Multi-armed Bandit Learning on a Graph

Tianpeng Zhang$^1$ Kasper Johansson$^2$ Na Li$^1$

Abstract—The multi-armed bandit (MAB) problem is a simple yet powerful framework that has been extensively studied in the context of decision-making under uncertainty. In many real-world applications, such as robotic applications, selecting an arm corresponds to a physical action that constrains the choices of the next available arms (actions). Motivated by this, we study an extension of MAB called the graph bandit, where an agent travels over a graph to maximize the reward collected from different nodes. The graph defines the agent’s freedom in selecting the next available nodes at each step. We assume the graph structure is fully available, but the reward distributions are unknown. Built upon an offline graph-based planning algorithm and the principle of optimism, we design a learning algorithm, G-UCB, that balances long-term exploration-exploitation using the principle of optimism. We show that our proposed algorithm achieves $O(\sqrt{SDT\log(T)} + D\log(T))$ learning regret, where $|S|$ is the number of nodes and $D$ is the diameter of the graph, which matches the theoretical lower bound $\Omega(\sqrt{SDT})$ up to logarithmic factors. To our knowledge, this result is among the first tight regret bounds in non-episodic, un-discounted learning problems with known determinisitic transitions. Numerical experiments confirm that our algorithm outperforms several benchmarks.

I. INTRODUCTION

The multi-armed bandit (MAB) problem is a popular framework for studying decision-making under uncertainty [1], [2]. The MAB consists of $n$ independent arms in the simplest setting, each providing a random reward from their corresponding probability distributions. Without knowing these distributions, an agent picks one arm for each round and tries to maximize the total expected reward in $T$ rounds. The agent needs to balance between exploring arms to learn the unknown distributions and exploiting the current knowledge by selecting the arm that has provided the highest reward.

The MAB problem enjoys a wide range of applications, such as digital advertising [3], portfolio selection [4], optimal design of clinical trials [5], etc. Many online learning algorithms have been developed and analyzed, such as confidence bound (UCB) methods and Thompson Sampling [1], [2].

One limiting assumption of the classical MAB is that the agent has access to all the arms at each time and can freely switch between different arms, which is not the case for many real-world applications. For instance, when a robot explores an unknown physical environment, the location of the robot will influence the locations that the robot can visit next. Such constraints on the available arms can often be modeled as a graph not captured in the MAB framework. This paper aims to understand how the agent could balance exploration and exploitation under such graph constraints.

Specifically, we consider the problem that an agent traverses an undirected, connected graph. Upon arrival at node $s$, the agent receives a random reward $r_s$ from a probability distribution $P(s)$. The objective is to maximize the expected accumulated reward over $T$ steps. The agent knows the graph structure but not the reward distributions. We refer to this problem as the graph bandit problem. There are many applications of the graph bandit problem, including street cleaning robots trying to maximize the trash-collection efficiency for a neighborhood [6], a mobile sensor moving between different spots to find the place that receives the strongest signals, or a drone that provides internet access flying over a network of rural/suburban locations to maximize the use of the communication channels [7]. The robot typically has a map (or a graph) of the environment in these applications. Still, each location’s quality, i.e., the signal strength in the mobile sensor application or the demand for internet access/communication in the drone application, is often stochastic with unknown distributions.

The graph bandit problem can be formulated as a reinforcement learning (RL) problem by modeling the underlying process as a Markov Decision Process (MDP) where the nodes are the states, and the next available nodes are the actions. Nevertheless, our setting is of additional interest because many robotic learning problems can be formulated as graph bandit problems, like those described previously. In these applications, directly applying generic RL algorithms may ignore the structure of the graph bandit problem, which could lead to high computational costs, high sample complexity, and/or high regrets, especially for large-size problems. Therefore, it is worth considering whether any improvement in computational efficiency and learning performance can be achieved by leveraging the graph bandit problem structure.

Contributions. In this paper, we propose a learning algorithm, G-UCB, that adopts the principle of optimism by establishing Upper Confidence Bounds (UCB) for each node and carefully controlling the number of visits of the destination nodes under the planned policy. We note that G-UCB is a single-trajectory learning scheme without forcing the agent to return to initial nodes and restarting the process. Compared with state-of-the-art UCB-based generic RL algorithms, like UCRL2 [8], we make two innovations that improve computational efficiency and learning performance using graph bandit structures: i) We propose a much more efficient offline planning algorithm by formulating offline planning as the shortest path problem (Remark 1). ii) We propose a different UCB definition leading to prominent empirical performance improvement (Remark 4).

