2024 American Control Conference (ACC)
July 8-12, 2024. Toronto, Canada

Cooperative Multi-Agent Graph Bandits: UCB Algorithm and Regret Analysis

Phevos Paschalidis Runyu Zhang Na Li

Abstract— **In this paper, we formulate the multi-agent graph bandit problem as a multi-agent extension of the graph bandit problem introduced in [1]. In our formulation, N cooperative agents travel on a connected graph G with K nodes. Upon arrival at each node, agents observe a random reward drawn from a node-dependent probability distribution. The reward of the system is modeled as a weighted sum of the rewards the agents observe, where the weights capture some transformation of the reward associated with multiple agents sampling the same node at the same time. We propose an Upper Confidence Bound (UCB)-based learning algorithm, `Multi-G-UCB`, and prove that its expected regret over T steps is bounded by $O(\gamma N \log(T)[\sqrt{KT}+DK])$, where D is the diameter of graph G and γ a boundedness parameter associated with the weight functions. Lastly, we numerically test our algorithm by comparing it to alternative methods.**

I. Introduction

The Multi-Armed Bandit (MAB) problem is a fundamental problem in the study of decision-making under uncertainty [2], [3], [4], [5]. In its simplest formulation, the MAB consists of K independent arms each with an associated reward probability distribution and a single agent that plays the MAB instance for T turns. On each turn, the agent selects one of the arms and receives a reward drawn from the arm's probability distribution. The agent's goal is to maximize the expected reward received over the course of its T turns. In order to do so, the agent must balance *exploring* arms it knows little about and *exploiting* arms already demonstrated to give high rewards.

One important limitation of the MAB framework is the assumption that at each time step the agent has access to all arms regardless of its previous action. Earlier work introduces a graph bandit problem where the agent traverses an undirected, connected graph, receiving random rewards upon arrival at each node [1]. The graph's nodes thus represent the K arms and the local connectivity between nodes imposes restrictions on which arms can be played sequentially. This framework has applications in robotics: consider a street-cleaning robot exploring an unknown physical environment or a mobile sensor covering some space to find the location that receives the strongest signals. However, [1] only examines the case of a single agent, whereas in most robotic applications it is useful to consider multiple agents exploring and exploiting the unknown graph together.

P. Paschalidis, R. Zhang, and N. Li are affiliated with Harvard J. Paulson School of Engineering and Applied Sciences. ppaschalidis@college.harvard.edu, runyuzhang@fas.harvard.edu, nali@seas.harvard.edu.

This work was funded by NSF AI institute: 2112085, NSF CNS: 2003111, NSF ECCS: 2328241, and Harvard College Research Program.

There are many works extending the standard MAB problem to a multi-agent setting. In most studies, agents observe an identical environment where their actions are independent of one another and there exist restrictions on agent communication [6], [7], [8], [9], [10], [11], [12], [13]. Other studies model the multi-agent extension such that two agents selecting the same action "collide" and observe no reward [14], [15], [16], [17]. Another line of related works focuses on the combinatorial bandit setting, where a single centralized decision maker chooses a "super-arm" from some feasibility set $\mathcal{S} \subseteq 2^{[K]}$ and receives a function of the random samples drawn from the arms that comprise the super-arm as a reward [18], [19], [20], [21], [22]. Intuitively, we could also consider each individual arm in the super-set as being chosen by a different agent, thus defining the combinatorial bandit framework as a type of multi-agent MAB problem. Importantly, however, none of the multi-agent or combinatorial bandit works consider a graph-like restriction on agent transitions.

Our Contributions. Motivated by the discussion above, we formulate the multi-agent graph bandit problem as a viable multi-agent extension of the graph bandit problem. Specifically, we consider N cooperative agents co-existing on the same undirected, connected graph G with the objective of optimizing a total reward that is a weighted sum of the rewards of arms that agents select, where the weight associated with each arm depends on the number of agents selecting it. We then propose a learning algorithm, `Multi-G-UCB`, which uses an Upper Confidence Bound (UCB)-based approach to managing the exploration/exploitation trade-off. In particular, the algorithm maintains a confidence radius for each arm that shrinks proportionally to the number of times the arm has been sampled and optimistically assumes that the mean reward for each arm is the maximum value within this radius. We show that `Multi-G-UCB` achieves a regret upper bounded by $O\left(\gamma N \log(T)\left[\sqrt{TK}+DK\right]\right)$, where D represents the diameter of the graph G and γ is a boundedness coefficient for the weight functions. We also demonstrate via simulations that our proposed algorithm works well empirically, achieving lower regret than several important benchmarks including a multi-agent version of G-UCB where each agent runs the G-UCB algorithm in isolation.

