Efficient Graph Bandit Learning with Side-Observations and Switching Constraints

Xueping Gong, Jiheng Zhang

The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong S.A.R., China.
xgongah@connect.ust.hk, jiheng@ust.hk

Abstract

This paper presents a novel framework for multi-armed bandit problems with side-observations and switching constraints, which arises in a range of real-world applications such as robotic. To address the challenges of effectively utilizing graph-structured observations while adhering to graph constraints, we design graph-agnostic and graph-aware algorithms tailored to this new setting. Specifically, our graph-agnostic algorithm selects nodes with the highest upper confidence bound without prior knowledge of feedback probabilities, while minimizing switching costs using offline shortest path planning and the doubling trick. If the graph structure and associated probability matrix are known, our graph-aware algorithm plans the exploration step using a linear programming approach and eliminates suboptimal nodes iteratively. We rigorously analyze the performance of our proposed algorithms, providing near-optimal minimax and instance-dependent regret upper bounds. Our analysis shows that our algorithms outperform generic reinforcement learning methods in terms of both regret and computational efficiency. Extensive numerical experiments on various types of graphs, including two real-world datasets, demonstrate the efficacy of our proposed methods and their advantages over benchmark methods in graph bandit settings.

1 Introduction

The multi-armed bandit (MAB) problem is a well-known framework for decision-making under uncertainty (Lattimore and Szepesvári 2020). In the classical stochastic MAB problem, an agent repeatedly selects from a finite set of actions to maximize total rewards, balancing exploration and exploitation. This framework has received significant attention from the online learning community and has been applied to various fields, including recommendation systems, portfolio selection, and clinical trials (Li et al. 2010).

Despite its popularity, the classical MAB framework has limitations in real-world applications that involve *side observations* and *switching constraints*. Side observations provide additional information about rewards (e.g., when certain actions yield similar outcome), such as multiple feedbacks (Mannor and Shamir 2011) or prior knowledge about the environment (Lykouris, Sridharan, and Tardos 2018; Seldin et al. 2014). Switching constraints refer to limitations on the ability of the agent to switch freely between different actions (Coquelin and Munos 2007; Lu et al. 2018; Zhang, Johansson, and Li 2022), which may have unit switching costs (Dekel et al. 2014; Arora, Marinov, and Mohri 2019; Rangi and Franceschetti 2019) or limited switching budgets (Chen et al. 2020; Simchi-Levi and Xu 2019).

To address these limitations, we propose a new framework, which incorporates both graph feedback and switching constraints. Specifically, we consider the problem that an agent traverses an undirected, connected graph $G_c = (V, E_c)$ that restricts its ability to move between different nodes, and receives a graph feedback from the whole reward graph $G_r = (V, E_r)$ upon arrival at a node. We refer to this MAB problem with graph feedback and graph constraint as the *graph bandit problem*.

The proposed framework is a suitable model for many real-world problems, as demonstrated in Appendix. For example, the graph bandit problem can be applied to optimize the collection efficiency of cleaning robots (Rayguru et al. 2021) who can observe the environments in a neighborhood (G_r) and should follow the route planning (G_c), to maximize the use of communication channels of a drone (Qiu et al. 2019) that can receive signals from geographically close sensors in a sensor network (G_r) and should move between different spots with a map (G_c), or to provide diverse items to users in a recommendation system (Li et al. 2010) which may infer the preferences of users to similar items (G_r) and follow certain switching rules (G_c).

Previously, these real-world problems were often reformulated as non-episodic, un-discounted learning problems with known deterministic transitions and then solved using generic reinforcement learning methods. These graph bandit problems have unique features that generic RL algorithms may not take into account, leading to high computational costs, sample complexity, and regrets, especially for large-scale problems. This is particularly relevant for robotic learning problems with extended sensors, as described in (Dann et al. 2020). Therefore, it is worth considering whether any improvement in computational efficiency and learning performance can be achieved by leveraging the graph bandit problem structure. Our proposed framework provides a powerful tool for addressing these problems that involve side observations and switching constraints, by uncovering their special hidden structure.

Contributions. Our work presents a new setting for stochastic online learning that incorporates both graph feedback and graph constraints. To the best of our knowledge, we are the first to consider both of these aspects simultaneously.

We design graph-agnostic and graph-aware algorithms tailored to this new setting and provide rigorous theoretical analysis of their performance. For graph-agnostic algorithms, we show general upper bounds in probabilistic cases (see Theorems 3.1 and 3.2). Compared with generic reinforcement learning methods in the graph bandit setting with deterministic feedback graph (Dann et al. 2020), our methods save a constant factor of $\sqrt{|V|}$ in regret order and reduce space complexity from $\mathcal{O}(|V|^2)$ to $\mathcal{O}(|V|)$. We have also developed a graph-aware algorithm (Algorithm 2), which employs an LP to fully utilize the graph structures and eliminate suboptimal choices iteratively. This approach provides much better upper bounds (Theorem 3.3).

Furthermore, we extend the switching condition presented in (Simchi-Levi and Xu 2019) to the probabilistic feedback setting. While (Simchi-Levi and Xu 2019) established the necessity of $\mathcal{O}(\log T)$ switches, our LP-GG algorithm introduces a more precise switching cost of $\mathcal{O}(\log(\Delta^{-1}))$. This refined switching cost takes into account the specific details and characteristics of the bandit instances, resulting in a more accurate and tailored approach to switching actions.

Finally, to demonstrate the efficacy of our methods, we conduct extensive numerical experiments on various types of graphs, including real-world datasets. Our results show that our methods have significant advantages over benchmarks in graph bandit settings, and are effective in practical scenarios.

Related Work

MAB models with side information, such as contextual bandits (Li et al. 2010; Abbasi-Yadkori, Pál, and Szepesvári 2011; Li, Lu, and Zhou 2017; Chu et al. 2011), are used in various problems. However, it is important to distinguish graph feedback from contextual side information, which is given to the agent before pulling an arm. The setting of graph feedback can be fit into a more general setting of partial monitoring (Lattimore and Szepesvári 2020), but algorithms for general partial monitoring games are not efficient to deal with graph feedback as these algorithms will require infinite or exponentially large feedback matrix.

