yadkori et al., 2011), the analysis in this paper focuses on coherent learning cases. However, we would like to emphasize that both of our proposed algorithms, `CombLinTS` and `CombLinUCB`, are also applicable to the agnostic learning cases. As is demonstrated in Section 6, `CombLinTS` performs well in the agnostic learning cases.

Finally, we define $\theta^* = \arg\min_\theta \|\bar{\mathbf{w}} - \Phi\theta\|_2$. Since $\text{rank}[\Phi] = d$, θ^* is uniquely defined. Moreover, in coherent learning cases, we have $\bar{\mathbf{w}} = \Phi\theta^*$.

3.3. Performance Metrics

Let $A^* = \texttt{ORACLE}(E, \mathcal{A}, \bar{\mathbf{w}})$. In this paper, we measure the performance loss of a learning algorithm with respect to A^*. Recall that the learning algorithm chooses A^t in episode t, we define $R_t = f(A^*, \mathbf{w}_t) - f(A^t, \mathbf{w}_t)$ as the *realized regret* in episode t. If the expected weight $\bar{\mathbf{w}}$ is fixed but unknown, we define the *expected cumulative regret* of the learning algorithm in n episodes as

$$R(n) = \sum_{t=1}^n \mathbb{E}\left[R_t | \bar{\mathbf{w}}\right], \quad (4)$$

where the expectation is over random weights and possible randomization in the learning algorithm. If necessary, we denote $R(n)$ as $R(n; \bar{\mathbf{w}})$ to emphasize the dependence on $\bar{\mathbf{w}}$. On the other hand, if $\bar{\mathbf{w}}$ is randomly generated or the agent has a prior belief in $\bar{\mathbf{w}}$, then from Russo & Van Roy (2013), the *Bayes cumulative regret* of the learning algorithm in n episodes is defined as

$$R_{\text{Bayes}}(n) = \mathbb{E}_{\bar{\mathbf{w}}}[R(n; \bar{\mathbf{w}})] = \sum_{t=1}^n \mathbb{E}\left[R_t\right], \quad (5)$$

where the expectation is also over $\bar{\mathbf{w}}$. That is, $R_{\text{Bayes}}(n)$ is a weighted average of $R(n; \bar{\mathbf{w}})$ under the prior on $\bar{\mathbf{w}}$.

4. Learning Algorithms

In this section, we propose two learning algorithms for combinatorial semi-bandits: Combinatorial Linear Thompson Sampling (`CombLinTS`) and Combinatorial Linear UCB (`CombLinUCB`), which are respectively motivated by Thompson sampling and `LinUCB`. Both algorithms maintain a mean vector $\bar{\theta}_t$ and a covariance matrix Σ_t, and use Kalman filtering to update $\bar{\theta}_t$ and Σ_t. They differ in how to choose A^t (i.e. how to explore) in each episode t: `CombLinTS` chooses A^t based on a randomly sampled coefficient vector θ_t, while CombLinUCB chooses A^t based on the *optimism in the face of uncertainty (OFU)* principle.

4.1. Combinatorial Linear Thompson Sampling

The psuedocode of CombLinTS is given in Algorithm 2, where $(E, \mathcal{A})$ is the combinatorial structure, Φ is the generalization matrix, ORACLE is a combinatorial optimization algorithm, and λ and σ are two algorithm parameters controlling the *learning rate*. Specifically, λ is an *inverse-regularization* parameter and smaller λ makes the covariance matrix Σ_t closer to 0. Thus, a too small λ will lead to insufficient exploration and significantly reduce the performance of CombLinTS. On the other hand, σ controls the decrease rate of the covariance matrix Σ_t. In particular, a large σ will lead to slow learning, while a too small σ will make the algorithm quickly converge to some sub-optimal coefficient vector.

In each episode t, Algorithm 2 consists of three steps. First, it randomly samples a coefficient vector θ_t from a Gaussian distribution. Second, it computes A^t based on θ_t and the pre-specified oracle. Finally, it updates the mean vector $\bar{\theta}_{t+1}$ and the covariance matrix Σ_{t+1} based on Kalman filtering (Algorithm 1).

It is worth pointing our that if (1) $\bar{\mathbf{w}} = \Phi\theta^*$, (2) the prior on θ^* is $N(0, \lambda^2 I)$, and (3) $\forall(t,e)$, the noise $\eta_t(e) = \mathbf{w}_t(e) - \bar{\mathbf{w}}(e)$ is independently sampled from $N(0,\sigma^2)$, then in each episode t, the CombLinTS algorithm samples θ_t from the posterior distribution of θ^*. We henceforth refer to a case satisfying condition (1)-(3) as a *coherent Gaussian case*. Obviously, the CombLinTS algorithm can be applied to more general cases, even to cases with no prior and/or agnostic learning cases.