We establish a minimax regret bound $O(\sqrt{|S|TD\log(T)} +$
that matches the theoretical lower bound $\Omega(\sqrt{TS})$ up to logarithmic factors (Remark 2). To our knowledge, this result is among the first tight regret bounds in non-episodic, un-discounted learning problems with known deterministic transitions. We also establish an instance-dependent bound $O\left(\frac{S^2 \log(T)}{\Delta}\right)$, where $\Delta$ is the ‘gap’ in mean rewards between the top two nodes. Our results are tighter than the results in [8] under the graph bandit setting (Remarks 3 and 5). Given the space limit, detailed proofs of the theorems are included in our technical report [9, Appendices A and B].

Lastly, we show in simulations that our proposed algorithm achieves lower learning regret than several benchmarks, including state-of-the-art algorithms like UCRL2 [8] and UCBH [10] (Section V-A). We also demonstrate the applicability of our framework in a synthetic robotic internet provision problem (Section V-B). More numerical studies can be found in our technical report [9, Appendix F].

A. Related work

MAB. The graph bandit problem is an extension of MAB to model the case where an arm pulled at one time will constrain the next available arms. Note that if the graph is fully connected, the graph bandit problem becomes the classical MAB problem. Recent MAB work has considered the effect among arms pulled at different times. For instance, [11], [12] assume there is a cost or constraint for switching from one arm to a different arm. We could view this as a soft constraint for switching between arms, while the graph bandit models the hard constraints in switching arms.

Reinforcement Learning. The graph bandit problem considered in this paper can be formulated as a non-episodic, undiscounted RL problem with deterministic state-transition dynamics but random rewards. There has been a quickly growing literature in designing efficient RL algorithms, especially for the episodic or infinite discounted MDPs, e.g., [10], [13]–[15]. In contrast, we focus on the non-episodic, un-discounted MDPs. Under this setting, [8], [16] develop RL algorithms with near-optimal regret bounds, but our lower regret bound is lower, especially concerning the problem size, since we exploit the known transitions in graph bandit. The line of work by [17] and [13] also consider RL problems under known transitions, but these studies are under the episodic setting, which is different from us. We are aware of only one work assuming both known transition and the non-episodic setting [18]. However, they assume the reward distribution changes adversarially in time, which is more complex than our setting. Their $O(S^{3/4}T^{2/3})$ regret is understandably worse than our result due to the additional complexity.

Stochastic shortest path. Our problem is also closely related to the stochastic shortest path (SSP) problems [19]. Indeed, our offline planning algorithm is built upon a deterministic shortest-path problem. However, SSP work mainly considers the episodic setting [20], [21]. Moreover, our graph bandit problem has unique features (deterministic node transitions but random rewards) that could facilitate the online learning algorithm design, making it different from SSP problems.

Other related work. The Canadian Traveller Problem could be an interesting extension to our work [22], where the graph is not fully known in advance but is gradually discovered as the agent travels. UCT [23] and AMS [24] also use optimism to guide exploration-exploitation. They can be viewed as special cases of graph bandit, where the graph is a tree. One crucial difference is that we consider the infinite horizon undiscounted reward, while UCT/AMS consider the finite horizon or discounted reward. These studies are rather different from this paper but are good inspirations for future work.

Notation: Let $x_{t:h}$ denote the ordered sequence of entities $(x_t, x_{t+1}, ..., x_h)$. If $x_{t:h}$ are real numbers, $x_{t:h} \leq x$ (or $\geq$) means $x_i \leq x$ (or $\geq$) for all $i = t, t+1, ..., h$. For $n \leq m$, $[n : m]$ denotes the set of integers $\{i : n \leq i \leq m\}$, and for $m \geq 0$, $[m] = [0 : m]$. Both $[p, q]$ and $[p, q)$ denote the concatenation of two finite sequences $p$ and $q$. Given a set $S$, $|S|$ denotes its cardinality. For two finite sets $U$ and $V$, let $V \times U = \{(u, v) : u \in U, v \in V\}$ denote the Cartesian product between $U$ and $V$.