The scenario of online learning with side information modeled by feedback graphs was first introduced by Mannor and Shamir (2011). This scenario just interpolates from the full information setting where the agent is informed of the all rewards at each round, to the bandit setting where only the reward of selected action is known to the agent. Online learning with feedback graphs has been further extensively analyzed by Alon et al. (2015, 2017) and several other authors Rouyer et al. (2022); Chen et al. (2021); Cohen, Hazan, and Koren (2016); Cesa-Bianchi, Gentile, and Zappella (2013); Atsidakou et al. (2022). Several novel methods, including UCB(Lykouris, Tardos, and Wali 2020; Caron et al. 2012), TS(Lykouris, Tardos, and Wali 2020), EXP(Rouyer et al. 2022; Chen et al. 2021; Alon et al. 2015; Cohen, Hazan, and Koren 2016), IDS(Liu, Buccapatnam, and Shroff 2018), and their variants, have been designed for this setting. Remarkably, Dann et al. (2020) extends it to reinforcement learning. A comprehensive survey (Valko 2016) of graph bandits is available in the literature.

Our work is closely relevant to bandits with switching costs, where the decision-maker should pay an extra penalty when the current action differs from the one on the previous round. Such switching costs can be modeled by our constraint graph G_c which is fully connected and has additional weights on edges representing switching costs. Dekel et al. (2014) show that algorithms for this setting in the adversarial regime will incur at least $\Omega(T^{\frac{2}{3}})$ regrets. Simchi-Levi and Xu (2019) consider a slightly different setting where the switching budget is limited, and Chen et al. (2020) extends it into online convex optimization setting. For strict switching constraints, we refer readers to (Coquelin and Munos 2007; Zhang, Johansson, and Li 2022).

As far as we know, only a few papers consider both graph feedback and switching costs (Arora, Marinov, and Mohri 2019; Rangi and Franceschetti 2019). However, most of these papers only consider *deterministic* feedback graphs rather than general *probabilistic* feedback graphs(Esposito et al. 2022; Cortes et al. 2020; Li et al. 2020; Kong, Zhou, and Li 2022). One reason is that these papers should make use of certain graph structure to collect sufficient information about rewards during the process of learning. Furthermore, their switching costs are fixed, which is a special case of our graph constraints. More generally, our graph constraints can be naturally applied to graph-structured switching costs which represent the scenario where moving to different nodes incurs different costs.

2 Problem Formulation

We study an agent traveling on an undirected, connected graph $G_c = (V, E_c)$ with 1-subgaussian reward distributions at each vertex $s \in V$, where rewards are independent and bounded in $[0, 1]$. The reward feedbacks have a graph structure $G_r = (V, E_r)$, where each edge $(i, j) \in E_r$ is associated with probability p_{ij}. The feedback graph G_r and the constraint graph G_c both have a self-loop (Alon et al. 2015; Cortes et al. 2020) at every vertex, allowing the agent to observe their own bandit information. We assume $p_{ij} > 0$ if and only if the edge (i, j) is in E_r, and thus $p_{ii} > 0$ for all nodes $i \in V$ for technical convenience. We also denote the neighbor of a node s as $\mathcal{N}_s(G)$ Whenever the agent visits a node a_t, they receive feedback from all neighboring nodes in G_r, denoted as $\{(s, r_s) : X_s = 1, s \in \mathcal{N}_{a_t}(G_r)\}$, where X_s is an independent Bernoulli random variable with parameter $p_{a_t,s}$. The agent receives reward feedback from node a_t and its neighborhood based on a probability matrix $\mathbf{P}$, and the minimum observation probability is denoted as $p_* = \min_{(i,j)\in E_r} p_{ij}$. The probabilistic feedback setting differs from deterministic graph feedback, as it is more general due to a gap between real and expected observations, whereas in deterministic cases, the number of observations can be well-controlled by selecting actions. Besides the side-observations, the agent must satisfy graph constraints, where it can only move to the neighborhood in G_c in the next round.

The reward feedback graph G_r and the constraint graph G_c share the same vertex set V. This is a new MAB setting with graph constraints and graph feedback, where the graph structure can be known or unknown to the agent.

Let μ_s be the expected reward at node s, i.e., $\mu_s = \mathbb{E}[r_s]$ and denote $\mu^* = \max_{s \in V} \mu_s$ the highest expected reward in the graph. Without loss of generality, we assume that there exists a unique node s^* such that $\mu_{s^*} = \mu^*$. The goal of the agent is to travel on the graph G_c with feedback graph in G_r to maximize the cumulative expected rewards over time horizon T. To evaluate the performance of the learning algorithm π, we define the regret of π with respect to the best single action in hindsight:

$$\mathcal{R}(T) = \mathbb{E}\left[T\mu^* - \sum_{t=1}^{T} r_{s_t}\right], \ \ s.t.\ (s_{t-1}, s_t) \in E_c, \forall t,$$

where s_t is the node where the agent reaches at the round t. Here, the expectation is taken w.r.t. the randomness of algorithms and reward noise.

During the algorithm π execution, adaptive decisions are made based on historical information. In presence of switching constraint, there is a cost to gather information — the agent must travel in a constraint graph to discover rewards. Side observation setting motivates the agent to exploit more as more information can be obtained compared with classical algorithms. The agent need to balance the exploration and exploitation while following the graph constraint and observing the graph feedback.

3 Methods

Offline planning. The goal of offline planning is to find optimal switching policy given that the mean rewards are known. This problem can be formulated as a shortest path problem, as described in (Zhang, Johansson, and Li 2022). A simple approach to finding the optimal policy is to move directly to the destination node using the shortest path algorithm. We can use either *Dijkstra* or *Bellman-Ford* to find a path from the starting vertex to the destination vertex.

