Bandits with an Edge

Dotan Di Castro[1], Claudio Gentile[2], and Shie Mannor[1]

[1]Technion, Israel Institute of Technology, Haifa, Israel
[2]Universita' dell'Insubria, Varese, Italy
`dot@tx.technion.ac.il`, `claudio.gentile@uninsubria.it`,
`shie@ee.technion.ac.il`

Abstract. We consider a bandit problem over a graph where the rewards are not directly observed. Instead, the decision maker can compare two nodes and receive (stochastic) information pertaining to the difference in their value. The graph structure describes the set of possible comparisons. Consequently, comparing between two nodes that are relatively far requires estimating the difference between every pair of nodes on the path between them. We analyze this problem from the perspective of sample complexity: How many queries are needed to find an approximately optimal node with probability more than $1 - \delta$ in the PAC setup? We show that the topology of the graph plays a crucial in defining the sample complexity: graphs with a low diameter have a much better sample complexity.

1 Introduction

We consider a graph where every edge can be sampled. When sampling an edge, the decision maker obtains a signal that is related to the value of the nodes defining the edge. The objective of the decision maker is to locate the node with the highest value. Since there is no possibility to sample the value of the nodes directly, the decision maker has to infer which is the best node by considering the differences between the nodes.

As a motivation, consider the setup where a user interacts with a webpage. In the webpage, several links or ads can be presented, and the response of the user is to click one or none of them. Essentially, in this setup we query the user to compare between the different alternatives. The response of the user is comparative: a preference of one alternative to the other will be reflected in a higher probability of choosing the alternative. It is much less likely to obtain direct feedback from a user, asking her to provide an evaluation of the worth of the selected alternative. In such a setup, not all pairs of alternatives can be directly compared or, even if so, there might be constraints on the number of times a pair of ads can be presented to a user. For example, in the context of ads it is reasonable to require that ads for similar items will not appear in the same page (e.g., two competing brands of luxury cars will not appear on the same page). In these contexts, a click on a particular link cannot be seen as an absolute relevance judgement (e.g., [13]), but rather as a relative preference. Moreover,

feedback can be noisy and/or inconsistent, hence aggregating the choices into a coherent picture may be a non-trivial task. Finally, in such contexts pairwise comparisons occur more frequently than multiple comparisons, and are also more natural from a cognitive point of view (e.g.,[22]).

We model this learning scenario as bandits on graphs where the information that is obtained is differential. We assume that there is an inherent and unknown value per node, and that the graph describes the allowed (pairwise) comparisons. That is, nodes i and j are connected by an edge if they can be compared by a single query. In this case, the query returns a random variable whose distribution depends, in general, on the values of i and j. For the sake of simplicity, we assume that the observation of the edge between nodes i and j is a random variable that depends only on the *difference* between the values of i and j. Since this assumption is restrictive in terms of applicability of the algorithms, we also consider the more general setup where contextual information is observed before sampling the edges. This is intended to model a more practical setting where, say, a web system has preliminary access to a set of user profile features.

In this paper, our goal is to identify the node with the highest value, a problem that has been studied extensively in the machine learning literature (e.g., [10,1]). More formally, our objective is to find an approximately optimal node (i.e., a node whose value is at most ϵ smaller than the highest value) with a given failure probability δ as quickly as possible. When contextual information is added, the goal becomes to progressively fasten the time needed for identifying a good node for the given user at hand, as more and more users interact with the system.

Related work.   There are two common objectives in stochastic bandit problems: minimizing the regret and identifying the "best" arm. While both objectives are of interest, regret minimization seems particularly difficult in our setup. In fact, a recent line of research related to our paper is the *Dueling Bandits Problem* of Yue et al. [24,25] (see also [11]). In the dualing bandit setting, the learner has at its disposal a *complete* graph of comparisons between pairs of nodes, and each edge (i, j) hosts an unknwon preference probability $p_{i,j}$ to be interpreted as the probability that node i will be preferred over node j. Further consistency assumptions (stochastic transitivity and stochastic triangle inequality) are added. The complete graph assumption allows the authors to define a well-founded notion of regret, and analyze a regret minimization algorithm which is further enhanced in [25] where the consistency assumptions are relaxed. Although at first look our paper seems to deal with the same setup, we highlight here the main differences. First, the setups are different with respect to the topology handled. In [24,25] the topology is always a complete graph which results in the possibility to directly compare between every two nodes. In our work (as in real life) the topology is *not* a complete graph, resulting in a framework where a comparison of two nodes requires sampling all the edges between the nodes. In the extreme case of a straight line we need to sample all the given edges in the graph in order to compare the two nodes that are farthest apart. Second, the objective of minimizing the regret is natural for a complete

graph where it amounts to comparing a choice of the best bandit repeatedly with the actual pairs chosen. In a topology other than the complete graph this notion is less clear since one has to restrict choices to edges that are available. Finally, the algorithms in [24,25] are geared towards the elimination of arms that are not optimal with high probability. In our setup one *cannot* eliminate such nodes and edges because it is crucial in comparing candidates for optimal nodes. Therefore, the resulting algorithms and analyses are quite different. On the other hand, constraining to a given set of allowed comparisons leads us to make less general statistical assumptions than [24,25], in that our algorithms are based on the ability to reconstruct the reward difference on adjacent nodes by observing their connecting edge.