II. PROBLEM FORMULATION

We consider an agent traveling over an undirected, connected graph, denoted by $G = (S, E)$ with vertices $S = \{1, 2, ..., n\}$ and edges $E \subseteq S \times S$. For each $s \in S$, denote $N_s = \{v \in S : (s, v) \in E\}$ as its neighboring nodes. Note that $s \in N_s$ for all $s \in S$. Each $s \in S$ is associated with a reward distribution $P(s)$. Whenever the agent visits a node $s$, it receives a random reward, $r_s$, sampled independently from $P(s)$. We assume all rewards are bounded in $[0, 1]$. Let $\mu_s$ be the expected reward at node $s$, i.e., $\mu_s = E[r_s]$ and denote $\mu^* = \max_s \mu_s$ the highest expected reward in the graph. Without loss of generality, we assume that there is a unique node $s^*$ such that $\mu_{s^*} = \mu^*$. Given a path of nodes $s_0:T = (s_0, s_1, ..., s_T)$ with $T \geq 1$, we say that the path is admissible if $(s_t, s_{t+1}) \in E$ for all $t = 0, 1, ..., T - 1$ and define its length as $T$. If a path contains only one node, the path is also admissible with length 0. For any two nodes $(s, s')$, let $l(s, s')$ be the length of the shortest path connecting them, i.e., $l(s, s') := \min \{T : s_0:T\}$ is an admissible path with $s_0 = s, s_T = s'$

$$D := \max_{s, s' \in S} l(s, s').$$

We let $V(s_0:T)$ be the expected cumulative reward for an admissible path $s_0:T$. 

$$V(s_0:T) := E[\sum_{t=0}^{T} r_t] = \sum_{t=0}^{T} \mu_{s_t},$$

where $r_t$ is the reward collected at node $s_t$ at time $t$.

Motivated by the applications in Section I, we assume the agent knows the graph $G$ but does not know the reward distributions $P$. The goal of the agent is to travel on the graph to maximize the expected reward $V(s_0:T)$ over the period $[0 : T]$. To do so, the agent must balance the exploration and exploitation while following the graph constraints. At each time $t$, a learning algorithm $A$ decides the next admissible node that the agent should visit based on the travel and
reward histories. To evaluate the performance of the learning algorithm, we define the regret of an algorithm \( A \) as follows:

\[
R(A, s_0, T) := \mathbb{E}[T\mu^* - \sum_{t=1}^{T} r_t],
\]

where \( r_t \) is the reward at time step \( t \) resulting from executing the algorithm \( A \) and \( s_0 \) is the initial node. The definition of \( R(A, s_0, T) \) follows the standard definition of regret in the undiscounted, infinite horizon RL literature [8], [16]. It measures the difference between the highest attainable expected reward, \( T\mu^* \), and the expected accumulated reward of the executed path. We want to highlight that we consider the online learning setting, where the learning is over a single trajectory without restarting the learning process by forcing the agent to return to \( s_0 \). Also, the regret definition is un-discounted over the trajectory, and \( T \) could be any positive integer.

III. OFFLINE PLANNING

This section studies the offline planning problem, where the goal is to find the optimal action policy given that the mean rewards \( \{\mu_s : s \in S\} \) are known. State-of-the-art RL algorithms [8], [16] often use value iteration methods in their planning components. However, it is well-known that value iterations are computationally expensive since the number of iterations can be very large. In this section, we propose a much more efficient planning algorithm by formulating the offline planning problem as a shortest path (SP) problem. This planning algorithm will be used in our learning algorithm.

Define the expected loss/cost of visiting a node \( s \) as

\[
c_a := \mu^* - \mu_a = \mathbb{E}(\mu^* - r_s).
\]

Let \( \pi : S \rightarrow S \) be a policy that maps one node \( s \) to a neighboring node \( \pi(s) \in N_s \). We consider an infinite-time-horizon un-discounted offline planning problem as follows.

\[
\min_{\pi} R_\infty(\pi, s_0) := \lim_{T \to \infty} \sum_{t=0}^{T} c_{s_t} \quad \text{such that } s_{t+1} = \pi(s_t) \in N_{s_t}, \forall t \in \{0, 1, 2, \ldots\}
\]

Note that any policy \( \pi \) that transits from \( s_0 \) to \( s^* \) in a finite number of steps has a finite \( R_\infty(\pi, s_0) \) value, so Problem (4) has a finite optimal value. Furthermore, for any path that does not end up at \( s^* \), \( R_\infty(\pi, s_0) \) is infinite. Hence, the optimal path ends up at \( s^* \). We could bound the optimal value of \( R_\infty(\pi, s_0) \) by \( D \) as shown in the following Lemma.