To describe our method clearly, we construct a directed graph $\hat{G}_c$ based on the original G_c. We define $\hat{E}_c$ as the directed edges converted from the undirected edges in E_c. The expected cost of visiting a node s is defined as the reward gap $\Delta_s = \mu^* - \mu_s$ of the node s. For all edges $(s, s') \in \hat{E}_c$, we define the distance $d_{ss'}$ from a node s to its neighboring node s' in $\mathcal{N}_s(G_c)$ as the reward gap of node s'. Note that the algorithm must take the agent to the destination in no greater than $|V|$ steps since it cannot revisit any sub-optimal node twice. (See implementation details of *OSSP* in Appendix.)

To quantify the total cost, we denote the maximal switching cost in the graph G_c as

$$D = \max_{s,s' \in V} \ell(s, s'), \tag{1}$$

where $\ell(s, s')$ is the distance of two nodes s, s' in G_c, i.e., the length of the shortest path in G_c from s to s'. Clearly, the total cost is bounded by D when the agent make a transition from one node to another.

Note that one famous setting is bandit feedback with switching cost (Arora, Marinov, and Mohri 2019; Rangi and Franceschetti 2019; Dekel et al. 2014), which can be incorporated into our graph constraint. The regret of bandits with switching is usually defined as $\mathcal{R}(T) = \mathbb{E}[T\mu^* - \sum_{t=1}^{T} r_{s_t} + \mathbf{I}\{s_t \neq s_{t-1}\}]$. Let G_c be a fully connected graph and add unit cost to each edge in $\hat{G}_c$, which indicates the agent will switch freely with unit cost. This conversion shows that this setting is a special case of ours, as G_c can be more general and forbids switching between some nodes. In this setting, $D = 1 + \max_{s \in V} \Delta_s$. For more general graph-structured switching cost $\tilde{G}$, we may rewrite (1) as $D = \max_{s,s' \in V} \ell(s, s') + \tilde{\ell}(s, s')$, where $\tilde{\ell}(s, s')$ is the length of the shortest path in $\tilde{G}$ from s to s'.

In summary, this model captures that the agent travels among several states and collects rewards from different states. The adjacent states share similar reward feedback, and switches between states should follow the graph constraint. For more motivating examples, we refer readers to Appendix.

Graph-agnostic algorithm. Adding switching costs, however, modifies the price of exploration. To control the overall switching cost, we apply the mini-batch technique and the doubling trick (Besson and Kaufmann 2018). These techniques force agents to stick to a valuable action in the current episodes. Our Algorithm 1 first finds the auxillary node $a'_m = \arg\max_{a \in V} U_a(m)$, and then decide the chosen node by $a_m = \arg\max_{a \in \mathcal{N}_{a'_m}(G_c) \cap \mathcal{N}_{a'_m}(G_r)} \hat{\mu}_a(m)$. Intuitively, the auxillary node has the highest UCB value, indicating that either its average estimate is very high, making it empirically the best node to visit, or its uncertainty is large, indicating that more information is needed. In the second case, the agent wants to observe samples to reduce the uncertainty of the node. However, in the side observation setting, the agent can obtain an observation by visiting any of the node's neighbors, especially one with a higher empirical reward. Meanwhile, in the first case, the chosen node is already the best one in its observation set. Hence, in Algorithm 1 (see UCB-GG in Appendix), the agent first selects the node it wants to observe based on its UCB value and then selects the node it wants to exploit from its neighbors based on its empirical mean only.

Due to the doubling trick, the total switching cost is bounded by $\mathcal{O}(MD)$, where M is the total number of episodes. The following Lemma 3.1 shows that M is logarithmic in T, which means that the total switching cost is controlled by a logarithmic factor. From (Simchi-Levi and Xu 2019) we know, it is necessary to make $\Omega(|V| \log T)$ transitions to achieve optimal regrets. We extend the leading quantities from $\mathcal{O}(|V| \log T)$ to $O(\chi / \log(1 + \frac{p_*}{2}) \log T)$ in graph feedback setting. Denote $\mathfrak{C}$ as the collection of cliques and $\mathcal{C}$ as the one of the cliques in $\mathfrak{C}$.

Lemma 3.1 *The total number of episodes M in Algorithm 1 is logarithmic in T, i.e.,*

$$M \leq \chi \log(2T + 3|V|) / \log(1 + \frac{p_*}{2}),$$

where χ is the clique-partition number of G_r, $\Delta_{\mathcal{C}} = \min_{s \in \mathcal{C}, s \neq s^*} \Delta_s$ and $\mathfrak{C}$ is the optimal clique-partition of G_r.

Moreover, the switching cost can be further reduced in the gap-dependent upper bound. This Lemma 3.2 extends the result of (Simchi-Levi and Xu 2019) in the gap-dependent settings. It indicates the less number of switches on easy instances.

Lemma 3.2 *The total number of episodes M in Algorithm 1 is logarithmic in T, i.e.,*

$$M \leq \sum_{\mathcal{C}\in\mathfrak{C}} \log(\frac{4\log T}{p_*\Delta_{\mathcal{C}}^2})/\log(1+\frac{p_*}{2}),$$

where χ is the clique-partition number of G_r.

Algorithm 1: UCB-GGmax

Input: An initial node s_0, a constraint graph $G_c = (V, E_c)$

Place the agent at s_0
Follow any path that visits all nodes at least once in G_c
Initialization: the empirical mean $\hat{\mu}_s$ and the number of observations n_s for each node $s \in V$
Let $m = 0$ and $a_0 = s_0$
for epoch $m = 1, 2, \cdots$ **do**
  Let t_m be the beginning round of the current episode m
  Compute $U_s(m) = \hat{\mu}_s(m) + \sqrt{\log t_m / n_s(m)}$ for all $s \in V$ at the episode m
  Find the node $a'_m = \arg\max_{a\in V} U_a(m)$
  Find $a_m = \arg\max_{a\in\mathcal{N}_{a'_m}(G_c)\cap\mathcal{N}_{a'_m}(G_r)} \hat{\mu}_a(m)$
  Use *OSSP*(G_c, $\hat{\mu}$, a_{m-1}, a_m) algorithm to find the shortest path $\mathcal{P}$ from a_{m-1} to a_m
  For each node $s \in \mathcal{P}$, collect the graph feedback $\mathcal{N}_s(G_r)$
  Update the empirical mean $\hat{\mu}_s$ and the number of observations n_s for visited nodes $s \in \mathcal{P}$
  Stay at the node a_m for $n_{a'_m}$ rounds and collect reward feedback at this node
  Update the empirical mean $\hat{\mu}_s$ and the number of observations n_s for each node $s \in \mathcal{N}_{a_m}(G_r)$
end for

