Thresholding Graph Bandits with GrAPL

Daniel LeJeune
Rice University

Gautam Dasarathy
Arizona State University

Richard G. Baraniuk
Rice University

Abstract

In this paper, we introduce a new online decision making paradigm that we call *Thresholding Graph Bandits*. The main goal is to efficiently identify a subset of arms in a multi-armed bandit problem whose means are above a specified threshold. While traditionally in such problems, the arms are assumed to be independent, in our paradigm we further suppose that we have access to the similarity between the arms in the form of a graph, allowing us to gain information about the arm means with fewer samples. Such a feature is particularly relevant in modern decision making problems, where rapid decisions need to be made in spite of the large number of options available. We present GrAPL, a novel algorithm for the thresholding graph bandit problem. We demonstrate theoretically that this algorithm is effective in taking advantage of the graph structure when the structure is reflective of the distribution of the rewards. We confirm these theoretical findings via experiments on both synthetic and real data.

1 INTRODUCTION

Systems that recommend products, services, or other attention-targets have become indispensable in the effective curation of information. Such personalization and recommendation techniques have become ubiquitous not only in product/content recommendation and ad placements but also in a wide range of applications like drug testing, spatial sampling, environmental monitoring, and rate adaptation in communication networks; see, e.g., Villar et al. (2015); Combes et al. (2014); Srinivas et al. (2010). These are often modeled as sequential decision making or *bandit problems*, where an algorithm needs to choose among a set of decisions (or arms) sequentially to maximize a desired performance criterion.

Recently, an important variant of the bandit problem was proposed by Locatelli et al. (2016) and Gotovos et al. (2013), where the goal is to rapidly identify all arms that are above (and below) a fixed threshold. This *thresholding bandit* framework, which may be thought of as a version of the combinatorial pure exploration problem (Chen et al., 2014), is useful in various applications like environmental monitoring, where one might want to identify the hypoxic (low-oxygen-content) regions in a lake; like crowd-sourcing, where one might want to keep all workers whose productivity trumps the cost to hire them; or like political polling, where one wants to identify which political candidate individual voting districts prefer. Such a procedure may even be considered in human-in-the-loop machine learning pipelines, where the algorithm might want to select a set of options that meet a certain cut-off for closer examination by a human expert.

In many important applications, however, one is faced with an enormous number of arms that need to sorted through almost instantaneously. This makes prior approaches untenable both from a computational and from a statistical viewpoint. However, when there is *information sharing* between these arms, one might hope that this situation can be improved.

In this paper, we consider the thresholding bandit problem in the setting where a graph describing the similarities between the arms is available (see Section 2). We show that if one leverages this graph information, and more importantly the *homophily* (that is, that strong connection implies similar behavior), then one can achieve significant gains over prior approaches. We develop a novel algorithm, GrAPL (see Section 3), that explicitly takes advantage of the graph structure and the homophily. We then characterize, using rigorous theoretical estimates of the error of GrAPL, how this algorithm indeed leverages this side information to improve upon prior algorithms in similar settings.

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

Finally, in Section 4, we confirm these theoretical findings via experiments on real and synthetic data.

2 THRESHOLDING GRAPH BANDITS

2.1 Thresholding Bandits

Let N denote the number of bandit arms, which are observable via independent samples of the corresponding *R-sub-Gaussian* distributions ν_i, $i \in [N]$. That is, each distribution ν_i satisfies the following condition for all $t \in \mathbb{R}$:

$$\mathbb{E}_{X\sim\nu_i}\left[\exp\{t(X-\mu_i)\}\right] \le \exp\{R^2t^2/2\}, \qquad (1)$$

where $\mu_i = \mathbb{E}_{X\sim\nu_i}[X]$. The goal of a learning algorithm in the thresholding bandit problem is to recover the superlevel set $\mathcal{S}_\tau = \{i : \mu_i \ge \tau\}$ from these noisy observations. The learning algorithm is allowed to run for T iterations, and at each iteration $t \in [T]$ it can select one arm $\pi_t \in [N]$ from which to receive an observation. At the end of the T iterations, the algorithm returns its estimate $\widehat{\mathcal{S}}$ of the superlevel set $\mathcal{S}_\tau$. This variant of the multi-armed bandit problem was introduced by Locatelli et al. (2016), who provided the Anytime Parameter-free Thresholding (APT) algorithm for solving the problem with matching upper and lower bounds. Mukherjee et al. (2017) and Zhong et al. (2017) have since provided algorithmic extensions to APT that incorporate variance estimates and provide guarantees in asynchronous settings. Recently, Tao et al. (2019) introduced the Logarithmic-Sample Algorithm and proved it to be instance-wise asymptotically optimal for minimizing aggregate regret.

The thresholding bandit problem can be thought of as a version of the combinatorial pure exploration (CPE) bandit problem described by Chen et al. (2014). As such, the appropriate performance loss measures the quality of the returned superlevel set estimate $\widehat{\mathcal{S}}$ at time T rather than a traditional notion of regret. We adopt a natural loss function for this setting (as done by Locatelli et al. (2016)):

$$\mathcal{L}_T = \mathbb{1}\left\{\left|(\mathcal{S}_{\tau+\varepsilon}\cap\widehat{\mathcal{S}}^c)\cup(\mathcal{S}^c_{\tau-\varepsilon}\cap\widehat{\mathcal{S}})\right| > 0\right\}, \qquad (2)$$

which for any $\varepsilon > 0$ is the indicator that at least one i such that $|\mu_i - \tau| > \varepsilon$ has been classified as being on the wrong side of the threshold.