4.2. Combinatorial Linear UCB

The pseudocode of CombLinUCB is given in Algorithm 3, where E, $\mathcal{A}$, Φ and ORACLE are defined the same as in Algorithm 2, and λ, σ, and c are three algorithm parameters. Similarly, λ is an inverse-regularization parameter, σ controls the decrease rate of the covariance matrix, and c controls the *degree of optimism* (exploration). Specifically, if c is too small, the algorithm might converge to some sub-optimal coefficient vector due to insufficient exploration; on the other hand, too large c will lead to excessive exploration and slow learning.

In each episode t, Algorithm 3 also consists of three steps. First, for each $e \in E$, it computes an upper confidence bound (UCB) $\hat{\mathbf{w}}_t(e)$. Second, it computes A^t based on $\hat{\mathbf{w}}_t$ and the pre-specified oracle. Finally, it updates $\bar{\theta}_{t+1}$ and Σ_{t+1} based on Kalman filtering (Algorithm 1).

5. Regret Bounds

In this section, we present a Bayes regret bound on CombLinTS, and a regret bound on CombLinUCB. We will also briefly discuss how these bounds are derived, as well as their tightness. The detailed proofs are left to the appendices. Without loss of generality, throughout this section, we assume that $\|\phi_e\|_2 \le 1, \forall e \in E$.

5.1. Bayes Regret Bound on CombLinTS

We have the following upper bound on $R_{\text{Bayes}}(n)$ when CombLinTS is applied to a coherent Gaussian case with the right parameter.

Theorem 1. *If (1) $\bar{\mathbf{w}} = \Phi\theta^*$, (2) the prior on θ^* is $N(0,\lambda^2 I)$, (3) the noises are i.i.d. sampled from $N(0,\sigma^2)$, and (4) $\lambda \ge \sigma$, then under* CombLinTS *algorithm with parameter (Φ,λ,σ), we have*

$$R_{\text{Bayes}}(n) \le \tilde{O}\left(K\lambda\sqrt{dn\min\{\ln(L), d\}}\right). \tag{6}$$

Notice that condition (1)-(3) ensure it is a coherent Gaussian case, and condition (4) almost always holds[1]. The $\tilde{O}$ notation hides the logarithm factors. We also note that Equation (6) is a minimum of two bounds. The first bound is L-dependent, but it is only $O(\sqrt{\ln(L)})$; on the other hand, the second bound is L-independent, but is $\tilde{O}(d)$ instead of $\tilde{O}(\sqrt{d})$. We would like to emphasize that Theorem 1 holds even if ORACLE is an approximation/randomized algorithm.

We now outline the proof of Theorem 1, which is motivated by Russo & Van Roy (2013) and Dani et al. (2008). Let $\mathcal{H}_t$ denote the "history" (i.e. all the available information) by the start of episode t. Note that from the Bayesian perspective, conditioning on $\mathcal{H}_t$, θ^* and θ_t are i.i.d. drawn from $N(\bar{\theta}_t, \Sigma_t)$ (Russo & Van Roy, 2013). This is because that conditioning on $\mathcal{H}_t$, the posterior belief in θ^* is $N(\bar{\theta}_t,\Sigma_t)$ and based on Algorithm 2, θ_t is independently sampled from $N(\bar{\theta}_t,\Sigma_t)$. Since ORACLE is a fixed combinatorial optimization algorithm (even though it can be independently randomized), and $E, \mathcal{A}, \Phi$ are all fixed, then conditioning on $\mathcal{H}_t$, A^* and A^t are also i.i.d., furthermore, A^* is conditionally independent of θ_t, and A^t is conditionally independent of θ^*.

To simplify the exposition, $\forall\theta \in \mathbb{R}^d$ and $\forall A \subseteq E$, we define

$$g(A,\theta) = \textstyle\sum_{e\in A}\langle\phi_e,\theta\rangle,$$

where $\langle\cdot,\cdot\rangle$ is an alternative notation for inner product. Thus we have $\mathbb{E}[R_t|\mathcal{H}_t] = \mathbb{E}[g(A^*,\theta^*) - g(A^t,\theta^*)|\mathcal{H}_t]$. We also define a UCB function $U_t : 2^E \to \mathbb{R}$ as

$$U_t(A) = \textstyle\sum_{e\in A}\left[\langle\phi_e,\bar{\theta}_t\rangle + c\sqrt{\phi_e^T\Sigma_t\phi_e}\right],$$

[1]Condition (4) is not essential, please refer to Theorem 3 in Appendix A for a Bayes regret bound without condition (4).