The exploration step can be omitted if we make the convention that $\sqrt{1/0} = +\infty$. In the following episodes, every node will have a chance to be visited. However, this step is still useful when the constraint graph G_c is unknown. When at a node s, the agent can discover its adjacent nodes. By keeping records of this information and following certain rules (e.g., DFS) to travel in the graph G_c, the agent can finally obtain G_c. This step will cause at most $\mathcal{O}(|V|D)$ regrets.

To illustrate the performance of Algorithm 1, we prove the following Theorem 3.1. The proof is challenging to incorporate graph structures. To address this problem, the visiting nodes are put into each layer that consists of an independence set of G_r. Summing over all layers, we can bound the regrets caused by suboptimal nodes.

Theorem 3.1 *The regret of Algorithm 1 is $\mathcal{O}(\sqrt{\frac{\alpha T}{p_*}}\log T + \chi D\log T/\log(1+\frac{p_*}{2}))$, where χ and α are the independence number and clique-partition number of G_r, respectively.*

The idea of the gap-dependent upper bound is to split all nodes into χ cliques, such that visiting a node in a clique reveals unbiased estimates of the rewards of all the other nodes in the clique. Our algorithm achieve it implicitly, compared with ExpBan in (Mannor and Shamir 2011), and simultaneously get low regret with respect to any node in that clique.

The gap-dependent upper bound of Algorithm 1 is shown in Theorem 3.2. The last term may actually be large for small values of T and pathological regret distributions. However, Algorithm 1 performs better than UCB-GG (see realization in Appendix) asymptotically, due to its greedy nature.

Theorem 3.2 *Denote $\mathfrak{C}$ as the optimal clique-partition of G_r and $\Delta_{\mathcal{C}} = \min_{s\in\mathcal{C}, s\neq s^*} \Delta_s$ as the clique optimality gap. Let $\delta_{min} = \min_{i\neq j}|\mu_i - \mu_j|$ be the minimum gap of a pair of nodes in V. The regret of the Algorithm 1 is*

$$2D\sum_{\mathcal{C}\in\mathfrak{C}} \log(\tfrac{4\log T}{p_*\Delta_{\mathcal{C}}^2})/\log(1+\tfrac{p_*}{2}) + \sum_{\mathcal{C}\in\mathfrak{C}} \tfrac{4\log T\max_{i\in\mathcal{C}}\Delta_i}{p_*\Delta_{\mathcal{C}}^2} - \mathcal{M}(T)\delta_{min} + \mathcal{O}(\log\log T),$$

where $\mathcal{M}(T)$ is the expected number that the node the agent want to observe and the node the agent actually choose are different.

The quantity $\mathcal{M}(T) \geq \sum_{m=1}^{M} \mathbf{I}\{a_m \neq a_{m-1}\}$ measures the benefits of exploitation by switching to the empirically better node nearby. Combining these facts, we can say that Algorithm 1 may outperform UCB-GG for sufficiently large T. Intuitively, the sample size becomes large in our setting, so it's better to exploit more compared with classical ones.

Connection to reinforcement learning. Our result shows the benefit of utilizing the graph structure compared to state-of-the-art generic RL algorithms. A $\tilde{\mathcal{O}}(\sqrt{\tilde{\alpha}T})$ regret bound is established by generic RL algorithms (Dann et al. 2020) in our graph bandit setting, where $\tilde{\alpha}$ is the independence number of state-action feedback graph G_{RL}. However, if the state-action feedback graph is constructed from the feedback graph G_r, the resulting graph G_{RL} is too sparse as $\tilde{\alpha} = \alpha|V|$ (see Proposition 3.1), where α is the independence number of G_r. This implies that the proposed method (Theorem 3.1) saves an extra $\sqrt{|V|}$ constant in regrets compared to (Dann et al. 2020; Sutton and Barto 2018).

Furthermore, traditional RL algorithms typically require $\mathcal{O}(|V|^2)$ space complexity in the graph bandit setting, but our method reduces it to $\mathcal{O}(|V|)$ storage. This reduction in storage complexity helps to reduce the variance in parameter estimation, which leads to lower regrets and faster learning rates. In numerical experiments, our algorithms are observed to be more stable and converge faster than Q-learning in almost all graph bandit instances.

Proposition 3.1 *Let $\tilde{\alpha}$ and α be the independence numbers of the feedback graph G_{RL} and G_r, respectively. Then, we have $\tilde{\alpha} = |V|\alpha$.*

Graph-aware algorithm. Algorithm 1 uses side observations efficiently, despite being agnostic to the feedback graph structure. Yet, sometimes, an alternative approach can be further beneficial if the whole reward graph is given as prior. Specifically, in some scenarios, certain nodes in the feedback graph may be highly informative due to their large

degrees. For example, Dekel et al. (2014) leverage the so-called "revealing actions" to obtain sufficient information and control the frequency of visiting such nodes to get low regrets; Dann et al. (2020) study the sample-complexity of reinforcement learning in MDPs with a small dominating set of the feedback graph. However, explicitly exploiting such nodes is typically not advantageous in regret terms as such node may yield low return. Furthermore, finding dominating sets or revealing actions in a large-scale graph is an NP-hard problem, and becomes ambiguous in probabilistic feedback cases.

