Combinatorial Cascading Bandits

Authored by:

Branislav Kveton
Csaba Szepesvari
Zheng Wen
Azin Ashkan

Abstract

We propose combinatorial cascading bandits, a class of partial monitoring problems where at each step a learning agent chooses a tuple of ground items subject to constraints and receives a reward if and only if the weights of all chosen items are one. The weights of the items are binary, stochastic, and drawn independently of each other. The agent observes the index of the first chosen item whose weight is zero. This observation model arises in network routing, for instance, where the learning agent may only observe the first link in the routing path which is down, and blocks the path. We propose a UCB-like algorithm for solving our problems, CombCascade; and prove gap-dependent and gap-free upper bounds on its n-step regret. Our proofs build on recent work in stochastic combinatorial semi-bandits but also address two novel challenges of our setting, a non-linear reward function and partial observability. We evaluate CombCascade on two real-world problems and show that it performs well even when our modeling assumptions are violated. We also demonstrate that our setting requires a new learning algorithm.

1 Paper Body

Combinatorial optimization [16] has many real-world applications. In this work, we study a class of combinatorial optimization problems with a binary objective function that returns one if and only if the weights of all chosen items are one. The weights of the items are binary, stochastic, and drawn independently of each other. Many popular optimization problems can be formulated in our setting. Network routing is a problem of choosing a routing path in a computer network that maximizes the probability that all links in the chosen path are up. Recommendation is a problem of choosing a list of items that minimizes the probability that none of the recommended items are attractive. Both of these problems are closely related and can be solved using similar techniques (Section 2.3). Combinatorial cascading bandits are a novel framework for online

learning of the aforementioned problems where the distribution over the weights of items is unknown. Our goal is to maximize the expected cumulative reward of a learning agent in n steps. Our learning problem is challenging for two main reasons. First, the reward function is non-linear in the weights of chosen items. Second, we only observe the index of the first chosen item with a zero weight. This kind of feedback arises frequently in network routing, for instance, where the learning agent may only observe the first link in the routing path which is down, and blocks the path. This feedback model was recently proposed in the so-called cascading bandits [10]. The main difference in our work is that the feasible set can be arbitrary. The feasible set in cascading bandits is a uniform matroid. 1

Stochastic online learning with combinatorial actions has been previously studied with semi-bandit feedback and a linear reward function [8, 11, 12], and its monotone transformation [5]. Established algorithms for multi-armed bandits, such as UCB1 [3], KL-UCB [9], and Thompson sampling [18, 2]; can be usually easily adapted to stochastic combinatorial semi-bandits. However, it is non-trivial to show that the algorithms are statistically efficient, in the sense that their regret matches some lower bound. Kveton et al. [12] recently showed this for CombUCB1, a form of UCB1. Our analysis builds on this recent advance but also addresses two novel challenges of our problem, a non-linear reward function and partial observability. These challenges cannot be addressed straightforwardly based on Kveton et al. [12, 10]. We make multiple contributions. In Section 2, we define the online learning problem of combinatorial cascading bandits and propose CombCascade, a variant of UCB1, for solving it. CombCascade is computationally efficient on any feasible set where a linear function can be optimized efficiently. A minor-looking improvement to the UCB1 upper confidence bound, which exploits the fact that the expected weights of items are bounded by one, is necessary in our analysis. In Section 3, we derive gap-dependent and gap-free upper bounds on the regret of CombCascade, and discuss the tightness of these bounds. In Section 4, we evaluate CombCascade on two practical problems and show that the algorithm performs well even when our modeling assumptions are violated. We also show that CombUCB1 [8, 12] cannot solve some instances of our problem, which highlights the need for a new learning algorithm.