Algorithm 1 Compute $\bar{\theta}_{t+1}$ and Σ_{t+1} based on Kalman Filtering

```
**Input:** $\bar{\theta}_t$, $\Sigma_t$, $\sigma$, and feature-observation pairs $\{(\phi_e, \mathbf{w}_t(e)) : e \in A^t\}$
Initialize $\bar{\theta}_{t+1} \leftarrow \bar{\theta}_t$ and $\Sigma_{t+1} \leftarrow \Sigma_t$
for $k = 1, \ldots, |A^t|$ do
    Update $\bar{\theta}_{t+1}$ and $\Sigma_{t+1}$ as follows, where $a_k^t$ is the $k$th element in $A^t$
    $$\bar{\theta}_{t+1} \leftarrow \left[ I - \frac{\Sigma_{t+1}\phi_{a_k^t}\phi_{a_k^t}^T}{\phi_{a_k^t}^T \Sigma_{t+1}\phi_{a_k^t} + \sigma^2} \right] \bar{\theta}_{t+1} + \left[ \frac{\Sigma_{t+1}\phi_{a_k^t}}{\phi_{a_k^t}^T \Sigma_{t+1}\phi_{a_k^t} + \sigma^2} \right] \mathbf{w}_t\left(a_k^t\right) \quad \text{and} \quad \Sigma_{t+1} \leftarrow \Sigma_{t+1} - \frac{\Sigma_{t+1}\phi_{a_k^t}\phi_{a_k^t}^T\Sigma_{t+1}}{\phi_{a_k^t}^T \Sigma_{t+1}\phi_{a_k^t} + \sigma^2},$$
end for
**Output:** $\bar{\theta}_{t+1}$ and $\Sigma_{t+1}$
```

Algorithm 2 Combinatorial Linear Thompson Sampling

```
**Input:** Combinatorial structure $(E, \mathcal{A})$, generalization matrix $\Phi \in \mathbb{R}^{L \times d}$, algorithm parameters $\lambda, \sigma > 0$, oracle ORACLE
Initialize $\Sigma_1 \leftarrow \lambda^2 I \in \mathbb{R}^{d \times d}$ and $\bar{\theta}_1 = 0 \in \mathbb{R}^d$
for all $t = 1, 2, \ldots, n$ do
    Sample $\theta_t \sim N\left(\bar{\theta}_t, \Sigma_t\right)$
    Compute $A^t \leftarrow$ ORACLE$(E, \mathcal{A}, \Phi\theta_t)$
    Choose set $A^t$, and observe $\mathbf{w}_t(e), \forall e \in A^t$
    Compute $\bar{\theta}_{t+1}$ and $\Sigma_{t+1}$ based on Algorithm 1
end for
```

Algorithm 3 Combinatorial Linear UCB

```
**Input:** Combinatorial structure $(E, \mathcal{A})$, generalization matrix $\Phi \in \mathbb{R}^{L \times d}$, algorithm parameters $\lambda, \sigma, c > 0$, oracle ORACLE
Initialize $\Sigma_1 \leftarrow \lambda^2 I \in \mathbb{R}^{d \times d}$ and $\bar{\theta}_1 = 0 \in \mathbb{R}^d$
for all $t = 1, 2, \ldots, n$ do
    Define the UCB weight vector $\hat{\mathbf{w}}_t$ as
    $$\hat{\mathbf{w}}_t(e) = \left\langle \phi_e, \bar{\theta}_t \right\rangle + c\sqrt{\phi_e^T \Sigma_t \phi_e} \quad \forall e \in E$$
    Compute $A^t \leftarrow$ ORACLE$(E, \mathcal{A}, \hat{\mathbf{w}}_t)$
    Choose set $A^t$, and observe $\mathbf{w}_t(e), \forall e \in A^t$
    Compute $\bar{\theta}_{t+1}$ and $\Sigma_{t+1}$ based on Algorithm 1
end for
```

where $c > 0$ is a constant to be specified. Notice that conditioning on $\mathcal{H}_t$, U_t is a deterministic function and A^*, A^t are i.i.d., then $\mathbb{E}[U_t(A^t) - U_t(A^*)|\mathcal{H}_t] = 0$ and

$$\begin{aligned}\mathbb{E}[R_t|\mathcal{H}_t] =& \mathbb{E}[g(A^*, \theta^*) - U_t(A^*)|\mathcal{H}_t] \\ &+ \mathbb{E}\left[U_t(A^t) - g(A^t, \theta^*)\middle|\mathcal{H}_t\right]. \end{aligned} \quad (7)$$

Theorem 1 follows by respectively bounding the two terms on the righthand side of Equation (7). Two key observations are (1) if $c = \tilde{O}\left(\sqrt{\min\{\ln(L), d\}}\right)$, then

$$\mathbb{E}[g(A^*, \theta^*) - U_t(A^*)|\mathcal{H}_t] = O(1),$$

and (2)

$$\mathbb{E}\left[U_t(A^t) - g(A^t, \theta^*)\middle|\mathcal{H}_t\right] = c\mathbb{E}\left[\textstyle\sum_{e \in A^t} \sqrt{\phi_e^T \Sigma_t \phi_e}\middle|\mathcal{H}_t\right],$$

and we have a worst-case bound (see Lemma 4 in Appendix A) on $\sum_{t=1}^n \sum_{e \in A^t} \sqrt{\phi_e^T \Sigma_t \phi_e}$. Please refer to Appendix A for the detailed proof for Theorem 1.