From a different perspective, the setup we consider is reminiscent of online learning with partial monitoring [19]. In the partial monitoring setup, one usually does not observe the reward directly, but rather a signal that is related (probabilistically) to the unobserved reward. However, as far we know, the alternatives (called arms usually) in the partial monitoring setup are separate and there is no additional structure: when sampling an arm a reward that is related to this arm alone is obtained but not observed. Our work differs in imposing an additional structure, where the signal is derived from the structure of the problem where the signal is always relative to adjacent nodes. This means that comparing two nodes that are not adjacent requires sampling all the edges on a path between the two nodes. So that deciding which of two remote nodes has higher value requires a high degree of certainty regarding all the comparisons on the path between them.

Another research area which is somewhat related to this paper is learning to rank via comparisons (a very partial list of references includes [7,13,8,4,14,5,12,23,18]). Roughly speaking, in this problem we have a collection of training instances to be associated with a finite set of possible alternatives or classes (the graph nodes in our setting). Every training example is assigned a set of (possibly noisy or inconsistent) pairwise (or groupwise) preferences between the classes. The goal is to learn a function that maps a new training example to a total order (or ranking) of the classes. We emphasize that the goal in this paper is different in that we work in the bandit setup with a given structure for the comparisons and, in addition, we are just aiming at identifying the (approximately) best class, rather than ranking them all.

Content of the paper. The rest of the paper is organized as follows. We start from the formal model in Section 2. We analyze the basic linear setup, where each node is comparable to at most two nodes in Section 3. We then move to the tree setup and analyze it in Section 4. The general setup of a network is treated in Section 5. Some experiments are then presented in Section 6 to elucidate the theoretical findings in previous sections. In Section 7 we discuss the more general setting with contextual information. We close with some directions for future research.

2   Model and Preliminaries

In this section we describe the classical Multi-Armed Bandit (MAB) setup, describe the Graphical Bandit (GB) setup, state/recall two concentration bounds for sequences of random variables, and review a few terms from graph theory.

2.1   The Multi-Armed Bandit Problem

The MAB model [16] is comprised of a set of *arms* $A \triangleq \{1, \ldots, n\}$. When sampling arm $i \in A$, a *reward* which is a random variable R_i, is provided. Let $r_i = \mathbb{E}[R_i]$. The goal in the MAB setup is to find the arm with the highest expected reward, denoted by r^*, where we term this arm's reward the *optimal reward*. An arm whose expected reward is strictly less than r^* is called a *non-best arm*. An arm i is called an ϵ-optimal arm if its expected reward is at most ϵ from the optimal reward, i.e., $\mathbb{E}[R_i] \geq r^* - \epsilon$. In some cases, the goal in the MAB setup is to find an ϵ-optimal arm.

   A typical algorithm for the MAB problem does the following. At each time step t it samples an arm i_t and receives a reward R_{i_t}. When making its selection, the algorithm may depend on the history (i.e., the actions and rewards) up to time $t-1$. Eventually the algorithm must commit to a single arm and select it. Next we define the desired properties of such an algorithm.

Definition 1. *(PAC-MAB) An algorithm is an (ϵ, δ)-probably approximately correct (or (ϵ, δ)-PAC) algorithm for the MAB problem with sample complexity T, if it terminates and outputs an ϵ-optimal arm with probability at least $1 - \delta$, and the number of times it samples arms before termination is bounded by T.*

In the case of standard MAB problems there is no structure defined over the arms. In the next section we describe the setup of our work where such a structure exists.

2.2   The Graphical Bandit Problem

Suppose that we have an undirected and connected graph $G = (V, E)$ with nodes $V = \{1, \ldots, n\}$ and edges E. The nodes are associated with reward values $r_1, \ldots, r_n$, respectively, that are unknown to us. We denote the node with highest value by i^* and, as before, $r^* = r_{i^*}$. Define $u \triangleq \min_{j \neq i^*} r_{i^*} - r_j$ to be the difference between the node with the highest value and the node with the second highest value. We call u the reward *gap*, and interpret it as a measure for how easy is to discriminate between the two best nodes in the network. As expected, the gap u has a significant influence on the sample complexity bounds (provided the accuracy parameter ϵ is not large). We say that nodes i and j are *neighbors* if there is an edge in E connecting them (and denote the edge random variable by E^{ij}). This edge value is a random variable whose distribution is determined by the nodes it is connecting, i.e., (i, j)'s statistics are determined by r_i and r_j. In

this work, we assume that[1] $\mathbb{E}\left[E^{ij}\right] = r_j - r_i$. Also, for the sake of concreteness, we assume the edge values are bounded in $[-1, 1]$.

In this model, we can only sample the graph edges E^{ij} that provide independent realizations of the node differences. For instance, we may interpret $E^{ij} = +1$ if the feedback we receive says that item j is preferred over item i, $E^{ij} = -1$ if i is preferred over j, and $E^{ij} = 0$ if no feedback is received. Then the reward difference $r_j - r_i$ becomes equal to the difference between the probability of preferring j over i and the probability of preferring i over j. Let us denote the realizations of E^{ij} by E_t^{ij} where the subscript t denotes time. Our goal is to find an ϵ-optimal node, i.e., a node i whose reward satisfies $r_i \geq r^* - \epsilon$.