Next, we need a notion of complexity that captures the statistical difficulty of performing the thresholding. Towards this end, we set $\Delta_i \triangleq \Delta_i^{\tau,\varepsilon} = |\mu_i - \tau| + \varepsilon$, where ε is the same quantity as in the definition of $\mathcal{L}_T$, and define the *complexity* of the thresholding problem as

$$H \triangleq H_{\tau,\varepsilon} = \sum_{i=1}^{N}\Delta_i^{-2}. \qquad (3)$$

This definition of complexity also plays a key role in the analysis of Locatelli et al. (2016). Intuitively, if there are values μ_i that are near the threshold, then the superlevel set will be "hard" to identify, and the problem complexity H will be correspondingly high. Conversely, if the values μ_i are far from the threshold, then the superlevel set will be "easy" to identify, and the problem complexity is correspondingly small.

2.2 Thresholding Graph Bandits

As discussed in the introduction, the main contribution of this paper is to present a new framework for such thresholding bandit problems where one has access to additional information about the similarities of arms. In particular, we will model this additional information as a weighted graph that describes the arm similarities. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ denote a similarity graph defined on the N arms such that each arm is a vertex in $\mathcal{V}$ and $\mathbf{W} \in \mathbb{R}^{N\times N}$ describes the weights of the edges $\mathcal{E}$ between these vertices. Let $\mathbf{L} = \mathbf{D} - \mathbf{W}$ denote the graph Laplacian, where $\mathbf{D} = \mathrm{diag}(\mathbf{W1})$ is a diagonal matrix containing the weighted degrees of each vertex. The graph Laplacian in this context is functionally quite similar to the precision matrix of a Gaussian graphical model defined on the same graph, where edges on the graph indicate conditional dependencies between two arms given all other arms, and the weight indicates the strength of the partial correlation.

The main idea behind leveraging this similarity graph is that, if the learning algorithm is aware of the similarity structure among arms through the graph $\mathcal{G}$, and if the rewards $\boldsymbol{\mu} = (\mu_i)_{i=1}^N$ vary smoothly among similar arms, then the learning algorithm can leverage the information sharing to avoid oversampling similar arms.

We capture the effectiveness of the graph in helping with the information sharing using two related notions of complexity. The first is $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda} = \sqrt{\boldsymbol{\mu}^\top\mathbf{L}_\lambda\boldsymbol{\mu}}$, the $\mathbf{L}_\lambda$ norm of $\boldsymbol{\mu}$, where $\mathbf{L}_\lambda = \mathbf{L} + \lambda\mathbf{I}$ for some $\lambda > 0$. It is not hard to check that this value is smaller for those $\boldsymbol{\mu}$'s that are smooth on the graph $\mathcal{G}$ (see, e.g., Ando and Zhang (2007)). The second notion of complexity, the *effective dimension*, characterizes the helpfulness of the graph itself.

Definition 2.1 (Valko et al., 2014, Def. 1)**.** For any $\gamma > 0$, $T \in \{1, 2, \ldots, N\}$, the **effective dimension** d_T of the regularized Laplacian $\mathbf{L}_\lambda$ is the largest d such that

$$(d-1)\gamma\lambda_d \le \frac{T}{\log(1+T/\gamma\lambda)}, \qquad (4)$$

where λ_d is the d-th eigenvalue of $\mathbf{L}_\lambda$ when $\lambda_1 \leq \ldots \leq \lambda_N$.

In Definition 2.1, T is the time horizon of the algorithm; if $T > N$, then one may use N instead of T on the right side of (4). $\gamma > 0$ is a free parameter that can be tuned in the algorithm design (see Section 3).

It can be checked readily that the effective dimension is no larger than N for any graph. In fact, as observed by Valko et al. (2014), for many graphs of interest the effective dimension turns out to be significantly smaller than N. As we will see in Section 3, this quantity plays a key role in capturing the effectiveness of our algorithm in leveraging the arm-similarity graph.[1]

2.3 A Non-adaptive Approach

Before introducing our algorithm for thresholding graph bandits, we first introduce a useful baseline.

Our algorithm for thresholding graph bandits has two primary components. The first of these is using the graph structure to regularize the estimate of the arm means using Laplacian regularization techniques, which have received considerable attention in recent decades (see Belkin et al., 2005; Zhu et al., 2003; Ando and Zhang, 2007). The second is an adaptive sampling strategy in the style of the Anytime Parameter-free Thresholding (APT) algorithm of Locatelli et al. (2016). In this section, we describe an algorithm which has only the first component—i.e., an algorithm which uses graph-regularized estimates of the arm means but selects which arm to sample next non-adaptively; see Algorithm 1.

Algorithm 1 Thresholding via non-adaptive graph-regularized estimation

```
 1: Input: $\tau, \varepsilon, \mathbf{L}, \gamma, T$
 2: $\mathbf{V}_0 \leftarrow \mathbf{L} + \lambda \mathbf{I}$
 3: $\widehat{\boldsymbol{\mu}}_0 \leftarrow \tau \mathbf{1}$
 4: $\mathbf{n}_0 \leftarrow \mathbf{0}$
 5: for $t$ in $1, \ldots, T$ do
 6:     Determine $\pi_t$ non-adaptively
 7:     Observe $x_t \sim \nu_{\pi_t}$
 8:     $\mathbf{V}_t \leftarrow \mathbf{V}_{t-1} + \gamma^{-1} \mathbf{e}_{\pi_t} \mathbf{e}_{\pi_t}^\top$
 9:     $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \gamma^{-1} x_t \mathbf{e}_{\pi_t}$
10:     $\widehat{\boldsymbol{\mu}}_t \leftarrow \mathbf{V}_t^{-1} \mathbf{x}_t$
11: end for
12: Output: $\widehat{\mathcal{S}} = \left\{ i : \widehat{\mu}_i^T \geq \tau \right\}$
```

At each iteration, Algorithm 1 first selects an arm to sample in a non-adaptive manner. This could be simply the selection of an arm at random or cycling through a permutation of the arms, for example.

