Laplacian-Regularized Graph Bandits: Algorithms and Theoretical Analysis

Kaige Yang
University College London

Xiaowen Dong
University of Oxford

Laura Toni
University College London

Abstract

We consider a stochastic linear bandit problem with multiple users, where the relationship between users is captured by an underlying graph and user preferences are represented as smooth signals on the graph. We introduce a novel bandit algorithm where the smoothness prior is imposed via the random-walk graph Laplacian, which leads to a single-user cumulative regret scaling as $\tilde{\mathcal{O}}(\Psi d\sqrt{T})$ with time horizon T, feature dimensionality d, and the scalar parameter $\Psi \in (0, 1)$ that depends on the graph connectivity. This is an improvement over $\tilde{\mathcal{O}}(d\sqrt{T})$ in **LinUCB** [Li et al., 2010], where user relationship is not taken into account. In terms of network regret (sum of cumulative regret over n users), the proposed algorithm leads to a scaling as $\tilde{\mathcal{O}}(\Psi d\sqrt{nT})$, which is a significant improvement over $\tilde{\mathcal{O}}(nd\sqrt{T})$ in the state-of-the-art algorithm **Gob.Lin** [Cesa-Bianchi et al., 2013]. To improve scalability, we further propose a simplified algorithm with a linear computational complexity with respect to the number of users, while maintaining the same regret. Finally, we present a finite-time analysis on the proposed algorithms, and demonstrate their advantage in comparison with state-of-the-art graph-based bandit algorithms on both synthetic and real-world data.

1 Introduction

In the classical multi-armed bandit (MAB) problem, an agent takes sequential actions, choosing one arm out of the k available ones and it receives an instantaneous payoff from the chosen arm only. The goal of the agent is to learn an action policy that maximizes the cumulative payoff over a course of T rounds [Robbins, 1952]. MAB problems formalize a trade-off between exploration and exploitation, and a particular solution is imposing the principle of optimism in face of uncertainty. Specifically, the agent assigns to each arm an index called the upper confidence bound (UCB) that with high probability is an overestimate of the unknown payoff, and selects the arm with the highest index.

Many variants of the basic MAB problem have been intensively studied, motivated by real-world applications such as ads placement and recommender systems. In the stochastic linear bandit [Auer, 2002], at each round, the agent receives a hint before taking the decision. Specifically, before choosing the arm, the agent is informed of a feature vector $\mathbf{x} \in \mathbb{R}^d$ associated with each arm, referred to as the 'context'. The payoff associated with each arm is modeled as a noisy linear function of $\mathbf{x}$ with an unknown coefficient vector $\boldsymbol{\theta} \in \mathbb{R}^d$ perturbed by a noise term η, i.e, $y = \mathbf{x}^T\boldsymbol{\theta} + \eta$, where the agent needs to learn $\boldsymbol{\theta}$ based on the context-payoff pair $\{\mathbf{x}, y\}$ and select an arm accordingly. This problem has been well understood in the literature and many studies have already proposed asymptotically optimal algorithms [Auer, 2002, Dani et al., 2008, Agrawal and Goyal, 2013, Chapelle and Li, 2011, Lattimore and Szepesvari, 2016].

The problem is less understood in the case of multiple users, as opposed to a single user, where we assume a central agent needs to select arms for multiple users in a sequential fashion. In this paper, we are interested in the setting where there are n users sharing the same set of arm choices $\mathcal{D}$ containing m arms. The agent faces a set of n independent instance of bandit characterized by an unknown $\boldsymbol{\theta}_i, i \in \{1, 2, ..., n\}$ specific to each user. At each round, one user out of n is selected uniformly at random, the agent then selects one arm from $\mathcal{D}$ for the user and receives an instantaneous pay-

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

off associated with the selected arm and the user. The overall goal is to minimize the cumulative regret (or equivalently, maximize the cumulative payoffs), which is defined as the sum of instantaneous regret experienced by the agent over a finite time horizon T.

In this setting, naively implementing bandit algorithms on each user independently will results in a cumulative regret that scales linearly with the number of users n. This is clearly infeasible in case of a large number of users. In many cases, however, the users are related in some way and this can be represented by a network (or graph) that encapsulates important additional source of information, such as similarities among users in terms of their preferences (user feature vectors). Exploiting this structure can mitigate the scalability problem. The key setup is therefore to construct a graph where each node represents a user and the edges identify the affinity between users. In real-world applications, such a graph can be a social network of users. This idea leads to a series of work on the so-called graph-based bandit problem [Cesa-Bianchi et al., 2013, Gentile et al., 2014, Liu et al., 2018, Vaswani et al., 2017].

Despite the previous effort, several important limitations still remain to be addressed. First, the graph Laplacian matrix is commonly used in graph-based algorithms, but the justification of its usage remains insufficient (and as to which version of the graph Laplacian leads to optimal policies). As a consequence, the advantage of graph-based bandit is largely shown empirically in previous works, without rigorous theoretical analysis. Furthermore, scalability remains a serious limitation of such algorithms. Involving user graph into bandit algorithms typically results in a computational complexity that scales quadratically with the number of users, which is clearly infeasible in case of large number of users. In this paper, we address the above limitations with the following main contributions:

- We propose a bandit algorithm **GraphUCB** based on the random-walk graph Laplacian, and show its theoretical advantages over other graph Laplacian matrices in reducing cumulative regret. We demonstrate empirically that **GraphUCB** outperforms state-of-the-art graph-based bandit algorithms in terms of cumulative regret.
- As a key ingredient of the proposed algorithm, we derive a novel UCB representing the single-user bound while embedding the graph structure, which reduces the size of the confidence set, in turn leading to lower regret;
- To improve scalability, we further propose a simplified algorithm **GraphUCB-Local** whose complexity scales linearly with respect to the number of users, yet still holding the same regret upper bound as **GraphUCB**;
- Finally, we derive a finite-time analysis on both algorithms and show a lower regret upper bound than other state-of-the-art graph-based bandit algorithms.

2 Related work

Graph-based bandit algorithms can be roughly categorized as: *i*) topology-based bandits, where the graph topology itself is exploited to improve learning performance, and *ii*) spectral bandits, where the user feature vectors $\boldsymbol{\theta}$ are modeled as signals defined on the underlying graph, whose characteristics are then exploited in the graph spectral domain via tools provided by graph signal processing [Shuman et al., 2013] to assist learning.