**Lemma 1.** The optimal value of Problem (4) is upper bounded by \( D \) where \( D \) is the diameter of the graph \( G \) as in (1).

**Proof.** The proof is straightforward by noticing that the shortest path policy \( \pi \) with \( s^* \) being the destination node incurs a bounded cost \( R_\infty(\pi, s_0) \leq D \) regardless of \( s_0 \). This is because \( r_s \in [0, 1] \) for every node \( s \) and \( D \) is the graph diameter.

By noting that the cost at \( s^* \) is 0, i.e., \( c_{s^*} = 0 \), it is not difficult to see that Problem (4) could be viewed as a deterministic shortest-path problem [19] with the destination node being \( s^* \). Algorithm 1 shows the pseudo-code for solving this shortest-path problem. We define a weighted directed graph \( \hat{G} = (S, \hat{E}, D) \) based on the original graph \( G \), where \( \hat{E} \) are the directed edges converted from the un-directed edges in \( E \). \( D \) stands for the distances for \((s, s') \in \hat{E}\), where the distance from node \( s \) to a neighboring node \( s' \in N_s \) is \( a_{ss'} := c_{ss'} \). We define the total distance of an admissible path as the sum of distances between the consecutive nodes. The optimal policy \( \pi_{\text{SP}} \) to Problem (4) is then the shortest path policy to \( s^* \) on \( \hat{G} \), which can be computed using any shortest-path algorithm such as Dijkstra’s algorithm and Bellman-Ford algorithm [25]. It is worth noting that \( \pi_{\text{SP}} \) must take the agent to \( s^* \) in no greater than \(|S|\) steps since it cannot revisit any sub-optimal nodes.

**Algorithm 1** Offline SP planning algorithm

**Input:** Graph \( G = (S, E) \), mean reward vector \( \mu = (\mu_1, \mu_2, \ldots, \mu_{|S|}) \)

**Output:** Policy \( \pi_{\text{SP}} : S \rightarrow S \)

1. \( \mu^* \leftarrow \max(\mu_1, \ldots, \mu_{|S|}) \)
2. \( s^* \leftarrow \arg \max_s (\mu_1, \ldots, \mu_{|S|}) \)
3. Define distance \( a_{ss'} := c_{ss'} = \mu^* - \mu_{s'} \) for all \((s, s') \in E\)
4. Let \( \hat{G} = (S, \hat{E}, D) \) be a directed version of \( G \) where \( D \) stands for the distances for \((s, s') \in \hat{E}\).
5. \( \pi_{\text{SP}} \leftarrow \text{Dijkstra}(\hat{G}, s^*) \), or Bellman-Ford(\( \hat{G}, s^* \))

**Remark 1.** The computation complexity of SP algorithms is \( O(|S| \cdot |E|) \) in the worst case, but in practice, the total computation is often much less. The technical report [9, Appendix F-B] shows that our algorithm under SP planning could run 1.6–11 times faster than the UCRL2 algorithm under value iteration.

IV. OPTIMISTIC GRAPH BANDIT LEARNING

This section presents a learning algorithm in graph bandit, where the agent knows the graph structure but not the reward distributions. The algorithm is built on the offline SP planning algorithm and the principle of optimism. We demonstrate that harnessing the graph bandit problem structure properly makes learning more efficient than generic RL algorithms.

**A. The Algorithm**

Algorithm 2 presents the pseudo-code for the online learning algorithm. Although the algorithm runs in a single trajectory without restarts, we divide the time steps into episodes with growing lengths for conceptual convenience. We define the UCB (upper-confidence-bound) value for each node according to the following rule.

\[
U_{m-1}(s) = \bar{\mu}_{m-1}(s) + \sqrt{\frac{2 \log(t_m)}{n_{m-1}(s)}},
\]

where \( t_m \) is the total number of time steps taken by the agent from episode 0 to \( m - 1 \) (episode 0 is the initialization), \( \bar{\mu}_{m-1}(s) \) is the average reward at \( s \) and \( n_{m-1}(s) \) is the number of samples of at node \( s \) during the same period.

The algorithm initializes the UCB values by visiting all nodes at least once. One possible method is to visit the nodes
Algorithm 2 G-UCB: Graph-UCB Algorithm

Input: The initial node \( s_0 \). The offline SP planning algorithm SP in Algorithm 1 that computes the optimal policy defined given the graph \( G \) and a set of reward values \( \{\mu_s\}_{s \in S} \) for each node: \( \pi \leftarrow \text{SP}(G, \{\mu_s\}_{s \in S}) \).