2

Combinatorial Cascading Bandits

This section introduces our learning problem, its applications, and also our proposed algorithm. We discuss the computational complexity of the algorithm and then introduce the co-called disjunctive variant of our problem. We denote random variables by boldface letters. The cardinality of set A is —A— and we assume that min ; = +1. The binary and operation is denoted by , and the binary or is _. 2.1

Setting

We model our online learning problem as a combinatorial cascading bandit. A combinatorial cascading bandit is a tuple B = (E, P, ?), where E = {1, . . . , L} is a finite set of L ground items, P E is a probability distribution over a

binary hypercube {0, 1} , ? ? ?? (E), and: ?? (E) = {(a1 , . . . , ak) : k

1, a1 , . . . , ak 2 E, ai 6= aj for any i 6= j}

is the set of all tuples of distinct items from E. We refer to ? as the feasible set and to A 2 ? as a feasible solution. We abuse our notation and also treat A as the set of items in solution A. Without loss of generality, we assume that the feasible set ? covers the ground set, E = [?. E

Let (wt)nt=1 be an i.i.d. sequence of n weights drawn from distribution P , where wt 2 {0, 1} . At time t, the learning agent chooses solution At = (at1 , . . . , at—At —) 2 ? based on its past observations and then receives a binary reward:

rt = min wt (e) = wt (e) e2At

e2At

as a response to this choice. The reward is one if and only if the weights of all items in At are one. The key step in our solution and its analysis is that the reward can be expressed as rt = f (At , wt), where f : ? ? [0, 1]E ! [0, 1] is a reward function, which is defined as: Y f (A, w) = w(e) , A 2 ? , w 2 [0, 1]E . e2A

At the end of time t, the agent observes the index of the first item in At whose weight is zero, and +1 if such an item does not exist. We denote this feedback by Ot and define it as: Ot = min 1 ? k ? —At — : wt (atk) = 0 .

Note that Ot fully determines the weights of the first min {Ot , —At —} items in At . In particular: wt (atk) = 1{k ¡ Ot }

k = 1, . . . , min {Ot , —At —} . 2

(1)

Accordingly, we say that item e is observed at time t if e = atk for some 1 ? k ? min {Ot , —At —}. Note that the order of items in At affects the feedback Ot but not the reward rt . This differentiates our problem from combinatorial semi-bandits. The goal of our learning agent is to maximize its expected cumulative reward. This is equivalent to minimizing the expected cumulative regret in n steps: Pn R(n) = E [t=1 R(At , wt)] , where R(At , wt) = f (A? , wt) f (At , wt) is the instantaneous stochastic regret of the agent at time t and A? = arg max A2? E [f (A, w)] is the optimal solution in hindsight of knowing P . For simplicity of exposition, we assume that A? , as a set, is unique.

A major simplifying assumption, which simplifies our optimization problem and its learning, is that the distribution P is factored: Q P (w) = e2E Pe (w(e)) , (2) where Pe is a Bernoulli distribution with mean w(e). ? We borrow this assumption from the work of Kveton et al. [10] and it is critical to our results. We would face computational difficulties without it. Under this assumption, the expected reward of solution A 2 ?, the probability that the weight of each item in A is one, can be written as E [f (A, w)] = f (A, w), ? and depends only on the expected weights of individual items in A. It follows that: A? = arg max A2? f (A, w) ? . In Section 4, we experiment with two problems that violate our independence assumption. We also discuss implications of this violation. Several interesting online learning problems can be formulated as combinatorial cascading bandits. Consider the problem of learning routing paths in Simple Mail Transfer Protocol (SMTP) that maximize the probability of e-mail delivery.

The ground set in this problem are all links in the network and the feasible set are all routing paths. At time t, the learning agent chooses routing path At and observes if the e-mail is delivered. If the e-mail is not delivered, the agent observes the first link in the routing path which is down. This kind of information is available in SMTP. The weight of item e at time t is an indicator of link e being up at time t. The independence assumption in (2) requires that all links fail independently. This assumption is common in the existing network routing models [6]. We return to the problem of network routing in Section 4.2.
2.2

CombCascade Algorithm

Our proposed algorithm, CombCascade, is described in Algorithm 1. This algorithm belongs to the family of UCB algorithms. At time t, CombCascade operates in three stages. First, it computes the upper confidence bounds (UCBs) Ut 2 [0, 1]E on the expected weights of all items in E. The UCB of item e at time t is defined as: ? Tt Ut (e) = min w

1 (e)

(e) + ct

1,Tt

1 (e)

,1 ,

(3)

? s (e) is the average of s observed where w weights of item e, Tt (e) is the number of times that item e p ? s (e) is observed in t steps, and ct,s = (1.5 log t)/s is the radius of a confidence interval around w ? s (e) ct,s , w ? s (e) + ct,s] holds with a high probability. After the after t steps such that w(e) ? 2 [w UCBs are computed, CombCascade chooses the optimal solution with respect to these UCBs: At = arg max A2? f (A, Ut) . Finally, CombCascade observes Ot and updates its estimates of the expected weights based on the weights of the observed items in (1), for all items atk such that k ? Ot .