In topology-based bandits, the key intuition is to achieve dimensionality reduction in the user space by exploiting the graph topology. Specifically, users can be clustered based on the graph topology and a per-cluster feature vector can be learned, substantially reducing the dimensionality of the problem as opposed to the case in which one vector is learned per user. For example, [Gentile et al., 2014] clusters users based on the connected components of the user graph, and [Li et al., 2016] generalizes it to consider both the user graph and item graph. On the other hand, [Yang and Toni, 2018] makes use of community detection techniques on graphs to find user clusters. More broadly, in the spirit of dimensionality reduction, even without constructing an explicit user graph, the work in [Korda et al., 2016] proposes a distributed clustering algorithm while [Nguyen and Lauw, 2014] applies *k-means* clustering to the user features. Despite the differences in the proposed techniques, these studies share two common drawbacks: 1) the learning performance depends on the clustering algorithm being used, which tends to be expensive for large-scale graphs; 2) learning a per-cluster (and not per-user) feature vector means ignoring the subtle difference between users within the same cluster. In short, clustering can reduce the dimensionality of the user space, but it does not necessarily preserve key users characteristics. To achieve both goals simultaneously, there is a need for a proper mathematical framework able to incorporate the user relationship into learning in a more direct way.

On the spectral bandit side, the strong assumption that users can be grouped into clusters is relaxed; users are assumed to be similar with their neighbors in the graph and such similarity is reflected by the weight of graph edges. [Wu et al., 2016] employs a graph

Laplacian-regularized estimator, which promotes similar feature vectors for users connected in the graph. In their setting, however, each arm is selected by all users jointly. This work results in a network regret scaling with $\tilde{\mathcal{O}}(dn\sqrt{T})$. [Vaswani et al., 2017] casts the same estimator as GMRF (Gaussian Markov Random Filed) and proposes a Thompson sampling algorithm, leading to a much simpler algorithmic implementation without a UCB evaluation. However, the regret bound remains $\tilde{\mathcal{O}}(dn\sqrt{T})$. To the best of our knowledge, an efficient algorithm able to address the multi-user MAB problem with a sub-linear regret bound is still missing.

A proper bound and mathematical derivation of spectral MAB are provided in [Valko et al., 2014], which represents the payoffs of arms as smooth signals on a graph with the arms being the nodes. Specifically, the arm features $\mathbf{x}$ are modeled as eigenvectors of the graph Laplacian and the sparsity of such eigenvectors is exploited to reduce the dimensionality of $\mathbf{x}$, to a so-called 'effective dimension' term $\tilde{d}$. This work shows an improved regret bound $\tilde{\mathcal{O}}(\tilde{d}\sqrt{T})$ where $\tilde{d}$ is significant less than d in **LinUCB** [Abbasi-Yadkori et al., 2011]. While interesting, the proposed solution applies to the single-user with high-dimensional arm set. Whereas, in our setting, the dimensionality issue is caused by the large number of users, leading to a completely different mathematical problem.

Among these works on graph bandit, the one that is most similar to our work in terms of problem definition and proposed solution is [Cesa-Bianchi et al., 2013]. In [Cesa-Bianchi et al., 2013], the graph is exploited such that each user shares instantaneous payoff with neighbors, which is promoted by a Laplacian-regularized estimator. This implicitly imposes smoothness among the feature vectors of users, resulting in the estimate of similar feature vectors for users connected by edges with strong weights in the graph.

In our paper, we proposed the **GraphUCB** algorithm that builds on and improves the work of **Gob.Lin** in a number of important ways:

- First, **Gob.Lin** employs the combinatorial Laplacian as a regularizer, whereas our algorithm **GraphUCB** makes use of the random-walk graph Laplacian. We prove theoretically that the combinatorial Laplacian results in a cumulative regret scaling with the number of users, which could be large. However, random-walk graph Laplacian overcomes this serious drawback and yields a sub-linear regret with the number of users.
- Second, UCB used in **Gob.Lin** results in a cumulative regret that scales with $\tilde{\mathcal{O}}(nd\sqrt{T})$ which is worse than **LinUCB** $\tilde{\mathcal{O}}(d\sqrt{nT})$. We propose a new UCB that leads to a cumulative regret scaling with $\tilde{\mathcal{O}}(\Psi d\sqrt{nT})$ where $\Psi \in (0,1)$.
- Finally, the computational complexity of **Gob.Lin** is quadratic with respect to the number of users. Our simplified algorithm **GraphUCB-Local** scales linearly with the number of users, and at the same time enjoys the same regret bound as **GraphUCB**. This significantly improves the scalability of the proposed graph bandit algorithm.

3 Setting

We consider a linear bandit problem with m arms and n users. We denote by $\mathcal{U}$ the user set with cardinality $|\mathcal{U}| = n$ and by $\mathcal{D}$ the arm set with $|\mathcal{D}| = m$. Each arm is described by a feature vector $\mathbf{x} \in \mathbb{R}^d$, while each user is described by a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$, with d being the dimension of both vectors. The affinity between users is encoded by an undirected and weighted graph $\mathcal{G} = (V, E)$, where $V = \{1, 2, .., n\}$ represents the node set for n users and E represents the edge set. The graph $\mathcal{G}$ is known a priori and identified by its adjacency matrix $\mathbf{W} \in \mathbb{R}^{n\times n}$, where $W_{ij} = W_{ji}$ captures the affinity between $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$. The combinatorial Laplacian of $\mathcal{G}$ is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_{i=1}^{n} W_{ii}$. The symmetric normalized Laplacian is defined as $\tilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$. In addition, the random-walk graph Laplacian is defined as $\boldsymbol{\mathcal{L}} = \mathbf{D}^{-1}\mathbf{L}$.

In our setting, the unknown user features $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_n]^T \in \mathbb{R}^{n\times d}$ are assumed to be smooth over $\mathcal{G}$. The smoothness of $\boldsymbol{\Theta}$ over graph $\mathcal{G}$ can then be quantified using the Laplacian quadratic form of any of the three Laplacian defined above. In this work, we choose the random-walk graph Laplacian $\boldsymbol{\mathcal{L}}$ because of its two unique properties $\mathcal{L}_{ii} = 1$ and $\sum_{j\neq i} \mathcal{L}_{ij} = -1$. The benefit of these properties will be clear after the introduce of our proposed bandit algorithm (see Remark 1). Mathematically, the Laplacian quadratic form based on $\boldsymbol{\mathcal{L}}$ is (see Appendix H for the derivation):

$$tr(\boldsymbol{\Theta}^T\boldsymbol{\mathcal{L}}\boldsymbol{\Theta}) = \frac{1}{4}\sum_{k=1}^{d}\sum_{i\sim j}\left(\frac{W_{ij}}{D_{ii}} + \frac{W_{ji}}{D_{jj}}\right)\left(\Theta_{ik} - \Theta_{jk}\right)^2 \tag{1}$$

where Θ_{ik} is the (i,k)-th element of $\boldsymbol{\Theta}$. The more the graph $\mathcal{G}$ reflects the similarity between users correctly, the smaller the quadratic term $tr(\boldsymbol{\Theta}^T\boldsymbol{\mathcal{L}}\boldsymbol{\Theta})$. Specifically, $tr(\boldsymbol{\Theta}^T\boldsymbol{\mathcal{L}}\boldsymbol{\Theta})$ is small when Θ_{ik} and Θ_{jk} are similar given a large weight $\frac{W_{ij}}{D_{ii}} + \frac{W_{ji}}{D_{jj}}$.