1: \( m \leftarrow 0 \)
2: Follow any path that visits all nodes at least once. //Initialize \( U_0 \),
3: Place the agent at \( s_0 \). \( s_{curr} \leftarrow s_0 \).
4: while The agent hasn’t received a stopping signal do
5: \( m \leftarrow m + 1 \)
6: Calculate the UCB values \( U_{m-1}(s) \) for all \( s \in S \) according to (5).
7: \( \pi_m \leftarrow \text{SP}(G, \{U_{m-1}(s)\}_{s \in S}) \)
8: while \( U_{m-1}(s_{curr}) < \max_s U_{m-1}(s) \) do
9: Execute \( \pi_m \) for one step. \( s_{curr} \leftarrow \pi_m(s_{curr}) \)
10: Collect reward at \( s_{curr} \).
11: end while
12: \( \text{dest}_m \leftarrow s_{curr} \). Keep collecting rewards at node \( \text{dest}_m \) until the number of its samples doubles compared to its sample number at the beginning of episode \( m \). //This operation ensures \( n_m(\text{dest}_m) = 2n_{m-1}(\text{dest}_m) \).
13: end while

Theorem 1 states the theoretical guarantee on Algorithm 2’s learning regret.

**Theorem 1.** Let \( T \geq 1 \) be any positive integer. Recall that \( |S| \) is the number of nodes, and \( D \) is the diameter of the graph \( G \) defined in (1). The regret of \( G-\text{UCB} \) (Algorithm 2), after taking \( T \) steps beyond the initialization is bounded by

\[
R^*(G-\text{UCB}, s, T) \leq O\left(\sqrt{|S|T \log(T)} + D|S| \log(T)\right)
\]

Note if furthermore \( T \geq D^2|S| \log(T) \), the above reduces to

\[
R^*(G-\text{UCB}, s, T) \leq O\left(\sqrt{|S|T \log(T)}\right)
\]

**Proof Sketch:** We describe the outline of the proof below. The detailed proof is in the technical report [9, Appendix A]. Define the random variable \( M \) as the number of episodes at the cutoff time \( T \). The doubling operation in line 12 ensures \( M \) is logarithmic in \( T \) almost surely, i.e., \( M \leq O(|S| \log(T)) \).

For episode \( m \geq 0 \), we define the clean event \( \mathcal{E}_m = \{\forall s \in S, \mu_s \in [\mu_m(s) - \text{rad}_m(s), \mu_m(s) + \text{rad}_m(s)]\} \), where \( \text{rad}_m(s) = \sqrt{\frac{2 \log(t_m)}{n_m(s)}} \) are the confidence radii. Define \( \sim \mathcal{E}_m \) as the bad event. We can decompose the regret via the clean/bad event distinction and use standard arguments involving Hoeffding’s inequality(such as in [1]) to show that the probabilities of the bad events are so small that the regret contributed by the bad events is bounded by \( O(\log(M)) = O(\log(|S|) + \log(\log(T))) \).

The regret under the clean events consists of two terms:

\[
2 \sum_{m=1}^{M} \sum_{t=1}^{H_m} \text{rad}_m(s_{t+m}) + \sum_{m=1}^{M} H_m \mu^* - U_{m-1}(s_{t+m})
\]

\[
= \sum_{m=1}^{M} \sum_{t=1}^{H_m} \mu^* - U_{m-1}(s_{t+m})
\]

Here, \( c_m(s) \) denotes the number of samples at node \( s \) during episode \( m \). The summation \( \sum_{s \in S} \frac{c_m(s)}{\sqrt{n_{m-1}(s)}} \) can be bounded using the inequality in [8, Appendix C.3], which states that for any sequence of numbers \( z_1, z_2, \ldots, z_n \) with \( 0 \leq z_k \leq Z_{k-1} := \max\{1, \sum_{i=1}^{k-1} z_i\} \), there is

\[
\sum_{k=1}^{n} \frac{z_k}{\sqrt{Z_{k-1}}} \leq (\sqrt{2} + 1)Z_n
\]

The proof of the eq. (8) is in the technical report [9, Appendix C]. In our case, \( c_{i-1}(s) \) corresponds to \( z_i \), while \( n_{i-1}(s) \) corresponds to \( Z_i \). Since the \( \text{SP} \) policy does not visit a suboptimal node twice, while the doubling operation ensures the samples at \( \text{dest}_m \) exactly double, there must be \( c_m(s) \leq n_{m-1}(s) \). Therefore we can apply eq. (8) and Jensen’s inequality to get

\[
\sum_{m=1}^{M} \sum_{s \in S} \frac{c_m(s)}{\sqrt{n_{m-1}(s)}} \leq \sum_{s \in S} \frac{c_{m}(s)}{\sqrt{n_{m-1}(s)}} \leq (\sqrt{2} + 1)\sqrt{|S|T}
\]

So the first term is upper bounded by \( O(\sqrt{|S|T \log(T)}) \).