Whereas neighboring nodes can be directly compared by sampling its connecting edge, if the nodes are far apart, a comparison between the two can only be done indirectly, by following a path connecting them. We denote a *path* between node i and node j by π_{ij}. Observe that there can be several paths in G connecting i to j. For a given path π from i to j, we define the *composed edge* value E_π^{ij} by $E_\pi^{ij} = \sum_{(k,l) \in \pi_{ij}} E^{kl}$, with $E_\pi^{ii} = 0$. By telescoping, the average value of a composed edge E^{ij} only depends on its endpoints, i.e.,

$$\mathbb{E}\left[E_\pi^{ij}\right] = \sum_{(k,l) \in \pi_{ij}} \mathbb{E}\left[E^{kl}\right] = \sum_{(k,l) \in \pi_{ij}} (r_l - r_k) = r_j - r_i, \tag{1}$$

independent of π. Similarly, define E_t^{ij} to be the time-t realization of the composed edge random variable E_π^{ij} when we pull once all the edges along the path π joining i to j. A schematic illustration of the the GB setup is presented in Figure 1.
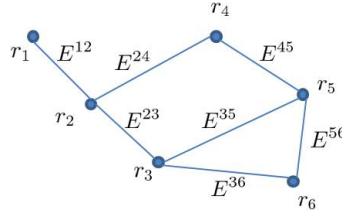

Fig. 1. Schematic illustration of the the GB setup for 6 nodes

The algorithms we present in the next sections hinge on constructing reliable estimates of edge reward differences, and then combining them into a suitable node selection procedure. This procedure heavily depends on the graph topology. In a tree-like (i.e., acyclic) structure no inconsistencies can arise due to the noise in the edge estimators. Hence the node selection procedure just aims at identifying the node with the largest reward gap to a given reference node. On

---

[1] Notice that, although the graph is undirected, we view edge (i, j) as a directed edge from i to j. It is understood that $E^{ji} = -E^{ij}$.

the other hand, if the graph has cycles, we have to rely on a more robust node elimination procedure, akin to the one investigated in [10] (see also the more recent [1]).

2.3   Large Deviations Inequalities

In this work we use Hoeffding's maximal inequality (e.g., [6]).

Lemma 1. *Let $X_1, \ldots, X_N$ be independent random variables with zero mean satisfying $a_i \leq X_i \leq b_i$ w.p. 1. Let $S_i = \sum_{j=1}^{i} X_j$. Then,*

$$P\left(\max_{1 \leq i \leq N} S_i > \epsilon\right) \leq \exp\left(-\frac{\epsilon^2}{\sum_{i=1}^{N} (b_i - a_i)^2}\right).$$

3   Linear topology and sample complexity

As a warm-up, we start by considering the GB setup in the case of a linear graph, i.e., $E = \{(i, i+1) : 1 \leq i \leq n-1\}$. We call it the *linear setup*. The algorithm for finding the highest node in the linear setup is presented in Algorithm 1. The algorithm samples all the edges, computes for each edge its empirical mean, and based on these statistics finds the highest edge. Algorithm 1 will also serve as a subroutine for the tree-topology discussed in Section 4. The following proposition

---

Algorithm 1 The algorithm for the linear setup

Input: $\epsilon > 0$, $\delta > 0$, line graph with edge set $E = \{(i, i+1) : 1 \leq i \leq n-1\}$

1: **for** $i = 1, \ldots, n-1$ **do**
2:     Pull edge $(i, i+1)$ for T^i times
3:     Let $\hat{E}^{i,i+1} = \frac{1}{T^i} \sum_{t=1}^{T^i} E_t^{i,i+1}$ be the empirical average of edge $(i, i+1)$
4:     Let $\hat{E}_{\pi_{1i}}^{1i} = \sum_{k=1}^{i-1} \hat{E}^{k-1,k}$ be the empirical average of the composed edge $E_{\pi_{1i}}^{1i}$, where π_{1i} is the (unique) path from 1 to i.
5: **end for**

Output: Node $k = \operatorname{argmax}_{i=1,\ldots,n} \hat{E}_{\pi_{1i}}^{1i}$.

---

gives the sample complexity of Algorithm 1 in the case when the edges are bounded.

Proposition 1. *If $-1 \leq E^{i,i+1} \leq 1$ holds, then Algorithm 1 operating on a linear graph with reward gap u is an (ϵ, δ)-PAC algorithm when the T^i satisfy*

$$\left(\sum_{i=1}^{n-1} \frac{4}{T^i}\right)^{-1} \geq \frac{1}{\max\{\epsilon, u\}^2} \log\left(\frac{2}{\delta}\right).$$

If $T^i = T$ then the sample complexity of each edge is $T \geq \frac{4n}{\max\{\epsilon, u\}^2} \log\left(\frac{2}{\delta}\right)$. Hence the sample complexity of the algorithm is $O\left(\frac{n^2}{\max\{\epsilon, u\}^2} \log\left(\frac{1}{\delta}\right)\right)$.