Next, the algorithm solves the following Laplacian-regularized least-squares optimization problem for some $\gamma > 0$:

$$\widehat{\boldsymbol{\mu}}_t = \arg\min_{\boldsymbol{\mu}} \sum_{s=1}^{t} (x_s - \mu_{\pi_s})^2 + \gamma \|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}^2. \tag{5}$$

This optimization problem is known to promote solutions that are *smooth* across the graph (see Ando and Zhang, 2007). In fact, let $\mathbf{e}_i$ denote the i-th standard basis vector and recall that π_t denotes the index of the arm pulled at time t. If we define the quantities

$$\mathbf{V}_t = \mathbf{L}_\lambda + \frac{1}{\gamma} \sum_{s=1}^{t} \mathbf{e}_{\pi_s} \mathbf{e}_{\pi_s}^\top, \tag{6}$$

$$\mathbf{x}_t = \frac{1}{\gamma} \sum_{s=1}^{t} x_s \mathbf{e}_{\pi_s}, \tag{7}$$

then the above optimization problem admits a solution of the form

$$\widehat{\boldsymbol{\mu}}_t = \mathbf{V}_t^{-1} \mathbf{x}_t. \tag{8}$$

We note that this solution also corresponds to *a posteriori* estimation of $\boldsymbol{\mu}$ under a Gaussian prior with precision matrix $\mathbf{L}_\lambda$ when the distributions ν_i are Gaussian with variance R^2. The following proposition characterizes the performance of Algorithm 1.

Proposition 2.2. *If Algorithm 1 is run using a sampling strategy where every N iterations all arms are sampled, and $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda} \leq \sqrt{\frac{T}{\gamma \widetilde{H}}}$, then for $T = kN$ for any positive integer k,*

$$\mathbb{E}\left[\mathcal{L}_T\right] \leq \exp\left\{ -\frac{\gamma^2}{2R^2} \left(\sqrt{\frac{T}{\gamma \widetilde{H}}} - \|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda} \right)^2 + d_T \log\left(1 + \frac{T}{\gamma\lambda}\right) \right\}, \tag{9}$$

where $\widetilde{H} \triangleq N / \min\left\{ |\mu_i - \tau|^2 : |\mu_i - \tau| \geq \varepsilon \right\}$.

Thus with a non-adaptive algorithm, the complexity depends only on the most difficult arm (the arm with μ_i closest to the threshold). As we will see next, with our adaptive approach, the complexity and therefore the algorithmic performance can be significantly improved when there are arms further away from the threshold.

3 GrAPL

In this section, we present our algorithm for thresholding graph bandits. Our algorithm is inspired in part

[1]We also note here that the same authors proposed an improved definition of effective dimension that is even smaller and remains applicable in our setting (see Section 1.3.1 of Valko (2016)).

by the Anytime Parameter-free Thresholding (APT) algorithm of Locatelli et al. (2016), and also by the work of Valko et al. (2014), who applied Laplacian regularization to the bandit estimator through the eigenvectors of $\mathbf{L}_\lambda$. Unlike Valko et al. (2014), however, we use the Laplacian directly, and we include the tunable regularization parameter γ. We dub our algorithm the **Gr**aph-based **A**nytime **P**arameter-**L**ight thresholding algorithm (**GrAPL**); see Algorithm 2.

Algorithm 2 GrAPL

```
1: Input: $\tau, \varepsilon, \mathbf{L}, \gamma, \alpha, \lambda, T$
2: $\mathbf{V}_0 \leftarrow \mathbf{L} + \lambda \mathbf{I}$
3: $\widehat{\boldsymbol{\mu}}_0 \leftarrow \tau \mathbf{1}$
4: $\widehat{\boldsymbol{\Delta}}_0 \leftarrow \varepsilon \mathbf{1}$
5: $\mathbf{n}_0 \leftarrow \mathbf{0}$
6: for $t$ in $1, \dots, T$ do
7:     $z_i^t \leftarrow \widehat{\Delta}_i^{t-1} \sqrt{n_i^{t-1} + \alpha}\ \forall i$
8:     $\pi_t \leftarrow \arg\min_i z_i^t$
9:     Observe $x_t \sim \nu_{\pi_t}$
10:    $\mathbf{V}_t \leftarrow \mathbf{V}_{t-1} + \gamma^{-1} \mathbf{e}_{\pi_t} \mathbf{e}_{\pi_t}^\top$
11:    $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} + \gamma^{-1} x_t \mathbf{e}_{\pi_t}$
12:    $\widehat{\boldsymbol{\mu}}_t \leftarrow \mathbf{V}_t^{-1} \mathbf{x}_t$
13:    $\widehat{\Delta}_i^t \leftarrow |\widehat{\mu}_i^t - \tau| + \varepsilon\ \forall i$
14:    $\mathbf{n}_t \leftarrow \mathbf{n}_{t-1} + \mathbf{e}_{\pi_t}$
15: end for
16: Output: $\widehat{\mathcal{S}} = \left\{ i : \widehat{\mu}_i^T \geq \tau \right\}$
```

At each iteration, GrAPL performs the same estimation routine as Algorithm 1. Where it differs is in the strategy for choosing the next arm to sample. To select the arm at iteration $t+1$, we estimate our distances from the threshold via

$$\widehat{\Delta}_i^t = |\widehat{\mu}_i^t - \tau| + \varepsilon. \tag{10}$$

We then use these to compute confidence proxies

$$z_i^{t+1} = \widehat{\Delta}_i^t \sqrt{n_i^t + \alpha}, \tag{11}$$

where n_i^t is the number of times arm i has been selected up to time t, and $\alpha > 0$ is some small quantity that keeps z_i^t from being equal to zero before arm i is sampled. Finally, the algorithm selects the next arm as

$$\pi_t = \arg\min_i z_i^t, \tag{12}$$

and the next sample is drawn as $x_t \sim \nu_{\pi_t}$. The algorithm then repeats the process in the subsequent iterations until stopped at time T.