To address these challenges, we consider the following LP problem

$$\begin{aligned} &\min \mathbf{1}^T \mathbf{z} \\ s.t. \quad &\mathbf{P}\mathbf{z} \geq \mathbf{1} \\ &\mathbf{z} \geq \mathbf{0}, \end{aligned} \tag{2}$$

where $\mathbf{P}$ is the associated probability matrix of the feedback graph G_r. The above LP computes the expected number of pulls to guarantee at least one observation for each action. Based on the above LP problem, we can identify informative nodes in the feedback graph. Intuitively, we can explore nodes in V according to the solution $\mathbf{z}^*$. In deterministic cases, nodes in dominating sets form a feasible solution to LP(2), so exploiting such nodes are less efficient compared with nodes found by LP problem. As noted in prior work (Buccapatnam et al. 2017; Li et al. 2020; Chen et al. 2021; He and Zhang 2022), controlling the frequency of exploration steps is crucial to achieving low regrets. Our method extends the idea of visiting "informative" nodes (Dekel et al. 2014) in the probabilistic setting.

The LP-GG policy is an extension of the UCB policy, which incorporates graph structure information. It follows a multi-round approach, where the algorithm estimates the gaps between arms in each epoch. In round m, the agent collects a sufficient number of observations for each remaining action, which is at least $\lceil 4^m \log(T/4^m) \rceil$ observations per action. The poorly performing actions are then eliminated based on their UCB indices. The agent utilizes the OSSP algorithm to switch between different actions.

Theorem 3.3 *The regret of the Algorithm 2 is upper bounded by*

$$\sum_{s\in V^*-B} \frac{2z_s^* \log(T/\Delta_s^2)}{\Delta_s} + \sum_{j\in V^*} \frac{20 \max_{i\in V} \Delta_i}{\Delta_j^2} + \sum_{s\in B} \frac{2\log(T/\Delta_s^2)}{\Delta_s} + (\|\mathbf{z}^*\|_0 \bar{m} + \sum_{s\in B} 2\log(\tfrac{1}{\Delta_s}))D,$$

where $m_s = \lceil \log_2(\frac{1}{\Delta_s}) \rceil$ *for any suboptimal nodes* $s \in V^* := V - \{s^*\}$, $\bar{m} = \min\{m : \|\mathbf{z}^*\|_1 > \sum_{s:m_s>m} 2^{-m+1}\}$ *and the set* $B = \{i \in V^* : m_i > \bar{m}\}$.

In the LP-GG algorithm, the switching cost can be further reduced to $\mathcal{O}(\log(\Delta^{-1}))$. To achieve optimal regrets, the gap Δ is chosen as $\Theta(\sqrt{\frac{1}{T}})$ in the minimax upper bound analysis. Consequently, the switching cost becomes $\mathcal{O}(\log T)$, which aligns with the results presented in (Simchi-Levi and Xu 2019). From (Buccapatnam, Eryilmaz, and Shroff 2014), we learn that the regret of the multi-armed bandit problem with graph feedback is lower bounded by $\Omega(\sqrt{\|\mathbf{z}^*\|_1 T})$.

Algorithm 2: LP-GG

Input: An initial node s_0, a constraint graph $G_c = (V, E_c)$, a reward feedback graph $G_r = (V, E_r)$ and its associated probability matrix $\mathbf{P}$, time horizon T

Solve the LP (2) for nodes in V, and obtain its solution as $\mathbf{z}$
Place the agent at s_0
Initialization: the empirical mean $\hat{\mu}_s = 0$, the number of observations $n_s = 0$ for each node $s \in V$, the exploration set $S_1 = V$ and the best arm candidate $B_1 = V$
for epoch $m = 1, 2, \cdots$ **do**
  if $|B_m| == 1$ **then**
    Use *OSPP* with input G_c, $\hat{\mu}$ to move from the current node to the only node in B_m
    Stay at the only node until the end of learning
  else if $\sum_{i\in V} z_i \leq |B_m| 2^{1-m}$ **then**
    for each node s in S_m and $z_s > 0$ **do**
      Use *OSPP* with input G_c, $\hat{\mu}$ to move from the current node to s
      Stay at the node for $\lceil z_s 4^m \log(T/4^m) \rceil - \lceil z_s 4^{m-1} \log(T/4^{m-1}) \rceil$ rounds
    end for
  else
    for each node s in B_m **do**
      Use *OSPP* with input G_c, $\hat{\mu}$ to move from the current node to s
      Stay at the node for $\lceil 4^m \log(T/4^m) \rceil - \lceil 4^{m-1} \log(T/4^{m-1}) \rceil$ rounds
    end for
  end if
  Update B_{m+1} by eliminating nodes in B_m with $\hat{\mu}_s + 2^{-m} < \max_{i\in B_m} \hat{\mu}_i$
  Update $S_{m+1} = \bigcup_{i\in B_{m+1}} \mathcal{N}_i(G_r)$
end for

Furthermore, (Simchi-Levi and Xu 2019) establishes that to achieve optimal regrets, the agent must incur a lower bound of $\Omega(\log T)$ switches. By combining these two facts, we can conclude that our LP-GG algorithm is near-optimal in terms of the time horizon T and the graph quantities. It achieves a regret that is close to the lower bound while minimizing the number of required switches.

4 Numerical Experiments

Experimental setup. We run 50 simulations for each algorithm under different conditions. We initialize the mean rewards from $\mu_s \sim \mathcal{U}(0, 1)$, for each node $s \in V$. In the simulations the reward distributions are normal random variables $\mathcal{N}(\mu_s, 0.1)$. We generate different types of the reward feedback graphs G_r and the constraint graphs G_c by a *graph generator* (see its realization details in Appendix). For simplicity, the agent is initially placed at the node with the smallest index in V. Additional numerical experiments can be found in Appendix.