Proof. Let $\tilde{E}_t^{i,i+1} \triangleq \frac{(r_{i+1}-r_i)-E_t^{i,i+1}}{T^i}$, $\quad t = 1,...T^i$. Each $\tilde{E}_t^{i,i+1}$ has zero mean with $-2/T^i \le \tilde{E}_t^{i,i+1} \le 2/T^i$. Hence

$$\tilde{E}_1^{1,2},\ldots,\tilde{E}_{T^1}^{1,2},\tilde{E}_1^{2,3},\ldots,\tilde{E}_{T^2}^{2,3},\ldots,\tilde{E}_1^{n-1,n},\ldots,\tilde{E}_{T^{n-1}}^{n-1,n} \tag{2}$$

is a sequence of $\sum_{i=1}^{n-1} T^i$ zero-mean and independent random variables. Set for brevity $\tilde{\epsilon} = \max\{\epsilon, u\}$, and suppose, without lost of generality, that some node j has the highest value. The probability that Algorithm 1 fails, i.e., returns a node whose value is ϵ below the optimal value is bounded by

$$\Pr\left(\exists i = 1,\ldots,n : \hat{E}_{\pi_{1i}}^{1i} - \hat{E}_{\pi_{1j}}^{1j} > 0 \text{ and } r_i < r_j - \tilde{\epsilon}\right). \tag{3}$$

We can write

$$(3) \le \Pr\left(\exists i = 1,\ldots,n : \hat{E}_{\pi_{1i}}^{1i} - \hat{E}_{\pi_{1j}}^{1j} - (r_i - r_j) > \tilde{\epsilon}\right)$$

$$= \Pr\left(\exists i = 1,\ldots,n : \sum_{k=1}^{i-1}\sum_{t=1}^{T^i} \tilde{E}_t^{k,k+1} > \tilde{\epsilon}\right)$$

$$\le \Pr\left(\exists \text{ partial sum in (2) with magnitude} > \tilde{\epsilon}\right)$$

$$\le 2\exp\left(-\frac{\tilde{\epsilon}^2}{\sum_{k=1}^{n-1}\sum_{t=1}^{T^k}(2/T^k)^2}\right),$$

where in the last inequality we used Lemma 1. Requiring this probability to be bounded by δ yields the claimed inequality. $\qquad\square$

The sample sizes T^i in Proposition 1 encode constraints on the number of times the edges $(i, i+1)$ can be sampled. Notice that the statement therein implies $T_i \ge \frac{4}{\max\{\epsilon,u\}^2}\log\left(\frac{2}{\delta}\right)$ for all i, i.e., we cannot afford in a line graph to undersample any edge. This is because every edge in a line graph is a *bridge*, hence a poor estimation of any such edge would affect the differential reward estimation throughout the graph. In this respect, this proposition only allows for a partial tradeoff among these numbers.

4   Tree topology and its sample complexity

In this section we investigate PAC algorithms for finding the best node in a tree. Let then $G = (V, E)$ be an n-node tree with diameter D and a set of leaves $L \subseteq V$. Without loss of generality we can assume that the tree is rooted at node 1 and that all edges are directed downwards to the leaves. Algorithm 2 considers all possible paths from the root to the leaves and treats each one of them as a line graph to be processed as in Algorithm 1. We have the following proposition where, for simplicity of presentation, we do no longer differentiate among the sample sizes $T^{i,j}$ associated with edges (i, j).

Algorithm 2 The algorithm for the tree setup

Input: $\epsilon > 0$, $\delta > 0$, tree graph with set of leaves $L \subseteq V$
1: **for** all leaves $k \in L$ **do**
2:     Pull each edge $(i,j) \in E$ for T times
3:     Let $\hat{E}^{ij} = \frac{1}{T} \sum_{t=1}^{T} E_t^{ij}$ be the empirical average of edge (i,j), and $m_k = \operatorname{argmax}_{(1,i) \in \pi_{1,k}} \hat{E}_{\pi_{1i}}^{1i}$ be the maximum empirical average along path π_{1k} (as in Algorithm 1)
4: **end for**
Output: Node $m = \operatorname{argmax}_{k \in L} m_k$.

Proposition 2. *If $-1 \leq E^{ij} \leq 1$ holds, then Algorithm 2 operating on a tree graph with reward gap u is an (ϵ, δ)-PAC algorithm when the sample complexity T of each edge satisfies $T \geq \frac{4D}{\max\{\epsilon, u\}^2} \log\left(\frac{2|L|}{\delta}\right)$. Hence the sample complexity of the algorithm is $O\left(\frac{nD}{\max\{\epsilon, u\}^2} \log\left(\frac{|L|}{\delta}\right)\right)$.*

Proof. The probability that Algorithm 2 returns a node whose average reward is ϵ below the optimal one coincides with the probability that there exists a leaf $k \in L$ such that Algorithm 1 operating on the linear graph $\pi_{1,k}$ singles out a node m_k whose average reward is more than ϵ from the optimal one within $\pi_{1,k}$. Setting $T = \frac{4|\pi_{1,k}|}{\tilde{\epsilon}^2} \log\left(\frac{2|L|}{\delta}\right)$, with $\tilde{\epsilon} = \max\{\epsilon, u\}$, ensures that the above happens with probability at most $\delta/|L|$. Hence each edge is sampled at most $\frac{4D}{\tilde{\epsilon}^2} \log\left(\frac{2|L|}{\delta}\right)$ times and the claim follows by a standard union bound over L. $\square$