Equipped with the above notation, we now introduce the multi-user bandit problem, in which an agent needs to take sequential decisions (e.g., recommendations) for a multitude of users appearing over time. At each

time $t = 1, \ldots, T$, the agent is informed about the user i_t to serve, with the user being selected uniformly at random from the user set $\mathcal{U}$. Then, the agent selects an arm $\mathbf{x}_t \in \mathcal{D}$ to be recommended to user i_t. Upon this selection, the agent observes a payoff y_t, which is assumed to be generated by noisy versions of linear functions of the users and item vectors. Namely,

$$y_t = \mathbf{x}_t^T \boldsymbol{\theta}_{i_t} + \eta_t \tag{2}$$

where the noise η_t is assumed to be σ-sub-Gaussian for any t.

The agent is informed about the graph $\mathcal{G}$ and the arm feature vectors $\mathbf{x}_a,\ a \in \{1, 2, ..., m\}$, while $\boldsymbol{\Theta}$ is unknown and needs to be inferred. The goal of the agent is to learn a selection strategy that minimizes the cumulative regret with respect to an optimal strategy, which always selects the optimal arm for each user. Formally, after a time horizon T, the cumulative (pseudo) regret is defined as:

$$R_T = \sum_{t=1}^{T} \left((\mathbf{x}_t^*)^T \boldsymbol{\theta}_{i_t} - \mathbf{x}_t^T \boldsymbol{\theta}_{i_t} \right) \tag{3}$$

where $\mathbf{x}_t$ and $\mathbf{x}_t^*$ are the arm selected by the agent and the optimal strategy at t, respectively. Note that the optimal choice depends on t as well as on the user i_t. For notation convenience in the rest of the paper, at each time step t, we use i to generally refer to the user appeared and $\mathbf{x}_t$ to represent the feature vector of the arm selected.

4 Laplacian-Regularized Estimator

To estimate the user parameter $\boldsymbol{\Theta}$ at time t, we make use of the Laplacian-regularized estimator:

$$\hat{\boldsymbol{\Theta}}_t = \arg \min_{\boldsymbol{\Theta} \in \mathbb{R}^{n \times d}} \sum_{i=1}^{n} \sum_{\tau \in \mathcal{T}_{i,t}} (\mathbf{x}_\tau^T \boldsymbol{\theta}_i - y_\tau)^2 + \alpha\ tr(\boldsymbol{\Theta}^T \boldsymbol{\mathcal{L}} \boldsymbol{\Theta}) \tag{4}$$

where $\boldsymbol{\theta}_i$ is the i-th row of $\boldsymbol{\Theta}$, $\mathcal{T}_{i,t}$ is the set of time steps at which user i is served up to time t. $\mathbf{x}_\tau$ is the feature of arm selected by the learner, y_τ is the payoff from user i at time τ, and α is the regularization parameter. Clearly, Eq. 4 is convex and can be solved via convex optimization techniques. Specifically, it has a closed form solution [Alvarez et al., 2012]:

$$vec(\hat{\boldsymbol{\Theta}}_t) = (\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^T + \alpha \boldsymbol{\mathcal{L}} \otimes \mathbf{I})^{-1} \boldsymbol{\Phi}_t \mathbf{Y}_t \tag{5}$$

where $\otimes$ is the Kronecker product, $vec(\hat{\boldsymbol{\Theta}}_t) \in \mathbb{R}^{nd}$ is the concatenation of the columns of $\hat{\boldsymbol{\Theta}}_t$, $\mathbf{I} \in \mathbb{R}^{d \times d}$ is the identity matrix, and $\mathbf{Y}_t = [y_1, y_2, ..., y_t]^T \in \mathbb{R}^t$ is the collection of all payoffs. Finally, $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, ..., \boldsymbol{\phi}_t] \in \mathbb{R}^{nd \times t}$, where $\boldsymbol{\phi}_t \in \mathbb{R}^{nd}$, is a long sparse vector indicating that the arm with feature $\mathbf{x}_t$ is selected for user i. Formally,

$$\boldsymbol{\phi}_t^T = (\ \underbrace{0, ..., 0}_{(i-1) \times d\ times}\ , \mathbf{x}_t^T, \ \underbrace{0, ...0}_{(n-i) \times d\ times}\ ). \tag{6}$$

While Eq. 4 provides the estimate of feature vectors of all users at t, i.e., $\hat{\boldsymbol{\Theta}}_t$, the agent is interested in the estimation of each single-user feature vector $\hat{\boldsymbol{\theta}}_{i,t}$. Mathematically, $\hat{\boldsymbol{\theta}}_{i,t}$ can be obtained by decoupling users in Eq. 5. This however is highly complex due to the inversion $(\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^T + \alpha \boldsymbol{\mathcal{L}} \otimes \mathbf{I})^{-1}$, which the agent needs to preform at each time step (we recall that the Laplacian is high-dimensional). We notice that the close-form solution of $\hat{\boldsymbol{\theta}}_{i,t}$ can be closely approximated via a Taylor expansion of $(\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^T + \alpha \boldsymbol{\mathcal{L}} \otimes \mathbf{I})^{-1}$, as stated in Lemma 1 and further commented and tested empirically in Appendix A.

Lemma 1. *$\hat{\boldsymbol{\Theta}}_t$ is obtained from Eq. 5, let $\hat{\boldsymbol{\theta}}_{i,t}$ be the i-th row of $\hat{\boldsymbol{\Theta}}_t$ which is the estimate of $\boldsymbol{\theta}_i$. $\hat{\boldsymbol{\theta}}_{i,t}$ can be approximated by[1]:*

$$\hat{\boldsymbol{\theta}}_{i,t} \approx \mathbf{A}_{i,t}^{-1} \mathbf{X}_{i,t} \mathbf{Y}_{i,t} - \alpha \mathbf{A}_{i,t}^{-1} \sum_{j=1}^{n} \mathcal{L}_{ij} \mathbf{A}_{j,t}^{-1} \mathbf{X}_{j,t} \mathbf{Y}_{j,t} \tag{7}$$

where $\mathbf{A}_{i,t} = \sum_{\tau \in \mathcal{T}_{i,t}} \mathbf{x}_\tau \mathbf{x}_\tau^T \in \mathbb{R}^{d \times d}$ is the Gram matrix related to the choices made by user i, $\mathcal{T}_{i,t}$ is the set of time at which user i is served up to time t, and $\mathcal{L}_{ij}$ is the (i,j)-th element in $\mathcal{L}$. $\mathbf{X}_{i,t} \in \mathbb{R}^{d \times |\mathcal{T}_{i,t}|}$ is the collection of features of arms that are selected for user i up to time t with $\{\boldsymbol{x}_\tau\}, \tau \in \mathcal{T}_{i,t}$ as columns. $\mathbf{Y}_{i,t} \in \mathbb{R}^{|\mathcal{T}_{i,t}|}$ is the collection of payoffs associated with user i up to time t, whose elements are $\{y_\tau\}, \tau \in \mathcal{T}_{i,t}$.