Finally, we briefly discuss the tightness of our bound. Without loss of generality, we assume that $\lambda = 1$. For the special case when $\Phi = I$ (i.e. no generalization), Russo & Van Roy (2014) provides an $O(\sqrt{LK \log(L/K)n})$ upper bound on $R_{\text{Bayes}}(n)$ when Thompson sampling is applied, and Audibert et al. (2014) provides an $\Omega(\sqrt{LKn})$ lower bound[2]. Since $L = d$ when $\Phi = I$, the above results indicate that for general Φ, the best upper bound one can hope is $O(\sqrt{Kdn})$. Hence, our bound is at most $\tilde{O}(\sqrt{K \min\{\ln(L), d\}})$ larger. It is well-known that the $O(\sqrt{d})$ factor is due to linear generalization (Dani et al., 2008; Abbasi-Yadkori et al., 2011), and as is discussed in the appendix (see Remark 1), the extra $O(\sqrt{K})$ factor is also due to linear generalization. They might be intrinsic, but we leave the final word and tightness analysis to future work.

5.2. Regret Bound on CombLinUCB

Under the assumptions that (1) the support of P is a subset of $[0, 1]^L$, (2) the stochastic item weights $\{\mathbf{w}(e)\}_{e \in E}$ are statistically independent under P, and (3) the oracle ORACLE *exactly* solves the offline optimization problem[3], we have the following upper bound on $R(n)$ when CombLinUCB is applied to coherent learning cases:

[2] Audibert et al. (2014) focuses on the adversarial setting but the lower bound is stochastic. So it is a reasonable lower bound to compare with.

[3] If ORACLE is an approximation algorithm, then a variant of Theorem 2 can be proved (see Appendix D).

Theorem 2. *For any $\lambda, \sigma > 0$, any $\delta \in (0,1)$, and any c satisfying*

$$c \geq \frac{1}{\sigma}\sqrt{d \ln\left(1 + \frac{nK\lambda^2}{d\sigma^2}\right) + 2\ln\left(\frac{1}{\delta}\right)} + \frac{\|\theta^*\|_2}{\lambda}, \quad (8)$$

if $\bar{\mathbf{w}} = \Phi\theta^*$ and the above three assumptions hold, then under `CombLinUCB` *algorithm with parameter $(\Phi, \lambda, \sigma, c)$, we have*

$$R(n) \leq 2cK\lambda\sqrt{\frac{dn\ln\left(1+\frac{nK\lambda^2}{d\sigma^2}\right)}{\ln\left(1+\frac{\lambda^2}{\sigma^2}\right)}} + nK\delta.$$

Generally speaking, the proof for Theorem 2 proceeds as follows. We first construct a confidence set G of θ^* based on the "self normalized bound" developed in Abbasi-Yadkori et al. (2011). Then we decompose the regret over the high-probability "good" event G and the low-probability "bad" event $\bar{G}$, where $\bar{G}$ is the complement of G. Finally, we bound the term associated with the event G based on the same worst-case bound on $\sum_{t=1}^{n}\sum_{e\in A^t}\sqrt{\phi_e^T\Sigma_t\phi_e}$ used in the analysis for `CombLinTS` (see Lemma 4 in Appendix A), and bound the term associated with the event $\bar{G}$ based on a naive bound. Please refer to Appendix B for the detailed proof of Theorem 2.

Notice that if we choose $\lambda = \sigma = 1$, $\delta = 1/(nK)$, and c as the lower bound specified in Inequality (8), then the regret bound derived in Theorem 2 is also $\tilde{O}(Kd\sqrt{n})$. Compared with the lower bound derived in Audibert et al. (2014), this bound is at most $\tilde{O}(\sqrt{Kd})$ larger. Similarly, the extra $O(\sqrt{K})$ and $O(\sqrt{d})$ factors are also due to linear generalization.

Finally, we would like to clarify that the assumption that the support of P is bounded is not essential. By slightly modifying the analysis, we can achieve a similar high-probability bound on the *realized cumulative regret* as long as P is *sub-Gaussian*. We also want to point out that the L-independent bounds derived in both Theorem 1 and 2 will still hold even if $L = \infty$.