For simplicity of exposition, we assume that CombCascade is initialized by one sample w0 ? P . If w0 is unavailable, we can formulate the problem of obtaining w0 as an optimization problem on ? with a linear objective [12]. The initialization procedure of Kveton et al. [12] tracks observed items and adaptively chooses solutions with the maximum number of unobserved items. This approach is computationally efficient on any feasible set ? where a linear function can be optimized efficiently. CombCascade has two attractive P properties. First, the algorithm is computationally efficient, in the sense that At = arg max A2? e2A log(Ut (e)) is the problem of maximizing a linear function on 3

Algorithm 1 CombCascade for combinatorial cascading bandits. // Initialization Observe w0 ? P 8e 2 E : T0 (e) 1 ? 1 (e) 8e 2 E : w w0 (e) for all t = 1, . . . , n do // Compute UCBs ? Tt 8e 2 E : Ut (e) = min w

1 (e)

(e) + ct

1,Tt

1 (e)

,1

// Solve the optimization problem and get feedback At arg max A2? f (A, Ut) Observe Ot 2 {1, . . . , —At — , +1}

// Update statistics 8e 2 E : Tt (e) Tt 1 (e) for all k = 1, . . . , min {Ot , —At —} do e atk Tt (e) Tt (e) + 1 ? Tt 1 (e) (e) + 1{k ¡ Ot } Tt 1 (e)w ? Tt (e) (e) w Tt (e) ?. This problem can be solved efficiently for various feasible sets ?, such as matroids, matchings, and paths. Second, CombCascade is sample efficient because the UCB of solution A, f (A, Ut), is a product of the UCBs of all items in A, which are estimated separately. The regret of CombCascade does not depend on —?— and is polynomial in all other quantities of interest. 2.3

Disjunctive Objective

Our reward model is conjuctive, the reward is one if and only if the weights W of all chosen items are one. A natural alternative is a disjunctive model rt = maxe2At wt (e) = e2At wt (e), the reward is one if the weight of any item in At is one. This model arises in recommender systems, where the recommender is rewarded when the user is satisfied with any recommended item. The feedback Ot is the index of the first item in At whose weight is one, as in cascading bandits [10]. Q Let f_ : ? ? [0, 1]E ! [0, 1] be a reward function, which is defined as f_ (A, w) = 1 e2A (1 w(e)). Then under the independence assumption in (2), E [f_ (A, w)] = f_ (A, w) ? and: Y A? = arg max f_ (A, w) ? = arg min (1 w(e)) ? = arg min f (A, 1 w) ? . A2?

A2?

A2?

e2A

Therefore, A can be learned by a variant of CombCascade where the observations are 1 each UCB Ut (e) is substituted with a lower confidence bound (LCB) on 1 w(e): ? ?

? Tt w

Lt (e) = max 1

1 (e)

(e)

ct

1,Tt

1 (e)

wt and

,0 .

Let R(At , wt) = f (At , 1 wt) f (A , 1 wt) be the instantaneous stochastic regret at time t. Then we can bound the regret of CombCascade as in Theorems 1 and 2. The only difference is that ? e,min and f are redefined as: w) ? f (A? , 1 w) ? , f ? = f (A? , 1 w) ? . e,min = minA2?:e2A, A ¿0 f (A, 1 ?

3

Analysis

We prove gap-dependent and gap-free upper bounds on the regret of CombCascade in Section 3.1. We discuss these bounds in Section 3.2. 3.1

Upper Bounds

We define the suboptimality gap of solution A = (a1 , . . . , a—A—) as A = f (A? , w) ? f (A, w) ? and Q—A— 1 the probability that all items in A are observed as pA = k=1 w(a ? k). For convenience, we define 4

? = E A? be the set of suboptimal items, the items shorthands f ? = f (A? , w) ? and p? = pA? . Let E ? ? is: that are not in A . Then the minimum gap associated with suboptimal item e 2 E e,min

= f (A? , w) ?

maxA2?:e2A,

A ¿0

f (A, w) ? .