While GrAPL has three parameters—namely, α, λ, and γ—and is therefore not truly parameter-free like APT, the only parameter that needs to be tuned to the specific problem instance is γ. A value such as 10^{-3} for λ is sufficient to stabilize the linear system solving in (8) for many problems. If we wish for the algorithm to sample all arms at least once before sampling an arm twice, we can let α be some very small value, such as 10^{-8}; otherwise, we can let α be a larger value such as 1. The parameter γ is the only parameter that we might wish to choose appropriately based on the graph and the properties of $\boldsymbol{\mu}$—see Section 3.3 for a deeper discussion. However, we note that our main result in Theorem 3.1 below is valid for *any* values of α, λ, and γ.

In terms of implementation, we note that while (8) involves solving a linear system which can be expensive in general, if the graph is sparse, then there exist techniques to solve this system efficiently (in time nearly linear in the number of edges in the graph). Even if the graph is not sparse, it can be "sparsified" so that the system can be approximately solved efficiently. We refer the reader to Vishnoi (2013) for more details. We believe this approach (solving the system with $\mathbf{V}_t$ directly) significantly reduces the complexity of implementing a graph-based bandit algorithm compared to the approach of Valko et al. (2014), which requires a computation of the eigenspace of $\mathbf{L}_\lambda$. While computing a restricted eigenspace can also be done efficiently using similar techniques, *GrAPL* can be implemented in only a few lines of code using a standard solver such as the conjugate gradient method, readily available in common scientific computing packages in most programming languages. We have found such an implementation[2] fast enough for our purposes when the solver is initialized with the solution from the previous iteration. Though we do not include very large graphs in our experiments in this paper, we have successfully applied GrAPL to sparse graphs with over 100,000 vertices with no major difficulty.

3.1 Error Upper Bounds

We present a bound on the error that quantifies the extent to which GrAPL is able to leverage both the graph structure itself and the smoothness of $\boldsymbol{\mu}$ on the graph.

Theorem 3.1. *If Algorithm 2 is run on a graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{W})$ with Laplacian $\mathbf{L}$ and effective dimension d_T, and $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda} \leq \frac{1}{3M+1}\sqrt{\frac{T}{\gamma H}}$, then*

$$\mathbb{E}\left[\mathcal{L}_T\right] \leq \exp\Bigg\{ -\frac{\gamma^2}{2R^2}\left(\frac{1}{3M+1}\sqrt{\frac{T}{\gamma H}} - \|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}\right)^2 + d_T \log\left(1 + \frac{T}{\gamma\lambda}\right)\Bigg\}, \tag{13}$$

[2]See `https://github.com/dlej/grapl`.

where $M \triangleq \max\left\{\sqrt{\alpha/\gamma\lambda}, \sqrt{1+\alpha}\right\}$.

Remark 3.2. While one must exercise caution when comparing upper bounds, we note that the primary difference between the performance bounds of Algorithm 1 and GrAPL is in the complexity quantities. The relationship between these two is given by

$$\widetilde{H} \geq \sum_{i=1}^{N} \left(\max\left\{|\mu_i - \tau|, \varepsilon\right\}\right)^{-2} \geq H. \tag{14}$$

That is, in the worst case, where all values μ_i are close to the threshold τ, we expect both Algorithm 1 and GrAPL to perform similarly, but when there are only a few values μ_i near τ, we expect GrAPL to have a significant advantage.

Remark 3.3. We can decompose T as $T = T_0 + T_1$, where

$$T_0 = \gamma H (3M+1)^2 \|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}^2 \tag{15}$$

is the iteration at which the condition for Theorem 3.1 is met, and T_1 is the number of iterations after T_0. Then for $T_1 \geq 8T_0$, the right-hand side of (13) can be upper bounded by

$$\exp\left\{-\frac{\gamma T_1}{4(3M+1)^2 R^2 H} + d_T \log\left(1 + \frac{T}{\gamma\lambda}\right)\right\}. \tag{16}$$

While this quantity is controllable by the parameter γ, this control is limited by the the dependence of T_0 on γ. However, with smaller values of $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}$—that is, a *smoother* graph signal—we may realize the faster convergence rates associated with larger values of γ.

Remark 3.4. If we consider the two summands in the exponent of (16), one of the form $-\Theta(T)$ and the other of the form $\Theta(d_T \log T)$, then we can define the *critical iteration* T_{crit} as the iteration at which point the first of these terms begins to dominate and the bound begins to rapidly decay with T. Specifically, T_{crit} is the iteration at which these two terms are equal in magnitude. If we allow the notation $\widetilde{\Theta}(\cdot)$ to absorb logarithmic factors, we have that $T_{\text{crit}} = \widetilde{\Theta}(d_T)$. This is already a significant improvement over the standard thresholding bandit problem, where every arm must be drawn at least once, so $T_{\text{crit}} = \widetilde{\Theta}(N)$.