Comparison between algorithms. We fix the number of nodes $|V| = 20$ in this section. Here, we let G_c to be a line and G_r be a star, k-tree or connected-star graph as an

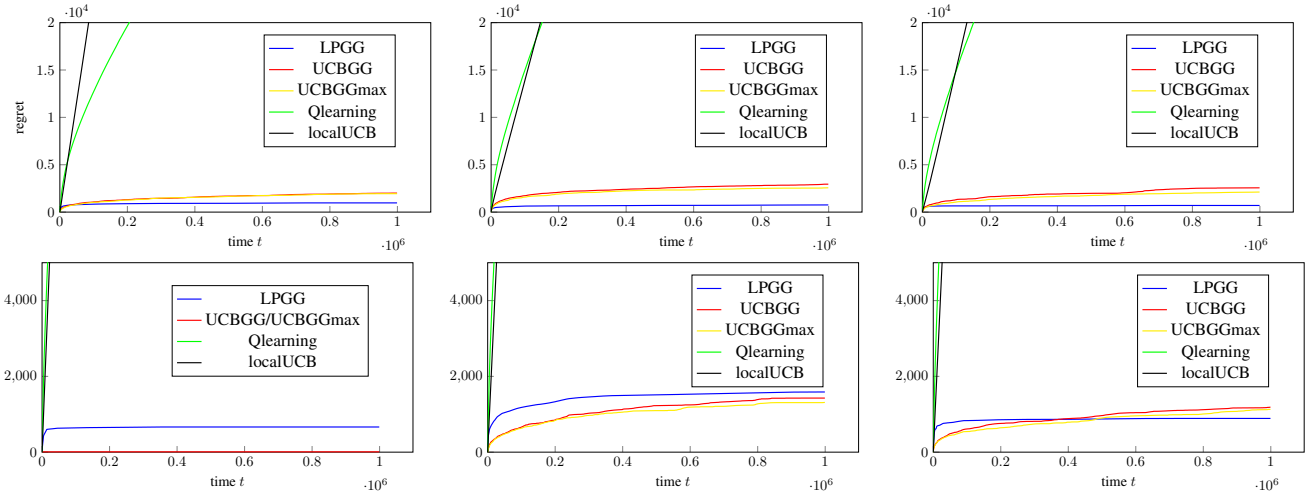

Figure 1: regrets in star, k-tree, connected-star, perfect, clique group and random graphs.

example of demonstration, while the results for other graph types are similar. We compare our proposed algorithms to the following benchmark algorithms:

- Local UCB (see Appendix): these benchmarks always move to the node with the highest UCB in the current neighborhood $\mathcal{N}_s(G_c)$ for current node s.
- Q-learning (Sutton and Barto 2018): we choose the learning rate 0.5, the discount factor 0.9 and the exploration probability $\min(1, 2|V|/\sqrt{t})$.

We implement LocalUCB to necessitate the application of UCB-GG, Algorithm 1 and Algorithm 2. In Figure 1, we observe that local UCB often gets trapped in the local maxima of rewards due to a lack of foresight, making its regrets grow almost linearly. On the other hand, Q-learning achieves sub-linear regrets, but it is not regret-optimal with respect to graph quantities in the graph bandit problem. Moreover, Q-learning has more unknown parameters to estimate and requires more samples than our algorithms to learn the optimal policy. Since Q-learning does not use the graph structure, the high variance of estimators in Q-learning makes it unstable and converge slowly. Finally, Q-learning requires tuning of three hyperparameters to achieve sub-linear regrets, highlighting the advantage of our proposed algorithms that do not require any modification of hyperparameters.

Since our algorithms are tailored to the considered setting, i.e., removing redundant states and decreasing the number of estimators, it is not surprising that they can achieve low regrets. LP-GG(Algorithm 2) can do better than UCB-GG and Algorithm 1 in graphs where there exist highly informative nodes. For example, in a k-tree, the independence number is equal to $|V| - 1$, but $\|\mathbf{z}^*\|_1$ is just equal to 1! As Theorem 3.3 indicates, the regrets can be improved approximately up to a constant $\sqrt{|V| - 1}$ if we use Algorithm 2 rather than UCB-GG in k-tree graphs.

We also notice that Algorithm 2 does not perform as well as UCB-GG and Algorithm 1 in dense graphs (e.g., perfect graphs or clique groups), where the independence number is almost equal to $\|\mathbf{z}^*\|_1$. In such graphs, the prior knowledge of G_r is not quite helpful, and Algorithm 2 may suffer from excessive exploration. We find that Algorithm 2 can achieve comparable regrets in k-tree, connected stars and perfect graphs, because $\|\mathbf{z}^*\|_1$ of these graphs can be rather smaller than the independence number (see Appendix).

Test on different p_*. We investigate the influence of p_* on random feedback and constraint graphs with a dense factor of 0.5 and a fixed number of nodes $|V| = 100$. To get smooth regret curves, we run simulations for 200 times. For small p_*, the feedback graph is rather empty, and using side observations is quite significant. Therefore, a small increase in p_* will lead to a large decrease in regrets when p_* is small. As p_* increases, the expected number of side-observations increase, and the performance of our algorithms that utilize side-observations, improves slightly compared with the cases when p_* is relatively small. Moreover, for any values of p_*, there is a noticeable gap between the performance of UCB-GG and UCB-GGmax.

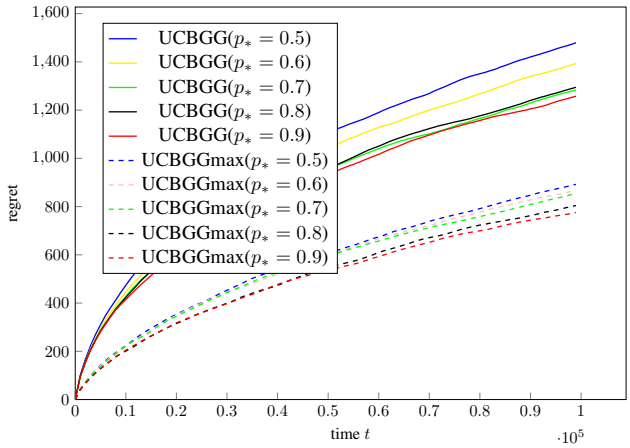

Figure 2: test on different p_*.