The second component, ‘the cost of destination switch’, measures the regret from visiting sub-optimal nodes during the transit to \( \text{dest}_m \) in episode \( m \). It can be shown that such transit induces at most \( O(D) \) regret per episode. So the total cost of the destination switch is bounded by \( O(MD) = O(D|S| \log(T)) \). The bound in the theorem is reached after combining all the components.

**Remark 2 (Tightness of the result).** In classical MAB, it is known that the regret is lower bounded by \( \Omega(\sqrt{KT}) \) [1], [26], where \( K \) is the number of arms. Note that on a fully
connected graph, the graph bandit becomes equivalent to the MAB problem. Therefore, \(\Omega(\sqrt{|S|T})\) is a lower bound to the graph bandit problem, and our regret \(O(\sqrt{|S|T}\log(T))\) matches the lower bound up to logarithmic factors. This means our regret is the best possible.

**Remark 3** (Comparison to state-of-the-art RL results). Our result shows the benefit of utilizing the graph structure compared to state-of-the-art generic RL algorithms. A \(O(D|S|\sqrt{|A|T\log(T)})\) regret bound is established in the general RL setting in [8], where \(|S|, |A|\) are the number of states and actions. This regret has a considerably worse dependence on \(|S|\) than our regret bound \(O(\sqrt{|S|T}\log(T))\), which is because their analysis has not utilized the graph bandit problem structures. We found that, if assuming known transitions, their regret can be improved to \(O((D + \sqrt{|S|A})\sqrt{T\log(T)})\) since the first term in their regret Eq. (19) vanishes. This regret is still worse than our \(O(\sqrt{|S|T}\log(T))\) regret due to the additional \(\sqrt{|A|}\) factor, but we believe it could be further improved if their analysis carefully utilizes other graph bandit properties. Meanwhile, [16] establishes a \(O(D\sqrt{|S|A}|T\log(T))\) regret. But in contrast to our work, they assume known rewards and unknown transition dynamics thus it is unclear what their regret will become under our settings.

**Remark 4** (Choice of Bonus function in UCB). The UCB in [8] is defined as

\[
\tilde{U}_{m-1}(s) = \tilde{\mu}_{m-1}(s) + \sqrt{\frac{7\log(2SA\tilde{m}/\delta)}{2n_{m-1}(s)}}, \tag{9}
\]

where \(S, A\) are the number of states and actions, and \(\delta \in (0, 1]\) is a confidence parameter. The UCB definition above has an explicit dependence on \(S\) and \(A\), which is absent in our UCB(5)). This dependence on \(S\) and \(A\) arises in the confidence radius due to the analysis under the general RL setting, but our analysis under the graph bandit setting does not hint at such dependence. Directly applying this UCB to graph bandit should result in more exploration and higher regret than our UCB. This intuition is confirmed in the numerical experiments of our technical report [9, Appendix F-C].

We also derive a regret bound that depends on the gap \(\Delta\) between the best and second-best mean rewards; see Theorem 2. If all mean rewards are not the same, we define \(\Delta := \mu^* - \hat{\mu}\), where \(\hat{\mu}\) is the second best mean reward. We also define \(L\) as the maximal length of cycle-free paths in the graph. Note that \(D \leq L \leq |S|\). Under the same assumptions as Theorem 1, the regret of G-UCB (Algorithm 2) is bounded by.

\[
R^*(G-UCB, s, T) \leq O\left(|S|(L - 1)\log(T) + \log\left(\frac{1}{\Delta}\right) + \frac{|S|\log(T)}{\Delta}\right)
\]

See the technical report [9, Appendix B] for detailed proof.

**Remark 5**. Instance-dependent bounds have been previously investigated under similar settings. In particular, the UCRL2 algorithm achieves regret \(O\left(D^\gamma|S|^\Delta\log(T)\right)\) [8] on connected graphs. If ignoring the \(\log\log(T)\) and \(\log\left(\frac{1}{\Delta}\right)\) terms, our results have much better dependence on \(|S|\) and \(D\) than [8].