Remark 3.5. The quantity $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}$ can be considered with respect to any reference offset used to estimate $\widehat{\boldsymbol{\mu}}$. For example, if we replaced (8) and (7) with

$$\widehat{\boldsymbol{\mu}}_t = \mathbf{V}_t^{-1}\mathbf{x}_t + \tau\mathbf{1} \tag{17}$$

$$\mathbf{x}_t = \frac{1}{\gamma}\sum_{s=1}^{t}(x_s - \tau)\mathbf{e}_{\pi_s}, \tag{18}$$

then the $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}$ quantities in the above bound would be replaced by $\|\boldsymbol{\mu} - \tau\mathbf{1}\|_{\mathbf{L}_\lambda}$.

3.2 Optimality

3.2.1 Oracle Sampling Strategy

Consider an oracle algorithm that uses the same estimation strategy as Algorithm 1 and GrAPL but has access to the values of $|\mu_i - \tau|$ and need only identify the sign of $\mu_i - \tau$. Instead of a non-adaptive sampling strategy, let this algorithm sample according to its knowledge of $|\mu_i - \tau|$. For such an algorithm, if we relax the notion of sampling to allow the algorithm to make non-integer sample allocations according to an allocation rule $\boldsymbol{\beta}$ (obeying $\beta_i \geq 0$ and $\sum_i \beta_i = 1$) such that $n_i^t = \beta_i t$, we obtain the following result.

Proposition 3.6. *For the oracle algorithm with sampling allocation* $\boldsymbol{\beta}$*, if* $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda} \leq \sqrt{\frac{T}{\gamma H_*}}$*, then*

$$\inf_{\boldsymbol{\beta}} \mathbb{E}\left[\mathcal{L}_T\right] \leq \exp\left\{-\frac{\gamma^2}{2R^2}\left(\sqrt{\frac{T}{\gamma H_*}} - \|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}\right)^2 + d_T \log\left(1 + \frac{T}{\gamma\lambda}\right)\right\}, \tag{19}$$

where $H_* \triangleq \sum_{j:|\mu_j - \tau| \geq \varepsilon} |\mu_j - \tau|^{-2}$.

We note the similarity between H and H_*. Using the fact that $|\mu_i - \tau| + \varepsilon \leq 2|\mu_i - \tau|$ for $|\mu_i - \tau| \geq \varepsilon$, we can relate the two by

$$4H \geq H_* \geq H - \varepsilon^{-2} N_{\text{small}}, \tag{20}$$

where $N_{\text{small}} = |\{i : |\mu_i - \tau| < \varepsilon\}|$. So, except in cases where there are many values μ_i that are near the threshold, the performance upper bound of GrAPL matches that of the oracle algorithm. However, in cases where there are many values μ_i that are within ε of the threshold, the oracle algorithm can have significantly lower complexity.

3.2.2 Lower Bound for Disconnected Cliques

Consider the following family of graphs of size N consisting of D disconnected K-cliques and associated graph signals $\boldsymbol{\mu}$ such that for each arm i belonging to clique j, $\mu_i = \mu_j$. For this family of graphs and signals, the thresholding graph bandit problem reduces to the thresholding bandit problem on D independent arms with complexity $H' \triangleq \sum_{j=1}^{D}(|\mu_j - \tau| + \varepsilon)^{-2} = H/K$. This gives us the following lower bound from Locatelli et al. (2016):

$$\mathbb{E}\left[\mathcal{L}_T\right] \geq \exp\left\{-\frac{3KT}{R^2 H} - 4\log(12(\log(T)+1)N)\right\}. \tag{21}$$

For the lower bound, then, $T_{\text{crit}} = \widetilde{\Theta}(R^2 H/K)$.

For this family of graphs, the graph Laplacian consists of a matrix with D blocks of the form $K\mathbf{I}_K - \mathbf{J}_K$, where $\mathbf{J}_K$ is the $K \times K$ matrix of all ones. Therefore, the eigenvalues of $\mathbf{L}_\lambda$ are λ with multiplicity D and $K + \lambda$ with multiplicity $N - D$. Thus, the effective dimension is the larger of

$$\min\left\{D, \left\lfloor 1 + \frac{T}{\gamma\lambda \log(1 + T/\gamma\lambda)} \right\rfloor\right\}$$

and

$$\min\left\{N, \left\lfloor 1 + \frac{T}{\gamma(K+\lambda) \log(1 + T/\gamma\lambda)} \right\rfloor\right\}.$$

For any desired time horizon (e.g., $T \leq 10,000$), for sufficiently small λ, this will result in $d_T \leq D$. We also note that for this class of signals, $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}^2 = \lambda\|\boldsymbol{\mu}\|_2^2$, so for sufficiently small λ, the bound in Theorem 3.1 holds for all T.

Considering the form of our upper bound in (16), we have for this problem class that $T_{\text{crit}} = \widetilde{\Theta}(DR^2H/\gamma) = \widetilde{\Theta}(NR^2H/\gamma K)$. So, considering a fixed N and γ, we can say that GrAPL has optimal T_{crit} (up to logarithmic factors) with respect to R, H, and K (equivalently, D) for this family of graphs and signals. With $\gamma = N$, this rate would also be optimal with respect to N if it were not for the condition in (15).