5   Network Sample Complexity

In this section we deal with the problem of finding the optimal reward in a general connected and undirected graph $G = (V, E)$, being $|V| = n$. We describe a node elimination algorithm that works in phases, sketch an efficient implementation and provide a sample complexity. The following ancillary definitions will be useful. We say that a node is a *local maximum* in a graph if all its neighboring nodes do not have higher expected reward than the node itself. The distance between node i and node j is the length of the shortest path between the two nodes. Finally, the diameter $D(G)$ of a graph G is the largest distance between any pair of nodes.

Our suggested Algorithm operates in $\log n$ phases. For notational simplicity, it will be convenient to use subscripts to denote the phase number. We begin with Phase 1, where the graph $G_1 = (V_1, E_1)$ is the original graph, i.e., at the beginning all nodes are participating, and $n_1 = |V_1| = n$. We then find a subgraph of G_1, which we call *sampled graph* denoted by G_1^S, that includes all the edges involved in shortest paths between all nodes in V_1. We sample each edge

in subgraph G_1^S for T_1-times, and compute the corresponding sample averages. Based on these averages, we find the local maxima[2] of G_1^S.

The key observation is that there can be at most $n_1/2$ maxima. Denote this set of maxima by V_2. Now, define a subgraph, denoted by G_2, whose nodes are V_2. We repeat the process of getting a sampled graph, denoted by G_2^S. We sample the edges of the sampled graph G_2^S for T_2-times and define, based on its maxima, a new subgraph. Denote the set of maxima by V_3, and the process continues until only one node is left. We call this algorithm NNE (*Network Node Elimination*), which is similar to the action elimination procedure of [10] (see also [1]). The algorithm is summarized in Algorithm 3. Two points should

---

Algorithm 3 The Network Node Elimination Algorithm

Input: $\epsilon > 0$, $\delta > 0$, graph $G = (V, E)$, $i = 1$
 1: Initialize $G_1 = G$, $V_1 = V$
 2: Compute the shortest path between all pairs of nodes of G_1, and denote each path by π_{ij}.
 3: Initialize the shortest path set by $SP_1 = \{\pi_{ij}|i, j \in V_1\}$
 4: **while** $|V_i| > 1$ **do**
 5:     $n_i = |V_i|$
 6:     Using the shortest paths in SP_i, find a sampled graph G_i^S of G_i
 7:     $D_i = D(G_i^S)$
 8:     Pull each edge in G_i^S for T_i times
 9:     Find the local maxima set, V_{i+1}, on G_i^S, and get a subgraph G_{i+1} that contains V_{i+1}
10:     $SP_{i+1} = \{\pi_{ij} \in SP_i|i, j \in V_{i+1}\}$
11:     $i \leftarrow i + 1$
12: **end while**
Output: The remaining node

---

be made regarding the NNE algorithm. First, as will be observed below, the sequence $\{D(G_i^S)\}_{i=1}^{\log n}$ of diameters is nonincreasing. Second, from the implementation viewpoint, a data-structure maintaining all shortest paths between nodes is crucial, in order to efficiently eliminate nodes while tracking the shortest paths between the surviving nodes of the graph. In fact, this data structure might just be a collection of n breadth-first spanning trees rooted at each node, that encode the shortest path between the root and any other node in the graph. When node i gets eliminated, we first eliminate the spanning tree rooted at i, but also prune all the other spanning trees where node i occurs as a leaf. If i is a (non-root) internal node of another tree, then i should not be eliminated from this tree since i certainly belongs to the shortest path between another pair of surviving nodes. Note that connectivity is maintained through the process.

The following result gives a PAC bound for Algorithm 3 in the case when the $E^{i,j}$ are bounded.

---
[2] Ties can be broken arbitrarily.

Proposition 3. *Suppose that $-1 \leq E^{i,j} \leq 1$ for every $(i,j) \in E$. Then Algorithm 3 operating on a general graph G with diameter D and reward gap u is an (ϵ, δ)-PAC algorithm with edge sample complexity*

$$T \leq \frac{\sum_{i=1}^{\log n} n_i D_i}{(\max\{\epsilon, u\}/\log n)^2} \, \log\left(\frac{n}{\delta/\log n}\right) \leq \frac{n\,D}{(\max\{\epsilon, u\}/\log n)^2} \, \log\left(\frac{n}{\delta/\log n}\right).$$

Proof. In each phase we have at most half the nodes of the previous phase, i.e., $n_{i+1} \leq n_i/2$. Therefore, the algorithm stops after at most $\log n$ phases. Also, because we retain shortest path between the surviving nodes, we also have $D_{i+1} \leq D_i \leq D$. At each phase, similar to the previous sections, we make sure that it is at most $\delta/\log n$ the probability of identifying an $\epsilon/\log n$-optimal node. Therefore, it suffices to pull the edges in each sampled graph G_i^S for $T_i \leq \frac{n_i D_i}{(\max\{\epsilon, u\}/\log(n))^2} \log\left(\frac{n_i}{\delta/\log n}\right)$ times. Hence the overall sample complexity for an (ϵ, δ)-PAC bound is at most $\sum_{j=1}^{\log n} T_j$, as claimed. The last inequality just follows from $n_{i+1} \leq n_i/2$ and $D_i \leq D$ for all i. $\qquad\square$