6. Experiments

In this section, we evaluate `CombLinTS` on three problems. The first problem is synthetic, but the last two problems are constructed based on real-world datasets. As we have discussed in Section 1, we only evaluate `CombLinTS` since in practice Thompson sampling algorithms usually outperform the UCB-like algorithms. Our experiment results in the synthetic problem demonstrate that `CombLinTS` is both scalable and robust to the choice of algorithm parameters. They also suggest the Bayes regret bound derived in Theorem 1 is likely to be tight. On the other hand, our experiment results in the last two problems show the *value of linear generalization* in real-world settings: with domain-specific but imperfect linear generalization (i.e. agnostic learning), `CombLinTS` can significantly outperform state-of-the-art learning algorithms that do not exploit linear generalization, which serve as baselines in these two problems.

In all three problems, the oracle `ORACLE` exactly solves the offline combinatorial optimization problem. Moreover, in the two real-world problems, we demonstrate the experiment results using a new performance metric, the *expected per-step return* in n episodes, which is defined as

$$\frac{1}{n}\mathbb{E}_{\mathbf{w}_1,\ldots,\mathbf{w}_n}\left[\sum_{t=1}^{n} f(A^t, \mathbf{w}_t)|\bar{\mathbf{w}}\right]. \quad (9)$$

Obviously, it is the expected cumulative return in n episodes divided by n. We demonstrate experiment results using expected cumulative return rather than $R(n)$ since it is more illustrative.

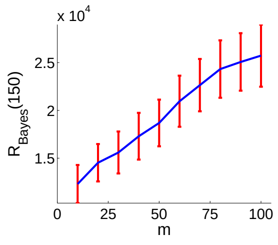
6.1. Longest Path

We first evaluate `CombLinTS` on a synthetic problem. Specifically, we experiment with a stochastic longest path problem on an $(m+1)\times(m+1)$ square grid[4]. The items in the ground set E are the edges in the grid, $L = 2m(m+1)$ in total. The feasible set $\mathcal{A}$ are all paths in the grid from the upper left corner to the bottom right corner that follow the directions of the edges. The length of these paths is $K = 2m$. In this problem, we focus on coherent Gaussian cases and randomly sample the linear generalization matrix $\Phi \in \mathbb{R}^{L\times d}$ to weaken the dependence on a particular choice of Φ.

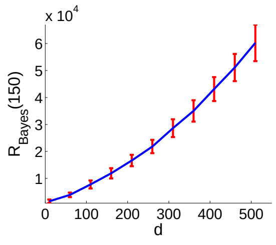
Our experiments are parameterized by a sextuple $(m, d, \lambda_{\text{true}}, \sigma_{\text{true}}, \lambda, \sigma)$, where m, d, λ, and σ are defined before and λ_{true} and σ_{true} are respectively the true standard deviations of θ^* and the observation noises. In each round of simulation, we first construct a problem instance as follows: (1) generate Φ by sampling each component of Φ i.i.d. from $N(0,1)$; (2) sample θ^* independently from $N(0, \lambda_{\text{true}}^2 I)$ and set $\bar{\mathbf{w}} = \Phi\theta^*$; and (3) $\forall(t,e)$, the observation noise $\eta_t(e) = \mathbf{w}_t(e) - \bar{\mathbf{w}}(e)$ is i.i.d. sampled from $N(0, \sigma_{\text{true}}^2)$. Then we apply `CombLinTS` with parameter (λ, σ) to the constructed instance for n episodes. Notice that in general $(\lambda, \sigma) \neq (\lambda_{\text{true}}, \sigma_{\text{true}})$. We average the experiment results over 200 simulations to estimate the Bayes cumulative regret $R_{\text{Bayes}}(n)$.

We start with a "default case" with $m = 30$, $d = 200$, $\lambda_{\text{true}} = \lambda = 10$ and $\sigma_{\text{true}} = \sigma = 1$. Notice in this case $L = 1860$ and $|\mathcal{A}| \approx 1.18 \times 10^{17}$. We choose $n = 150$ since in the default case, the Bayes per-episode regret of `CombLinTS` vanishes far before period 150. In the default case $R_{\text{Bayes}}(150) \approx 1.56 \times 10^4$. In the experiments, we

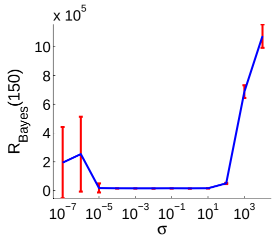
[4]That is, each side has m edges and $m+1$ nodes. Notice that the longest path problem and the shortest path problem are mathematically equivalent.



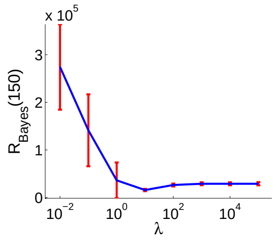
(a) R_{Bayes} vs. m



(b) R_{Bayes} vs. d



(c) R_{Bayes} vs. σ



(d) R_{Bayes} vs. λ

Figure 1. Experiment results in the longest path problem

vary one and only one parameter while keeping all the other parameters fixed to their "default values" specified above to demonstrate the *scalability* and *robustness* of CombLinTS.