3.2.3 Linear Bandits

As pointed out by Valko et al. (2014), if $\boldsymbol{\mu}$ lies in the span of D eigenvectors of $\mathbf{L}$, then the graph bandit problem reduces to the problem of thresholding *linear bandits* (Auer, 2003). Results from the *best arm identification* problem in linear bandits (Soare et al., 2014; Tao et al., 2018), another example of pure exploration bandits, suggest that the optimal sample complexity is linear in the underlying dimension D. In the above example with graphs consisting of D cliques, signal $\boldsymbol{\mu}$ lies in the span of the D eigenvectors corresponding to the smallest eigenvalues of $\mathbf{L}$, and so our result that $T_{\text{crit}} = \widetilde{\Theta}(D)$ in this setting is consistent with results from linear bandits.

3.3 Choice of Regularization Parameter

GrAPL has a free parameter γ which can be tuned to optimize T_{crit}, which we discuss in this section. T_{crit} will be on the order of the larger of T_0 and T_1, so to optimize T_{crit}, we must fix T_0 and T_1 to be of the same order. Here, we simply set $T_1 = 8T_0$. Then our optimal choice of γ is that which satisfies (15) and

$$\frac{\gamma T_1}{4(3M+1)^2R^2H} = d_{T_0+T_1} \log\left(1 + \frac{T_0 + T_1}{\gamma\lambda}\right).$$

After some algebra, we obtain

$$\gamma^* = \frac{2R}{\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}} \sqrt{d' \log\left(1 + \frac{9H(3M+1)^2\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}^2}{\lambda}\right)}, \tag{22}$$

where d', the effective dimension at time $T_0 + T_1$ for this choice of γ, is the largest d such that

$$(d-1)\lambda_d \leq \frac{9H(3M+1)^2\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}^2}{\log\left(1 + \frac{9H(3M+1)^2\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}^2}{\lambda}\right)}.$$

As we would expect, the smoother the graph signal is (smaller $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}$) and the larger the amount of noise, the larger γ (the more smoothing) we will require. All together, this gives us $T_{\text{crit}} = \widetilde{\Theta}(\sqrt{d'}R\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda}H)$. In the worst case, when the graph structure is unhelpful (i.e., when $d' = N$) and the signal is not smooth on the graph (i.e., $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda} = \Theta(\sqrt{N})$), this gives T_{crit} a linear dependence on N, as we would expect. On the other hand, in the setting of D cliques, where for sufficiently small λ we can consider $\|\boldsymbol{\mu}\|_{\mathbf{L}_\lambda} = \Theta(\sqrt{D})$, we again obtain $T_{\text{crit}} = \widetilde{\Theta}(D)$.

4 EXPERIMENTS

In experiments on both artificial and real data we demonstrate the advantage of GrAPL over the APT algorithm of Locatelli et al. (2016), which does not utilize the graph information, and over Algorithm 1, which uses non-adaptive random arm sampling. We demonstrate that exploiting the graph structure can significantly reduce the number of samples necessary to obtain a good estimate of the superlevel set, and that the adaptive arm selection rule of GrAPL further reduces the number of samples necessary over non-adaptive sampling with the same graph-regularized estimator.

4.1 Stochastic Block Model

In our first experiment, we let $N = 1000$ and sample an unweighted, undirected graph from a stochastic block model with two communities of size $N/2$, with within-community edge probability $\log(N/2)/(N/2)$ and between-community edge probability $\log(N/2)/(N/2)^{3/2}$. We let

$$\mu_i = \begin{cases} 1 & i \leq N/2 \\ -1 & \text{otherwise,} \end{cases}$$

and we make the distribution of each arm Gaussian with $\sigma = 2$. For GrAPL, we let $\lambda = 10^{-3}$ and $\alpha = 1$. With $\tau = 0$ and $\varepsilon = 0.01$, we run the algorithms for

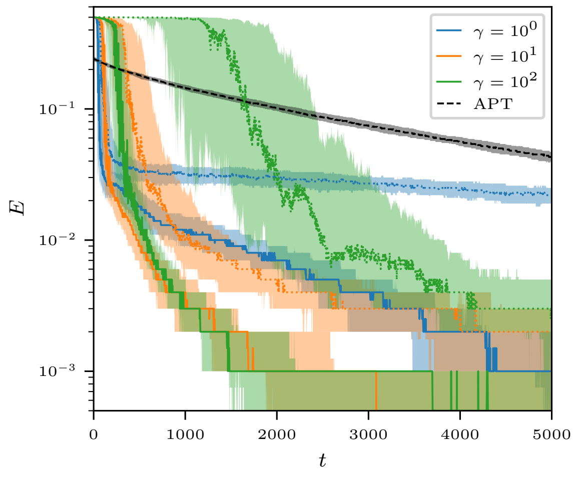

Figure 1: Misclassification error E vs. iteration t on the stochastic block model problem for GrAPL (solid), APT (dashed), and Algorithm 1 (dotted). Lines indicate the median error, and shaded areas around the lines indicate the interquartile range. Solid and dotted lines of the same color use the same value of γ for GrAPL and Algorithm 1, respectively.

$T = 5000$ iterations and compute the misclassification error E at each iteration t, defined as

$$E = \frac{\left|(\mathcal{S}_{\tau+\varepsilon} \cap \widehat{\mathcal{S}}^c) \cup (\mathcal{S}^c_{\tau-\varepsilon} \cap \widehat{\mathcal{S}})\right|}{\left|\mathcal{S}_{\tau+\varepsilon} \cup \mathcal{S}^c_{\tau-\varepsilon}\right|}. \tag{23}$$

Figure 1 shows the median misclassification error for each algorithm and choice of γ over 100 trials along with the interquartile range. We note that APT is initialized with an additional $2N = 2000$ samples before its first iteration, so for APT the actual number of samples collected is higher than the iteration counter. Both GrAPL and Algorithm 1 (for sufficiently large γ) are able to exploit the graph structure and converge to the correct superlevel set much more quickly than APT. However, consistently across values of γ, GrAPL converges in turn much more quickly than its non-adaptive counterpart. In particular, GrAPL makes significant gains in early iterations and appears to be more robust to the choice of γ. We also computed γ^* according to (22) for this problem and found the average γ^* to be 28.72 with a standard deviation of 1.15 over 100 trials, which agrees with the good performance of GrAPL with $\gamma = 10$ and $\gamma = 100$.