Let K = max {—A— : A 2 ?} be the maximum number of items in any solution and f ? ¿ 0. Then the regret of CombCascade is bounded as follows. K X 4272 ?2 Theorem 1. The regret of CombCascade is bounded as R(n) ? ? log n + L. f 3 e,min ? e2E

Proof. The proof is in Appendix A. The main idea is to reduce our analysis to that of CombUCB1 in stochastic combinatorial semi-bandits [12]. This reduction is challenging for two reasons. First, our reward function is non-linear in the weights of chosen items. Second, we only observe some of the chosen items. Our analysis can be trivially reduced to semi-bandits by conditioning on the event of observing all items. In particular, let Ht = (A1 , O1 , . . . , At 1 , Ot 1 , At) be the history of CombCascade up to choosing solution At , the first t 1 observations and t actions. Then we can express the expected regret at time t conditioned on Ht as: E [R(At , wt) — Ht] = E [

At (1/pAt)1{

At

¿ 0, Ot

—At —} — Ht]

and analyze our problem under the assumption that all items in At are observed. This reduction is problematic because the probability pAt can be low, and as a result we get a loose regret bound. We address this issue by formalizing the following insight into our problem. When f (A, w) ? ? f ?, ? CombCascade can distinguish A from A without learning the expected weights of all items in A. In particular, CombCascade acts implicitly on the prefixes of suboptimal solutions, and we choose them in our analysis such that the probability of observing all items in the prefixes is ?close? to f ? , and the gaps are ?close? to those of the original solutions. Lemma 1. Let A = (a1 , . . . , a—A—) 2 ? be a feasible solution and Bk = (a1 , . . . , ak) be a prefix of 1 1 ? k ? —A— items of A. Then k can be set such that Bk 2 A and pBk 2f . Then we count the number of times that the prefixes can be chosen instead of A? when all items in the prefixes are observed. The last remaining issue is that f (A, Ut) is non-linear in the confidence radii of the items in A. Therefore, we bound it from above based on the following lemma. Lemma 2. Let 0 ? p1 , . . . , pK ? 1 and u1 , . . . , uK 0. Then: QK QK PK k=1 min {pk + uk , 1} ? k=1 pk + k=1 uk . This bound is tight when p1 , . . . , pK = 1 and u1 , . . . , uK = 0.

The rest of our analysis is along the lines of Theorem 5 in Kveton et al. [12]. We can achieve linear dependency on K, in exchange for a multiplicative factor of 534 in our upper bound. We also prove the following gap-free bound. Theorem 2. The regret of CombCascade is bounded as R(n) ? 131

s

KLn log n ? 2 + L. f? 3

Proof. The proof is in Appendix B. The key idea is to decompose the regret of CombCascade into two parts, where the gaps At are at most ? and larger than ?. We analyze each part separately and then set ? to get the desired result. 3.2

Discussion

In Section 3.1, we prove two upper bounds on the n-step regret of CombCascade: p Theorem 1: O(KL(1/f ?)(1/) log n) , Theorem 2: O(KL(1/f ?)n log n) ,

where = mine2E? e,min . These bounds do not depend on the total number of feasible solutions —?— and are polynomial in any other quantity of interest. The bounds match, up to O(1/f ?) factors, 5

w 7 = (0:4; 0:4; 0:2; 0:2)

500

100

20

CombCascade CombUCB1 2k

4k 6k Step n

8k

10k

300 200 100 0

w 7 = (0:4; 0:4; 0:3; 0:3)

80 Regret

40

0

w 7 = (0:4; 0:4; 0:9; 0:1)

400

60

Regret

Regret

80

60 40 20

2k

4k 6k Step n

8k

10k

0

2k

4k 6k Step n

8k

10k

Figure 1: The regret of CombCascade and CombUCB1 in the synthetic experiment (Section 4.1). The results are averaged over 100 runs. the upper bounds of CombUCB1 in stochastic combinatorial semi-bandits [12]. Since CombCascade receives less feedback than CombUCB1, this is rather surprising and unexpected. The upper bounds of Kveton et al. [12] are known to be tight up to polylogarithmic factors. We believe that our upper bounds are also tight in the setting similar to Kveton et al. [12], where the expected weight of each item is close to 1 and likely to be observed. The assumption that f ? is large is often reasonable. In network routing, the optimal routing path is likely to be reliable. In recommender systems, the optimal recommended list often does not satisfy a reasonably large fraction of users.