Proof. See Appendix A. □

4.1 Construction of Confidence Set

For the agent to balance exploration and exploration in sequential decisions, we need to quantify the uncertainty over the estimation of $\hat{\boldsymbol{\theta}}_{i,t}$. This is possible by defining a confidence set around $\hat{\boldsymbol{\theta}}_{i,t}$ based on Mahalanobis distance using its precision matrix, as commonly adopted in bandit literature [Lattimore and Szepesvári, 2018]. Let $\boldsymbol{\Lambda}_{i,t} \in \mathbb{R}^{d \times d}$ be the precision matrix of $\hat{\boldsymbol{\theta}}_{i,t}$, the confidence set is formally defined as:

$$\mathcal{C}_{i,t} = \{\boldsymbol{\theta}_{i,t} : ||\hat{\boldsymbol{\theta}}_{i,t} - \boldsymbol{\theta}_i||_{\boldsymbol{\Lambda}_{i,t}} \leq \beta_{i,t}\} \tag{8}$$

where $\beta_{i,t}$ is the upper bound of $||\hat{\boldsymbol{\theta}}_{i,t} - \boldsymbol{\theta}_i||_{\boldsymbol{\Lambda}_{i,t}}$ which is what we are interested in for the bandit algorithm.

[1] $\mathbf{A}_{i,t}$(and $\mathbf{A}_{j,t}$) is not full-rank when $|\mathcal{T}_{i,t}| < d$. To guarantee inversion, in practice we set $\mathbf{A}_{i,t} = \sum_{\tau \in \mathcal{T}_{i,t}} \boldsymbol{x}_\tau \boldsymbol{x}_\tau^T + \lambda \mathbf{I}_d$ with $\lambda = 0.01$.

With this goal in mind, we seek an expression for $\mathbf{\Lambda}_{i,t}$. Let $\mathbf{\Lambda}_t \in \mathbb{R}^{nd\times nd}$ denote the precision matrix of $vec(\hat{\mathbf{\Theta}}_t) \in \mathbb{R}^{nd}$, where $\mathbf{\Lambda}_{i,t} \in \mathbb{R}^{d\times d}$ is the i-th block matrix along the diagonal of $\mathbf{\Lambda}_t$. Defining the precision matrix of $vec(\hat{\mathbf{\Theta}}_t) \in \mathbb{R}^{nd}$ as:

$$\mathbf{\Lambda}_t = \mathbf{M}_t\mathbf{A}_t^{-1}\mathbf{M}_t \tag{9}$$

with $\mathbf{A}_t = \mathbf{\Phi}_t\mathbf{\Phi}_t^T$, $\boldsymbol{\mathcal{L}}_\otimes = \boldsymbol{\mathcal{L}} \otimes \mathbf{I}$, and $\mathbf{M}_t = \mathbf{A}_t + \alpha\boldsymbol{\mathcal{L}}_\otimes$, we have:

$$\mathbf{\Lambda}_{i,t} = \mathbf{A}_{i,t} + 2\alpha\mathcal{L}_{ii}\mathbf{I} + \alpha^2\sum_{j=1}^{n}\mathcal{L}_{ij}^2\mathbf{A}_{j,t}^{-1} \tag{10}$$

where $\mathbf{A}_{i,t}$ and $\mathbf{A}_{j,t}$ are defined in Lemma 1, and $\mathcal{L}_{ij}$ is the (i,j)-th element in $\mathcal{L}$. A detailed derivation of Eq. 10 is presented in Appendix B and Appendix C. Given Eq. 10, we can upper bound the size of the confidence set, which provides the value of $\beta_{i,t}$.

Lemma 2. *Let $\mathbf{V}_{i,t} = \mathbf{A}_{i,t} + \alpha\mathcal{L}_{ii}\mathbf{I}$, and $\mathbf{I} \in \mathbb{R}^{d\times d}$ is the identity matrix. Given a scalar $\delta \in [0,1]$, and by defining $\boldsymbol{\Delta}_i = \sum_{j=1}^{n}\mathcal{L}_{ij}\boldsymbol{\theta}_j$, the size of the confidence set defined in Eq. 8 is upper bounded with probability $1-\delta$ by $\beta_{i,t}$:*

$$\beta_{i,t} = \sigma\sqrt{2\log\frac{|\mathbf{V}_{i,t}|^{1/2}}{\delta|\alpha\mathbf{I}|^{1/2}}} + \sqrt{\alpha}||\boldsymbol{\Delta}_i||_2 \tag{11}$$

Proof. See Appendix D. □

It is worth mentioning that the bound $\beta_{i,t}$ depends on $\boldsymbol{\Delta}_i$, which reflects information contained in the graph structure. In the case of random-walk Laplacian in which $\mathcal{L}_{ii} = 1$ and $\sum_{j\neq i} -\mathcal{L}_{ij} = 1$, $\boldsymbol{\Delta}_i = \boldsymbol{\theta}_i - (\sum_{j\neq i} -\mathcal{L}_{ij}\boldsymbol{\theta}_j)$ measures the difference between user feature $\boldsymbol{\theta}_i$ and the weighted average of its neighbors. To show the effect of graph structure on $\boldsymbol{\theta}_i$, we consider two extreme cases: *a)* an empty graph[2], i.e., $\mathcal{L}_{ii} = 1$ and $\mathcal{L}_{ij} = 0$. In this case, $\boldsymbol{\Delta}_i = \boldsymbol{\theta}_i$, which recovers the confidence set used in **LinUCB** [Abbasi-Yadkori et al., 2011]; *b)* a fully connected graph with $W_{ij} = 1$ which means $\mathcal{L}_{ii} = 1$, $\mathcal{L}_{ij} = \frac{1}{n-1}$ and $\boldsymbol{\theta}_i = \boldsymbol{\theta}_j$. In this case, $\boldsymbol{\Delta}_i = \boldsymbol{\theta}_i - \frac{n-1}{n-1}\boldsymbol{\theta}_i = \mathbf{0}$, leading to a much lower bound than the one in case *a)* and with no graph structure to be exploited. In between, $\boldsymbol{\Delta}_i$ depends on the similarity between $\boldsymbol{\theta}_i$ and its neighbors $\boldsymbol{\theta}_j, j \neq i$. In general, the smoother the signal is on the graph (in the sense of a small Laplacian quadratic in Eq. 1), the lower the $||\boldsymbol{\Delta}_i||_2$. This has been empirically shown in Figure 1(a), where we depict $||\boldsymbol{\Delta}_i||_2$ as a function of the level of smoothness quantified by $tr(\mathbf{\Theta}^T\boldsymbol{\mathcal{L}}\mathbf{\Theta})$ where smaller value means smoother $\mathbf{\Theta}$ over the graph.