Being more general, the bound contained in Proposition 3 is weaker than the ones in previous sections when specialized to line graphs or trees. In fact, one is left wondering whether it is always convenient to reduce the identification problem on a general graph G to the identification problem on trees by, say, extracting a suitable spanning tree of G and then invoking Algorithm 2 on it. The answer is actually negative, as the set of simulations reported in the next section show.

6   Simulations

In this section we briefly investigate the role of the graph topology in the sample complexity.

In our simple experiment we compare Algorithm 2 (with two types of spanning trees) to Algorithm 3 over the "spider web graph" illustrated in Figure 2 (a). This graph is made up of 15 nodes arranged in 3 concentric circles (5 nodes each), where the circles are connected so as to resemble a spider web. Node rewards are independently generated from the uniform distribution on [0,1], edge rewards are just uniform in [-1,+1]. The two mentioned spanning trees are the longest diameter spanning tree (diameter 14) and the shortest diameter spanning tree (diameter 5). As we see from Figure 2 (b), the latter tends to outperform the former. However, both spanning tree-based algorithms are eventually outperformed by NNE on this graph. This is because in later stages NNE tends to handle smaller subgraphs, hence it needs only compare subsets of "good nodes".

7   Extensions

We now sketch an extension of our framework to the case when the algorithm receives contextual information in the form of feature vectors before sampling
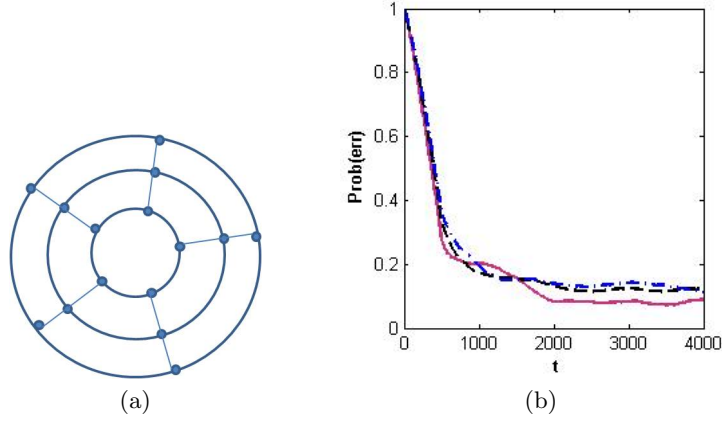
Fig. 2. (*a*) The spider-web topology. (*b*) Empirical error vs. time for the graph setup in (a) and spanning trees thereof. Three algorithms are compared: NNE (red solid line), the tree-based algorithm operating on a smallest diameter spanning tree (black dashed line), and the tree-based algorithm operating on a largest diameter spanning tree (blue dash-dot line). The parameters are $n = 15$ and $\epsilon = 0$. Average of 200 runs.

the edges. This is intended to model a more practical setting where, say, a web system has preliminary access to a set of user profile features.

This extension is reminiscent of the so-called *contextual bandit* learning setting (e.g., [17]), also called *bandits with covariates* (e.g., [21]). In such a setting, it is reasonable to assume that different users $\boldsymbol{x}_s$ have different preferences (i.e., different best nodes associated with), but also that similar users tend to have similar preferences. A simple learning model that accommodates the above (and is also amenable to theoretical analysis) is to assume each node i of G to host a linear function $\boldsymbol{u}_i \,:\, \boldsymbol{x} \to \boldsymbol{u}_i^\top \boldsymbol{x}$ where, for simplicity, $||\boldsymbol{u}_i|| = ||\boldsymbol{x}|| = 1$ for all i and $\boldsymbol{x}$. The optimal node $i^*(\boldsymbol{x})$ corresponding to vector $\boldsymbol{x}$ is $i^*(\boldsymbol{x}) = \arg\max_{i \in V} \boldsymbol{u}_i^\top \boldsymbol{x}$. Our goal is to identify, for the given $\boldsymbol{x}$ at hand, an ϵ-optimal node j such that $\boldsymbol{u}_j^\top \boldsymbol{x} \geq \boldsymbol{u}_{i^*}^\top \boldsymbol{x} - \epsilon$. Again, we do not directly observe node rewards, but only the differential rewards provided by edges.[3] When we operate on input $\boldsymbol{x}$ and pull edge (i,j), we receive an independent observation of random variable $E^{ij}(\boldsymbol{x})$ such that $\mathbb{E}[E^{ij}(\boldsymbol{x})] = \boldsymbol{u}_j^\top \boldsymbol{x} - \boldsymbol{u}_i^\top \boldsymbol{x}$.