First, we study how the Bayes cumulative regret of CombLinTS scales with the size of the problem by varying $m = 10, 20, \dots, 100$, and show the result in Figure 1(a). The experiment results show that $R_{\text{Bayes}}(150)$ roughly increases linearly with m, which indicates that CombLinTS is scalable with respect to the problem size m. We also experiment with $m = 250$, in this case we have $L \approx 125k$, $|\mathcal{A}| \approx 1.17 \times 10^{149}$, and $R_{\text{Bayes}}(150) \approx 6.56 \times 10^4$, which is only 4.2 times of $R_{\text{Bayes}}(150)$ in the default case. It is worth mentioning that this result also suggests that the Bayes regret bound derived in Theorem 1 is (almost) tight in this problem[5]. To see it, notice that $K = 2m$ and $L = O(m^2)$, and hence the Bayes regret bound derived in Theorem 1 is $\tilde{O}(m)$.

Second, we study how the Bayes cumulative regret of CombLinTS scales with d, the dimension of the feature vectors, by varying $d = 10, 60, 110, \dots, 510$, and demonstrate the result in Figure 1(b). The experiment results indicate that $R_{\text{Bayes}}(150)$ also roughly increases linearly with d, and hence CombLinTS is also scalable with the feature dimension d. This result also suggests that the $\tilde{O}(\sqrt{d})$ bound in Theorem 1 is (almost) tight[5].

[5]Recall that Theorem 1 requires $\max_{e \in E} \|\phi_e\|_2 \leq 1$. It can be easily extended to cases with $\max_{e \in E} \|\phi_e\|_2 \leq M$ by scaling the Bayes regret bound by M. However, in this problem ϕ_e is not bounded since it is sampled from a Gaussian distribution. We believe that Theorem 1 can be extended to this case by exploiting the properties of Gaussian distribution. Roughly speaking, in this problem, with high probability, $\|\phi_e\|_2 = O(\sqrt{d})$.

Finally, we study the robustness of CombLinTS with respect to the algorithm parameters σ and λ. In Figure 1(c), we vary $\sigma = 10^{-7}, 10^{-6}, \dots, 10^4$ and in Figure 1(d), we vary $\lambda = 10^{-2}, 10^{-1}, \dots, 10^5$. We would like to emphasize again that we only vary the algorithm parameters and fix $\sigma_{\text{true}} = 1$ and $\lambda_{\text{true}} = 10$. The experiment results show that CombLinTS is robust to the choice of algorithm parameters and performs well for a wide range of σ and λ. However, too small or too large σ, or too small λ, can significantly reduce the performance of CombLinTS, as we have discussed in Section 4.1.

6.2. Online Advertising

In the second experiment, we evaluate CombLinTS on an advertising problem. Our objective is to identify 100 people that are most likely to accept an advertisement offer, subject to the targeting constraint that exactly half of them are females. Specifically, the ground set E includes 33k representative people from Adult dataset (Asuncion & Newman, 2007), which was collected in the 1994 US census. A feasible solution A is any subset of E with $|A| = 100$ and satisfying the targeting constraint mentioned above. We assume that person e accepts an advertisement offer with probability

$$\bar{\mathbf{w}}(e) = \begin{cases} 0.15 & \text{income is at least 50k} \\ 0.05 & \text{otherwise,} \end{cases}$$

and people accept offers independently of each other. The features in the generalization matrix Φ are the age, which is binned into 7 groups; gender; whether the person works more than 40 hours per week; and the length of education in years. All these features can be constructed based on the Adult dataset.

CombLinTS is compared to three baselines. The first baseline is the optimal solution A^{opt}. The second baseline is CombUCB1 (Kveton et al., 2015b). This algorithm estimates the probability that person e accepts the offer $\bar{\mathbf{w}}(e)$ independently of the other probabilities. The third baseline is CombLinTS without linear generalization, which we simply refer to as CombTS. As in CombUCB1, this algorithm estimates the probability that person e accepts the offer $\bar{\mathbf{w}}(e)$ independently of the other probabilities. The posterior of $\bar{\mathbf{w}}(e)$ is modeled as a beta distribution.