V. NUMERICAL EXPERIMENTS

A. Benchmark Experiments

The simulation code can be found at [27]. Due to the space limit, we defer most numerical experiments to our technical report [9, Appendix F]. Here, we only use the experiment on a \(10 \times 10\) grid graph as an example of demonstration, while the results for other graph types are similar and can be found in the technical report [9, Appendix F-A]. We compare our proposed algorithm to the following benchmark algorithms.

- Local UCB and Local TS. These benchmarks always move to the node with the highest UCB or posterior sampling value in the current neighborhood.
- QL-\(\epsilon\)-greedy [28] and QL-UCB-H [10]. Q-learning algorithms with \(\epsilon\)-greedy exploration and state-of-the-art Hoeffding-style exploration bonus.
- UCRL2. For a fair comparison, we adjusted the original UCRL2 algorithm [8] so that this benchmark knows the deterministic transition dynamics.

We run 100 simulations for each algorithm on the \(10 \times 10\) grid graph, with \(T = 2 \times 10^4\) steps per simulation. We initialize the mean rewards from \(\mu_s \sim U(0.5, 9.5)\), for each node \(s \in S\). In the simulations the reward distributions are defined as \(P(s) = U(\mu_s - 0.5, \mu_s + 0.5)\), \(\forall s \in S\). Fig. 1 demonstrates a clear advantage of G-UCB over the benchmarks. UCRL2 performs much better than other benchmarks but is still not as good as G-UCB. The technical report [9, Appendix F-C] shows that the advantage of G-UCB over UCRL2 is mostly due to our more proper definition of UCB as in Remark 4.

![Fig. 1: Learning regrets of the benchmarks on the grid graph. All regret curves in the figure indicate the average across 100 simulations, and error bars indicate one standard deviation. We plot the error bars only for G-UCB for visualization clarity.](image)

B. Synthetic robotic application

This section presents a synthetic robotic application modeled as a graph bandit problem. Consider a drone traveling
over a network of rural/suburban locations to provide internet access, as illustrated in Fig. 2a. The robot serves one location per period. The reward for serving the location is the communication traffic carried by the robot in gigabytes, sampled independently from a distribution associated with the location. The robot may stay at the current location or move to a neighboring location for the next period. It has a map of its service area but does not know the reward distributions. The goal is to maximize the total reward before the robot is called back and recharged. We perform numerical experiments on the graph defined in Fig. 2a, representing ten counties with lower than 50% broadband coverage in AR, USA. We run our algorithm for 100 simulations on this graph with the reward distributions initialized as in Fig. 1. Fig. 2b shows the robot consistently achieves sub-linear regrets in the simulations.

VI. CONCLUSION AND FUTURE WORK

This paper studies an extension of the MAB problem: the graph bandit problem, where the current selection determines the set of available arms through a graph structure. We show that offline planning can be converted into a shortest-path problem and solved efficiently. Based on the offline planning algorithm and the principle of optimism, we develop a UCB-based graph bandit learning algorithm, G-UCB, and establish a minimax regret bound for the algorithm that matches the theoretical lower bound. Extensive numerical experiments verify our theoretical findings and show that our algorithm achieves better learning performance than the benchmarks. Our synthetic robotic application illustrates the real-world applicability of the graph bandit framework. Future directions include introducing time-varying reward distributions and/or extensions to the multi-agent settings.