4.2 Small-World Graph

In our next experiment, we again let $N = 1000$ and sample small-world graphs according to the model of Newman and Watts (1999) with new-edge probability 0.01 and ring initialized with 4 neighbors. To generate our smooth signal, we first generate an i.i.d. Gaussian vector $\mathbf{y} \in \mathbb{R}^N$ and compute

$$\boldsymbol{\mu}_0 = (\mathbf{L} + \mathbf{I}/N^2)^{-1}\mathbf{y},$$

which we then normalize to have zero median and standard deviation 0.2. The multiplication by $(\mathbf{L} + \mathbf{I}/N^2)^{-1}$ serves essentially to project $\mathbf{y}$ onto the eigenspace of $\mathbf{L}$ corresponding to its smallest eigenvalues, which vary smoothly along the graph. Following this, we obtain $\boldsymbol{\mu}$ by adding 0.5 to the signal and clipping the values to be between 0 and 1. The distribution of each arm is Bernoulli with probability μ_i. With $\tau = 0.5$ and $\varepsilon = 0.01$, the problem is quite difficult.

Figure 2 shows the misclassification error for this problem when the algorithms are run over 100 trials for $T = 5000$ iterations. As before, we show the median error and interquartile range. For GrAPL, we let $\lambda = 10^{-3}$ and $\alpha = 10^{-8}$, and we estimate $\widehat{\boldsymbol{\mu}}$ with respect to the offset τ as described in Remark 3.5. On this much more difficult problem, we have selected a wider range of values for γ. Here we again see that although with the best choice of γ the advantage of GrAPL is only slight over Algorithm 1, GrAPL is much more robust to the choice of γ, and for poorly chosen γ the non-adaptive algorithm provides almost no advantage over APT. We found the average γ^* to be 227.9 with a standard deviation of 50.9 over 100 trials for this problem, which agrees with our finding the best performance at $\gamma = 100$. Lastly, we note that an artifact of the choice of very small α is that there is a spike in error around $t = N$ which corresponds to GrAPL prioritizing sampling each arm at least once over the adaptive strategy.

4.3 Political Blogs

In our experiment on real-world data, we use the political blogs graph from Adamic and Glance (2005). The vertices in the graph correspond to political blogs commenting on US politics around the time of the 2004 U.S. presidential campaign, and edges denote links from one blog to another. The signal $\boldsymbol{\mu}$ associated with this graph is

$$\mu_i = \begin{cases} 1 & \text{blog } i \text{ is conservative-leaning} \\ 0 & \text{blog } i \text{ is liberal-leaning.} \end{cases}$$

We make the edges undirected and set the edge weight equal to the total number of links from one blog to the other, and then take the largest connected component, which contained 1222 blogs. The problem we simulate then is that we would like to identify which of these blogs are conservative and liberal without actually having to visit and read each blog (expensive sampling), and we have access to this additional graph

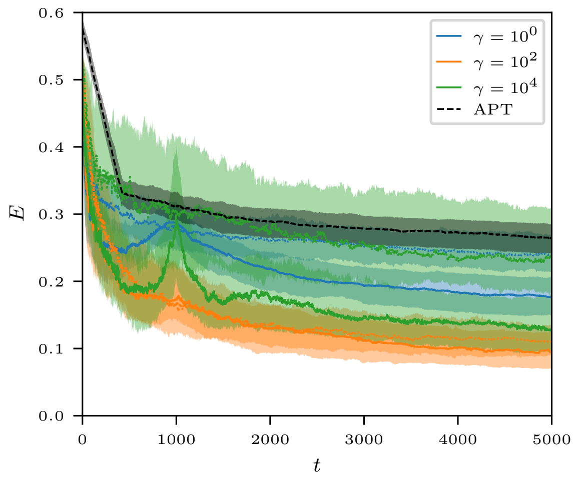

Figure 2: Misclassification error E vs iteration t on the small-world graph problem for GrAPL (solid), APT (dashed), and Algorithm 1 (dotted). Lines indicate the median error, and shaded areas around the lines indicate the interquartile range. Solid and dotted lines of the same color use the same value of γ for GrAPL and Algorithm 1, respectively.

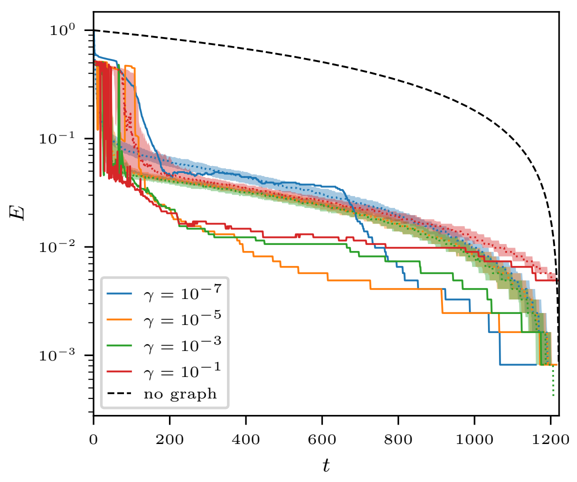

Figure 3: Misclassification error E vs iteration t on the political blogs problem for GrAPL (solid), Algorithm 1 (dotted), and using no graph (dashed). For Algorithm 1, lines indicate median error, and shaded areas around the lines indicate the interquartile range. Solid and dotted lines of the same color use the same value of γ.

information (and cheap computation compared to the time it would take to visit a blog). We make the distribution of each arm non-random and let the algorithms take at most N samples. Since APT requires $2N$ samples for initialization, we do not compare against APT.