[2]For isolated node, we set $\mathcal{L}_{ii} = 1$.

Algorithm 1: GraphUCB

Input : α, T, $\boldsymbol{\mathcal{L}}$, δ
Initialization : For any $i \in \{1,2,...,n\}$
$\hat{\boldsymbol{\theta}}_{0,i} = \mathbf{0} \in \mathbb{R}^d$, $\mathbf{\Lambda}_{0,i} = \mathbf{0} \in \mathbb{R}^{d\times d}$,
$\mathbf{A}_{0,i} = \mathbf{0} \in \mathbb{R}^{d\times d}$, $\beta_{i,t} = 0$.
for $t \in [1,T]$ **do**
  User index i is selected
  1. $\mathbf{A}_{i,t} \leftarrow \mathbf{A}_{i,t-1} + \mathbf{x}_{t-1}\mathbf{x}_{t-1}^T$.
  2. $\mathbf{A}_{j,t} \leftarrow \mathbf{A}_{j,t-1}$, $\forall j \neq i$.
  3. Update $\mathbf{\Lambda}_{i,t}$ via Eq. 10.
  4. Select $\mathbf{x}_t$ via Eq. 12 where $\beta_{i,t}$ is defined in Eq. 11
  5. Receive the payoff y_t
  6. Update $\hat{\mathbf{\Theta}}_t$ via Eq. 4
end

Now that we have introduced $\boldsymbol{\Delta}_i$, we are ready to motivate the choice of the random-walk graph Laplacian instead of other commonly used graph Laplacians.

Remark 1. *The two unique properties $\mathcal{L}_{ii} = 1$ and $\sum_{j\neq i} -\mathcal{L}_{ij} = 1$ of the random-walk graph Laplacian $\boldsymbol{\mathcal{L}}$ ensure a bounded regret and lower regret with more similar users. The same cannot be guaranteed with the combinatorial or normalized Laplacian.*

To see this more clearly, if the combinatorial Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is used in the Laplacian-regularized estimator (Eq. 4), the term $\boldsymbol{\Delta}_i$ becomes $\boldsymbol{\Delta}_i = D_{ii}(\boldsymbol{\theta}_i + \sum_{j\neq i}\frac{W_{ij}}{D_{ii}}\boldsymbol{\theta}_j)$. This term scales linearly with D_{ii}, the degree of each user i, resulting in a regret that also scales with D_{ii} and may become rather large for densely connected graphs. On the other hand, if the symmetric normalized Laplacian $\tilde{\mathbf{L}} = \mathbf{D}^{-1}\mathbf{L}\mathbf{D}^{-1}$ is used in Eq. 4, we have $\sum_{j\neq i} -\tilde{L}_{ij} \neq 1$. It follows that $\sum_{j\neq i} -\tilde{L}_{ij}\boldsymbol{\theta}_j$ will not be a convex combination of $\boldsymbol{\theta}_j, j \neq i$, which means that there is no guarantee for $\boldsymbol{\Delta}_i$ to be located inside the Euclidean ball defined by $||\boldsymbol{\theta}_i||_2 \leq 1, \forall i \in [1,...,n]$, leading to an unbounded regret. By contrast, the two unique properties $\mathcal{L}_{ii} = 1$ and $\sum_{j\neq i} -\mathcal{L}_{ij}$ of the random-walk normlaized Laplaican $\boldsymbol{\mathcal{L}}$ ensure a bounded regret and less regret if users are similar.

5 Algorithms

We now introduce our proposed **GraphUCB** bandit algorithm, sketched in **Algorithm 1** and based on the principle of *optimism in face of uncertainty*:

$$\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{D}}\left(\mathbf{x}^T\hat{\boldsymbol{\theta}}_{i,t} + \beta_{i,t}||\mathbf{x}||_{\mathbf{\Lambda}_{i,t}^{-1}}\right). \tag{12}$$

GraphUCB is designed based on the Laplacian regularized estimator Eq. 4 and the arm selection principle

in Eq. 12. Formally, at each time t, an user index i is selected randomly from the user set $\mathcal{U}$. The algorithm first updates $\mathbf{\Lambda}_{i,t}$ based on the Gram matrix $\mathbf{A}_{i,t}$ and $\mathbf{A}_{j,t}$. Then, it selects the arm $\mathbf{x}_t$ from the arm set $\mathcal{D}$ following Eq. 12. Upon receiving the instantaneous payoff y_t, it updates the features of all users $\hat{\mathbf{\Theta}}_t$ by solving Eq. 4. The process then continues to time $t+1$, and is repeated until T. In Eq. 11 the $\mathbf{\Delta}_i$ is based on the unknown ground-truth $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$. In practice, it is replaced by its empirical counterpart $\hat{\mathbf{\Delta}}_i = \sum_{i=1}^{n} \mathcal{L}_{ij}\hat{\boldsymbol{\theta}}_{i,t}$ where $\hat{\boldsymbol{\theta}}_{i,t}$ is the i-th row of $\hat{\mathbf{\Theta}}_t$.

One limitation of **GraphUCB** is its high computational complexity. Specifically, in solving Eq. 4, the running time is dominated by the inversion $(\mathbf{\Phi}_t\mathbf{\Phi}_t^T + \alpha\boldsymbol{\mathcal{L}} \otimes \mathbf{I})^{-1}$, which is in the order $\mathcal{O}(n^2d^2)$. This is impractical when the user number n is large. Recall that in the learning setting, at each time t, only one user is selected. Thus, it suffices to only update $\hat{\boldsymbol{\theta}}_{i,t}$ (i.e., a local rather than global update). Therefore, we propose to make use of Lemma 1 instead of Eq. 4 to estimate $\hat{\boldsymbol{\theta}}_{i,t}$, which results in a significant reduction in computational complexity. Clearly, the complexity of the approximation in Lemma 1 is dominated by the inversion of $(\mathbf{A}_{i,t} + \alpha\mathcal{L}_{ii}\mathbf{I})^{-1}$ or $\mathbf{A}_{j,t}^{-1}$, which is in the order $\mathcal{O}(d^2)$. Since the approximation involves n such inversions, the total complexity is $\mathcal{O}(nd^2)$, i.e., it scales linearly (rather than quadratically) with n.