The remainder of the paper is organized as follows. In Section II, we formulate the multi-agent graph bandit problem and provide examples of potential applications. Section III introduces the `Multi-G-UCB` algorithm, IV provides the theoretical analysis of `Multi-G-UCB`, and V details the experimental simulation result.

979-8-3503-8265-5/$31.00 ©2024 AACC

II. Problem Formulation

A. Multi-Agent Graph Bandit

We define the multi-agent graph bandit problem as follows. Consider an undirected, connected graph with self-loops $G = ([K], \mathcal{E})$ and diameter D, where the diameter of a graph refers to the maximum distance between any two of its nodes. For each node $k \in \{1, \dots, K\} = [K]$, there exists an associated probability distribution for its reward, $P_k(\cdot)$, with support $[0,1]$ and mean μ_k; the nodes of the graph thus represent the K arms of the bandit problem. Now, let there exist N individual agents, each having knowledge of the graph G and able to communicate with the other agents. Let T be an integer representing the number of steps in the multi-armed bandit problem. We initialize each agent i to start on some initial node $k_{i,0}$. Then, at each time step $t \leq T$, each agent i chooses an arm $k_{i,t}$ such that $(k_{i,t-1}, k_{i,t}) \in \mathcal{E}$. That is, the agent traverses an edge in the graph to a neighboring node/arm. For each time step t, we define $c_{k,t}$ as a count of the number of agents sampling node k at time t, and further define $C_t = (c_{1,t}, \dots, c_{K,t})$. Then, for each k such that $c_{k,t} > 0$, a reward $X_{k,t}$ is drawn independently from $P_k(\cdot)$. The algorithm observes a system-wide reward of

$$R_t := \sum_{k \in [K]} f_k(c_{k,t}) X_{k,t},$$

where for each arm k, f_k is defined such that $f_k(1) = 1$ and $f_k(c) \leq \gamma \cdot c$ for some constant γ independent of c and k. The f_k's thus represent some general transformation of the reward for multiple selections of the same arm at the same time. Note that f_k is not necessarily equivalent to $f_{k'}$ for $k \neq k'$. At the end of each time step t, $X_{k,t}$ is revealed for all arms k with $c_{k,t} > 0$. The goal of the N agents is thus to travel the graph so as to maximize the net expected reward received by the system of agents over the course of the T time steps.

Equivalently, we can define the goal as a minimization of expected regret. In order to do so, we first need to define the optimal allocation of agents over the graph. Importantly, since the rewards the system observes at each node are dependent only on the number of agents sampling the node and not the location of specific agents, it suffices to specify the optimal allocation only in terms of the count of agents at each node. We can thus define the set of possible agent allocations as $\mathcal{C} = \{C = (c_1, \dots, c_K) : \sum_{k \in [K]} c_k = N\}$ and the optimal allocation

$$C^* := \arg\max_{C \in \mathcal{C}} \sum_{k \in [K]} f_k(c_k) \mu_k$$

as the count vector that maximizes the expected reward of the system. We can then define the regret of the system at time t as

$$\mathcal{R}_t := \sum_{k \in [K]} f_k(c_k^*) \mu_k - R_t.$$

B. Examples

In this section, we provide two motivating examples that fit our formulation.

1) Drone-Enabled Internet Access: Consider N cooperative drones deployed over a network of K rural communities to provide internet access. Each drone can serve only one community per time period and, between time periods, only has time to move from its current location to an adjacent community. At the end of each time period, the fleet of drones receives a reward proportional to the communication traffic it serviced. Importantly, for any time period, the demand of each community is sampled independently from a distribution associated with that location; the demand of a community at any time step is stochastic, but different communities have different average internet needs. If multiple agents are positioned over the same community, the reward they observe is modified. It may be the case that the marginal benefit decreases with each additional agent, in which case the f_k's would be concave. Our formulation also extends to more complicated interactions, though, for example with agents that can amplify each other's effect if positioned over the same community. The goal of the drone fleet is to maximize the total reward before the robots' batteries run out and they are recalled. Note that this problem is also related to the "sensor coverage" formulations of [23], [24], [25], [26], [27].