Test dependence on α in clique group graphs. In the analysis of UCB-GG, the upper bounds for regret are dependent on the independence number of G_r. However, determining the independence number of certain graphs is an NP-hard problem. To overcome this issue, we conduct experiments on clique group graphs (see more details in Appendix), where the independence number is equal to the minimum number of cliques that can cover the whole graphs. Specifically, we set the clique-partition number to be 10 and vary the number of nodes $|V|$ from 400 to 900. To maintain an approximately equal value of D, we choose $G_c = G_r$ so that the maximal switching cost does not increase with $|V|$. We observe that after exploration steps, the regrets of UCB-GG are nearly parallel. Based on these observations, we conclude that our algorithm does not scale with $|V|$.

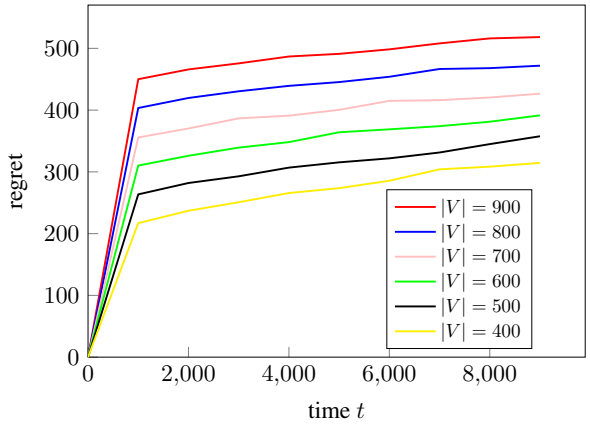

Figure 3: regrets in clique groups.

Test dependence on D. We also experiment with the influence of maximal weighted path in G_c on our algorithm. To conduct this experiment, we set G_r as a fully connected graph and G_c as a line. The number of nodes is fixed to $|V| = 50$, and D is systematically increased. Figure 4 shows that, on average, the regret at the round 5000 grows relatively linearly in D, which agrees with the regret bound in Theorem 3.2.

Test on real-world datasets. This section presents two synthetic applications that are modeled as a graph bandit problem. Since the datasets are too large, we use union-find technique to find connected subgraphs in the whole graphs.

The first application involves a drone that travels over a network of rural/suburban locations to provide internet access. The drone serves several neighboring locations in California per period, and the reward for serving each location is the communication traffic carried by the drone in gigabytes. This reward is independently sampled from a distribution associated with the location. The drone can either stay at the current location or move to a neighboring location for the next period. The drone has a map of the service area but does not know the reward distributions. The objective is to maximize the total reward before the drone is called back and recharged. To evaluate the performance of the algorithm, we conduct numerical experiments on the dataset of roads in a small area of California (Leskovec and Krevl 2014). The results, as shown in Figure 5, indicate that the drone consistently achieves sub-linear regrets in the simulations.

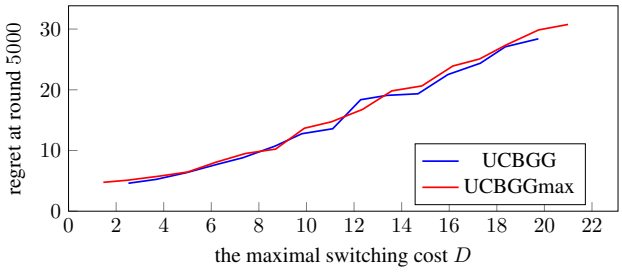

Figure 4: dependence on D.

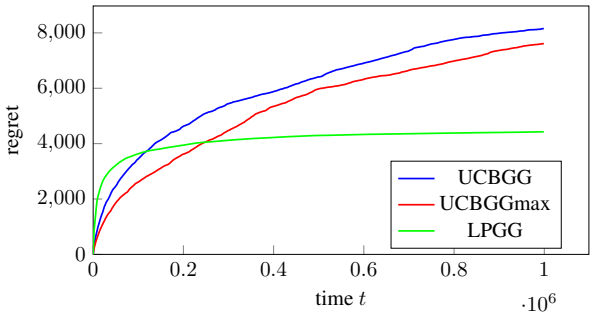

Figure 5: Robotic application in California road dataset.

The second application involves a recommendation system that recommends different products in Amazon to users and receives feedback from users on the chosen item and similar products. The rewards are independently sampled from unknown distributions associated with the products. The system may be subject to certain rules that constrain the recommendations, which can be described as the graph constraint. We conduct numerical experiments on the dataset of products in Amazon (Leskovec and Krevl 2014) and present our results in Figure 6. The results indicate that the system can achieve sub-linear regrets in the simulations.

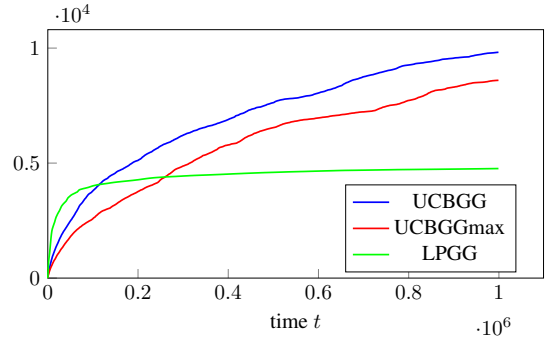

Figure 6: Recommendation system in Amazon dataset.

References

Abbasi-Yadkori, Y.; Pál, D.; and Szepesvári, C. 2011. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24.

Alon, N.; Cesa-Bianchi, N.; Dekel, O.; and Koren, T. 2015. Online learning with feedback graphs: Beyond bandits. In *Conference on Learning Theory*, 23–35. PMLR.

Alon, N.; Cesa-Bianchi, N.; Gentile, C.; Mannor, S.; Mansour, Y.; and Shamir, O. 2017. Nonstochastic multi-armed bandits with graph-structured feedback. *SIAM Journal on Computing*, 46(6): 1785–1826.

Arora, R.; Marinov, T. V.; and Mohri, M. 2019. Bandits with feedback graphs and switching costs. *Advances in Neural Information Processing Systems*, 32.

Atsidakou, A.; Papadigenopoulos, O.; Caramanis, C.; Sanghavi, S.; and Shakkottai, S. 2022. Asymptotically-optimal Gaussian bandits with side observations. In *International Conference on Machine Learning*, 1057–1077. PMLR.