Our experiment results are reported in Figure 2. We observe two major trends. First, CombLinTS learns extremely quickly. In particular, its per-step return at episode 100 is 70% of the optimum, and its per-step return at episode 1k is 80% of the optimum. These results are remarkable since the linear generalization is imperfect in this problem. Second, both CombUCB1 and CombTS perform poorly due to insufficient observations with respect to the model complexity. Specifically, in 1k episodes, the people in E are observed 100k times, which implies that each person is ob-

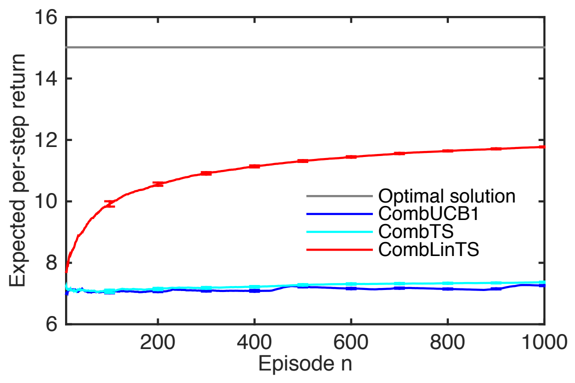


Figure 2. Experiment results in the online advertising problem

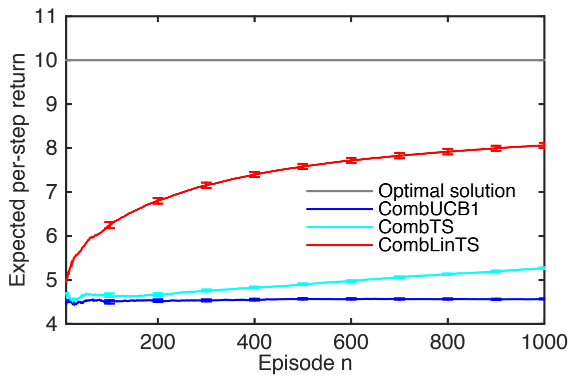


Figure 3. Experiment results in the artist recommendation problem

served only 3 times on average. This is not enough to discriminate the people who are likely to accept the advertisement offer from those that are not.

6.3. Artist Recommendation

In the last experiment, we evaluate `CombLinTS` on a problem of recommending $K = 10$ music artists that are most likely to be chosen by an average user of a music recommendation website. Specifically, the ground set E include artists from the *Last.fm* music recommendation dataset (Cantador et al., 2011). The dataset contains tagging and music artist listening information from a set of 2k users from Last.fm online music system[6]. The tagging part includes the tag assignments of all artists provided by the users. For each user, the artists to whom she listened and the number of listening events are also available in the dataset.

We choose E as the set of artists that were listened by at least two users and had at least one tag assignment among the top 20 most popular tags, and $|E| \approx 6$k. For each artist e, we construct its feature vector $\phi_e \in [0,1]^{20}$ by setting its jth component as the fraction of users who assigned tag j to this artist. We assume that each artist e is chosen by an average user with probability $\bar{\mathbf{w}}(e) = \frac{1}{|U_e|}\sum_{u \in U_e} \bar{\mathbf{w}}_u(e)$, where U_e is the set of users that listened to artist e, and $\bar{\mathbf{w}}_u(e)$ is the probability that user u likes artist e. We estimate $\bar{\mathbf{w}}_u(e)$ based on a Naïve Bayes classifier with respect to the number of person/artist listening events.

Like Section 6.2, we also compare `CombLinTS` to three baselines: the optimal solution A^{opt}, the `CombUCB1` algorithm and the `CombTS` algorithm. Our experiment results are reported in Figure 3. Similarly as Figure 2, the expected per-step return of `CombLinTS` approaches that of A^{opt} much faster than `CombUCB1` and `CombTS`. Moreover, both `CombUCB1` and `CombTS` perform poorly due to the insufficient observations with respect to the model complexity: In 1k episodes, each artist is observed less than 2 times on average, which is not enough to discriminate most popular artists from less popular artists.

7. Conclusion

We have proposed two learning algorithms, `CombLinTS` and `CombLinUCB`, for stochastic combinatorial semi-bandits with linear generalization. The main contribution of this work is two-fold: First, we have established L-independent regret bounds for these two algorithms under reasonable assumptions, where L is the number of items. Second, we have also evaluated `CombLinTS` on a variety of problems. The experiment results in the first problem show that `CombLinTS` is scalable and robust, and the experiment results in the other two problems demonstrate the value of exploiting linear generalization in real-world settings.

It is worth mentioning that our results can be easily extended to the *contextual combinatorial semi-bandits* with linear generalization. In a contextual combinatorial semi-bandit, the probability distribution P (and hence the expected weight $\bar{\mathbf{w}}$) also depends on a context x, which either follows an exogenous stochastic process or is adaptively chosen by an adversary. Assume that each state-item pair (x, e) is associated with a feature vector $\phi_{x,e}$, then similar to Agrawal & Goyal (2013), both `CombLinTS` and `CombLinUCB`, as well as their analyses, can be generalized to the contextual combinatorial semi-bandits.

We leave open several questions of interest. One interesting open question is how to derive regret bounds for `CombLinTS` and `CombLinUCB` in the agnostic learning cases. Another interesting open question is how to extend the results to combinatorial semi-bandits with nonlinear generalization. We believe that our results can be extended to combinatorial semi-bandits with *generalized linear generalization*[7], but leave it to future work.