Learning proceeds in a sequence of *stages* $s = 1, \ldots, S$, each stage being in turn a sequence of time steps corresponding to the edge pulls taking place in that stage. In Stage 1 the algorithm gets input $\boldsymbol{x}_1$, is allowed to pull (several times) the graph edges $E^{ij}(\boldsymbol{x}_1)$, and is required to output an ϵ-optimal node for $\boldsymbol{x}_1$. Let $T(\boldsymbol{x}_1)$ be the sample complexity of this stage. In Stage 2, we retain the information gathered in Stage 1, receive a new vector $\boldsymbol{x}_2$ (possibly close to $\boldsymbol{x}_1$) and repeat the same kind of inference, with sample complexity $T(\boldsymbol{x}_2)$. The game continues until S stages have been completed.

---

[3]   For simplicity of presentation, we disregard the reward gap here.

For any given sequence $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_S$, one expects the cumulative sample size $\sum_{s=1}^{S} T(\boldsymbol{x}_s)$ to grow *less than linearly* in S. In other words, the additional effort the algorithm makes in the identification problem diminishes with time, as more and more users are interacting with the system, especially when these users are similar to each other, or even occur more than once in the sequence $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_S$. In fact, we can prove stronger results of the following kind. Notice that the bound does not depend on the number S of stages, but only on the dimension of the input space.[4]

Proposition 4. *Under the above assumptions, if $G = (V, E)$ is a connected and undirected graph, with n nodes and diameter D, and $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_S \in R^d$ is any sequence of unit-norm feature vectors, then with probability at least $1 - \delta$ a version of the NNE algorithm exists which outputs at each stage s an ϵ-optimal node for $\boldsymbol{x}_s$, and achieves the following cumulative sample size*

$$\sum_{s=1}^{S} T(\boldsymbol{x}_s) = O\left(B \log^2 B\right),$$

where $B = \frac{n\,D}{(\epsilon/\log n)^2} \log\left(\frac{n}{\delta/\log n}\right) d^2$.

Proof (Sketch). The algorithm achieving this bound combines linear-regression-like estimators with NNE. In particular, every edge of G maintains a linear estimator $\hat{\boldsymbol{u}}^{ij}$ intended to approximate the difference $\boldsymbol{u}_j - \boldsymbol{u}_i$ over both stages and sampling times within each stage. At stage s and sampling time t within stage s, the vector $\hat{\boldsymbol{u}}_{s,t}^{ij}$ suitably stores all past feature vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_s$ observed so far, along with the corresponding edge reward observations. By using tools from ridge regression in adversarial settings (see, e.g., [9]), one can show high-probability approximation results of the form

$$(\hat{\boldsymbol{u}}_{s,t}^{ij\,\top} \boldsymbol{x} - (\boldsymbol{u}_j - \boldsymbol{u}_i)^\top \boldsymbol{x})^2 \leq \boldsymbol{x}^\top A_{s,t}^{-1} \boldsymbol{x} \left(d \log \Sigma_{s,t} + \log \frac{1}{\delta}\right), \qquad (4)$$

being $\Sigma_{s,t} = \sum_{k \leq s-1} T(\boldsymbol{x}_k) + t$, and $A_{s,t}$ the matrix

$$A_{s,t} = I + \sum_{k \leq s-1} T(\boldsymbol{x}_k)\boldsymbol{x}_k\boldsymbol{x}_k^\top + t\,\boldsymbol{x}_s\boldsymbol{x}_s^\top .$$

In stage s, NNE is able to output an ϵ-optimal node for input $\boldsymbol{x}_s$ as soon as the RHS of (4) is as small as $c\epsilon^2$, for a suitable constant c depending on the current graph topology NNE is operating on. Then the key observation is that in stage s the number of times we sample an edge (i, j) such that the above is false cannot be larger than

$$\frac{1}{c\epsilon^2} \log \frac{|A_{s,T(\boldsymbol{x}_s)}|}{|A_{s,0}|} \left(d \log \Sigma_{s,T(\boldsymbol{x}_s)} + \log \frac{1}{\delta}\right),$$

---

[4] A slightly different statement holds in the case when the input dimension is infinite. This statement quantifies the cumulative sample size w.r.t. the amount to which the vectors $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_S$ are close to each other. Details are omitted due to lack of space.

where $|\cdot|$ is the determinant of the matrix at argument. This follows from standard inequalities of the form $\sum_{t=1}^{T(\boldsymbol{x}_s)} \boldsymbol{x}_s^\top A_{s,t}^{-1} \boldsymbol{x}_s \leq \log \frac{|A_{s,T(\boldsymbol{x}_s)}|}{|A_{s,0}|}$. $\qquad\square$

8   Discussion

This paper falls in the research thread of analyzing online decision problems where the information that is obtained is comparative between arms. We analyzed a simple setup where the structure of comparisons is provided by a given graph which, unlike previous works on this subject [24,25], lead us focus on the notion of finding an ϵ-optimal arm with high probability. We then described an extension to the important contextual setup. There are several issues that call for further research that we outline below.