By using Lemma 1, we therefore propose a second algorithm **GraphUCB-Local** serving as a simplified version of **GraphUCB**. The only difference lies in the number of users updated per round. **GraphUCB** updates all users $\hat{\mathbf{\Theta}}_t$ via Eq. 4 (closed-form solution), while **GraphUCB-Local** only updates one user $\hat{\boldsymbol{\theta}}_{i,t}$ via Lemma 1 (approximated solution). Pseudocode of **GraphUCB-Local** is presented in Appendix E.

6 Analysis

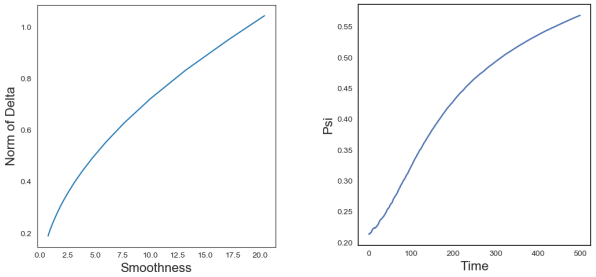
Before providing the finite-time analysis on the proposed algorithms, we define:

$$\Psi_{i,T} = \frac{\sum_{\tau\in\mathcal{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\mathbf{\Lambda}^{-1}_{i,\tau}}}{\sum_{\tau\in\mathcal{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\mathbf{V}^{-1}_{i,\tau}}} \tag{13}$$

where $\mathcal{T}_{i,T}$ is the set of time steps in which user i is served up to time T, $\mathbf{A}_{i,\tau} = \sum_{\ell\in\mathcal{T}_{i,\tau}} \boldsymbol{x}_\ell\boldsymbol{x}_\ell^T$, $\mathbf{V}_{i,\tau} = \mathbf{A}_{i,\tau} + \alpha\mathcal{L}_{ii}\mathbf{I}$ and $\mathbf{\Lambda}_{i,\tau}$ is defined as in Eq. 10. Moreover, $||\mathbf{x}_\tau||^2_{\mathbf{\Lambda}^{-1}_{i,\tau}}$ and $||\mathbf{x}_\tau||^2_{\mathbf{V}^{-1}_{i,\tau}}$ quantify the variance of predicted payoff $\hat{y}_\tau = \hat{\boldsymbol{\theta}}_{i,\tau}^T\mathbf{x}_\tau$ in the cases where the graph structured is exploited or ignored, respectively.

Lemma 3. *Let $\Psi_{i,T}$ be as defined in Eq. 13 and $||\mathbf{x}_\tau||_2 \leq 1$ for any $\tau \leq T$, then:*

$$\Psi_{i,T} \in (0, 1)$$



(a) (b)

Figure 1: (a) $||\mathbf{\Delta}_i||_2$ vs. smoothness ($tr(\mathbf{\Theta}^T\boldsymbol{\mathcal{L}}\mathbf{\Theta})$), (b) $\Psi_{i,T}$ vs. time.

and as $T \to \infty$, $\Psi_{i,T} \to 1$. This implies:

$$\sum_{\tau\in\mathcal{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\mathbf{\Lambda}^{-1}_{i,\tau}} \leq \sum_{\tau\in\mathcal{T}_{i,T}} ||\mathbf{x}_\tau||^2_{\mathbf{V}^{-1}_{i,\tau}}.$$

Proof on the bound $\Psi_{i,T}$ is provided in Appendix F. The Lemma highlights the importance in taking into account the graph structure in the payoff estimation, showing that the uncertainty of $\hat{y}_\tau$ is reduced when the graph structure is exploited. This effect diminishes with time: in Fig. 1(b), we see that as more data are collected, the graph-based estimator approaches the estimator in which users parameters are estimated independently (since $\Psi_{i,T} \to 1$).

6.1 Regret Upper Bound

We present the cumulative regret upper bounds satisfied by both **GraphUCB** and **GraphUCB-Local**.

Theorem 1. *$\Psi_{i,T}$ is defined in Eq. 13, $\mathbf{\Lambda}_{i,T}$ defined in Eq. 10 and $\mathbf{\Delta}_i = \sum_{j=1}^{n} \mathcal{L}_{ij}\boldsymbol{\theta}_j$. Without loss of generality, assume $||\boldsymbol{\theta}_i||_2 \leq 1$ for any $i \in \{1, 2, ..., n\}$ and $||\mathbf{x}_\tau||_2 \leq 1$ for any $\tau \leq T$. Then, for $\delta \in [0, 1]$, for any user $i \in \{1, 2, ..., n\}$ the cumulative regret over time horizon T satisfies the following upper bound with probability $1-\delta$:*

$$R_{i,T} = \sum_{\tau\in\mathcal{T}_{i,T}} r_\tau = \mathcal{O}\Big(\big(\sqrt{d\log(|\mathcal{T}_{i,T}|)} + \sqrt{\alpha}||\mathbf{\Delta}_i||_2\big) \times \Psi_{i,T}\sqrt{d|\mathcal{T}_{i,T}|\log(|\mathcal{T}_{i,T}|)}\Big). \tag{14}$$

Assuming that users are served uniformly up to time horizon T, i.e., $|\mathcal{T}_{i,T}| = T/n$, the network regret (the total cumulative regret experienced by all users) satisfies the following upper bound with probability $1-\delta$:

$$\begin{aligned} R_T &= \sum_{i=1}^{n} R_{i,T} = \sum_{i=1}^{n} \tilde{\mathcal{O}}\Big(\Psi_{i,T} d\sqrt{T/n}\Big) \\ &= \tilde{\mathcal{O}}\Big(d\sqrt{Tn}\max_{i\in\mathcal{U}} \Psi_{i,T}\Big). \end{aligned} \tag{15}$$

Proof. See Appendix G. □

Remark 2. *Both* ***GraphUCB*** *and* ***GraphUCB-Local*** *satisfy Theorem 1.*

The regret upper bound in Theorem 1 is derived based on **GraphUCB-Local** algorithm (Appendix G). Due to the approximation error introduced in the Taylor expansion, the regret of **GraphUCB-Local** is worse than that of **GraphUCB**. Therefore, Theorem 1 is also an upper bound of **GraphUCB**.

6.2 Comparison with LinUCB and Gob.Lin

Under the same setting, the single-user regret upper bound of **LinUCB** [Li et al., 2010] is:

$$R_{i,T} = \mathcal{O}\Big(\big(\sqrt{d\log(|\mathcal{T}_{i,T}|)} + \sqrt{\alpha}||\boldsymbol{\theta}_i||_2\big) \times \sqrt{d|\mathcal{T}_{i,T}|\log(|\mathcal{T}_{i,T}|)} \Big) \tag{16}$$

Since $||\boldsymbol{\Delta}_i||_2 \leq ||\boldsymbol{\theta}_i||_2$ and $\Psi_{i,T} \in (0,1)$ (Lemma 2 and Lemma 3), **GraphUCB** (and **GraphUCB-Local**) leads to a lower regret −Eq. 14− than **LinUCB** −Eq. 16.