Figure 3 shows the misclassification error for $\tau = 0.5$ and $\varepsilon = 0.01$, with median error and interquantile range over 100 trials for Algorithm 1. We run GrAPL with $\lambda = 10^{-3}$ and $\alpha = 10^{-8}$, using offset τ, and vary γ, but we do not run repeated trials since the observations are non-random. The results are similar to before, in that using the graph structure provides much better results than not using the graph, and in that we see GrAPL consistently outperforming Algorithm 1. For instance, with $\gamma = 10^{-5}$, GrAPL is able to reach 1% error at $t \approx 400$, while its random counterpart over the majority of trials does not do the same until $t > 1000$. We would expect the optimal γ to be the smallest γ possible based on (22), since there is no noise in the problem. However, for $\gamma = 10^{-7}$, floating point rounding begins to become an issue—effectively, there is a small nonzero amount of noise due to rounding—and the performance of GrAPL is worse than with the larger values of γ.

5 CONCLUDING REMARKS

In this paper we have introduced a new paradigm of online sequential decision making that we call Thresholding Graph Bandits, where the main objective is the identification of the superlevel set of arms whose means are above a given threshold in a multi-armed bandit setting. Importantly, in our framework, we have supposed that we have access to a graph that encodes the similarity between the arms. We have developed GrAPL, a novel algorithm for this thresholding graph bandits problem, along with theoretical results that show the relationship between the misclassification rate of GrAPL, the number of arm pulls, the graph structure, and the smoothness of the reward function with respect to the given graph. We have also demonstrated that GrAPL is optimal in terms of the number of arm pulls, the statistical hardness, and the dimensionality of the problem. Finally, we have confirmed our theoretical results via experiments on synthetic and real data, highlighting the significant gains to be had in leveraging the graph information with an adaptive algorithm.

Acknowledgements

This work was supported by NSF grants CCF-1911094, IIS-1838177, and IIS-1730574; ONR grants N00014-18-12571 and N00014-17-1-2551; AFOSR grant FA9550-18-1-0478; DARPA grant G001534-7500; and a Vannevar Bush Faculty Fellowship, ONR grant N00014-18-1-2047.

References

Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári. Improved algorithms for linear stochastic bandits. In

Advances in Neural Information Processing Systems 24, pages 2312–2320, 2011.

L. A. Adamic and N. Glance. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, pages 36–43. ACM, 2005.

R. K. Ando and T. Zhang. Learning on graph with laplacian regularization. In *Advances in Neural Information Processing Systems 19*, pages 25–32, 2007.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2003.

M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 17–24, 2005.

S. Chen, T. Lin, I. King, M. R. Lyu, and W. Chen. Combinatorial pure exploration of multi-armed bandits. In *Advances in Neural Information Processing Systems 27*, pages 379–387, 2014.

R. Combes, A. Proutiere, D. Yun, J. Ok, and Y. Yi. Optimal rate sampling in 802.11 systems. In *Proceedings of IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 2760–2767, 2014.

A. Gotovos, N. Casati, G. Hitz, and A. Krause. Active learning for level set estimation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 1344–1350, 2013.

A. Locatelli, M. Gutzeit, and A. Carpentier. An optimal algorithm for the thresholding bandit problem. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1690–1698, 2016.

S. Mukherjee, N. K. Purushothama, N. Sudarsanam, and B. Ravindran. Thresholding bandits with augmented UCB. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 2515–2521, 2017.

M. E. Newman and D. J. Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4-6):341–346, 1999.

M. Soare, A. Lazaric, and R. Munos. Best-arm identification in linear bandits. In *Advances in Neural Information Processing Systems 27*, pages 828–836. 2014.

N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on Machine Learning*, pages 1015–1022, 2010.

C. Tao, S. Blanco, and Y. Zhou. Best arm identification in linear bandits with linear dimension dependency. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4877–4886, 2018.

C. Tao, S. Blanco, J. Peng, and Y. Zhou. Thresholding bandit with optimal aggregate regret. In *Advances in Neural Information Processing Systems 32*, pages 11664–11673. 2019.

M. Valko. *Bandits on graphs and structures*. habilitation thesis, École normale supérieure de Cachan, 2016.

M. Valko, R. Munos, B. Kveton, and T. Kocák. Spectral bandits for smooth graph functions. In *Proceedings of the 31st International Conference on Machine Learning*, pages 46–54, 2014.

S. S. Villar, J. Bowden, and J. Wason. Multi-armed bandit models for the optimal design of clinical trials: benefits and challenges. *Statistical science: a review journal of the Institute of Mathematical Statistics*, 30(2):199–215, 2015.

N. K. Vishnoi. Lx=b. Laplacian solvers and their algorithmic applications. *Foundations and Trends® in Theoretical Computer Science*, 8(1–2):1–141, 2013.

J. Zhong, Y. Huang, and J. Liu. Asynchronous parallel empirical variance guided algorithms for the thresholding bandit problem. *arXiv preprint arXiv:1704.04567*, 2017.

X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*, pages 912–919, 2003.