Besson, L.; and Kaufmann, E. 2018. What doubling tricks can and can't do for multi-armed bandits. *arXiv preprint arXiv:1803.06971*.

Buccapatnam, S.; Eryilmaz, A.; and Shroff, N. B. 2014. Stochastic bandits with side observations on networks. In *The 2014 ACM international conference on Measurement and modeling of computer systems*, 289–300.

Buccapatnam, S.; Liu, F.; Eryilmaz, A.; and Shroff, N. B. 2017. Reward maximization under uncertainty: Leveraging side-observations on networks. *arXiv preprint arXiv:1704.07943*.

Caron, S.; Kveton, B.; Lelarge, M.; and Bhagat, S. 2012. Leveraging side observations in stochastic bandits. *arXiv preprint arXiv:1210.4839*.

Cesa-Bianchi, N.; Gentile, C.; and Zappella, G. 2013. A gang of bandits. *Advances in neural information processing systems*, 26.

Chen, H.; Li, S.; Zhang, C.; et al. 2021. Understanding Bandits with Graph Feedback. *Advances in Neural Information Processing Systems*, 34: 24659–24669.

Chen, L.; Yu, Q.; Lawrence, H.; and Karbasi, A. 2020. Minimax regret of switching-constrained online convex optimization: No phase transition. *Advances in Neural Information Processing Systems*, 33: 3477–3486.

Chu, W.; Li, L.; Reyzin, L.; and Schapire, R. 2011. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 208–214. JMLR Workshop and Conference Proceedings.

Cohen, A.; Hazan, T.; and Koren, T. 2016. Online learning with feedback graphs without the graphs. In *International Conference on Machine Learning*, 811–819. PMLR.

Coquelin, P.-A.; and Munos, R. 2007. Bandit algorithms for tree search. *arXiv preprint cs/0703062*.

Cortes, C.; DeSalvo, G.; Gentile, C.; Mohri, M.; and Zhang, N. 2020. Online learning with dependent stochastic feedback graphs. In *International Conference on Machine Learning*, 2154–2163. PMLR.

Dann, C.; Mansour, Y.; Mohri, M.; Sekhari, A.; and Sridharan, K. 2020. Reinforcement learning with feedback graphs. *Advances in Neural Information Processing Systems*, 33: 16868–16878.

Dekel, O.; Ding, J.; Koren, T.; and Peres, Y. 2014. Bandits with switching costs: T 2/3 regret. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, 459–467.

Esposito, E.; Fusco, F.; van der Hoeven, D.; and Cesa-Bianchi, N. 2022. Learning on the edge: Online learning with stochastic feedback graphs. *arXiv preprint arXiv:2210.04229*.

He, Y.; and Zhang, C. 2022. Improved Algorithms for Bandit with Graph Feedback via Regret Decomposition. *arXiv preprint arXiv:2205.15076*.

Kong, F.; Zhou, Y.; and Li, S. 2022. Simultaneously learning stochastic and adversarial bandits with general graph feedback. In *International Conference on Machine Learning*, 11473–11482. PMLR.

Lattimore, T.; and Szepesvári, C. 2020. *Bandit algorithms*. Cambridge University Press.

Leskovec, J.; and Krevl, A. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

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*, 661–670.

Li, L.; Lu, Y.; and Zhou, D. 2017. Provably optimal algorithms for generalized linear contextual bandits. In *International Conference on Machine Learning*, 2071–2080. PMLR.

Li, S.; Chen, W.; Wen, Z.; and Leung, K.-S. 2020. Stochastic online learning with probabilistic graph feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 4675–4682.

Liu, F.; Buccapatnam, S.; and Shroff, N. 2018. Information directed sampling for stochastic bandits with graph feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Lu, Y.; Wu, X.; Meng, J.; Fu, L.; and Wang, X. 2018. BLAG: Bandit On Large Action Set Graph. *arXiv preprint arXiv:1809.02711*.

Lykouris, T.; Sridharan, K.; and Tardos, É. 2018. Small-loss bounds for online learning with partial information. In *Conference on Learning Theory*, 979–986. PMLR.

Lykouris, T.; Tardos, E.; and Wali, D. 2020. Feedback graph regret bounds for Thompson Sampling and UCB. In *Algorithmic Learning Theory*, 592–614. PMLR.

Mannor, S.; and Shamir, O. 2011. From bandits to experts: On the value of side-observations. *Advances in Neural Information Processing Systems*, 24.

Qiu, J.; Grace, D.; Ding, G.; Zakaria, M. D.; and Wu, Q. 2019. Air-ground heterogeneous networks for 5G and beyond via integrating high and low altitude platforms. *IEEE Wireless Communications*, 26(6): 140–148.

Rangi, A.; and Franceschetti, M. 2019. Online learning with feedback graphs and switching costs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2435–2444. PMLR.

Rayguru, M. M.; Mohan, R. E.; Parween, R.; Yi, L.; Le, A. V.; and Roy, S. 2021. An output feedback based robust saturated controller design for pavement sweeping self-reconfigurable robot. *IEEE/ASME Transactions on Mechatronics*, 26(3): 1236–1247.

Rouyer, C.; van der Hoeven, D.; Cesa-Bianchi, N.; and Seldin, Y. 2022. A Near-Optimal Best-of-Both-Worlds Algorithm for Online Learning with Feedback Graphs. *arXiv preprint arXiv:2206.00557*.

Seldin, Y.; Bartlett, P.; Crammer, K.; and Abbasi-Yadkori, Y. 2014. Prediction with limited advice and multiarmed bandits with paid observations. In *International Conference on Machine Learning*, 280–287. PMLR.

Simchi-Levi, D.; and Xu, Y. 2019. Phase transitions and cyclic phenomena in bandits with switching constraints. *Advances in Neural Information Processing Systems*, 32.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Valko, M. 2016. *Bandits on graphs and structures*. Ph.D. thesis, École normale supérieure de Cachan-ENS Cachan.

Zhang, T.; Johansson, K.; and Li, N. 2022. Multi-armed Bandit Learning on a Graph. *arXiv preprint arXiv:2209.09419*.