4

Experiments

We evaluate CombCascade in three experiments. In Section 4.1, we compare it to CombUCB1 [12], a state-of-the-art algorithm for stochastic combinatorial semi-bandits with a linear reward function. This experiment shows that CombUCB1 cannot solve all instances of our problem, which highlights the need for a new learning algorithm. It also shows the limitations of CombCascade. We evaluate CombCascade on two real-world problems in Sections 4.2 and 4.3. 4.1

Synthetic

In the first experiment, we compare CombCascade to CombUCB1 [12] on a synthetic problem. This problem is a combinatorial cascading bandit with L = 4 items and ? = {(1, 2), (3, 4)}. CombUCB1 is a popular algorithm for stochastic combinatorial semi-bandits with a linear reward function. We P approximate maxA2? f Q (A, w) by minA2? P e2A (1 w(e)). This approximation is motivated by the fact that f (A, w) = e2A w(e) ? 1 w(e)) as mine2E w(e) ! 1. We update the e2A (1 estimates of w ? in CombUCB1 as in CombCascade, based on the weights of the observed items in (1). We experiment with three different settings of w ? and report our results in Figure 1. The settings of w ? are reported in our plots. We assume that wt (e) are distributed independently, except for the last plot where wt (3) = wt (4). Our plots represent three common scenarios P that we encountered in our experiments. In the first plot, arg max A2? f (A, w) ? = arg min A2? e2A (1 w(e)). ? In this case, both CombCascade and CombUCB1 can learn A? . The regret of CombCascade P is slightly lower than that of CombUCB1. In the second plot, arg max A2? f (A, w) ? 6= arg min A2? e2A (1 w(e)). ? In this case, CombUCB1 cannot learn A? and therefore suffers linear regret. In the third plot, we violate our modeling assumptions. Perhaps surprisingly, CombCascade can still learn the optimal solution A? , although it suffers higher regret than CombUCB1. 4.2

Network Routing

In the second experiment, we evaluate CombCascade on a problem of network routing. We experiment with six networks from the RocketFuel dataset [17], which are described in Figure 2a. Our learning problem is formulated as follows. The ground set E are the links in the network. The feasible set ? are all paths in the network. At time t, we generate a random pair of starting and end nodes, and the learning agent chooses a routing path between these nodes.

The goal of the agent is to maximizes the probability that all links in the path are up. The feedback is the index of the first link in the path which is down. The weight of link e at time t, wt (e), is an indicator of link e being 6

8k

30k Regret

1221 1755 3967

6k Regret

Network Nodes Links 1221 108 153 1239 315 972 1755 87 161 3257 161 328 3967 79 147 6461 141 374

4k 2k 0

60k

120k 180k 240k 300k Step n

1239 3257 6461

20k 10k 0

60k

120k 180k 240k 300k Step n

(a) (b) Figure 2: a. The description of six networks from our network routing experiment (Section 4.2). b. The n-step regret of CombCascade in these networks. The results are averaged over 50 runs. ? up at time t. We model wt (e) as an independent Bernoulli random variable wt (e) ? B(w(e)) with ? mean w(e) = 0.7 + 0.2 local(e), where local(e) is an indicator of link e being local. We say that the link is local when its expected latency is at most 1 millisecond. About a half of the links in our networks are local. To summarize, the local links are up with probability 0.9; and are more reliable than the global links, which are up only with probability 0.7. Our results are reported in Figure 2b. We observe that the n-step regret of CombCascade flattens as time n increases. This means that CombCascade learns near-optimal policies in all networks. 4.3