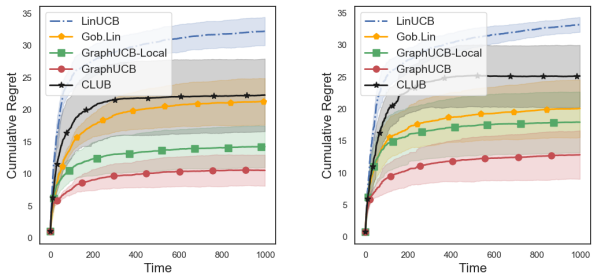
The cumulative regret experienced by all users in **Gob.Lin** [Cesa-Bianchi et al., 2013] is upper bounded by:

$$R_T = 4\sqrt{T(\sigma^2 \log\frac{|\mathbf{M}_t|}{\delta} + L(\boldsymbol{\theta})\log|\mathbf{M}_t|} = \tilde{\mathcal{O}}(nd\sqrt{T}) \tag{17}$$

where $L(\boldsymbol{\theta}) = \sum_{i=1}^{n}||\boldsymbol{\theta}_i||_2 + \sum_{(i,j)\in E}||\boldsymbol{\theta}_i - \boldsymbol{\theta}_j||_2$. Clearly, the cumulative regret achieved by **GraphUCB** in Eq. 15 is in the order of $\tilde{\mathcal{O}}(\sqrt{n})$ which is better than that in Eq. 17. This is mainly due to the different UCBs used in these algorithms. Specifically, **Gob.Lin** proposed the following bound:

$$\beta_t = \sigma\sqrt{\log\frac{|\mathbf{M}_t|}{\delta} + L(\boldsymbol{\theta})} = \tilde{\mathcal{O}}(\sqrt{nd}), \tag{18}$$

while we propose a lower single-user bound $\beta_{i,t} = \tilde{\mathcal{O}}(\sqrt{d})$ (Eq. 11). As described in Remark 1, the bound in Eq. 18 grows with the degree of the network, which in the bound is hidden in $L(\boldsymbol{\theta})$. We emphasize that in practice **GOB.Lin** could perform much better than its regret upper bound Eq. 17, if β_t defined in Eq. 18 is replaced by $\lambda\sqrt{log(t+1)}$ where λ is a tunable parameter. This is exactly what the authors of **Gob.Lin** did in their original paper when reporting empirical results. We follow the same trick when implementing **Gob.Lin** in our experiments, and show that **GraphUCB** still achieves better performance empirically.



(a) RBF (s=0.5) (b) ER (p=0.4)

Figure 2: Cumulative regret vs. time for different type of graphs (ER and RBF) consistently generated with the same level of smoothness and sparsity between graphs.

7 Experiment Results

We evaluate the proposed algorithms and compare them to **LinUCB** (no graph information exploited in the bandit), **Gob.Lin** (graph exploited in the features estimation) and **CLUB** (graph exploited to cluster users). All results reported are averaged across 20 runs. In all experiments, we set confidence probability parameter $\delta = 0.01$, noise variance $\sigma = 0.01$, and regularization parameter $\alpha = 1$. For **Gob.Lin**, we use $\beta_{i,t} = \lambda\sqrt{\log(t+1)}$, and λ is set using the best value in range $[0, 1]$. For **CLUB**, the edge deletion parameter α_2 is tuned to its best value.

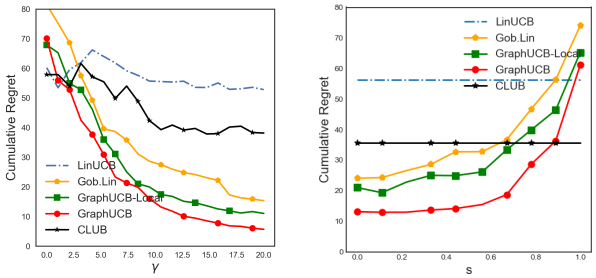
7.1 Experiments on Synthetic Data

In the synthetic simulations, we first generate a graph $\mathcal{G}$ and then generate a smooth $\boldsymbol{\Theta}$ via Eq. 19 which is proposed in [Yankelevsky and Elad, 2016]:

$$\boldsymbol{\Theta} = \arg\min_{\boldsymbol{\Theta}\in\mathbb{R}^{n\times d}} ||\boldsymbol{\Theta} - \boldsymbol{\Theta}_0||_F^2 + \gamma tr(\boldsymbol{\Theta}^T\mathcal{L}\boldsymbol{\Theta}) \tag{19}$$

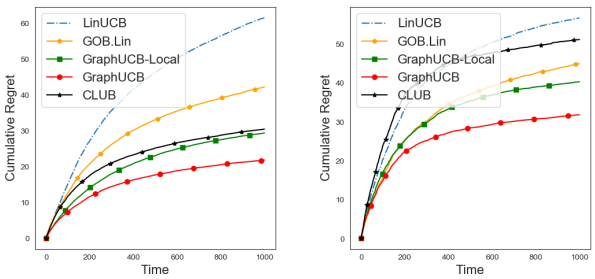
where $\boldsymbol{\Theta_0} \in \mathbb{R}^{n\times d}$ is a randomly initialized matrix, and $\boldsymbol{\mathcal{L}}$ is the random-walk graph Laplacian of $\mathcal{G}$. The second term in Eq. 19 promotes the smoothness of $\boldsymbol{\Theta}$: the larger the γ, the smoother the $\boldsymbol{\Theta}$ over the graph[3]. In all experiments, $n = 20, d = 5$. To simulate $\mathcal{G}$, we follow two random graph models commonly used in the network science community: 1) Radial basis function (RBF) model, a weighted fully connected graph, with edge weights $W_{ij} = exp(-\rho||\boldsymbol{\theta_i} - \boldsymbol{\theta_j}||^2)$; 2) Erdős Rényi (ER) model, an unweighted graph, in which each edge is generated independently and randomly with probability p.

[3]The regularization parameter γ in Eq. 19 is used to generate a smooth function in the synthetic settings, while the parameter α in Eq. 4 is used in the bandit algorithm to infer the smooth prior when estimating user features.



(a) (b)

Figure 3: Cumulative regret for RBF graphs with different level of smoothness (a) and sparsity (b).



(a) MovieLens (b) Netflix

Figure 4: Performance on Real-World data.