REFERENCES

[1] A. Slinkin et al., “Introduction to multi-armed bands,” Foundations and Trends® in Machine Learning, vol. 12, no. 1-2, pp. 1–286, 2019.
[2] T. Lattimore and C. Szepesvári, Bandit algorithms. Cambridge University Press, 2020.
[3] E. M. Schwartz, E. T. Bradlow, and P. S. Fader, “Customer acquisition via display advertising using multi-armed bandit experiments,” Marketing Science, vol. 36, no. 4, pp. 500–522, 2017.
[4] X. Huo and F. Fu, “Risk-aware multi-armed bandit problem with application to portfolio selection,” Royal Society open science, vol. 4, no. 11, p. 171377, 2017.
[5] S. S. Villar, J. Bowden, and J. Wason, “Multi–arm bandit models for the optimal design of clinical trials: benefits and challenges,” Statistical science: a review journal of the Institute of Mathematical Statistics, vol. 30, no. 2, p. 199, 2015.
[6] M. M. Rayguru, R. E. Mohan, R. Parween, L. Yi, A. V. Le, and S. Roy, “An output feedback based robust saturated controller design for pavement sweeping self-reconfigurable robot,” IEEE/ASME Transactions on Mechatronics, vol. 26, no. 3, pp. 1236–1247, 2021.
[7] J. Qiu, D. Grace, G. Ding, M. D. Zakaria, and Q. Wu, “Air-ground heterogeneous networks for 5G and beyond via integrating high and low altitude platforms,” IEEE Wireless Communications, vol. 26, no. 6, pp. 140–148, 2019.
[8] T. Jaksch, R. Ortner, and P. Auer, “Near-optimal regret bounds for reinforcement learning,” Journal of Machine Learning Research, vol. 11, no. 51, pp. 1563–1600, 2010. [Online]. Available: http://jmlr.org/papers/v11/jaksch10a.html
[9] T. Zhang, K. Johansson, and N. Li, “Multi–armed bandit learning on a graph,” 2022. [Online]. Available: https://arxiv.org/abs/2209.09419
[10] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, “Is q-learning provably efficient?” in Advances in Neural Information Processing Systems, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.
[11] A. Slivkins, “Phase transitions and cyclic phenomena in bandits with switching constraints,” Advances in Neural Information Processing Systems, vol. 32, 2019.
[12] L. Chen, Q. Yu, H. Lawrence, and A. Karbasi, “Minimax regret of switching-constrained online convex optimization: No phase transition,” Advances in Neural Information Processing Systems, vol. 33, pp. 3477–3486, 2020.
[13] A. Zimin and G. Neu, “Online learning in episodic markovian decision processes by relative entropy policy search,” in Advances in Neural Information Processing Systems, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013.
[14] J. He, D. Zhou, and Q. Gu, “Nearly minimax optimal reinforcement learning for discounted mdps,” Advances in Neural Information Processing Systems, vol. 34, pp. 22288–22300, 2021.
[15] M. G. Azar, I. Osband, and R. Munos, “Minimax regret bounds for reinforcement learning,” in Proceedings of the 34th International Conference on Machine Learning - Volume 70, ser. ICML’17. JMLR.org, 2017, p. 263–272.
[16] S. Agrawal and R. Jia, “Optimistic posterior sampling for reinforcement learning: worst-case regret bounds,” Advances in Neural Information Processing Systems, vol. 30, 2017.
[17] G. Neu, A. György, C. Szepesvári et al., “The online loop-free stochastic shortest-path problem,” in COLT, vol. 2010. Citeseer, 2010, pp. 231–243.
[18] G. Neu, A. Antos, A. György, and C. Szepesvári, “Online markov decision processes under bandit feedback,” in Advances in Neural Information Processing Systems, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23. Curran Associates, Inc., 2010.
[19] D. P. Bertsekas, “Dynamic programming and optimal control 4th edition, volume ii,” Athena Scientific, 2015.
[20] A. Rosenberg, A. Cohen, Y. Mansour, and H. Kaplan, “Near-optimal regret bounds for stochastic shortest path,” in International Conference on Machine Learning. PMLR, 2020, pp. 8210–8219.
[21] J. Tarbouriech, R. Zhou, S. S. D. Mirotta, M. Valko, and A. Lazaric, “Stochastic shortest path: Minimax, parameter-free and towards horizon-free regret,” Advances in Neural Information Processing Systems, vol. 34, pp. 6843–6855, 2021.
[22] E. Nikolova and D. R. Karger, “Route planning under uncertainty: The canadian traveller problem,” in AAAI, 2008, pp. 969–974.
[23] P.-A. Coquelin and R. Munos, “Bandit algorithms for tree search,” arXiv preprint cs/0703002, 2007.
[24] H. S. Chang, M. C. Fu, J. Hu, and S. I. Marcus, “An adaptive sampling algorithm for solving markov decision processes,” Operations Research, vol. 53, no. 1, pp. 126–139, 2005.
[25] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to algorithms. MIT press, 2022.
[26] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, “The nonstochastic multiarmed bandit problem,” SIAM Journal on Computing, vol. 32, no. 1, pp. 48–77, 2002. [Online]. Available: https://doi.org/10.1137/S0097539799367375
[27] T. Zhang, “Graph-Bandit,” 2022. [Online]. Available: https://github.com/lina-robotics-lab/Graph-Bandit
[28] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.