[6]http://www.lastfm.com

[7]That is, $\bar{\mathbf{w}}(e) = h(\phi_e^T \theta^*)$, where $h : \mathbb{R} \to \mathbb{R}$ is a strictly monotone function.

References

Abbasi-Yadkori, Yasin, Pál, Dávid, and Szepesvári, Csaba. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems 24*, pp. 2312–2320, 2011.

Agrawal, Shipra and Goyal, Navin. Analysis of thompson sampling for the multi-armed bandit problem. In *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, pp. 39.1–39.26, 2012.

Agrawal, Shipra and Goyal, Navin. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pp. 127–135, 2013.

Asuncion, A. and Newman, D.J. UCI machine learning repository, 2007. URL `http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html`.

Audibert, Jean-Yves, Bubeck, Sebastien, and Lugosi, Gabor. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1):31–45, 2014.

Auer, Peter. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2002.

Cantador, Iván, Brusilovsky, Peter, and Kuflik, Tsvi. Second workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011. ACM, 2011.

Cesa-Bianchi, Nicolò and Lugosi, Gábor. Combinatorial bandits. *Journal of Computer and System Sciences*, 78 (5):1404–1422, 2012.

Chapelle, Olivier and Li, Lihong. An empirical evaluation of Thompson sampling. In *Neural Information Processing Systems*, pp. 2249–2257, 2011.

Chen, Wei, Wang, Yajun, and Yuan, Yang. Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 151–159, 2013.

Dani, Varsha, Hayes, Thomas, and Kakade, Sham. Stochastic linear optimization under bandit feedback. In *Proceedings of the 21st Annual Conference on Learning Theory*, pp. 355–366, 2008.

Gabillon, Victor, Kveton, Branislav, Wen, Zheng, Eriksson, Brian, and Muthukrishnan, S. Adaptive submodular maximization in bandit setting. In *Advances in Neural Information Processing Systems 26*, pp. 2697–2705, 2013.

Gabillon, Victor, Kveton, Branislav, Wen, Zheng, Eriksson, Brian, and Muthukrishnan, S. Large-scale optimistic adaptive submodularity. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.

Gai, Yi, Krishnamachari, Bhaskar, and Jain, Rahul. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking*, 20(5):1466–1478, 2012.

Koren, Yehuda, Bell, Robert, and Volinsky, Chris. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

Kveton, Branislav, Wen, Zheng, Ashkan, Azin, and Eydgahi, Hoda. Matroid bandits: Practical large-scale combinatorial bandits. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014a.

Kveton, Branislav, Wen, Zheng, Ashkan, Azin, Eydgahi, Hoda, and Eriksson, Brian. Matroid bandits: Fast combinatorial optimization with learning. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, pp. 420–429, 2014b.

Kveton, Branislav, Wen, Zheng, Ashkan, Azin, Eydgahi, Hoda, and Valko, Michal. Learning to act greedily: Polymatroid semi-bandits. *CoRR*, abs/1405.7752, 2014c.

Kveton, Branislav, Szepesvari, Csaba, Wen, Zheng, and Ashkan, Azin. Cascading bandits: Learning to rank in the cascade model. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015a.

Kveton, Branislav, Wen, Zheng, Ashkan, Azin, and Szepesvari, Csaba. Tight regret bounds for stochastic combinatorial semi-bandits. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015b.

Neu, Gergely and Bartók, Gábor. An efficient algorithm for learning with semi-bandit feedback. In Jain, Sanjay, Munos, Rémi, Stephan, Frank, and Zeugmann, Thomas (eds.), *Algorithmic Learning Theory*, volume 8139 of *Lecture Notes in Computer Science*, pp. 234–248. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-40934-9.

Papadimitriou, Christos and Steiglitz, Kenneth. *Combinatorial Optimization*. Dover Publications, Mineola, NY, 1998.

Russo, Daniel and Van Roy, Benjamin. Learning to optimize via posterior sampling. *CoRR*, abs/1301.2609, 2013.

Russo, Daniel and Van Roy, Benjamin. An information-theoretic analysis of thompson sampling. *CoRR*, abs/1403.5341, 2014.

Thompson, W.R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Van Roy, Benjamin and Wen, Zheng. Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635*, 2014.

Wen, Zheng and Van Roy, Benjamin. Efficient exploration and value function generalization in deterministic systems. In *Advances in Neural Information Processing Systems 26*, pp. 3021–3029, 2013.

Wen, Zheng, Kveton, Branislav, Eriksson, Brian, and Bhamidipati, Sandilya. Sequential Bayesian search. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 977–983, 2013.

Yue, Yisong and Guestrin, Carlos. Linear submodular bandits and their application to diversified retrieval. In *Advances in Neural Information Processing Systems 24*, pp. 2483–2491, 2011.