Diverse Recommendations

In our last experiment, we evaluate CombCascade on a problem of diverse recommendations. This problem is motivated by on-demand media streaming services like Netflix, which often recommend groups of movies, such as ?Popular on Netflix? and ?Dramas?. We experiment with the MovieLens dataset [13] from March 2015. The dataset contains 138k people who assigned 20M ratings to 27k movies between January 1995 and March 2015. Our learning problem is formulated as follows. The ground set E are 200 movies from our dataset: 25 most rated animated movies, 75 random animated movies, 25 most rated non-animated movies, and 75 random non-animated movies. The feasible set ? are all K-permutations of E where K/2 movies are animated. The weight of item e at time t, wt (e), indicates that item e attracts the user at time t. We assume that wt (e) = 1 if and only if the user rated item e in our dataset. This indicates that the user watched movie e at some point in time, perhaps because the movie was attractive. The user at time t is drawn randomly from our pool of users. The goal of the learning agent is to learn a list of items A? = arg max A2? E [f_ (A, w)] that maximizes the probability that at least one item is attractive. The feedback is the index of the first attractive item in the list (Section 2.3). We would like to point out that our modeling assumptions are

violated in this experiment. In particular, wt (e) are correlated across items e because the users do not rate movies independently. The result is that A? 6= arg max A2? f_ (A, w). ? It is NP-hard to compute A? . However, E [f_ (A, w)] is submodular and monotone in A, and therefore a (1 1/e) approximation to A? can be computed greedily. We denote this approximation by A? and show it for K = 8 in Figure 3a. Our results are reported in Figure 3b. Similarly to Figure 2b, the n-step regret of CombCascade is a concave function of time n for all studied K. This indicates that CombCascade solutions improve over time. We note that the regret does not flatten as in Figure 2b. The reason is that CombCascade does not learn A? . Nevertheless, it performs well and we expect comparably good performance in other domains where our modeling assumptions are not satisfied. Our current theory cannot explain this behavior and we leave it for future work.

5

Related Work

Our work generalizes cascading bandits of Kveton et al. [10] to arbitrary combinatorial constraints. The feasible set in cascading bandits is a uniform matroid, any list of K items out of L is feasible. Our generalization significantly expands the applicability of the original model and we demonstrate this on two novel real-world problems (Section 4). Our work also extends stochastic combinatorial semi-bandits with a linear reward function [8, 11, 12] to the cascade model of feedback. A similar model to cascading bandits was recently studied by Combes et al. [7]. 7

8k

K=8 K = 12 K = 16

6k Regret

Movie title Animation Pulp Fiction No Forrest Gump No Independence Day No Shawshank Redemption No Toy Story Yes Shrek Yes Who Framed Roger Rabbit? Yes Aladdin Yes

4k 2k 0

20k

40k

60k

80k

100k

Step n

(a) (b) Figure 3: a. The optimal list of 8 movies in the diverse recommendations experiment (Section 4.3). b. The n-step regret of CombCascade in this experiment. The results are averaged over 50 runs. Our generalization is significant for two reasons. First, CombCascade is a novel learning algorithm. CombUCB1 [12] chooses solutions with the largest sum of the UCBs. CascadeUCB1 [10] chooses K items out of L with the largest UCBs. CombCascade chooses solutions with the largest product of the UCBs. All three algorithms can find the optimal solution in cascading bandits. However, when the feasible set is not a matroid, it is critical to maximize the product of the UCBs. CombUCB1 may learn a suboptimal solution in this setting and we illustrate this in Section 4.1.

Second, our analysis is novel. The proof of Theorem 1 is different from those of Theorems 2 and 3 in Kveton et al. [10]. These proofs are based on counting the number of times that each suboptimal item is chosen instead of any optimal item. They can be only applied to special feasible sets, such a matroid, because they require that the items in the feasible solutions are exchangeable. We build on the recent work of Kveton et al. [12] to achieve linear dependency on K in Theorem 1. The rest of our analysis is novel. Our problem is a partial monitoring problem where some of the chosen items may be unobserved. Agrawal et al. [1] and Bartok et al. [4] studied partial monitoring problems and proposed learning algorithms for solving them. These algorithms are impractical in our setting. As an example, if we formulate our problem as in Bartok et al. [4], we get —?— actions and 2L unobserved outcomes; and 2 the learning algorithm reasons over —?— pairs of actions and requires O(2L) space. Lin et al. [15] also studied combinatorial partial monitoring. Their feedback is a linear function of the weights of chosen items. Our feedback is a non-linear function of the weights. Our reward function is non-linear in unknown parameters. Chen et al. [5] studied stochastic combinatorial semi-bandits with a non-linear reward function, which is a known monotone function of an unknown linear function. The feedback in Chen et al. [5] is semi-bandit, which is more informative than in our work. Le et al. [14] studied a network optimization problem where the reward function is a non-linear function of observations.