In Figure 2, we depict the cumulative per-user regret as a function of time for both RBF and ER graphs for both our proposed algorithms and competitors. The regret is averaged over all users and over all runs. Under all graph models, **GraphUCB** outperforms its competitors consistently with a large margin. Also **GraphUCB-Local** consistently outperform competitor algorithms, with however slightly degradated performance with respect to **GraphUCB**. This is due to the approximation introduced by Eq. 7. **Gob.Lin** is a close runner since it is also based on the Laplacian regularized estimator, but it performs worse than the proposed algorithms, as already explained in the previous section. **CLUB** performs relative poor since there is no clear clusters in the graph. Nevertheless, it still outperforms **LinUCB** by grouping users into clusters in the early stage which speeds up the learning process. It is worth noting that the two subfigures depict the same algorithm for two different graph models (RBF and ER) with the same level of smoothness and sparsity. The trend of the cumulative regret is the same, highlighting that the algorithm is not affected by the graph model. This behavior is reinforced in Appendix I where we provide further results.

We are now interested in evaluating the performance of the proposed algorithms against different graph topologies, by varying signal smoothness and sparsity of graph (edge density) as follows:
Smoothness [γ]**:** We first generate a *RBF* graph. To control the smoothness, we vary $\gamma \in [0, 10]$.
Sparsity [s]**:** We first generate a *RBF* graph, then generate a smooth $\mathbf{\Theta}$ via Eq. 19. To control the sparsity, we set a threshold $s \in [0, 1]$ on edge weights W_{ij} such that W_{ij} less than s are removed.

Figure 3 depicts the cumulative regret for different level of smoothness and sparsity of $\mathcal{G}$. **GraphUCB** and **GraphUCB-Local** show similar patterns (with **GraphUCB-Local** leading to higher regret due to the already commented approximation): *(i)* the smoother $\mathbf{\Theta}$ the lower the regret, which is consistent with the Laplacian-regualrized estimator of Eq. 4; *(ii)* denser graphs lead to lower regret since more connectivity provides more graph information which speeds up the learning process.

7.2 Experiments on Real-World Data

We then carry out experiments on two real-world datasets : **Movielens** [Lam and Herlocker, 2006] and **Netflix** [Bennett et al., 2007]. We follow the data pre-processing steps in [Valko et al., 2014], described in details in Appendix K. The cumulative regret over time is depicted in Figure 4 for both datasets. Both **GraphUCB** and **GraphUCB-Local** outperform baseline algorithms in all cases. Similarly to the synthetic experiments, **LinUCB** performs poorly, while **Gob.Lin** shows a regret behavior more similar to the proposed algorithms. In the case of **Movielens**, **CLUB** outperforms **Gob.Lin**. A close inspection of the data reveals that ratings provided by all users are highly concentrated. It means most users like a few set of movies. This is a good model for the clustering algorithm implemented in **CLUB**, hence the gain.

8 Conclusion

In this work, we propose a graph-based bandit algorithm **GraphUCB** and its scalable version **GraphUCB-Local**, both of which outperform the state-of-art bandit algorithms in terms of cumulative regret. On the theoretical side, we introduce a novel UCB embedding the graph structure in a natural way and show clearly that exploring the graph prior could reduce the cumulative regret. We demonstrate that the graph structure helps reduce the size of confidence set of the estimation of user features and the uncertainty of predicted payoff. As for future research directions, one possibility is to relax the assumption that the user graph is available and infer the graph from the data, ideally in a dynamic fashion.

Acknowledge

This work has been partially funded by Cisco Research Center.

References

[Abbasi-Yadkori et al., 2011] Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320.

[Agrawal and Goyal, 2013] Agrawal, S. and Goyal, N. (2013). Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135.

[Alvarez et al., 2012] Alvarez, M. A., Rosasco, L., Lawrence, N. D., et al. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine Learning*, 4(3):195–266.

[Auer, 2002] Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422.

[Bennett et al., 2007] Bennett, J., Lanning, S., et al. (2007). The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA.

[Cesa-Bianchi et al., 2013] Cesa-Bianchi, N., Gentile, C., and Zappella, G. (2013). A gang of bandits. In *Advances in Neural Information Processing Systems*, pages 737–745.

[Chapelle and Li, 2011] Chapelle, O. and Li, L. (2011). An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257.

[Dani et al., 2008] Dani, V., Hayes, T. P., and Kakade, S. M. (2008). Stochastic linear optimization under bandit feedback.

[Gentile et al., 2014] Gentile, C., Li, S., and Zappella, G. (2014). Online clustering of bandits. In *International Conference on Machine Learning*, pages 757–765.

[Korda et al., 2016] Korda, N., Szörényi, B., and Shuai, L. (2016). Distributed clustering of linear bandits in peer to peer networks. In *Journal of machine learning research workshop and conference proceedings*, volume 48, pages 1301–1309. International Machine Learning Societ.

[Lam and Herlocker, 2006] Lam, S. and Herlocker, J. (2006). Movielens data sets. *Department of Computer Science and Engineering at the University of Minnesota*.

[Lattimore and Szepesvari, 2016] Lattimore, T. and Szepesvari, C. (2016). The end of optimism? an asymptotic analysis of finite-armed linear bandits. *arXiv preprint arXiv:1610.04491*.

[Lattimore and Szepesvári, 2018] Lattimore, T. and Szepesvári, C. (2018). Bandit algorithms. *preprint*.

[Li et al., 2010] Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM.

[Li et al., 2016] Li, S., Karatzoglou, A., and Gentile, C. (2016). Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 539–548. ACM.

[Liu et al., 2018] Liu, B., Wei, Y., Zhang, Y., Yan, Z., and Yang, Q. (2018). Transferable contextual bandit for cross-domain recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Nguyen and Lauw, 2014] Nguyen, T. T. and Lauw, H. W. (2014). Dynamic clustering of contextual multi-armed bandits. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1959–1962. ACM.

[Robbins, 1952] Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535.

[Shuman et al., 2013] Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98.

[Valko et al., 2014] Valko, M., Munos, R., Kveton, B., and Kocák, T. (2014). Spectral bandits for smooth graph functions. In *International Conference on Machine Learning*, pages 46–54.

[Vaswani et al., 2017] Vaswani, S., Schmidt, M., and Lakshmanan, L. V. (2017). Horde of bandits using gaussian markov random fields. *arXiv preprint arXiv:1703.02626*.

[Wu et al., 2016] Wu, Q., Wang, H., Gu, Q., and Wang, H. (2016). Contextual bandits in a collaborative environment. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 529–538. ACM.

[Yang and Toni, 2018] Yang, K. and Toni, L. (2018). Graph-based recommendation system. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 798–802. IEEE.

[Yankelevsky and Elad, 2016] Yankelevsky, Y. and Elad, M. (2016). Dual graph regularized dictionary learning. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):611–624.