First, we only addressed the exploratory bandit problem. It would be interesting to consider the regret minimization version of the problem. While naively one can think of it as a problem with an arm per edge of the graph, this may not be a very effective model because the number of arms may go as n^2 but the number of parameters grows like n. On top of this, definining a meaningful notion of regret may not be trivial (see the discussion in the introductory section). Second, we only considered graphs as opposed to hypergraphs. Considering comparisons of more than two nodes raises interesting modeling issues and well as computational issues. Third, we assumed that all samples are equivalent in the sense that all the pairs we can compare have the same cost. This is not a realistic assumption in many applications. An approach akin to budgeted learning [20] would be interesting here. Fourth, we focused on upper bounds and constructive algorithms. Obtaining lower bounds that depend on the network topology would be interesting. The upper bounds we have provided are certainly loose for the case of a general network. Furthermore, more refined upper bounds are likely to exist which take into account the distance on the graph between the good nodes (e.g., between the best and the second best ones). In any event, the algorithms we developed for the network case are certainly not optimal. There is room for improvement by reusing information better and by adaptively selecting which portions of the network to focus on. This is especially interesting under smoothness assumptions on the expected rewards. Relevant references in the MAB setting to start off with include [2,15,3].

References

1. J.Y. Audibert, S. Bubeck, R. Munos (2010). Best Arm Identification in Multi-Armed Bandits. *Conference on Learning Theory* (COLT 2010). pp. 41–53
2. P. Auer, R. Ortner, C. Szepesvri (2007). Improved Rates for the Stochastic Continuum-Armed Bandit Problem. *Conference on Learning Theory* (COLT). pp. 454–468
3. S. Bubeck, R. Munos, G. Stoltz, C. Szepesvari (2008). Online Optimization in X-Armed Bandits. *Advances in Neural Information Processing Systems* (NIPS). pp. 201–208

4. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender (2005). Learning to Rank Using Gradient Descent. *International Conference on Machine Learning* (ICML), 89–96

5. Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, H. Li (2007). Learning to Rank: From Pairwise to Listwise Approach. *International Conference on Machine Learning* (ICML). pp. 129–136

6. N. Cesa-Bianchi, G. Lugosi (2006). *Prediction, Learning, and Games*. Cambridge University Press.

7. W. Cohen, R. Schapire, Y. Singer (1999). Learning to order things. *Journal of Artificial Inteligence Research* (JAIR). **10**, pp. 243–270

8. O. Dekel C. Manning, Y. Singer (2003). Log-Linear Models for Label Ranking, *Advances in Neural Information Processing Systems* (NIPS).

9. O. Dekel, C. Gentile, K. Sridharan (2010). Robust Selective Sampling From Single and Multiple Teachers. *Conference on Learning Theory* (COLT 2010) pp. 346–358

10. E. Even-Dar, S. Mannor, Y. Mansour (2006). Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems. *Journal of Machine Learning Research* (JMLR), **7**: 1079–1105, MIT Press

11. U. Feige, P. Raghavan, D. Peleg, and E. Upfal (1994). Computing with Noisy Information. *SIAM J. Comput.*, pp.1001–1018

12. E. Hullermeier, J. Furnkranz, W. Cheng, K. Brinker (2008). Label Ranking by Learning Pairwise Preferences. *Artificial Intelligence*, **172**, pp. 897–1916

13. T. Joachims (2002). Optimizing Search Engines Using Clickthrough Data. *Eighth ACM International Conference on Knowledge Discovery and Data Mining* (SIGKDD), pp. 133–142

14. T. Joachims, F. Radlinski (2005). Query Chains: Learning to Rank from Implicit Feedback. *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (ACM KDD), pp. 239–248

15. R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandit problems in metric spaces. *In Proc. 40th ACM Symposium on Theory of Computing* (STOC 2008), pp. 681–690.

16. T.L. Lai, H. Robbins (1985). Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics Elsevier*, **6:1**, 4–22

17. J. Langford, T. Zhang (2007). The Epoch-Greedy Algorithm for Contextual Multi-Armed Bandits. *Advances in Neural Information Processing Systems*

18. T.Y. Liu (2009). Learning to Rank for Information Retrieval. *Foundations and trends in Information Retrieval*, **3**, pp. 225–331

19. G. Lugosi, S. Mannor, G. Stoltz (2008). Strategies for Prediction Under Imperfect Monitoring. *Mathematics of Operations Research*, **33:3**, pp. 513–528

20. O. Madani, D.L.J Lizotte, R. Greiner (2004). The Budgeted Multi-Armed Bandit Problem. *Conference on Learning Theory* (COLT), pp. 643–645.

21. P. Rigollet, A. Zeevi (2010). Nonparametric Bandits with Covariates. *Conference on Learning Theory* (COLT 2010) pp. 54–66

22. L. L. Thurstone (1927). A Law of Comparative Judgement. *Psychological Review*, **34**, 278–286

23. F. Xia, T.Y. Liu, J. Wang, W. Zhang, H. Li (2008). Listwise Approach to Learning to Rank - Theory and Algorithm. *International Conference on Machine Learning* (ICML), pp. 1192–1199

24. Y. Yue, J. Broder, R. Kleinberg, T. Joachims (2011). The K-armed Dueling Bandits Problem. *Journal of Computer and System Sciences* (JCSS), Special Issue on Learning Theory, to appear

25. Y. Yue, T. Joachims (2011). Beat the Mean Bandit. *International Conference on Machine Learning* (ICML), to appear