6

Conclusions

We propose combinatorial cascading bandits, a class of stochastic partial monitoring problems that can model many practical problems, such as learning of a routing path in an unreliable communication network that maximizes the probability of packet delivery, and learning to recommend a list of attractive items. We propose a practical UCB-like algorithm for our problems, CombCascade, and prove upper bounds on its regret. We evaluate CombCascade on two real-world problems and show that it performs well even when our modeling assumptions are violated. Our results and analysis apply to any combinatorial action set, and therefore are quite general. The strongest assumption in our work is that the weights of items are distributed independently of each other. This assumption is critical and hard to eliminate (Section 2.1). Nevertheless, it can be easily relaxed to conditional independence given the features of items, along the lines of Wen et al. [19]. We leave this for future work. From the theoretical point of view, we want to derive a lower bound on the n-step regret in combinatorial cascading bandits, and show that the factor of f ? in Theorems 1 and 2 is intrinsic. 8

2 References

[1] Rajeev Agrawal, Demosthenis Teneketzis, and Venkatachalam Anantharam. Asymptotically efficient adaptive allocation schemes for controlled i.i.d. processes: Finite parameter space. IEEE Transactions on Automatic Control,

34(3):258?267, 1989. [2] Shipra Agrawal and Navin Goyal. Analysis of Thompson sampling for the multi-armed bandit problem. In Proceeding of the 25th Annual Conference on Learning Theory, pages 39.1?39.26, 2012. [3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. Machine Learning, 47:235?256, 2002. [4] Gabor Bartok, Navid Zolghadr, and Csaba Szepesvari. An adaptive algorithm for finite stochastic partial monitoring. In Proceedings of the 29th International Conference on Machine Learning, 2012. [5] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework, results and applications. In Proceedings of the 30th International Conference on Machine Learning, pages 151?159, 2013. [6] Baek-Young Choi, Sue Moon, Zhi-Li Zhang, Konstantina Papagiannaki, and Christophe Diot. Analysis of point-to-point packet delay in an operational network. In Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, 2004. [7] Richard Combes, Stefan Magureanu, Alexandre Proutiere, and Cyrille Laroche. Learning to rank: Regret lower bounds and efficient algorithms. In Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2015. [8] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. 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. [9] Aurelien Garivier and Olivier Cappe. The KL-UCB algorithm for bounded stochastic bandits and beyond. In Proceeding of the 24th Annual Conference on Learning Theory, pages 359? 376, 2011. [10] Branislav Kveton, Csaba Szepesvari, Zheng Wen, and Azin Ashkan. Cascading bandits: Learning to rank in the cascade model. In Proceedings of the 32nd International Conference on Machine Learning, 2015. [11] Branislav Kveton, Zheng Wen, Azin Ashkan, Hoda Eydgahi, and Brian Eriksson. Matroid bandits: Fast combinatorial optimization with learning. In Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence, pages 420?429, 2014. [12] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. Tight regret bounds for stochastic combinatorial semi-bandits. In Proceedings of the 18th International Conference on Artificial Intelligence and Statistics, 2015. [13] Shyong Lam and Jon Herlocker. MovieLens Dataset. http://grouplens.org/datasets/movielens/, 2015. [14] Thanh Le, Csaba Szepesvari, and Rong Zheng. Sequential learning for multi-channel wireless network monitoring with channel switching costs. IEEE Transactions on Signal Processing, 62(22):5919?5929, 2014. [15] Tian Lin, Bruno Abrahao, Robert Kleinberg, John Lui, and Wei Chen. Combinatorial partial monitoring game with linear feedback and its applications. In Proceedings of the 31st International Conference on Machine Learning, pages 901?909, 2014. [16] Christos Papadimitriou and Kenneth Steiglitz. Combinatorial Optimization. Dover Publications, Mineola, NY, 1998. [17] Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with Rocketfuel. IEEE / ACM Transactions on Networking, 12(1):2?16, 2004. [18] William. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika, 25(3-4):285?294, 1933. [19] Zheng Wen, Branislav Kveton,

and Azin Ashkan. Efficient learning in large-scale combinatorial semi-bandits. In Proceedings of the 32nd International Conference on Machine Learning, 2015. 9