2) Factory Production: Consider a factory with N production lines each of which can manufacture a total of K different products, and assume that there are restrictions regarding which products can be manufactured sequentially (depending on the raw materials required, for example). Then, we can also model this problem as an instance of the multi-agent graph bandit problem where each production line is an agent and the actions they take are the products they choose to manufacture at each time step. Each product k is associated with some reward dependent on its stochastic demand market as represented by the reward probability distribution $P_k(\cdot)$. The more production lines that choose to manufacture product k, the greater the supply which negatively affects the price of the commodity; thus we have diminishing marginal return for additional lines producing k which can be modelled with concave choice of $f_k(\cdot)$ (though, again, the formulation also extends to more general f_k's).

A related problem in which transitioning between products is subjected not only to binary yes/no constraint but also associated with some cost can be represented by a framework extremely similar to ours with the addition of non-uniform edge weights to G. The algorithm we introduce, `Multi-G-UCB`, can be modified to address this formulation with only a slight modification to the offline planning component and the regret analysis (see Remarks 1 and 3).

III. Multi-Agent Graph Bandit Learning

In this section, we present a learning algorithm for the multi-agent graph bandit problem. In Section III-A we outline the algorithm and provide intuition for its structure; in Section III-B we explore in depth the offline planning algorithm we use as a subroutine; and in Section III-C we discuss the algorithm's initialization phase.

A. Algorithm

The formulation given in Section II introduces a few key complications with respect to the current literature. If we merely apply MAB algorithms to the graph setting, then at any time step the intended arm for an agent can be far away on the action graph G, requiring multiple time steps of sub-optimal actions as the agent traverses the path between the initial and desired arm. Furthermore, in comparison to the single-agent graph bandit setting, the existence of multiple agents and the non-linear weight functions necessitates communication and sophisticated planning since the reward of each agent is dependent on the actions of other agents.

With these in mind, we introduce the `Multi-G-UCB` algorithm which attempts to minimize this transition cost by dividing the time horizon T into E episodes and emphasizes agent communication through communal UCB values. In each episode, the algorithm performs four major steps: (i) integrating agent information to compute the UCB values for each node, (ii) calculating the desired allocation using the UCB values, (iii) transitioning to the destination allocation using an offline planning algorithm, `SPMatching`, and (iv) collecting new samples at the destination nodes until samples double.

Specifically, `Multi-G-UCB` calculates for each arm the mean observed reward $\{\hat{\mu}_{k,t_e}\}_{a\in[K]}$ as well as a confidence radius and upper confidence bound

$$b_{k,t_e} := \sqrt{\frac{2\log(t_e)}{n_{k,t_e}}} \quad \text{and} \quad U_{k,t_e} := \hat{\mu}_{k,t_e} + b_{k,t_e}, \tag{1}$$

where we denote by t_e the time at which episode e starts and by n_{k,t_e} the number of times arm k has been sampled in the first t_e time steps. Importantly, in calculating the mean reward, the algorithm uses the X_{k,t_e} drawn from the arm's reward distribution rather than the observed reward weighted by $f_k(\cdot)$ and utilizes the collective experience of all agents. Similarly, we update the sample counts as

$$n_{k,t+1} := \begin{cases} n_{k,t}, & \text{if } c_{k,t} = 0, \\ n_{k,t}+1, & \text{otherwise,} \end{cases} \tag{2}$$

so that they reflect the number of samples drawn from the reward distribution.

In order to calculate the optimal allocation of agents, `Multi-G-UCB` optimistically assumes that the true mean reward of each arm is equal to its upper confidence bound. Thus, we define the estimated optimal count vector of episode e as

$$\hat{C}_e := \arg\max_{C\in\mathcal{C}} \sum_{k\in[K]} f_k(c_k) U_{k,t_e}. \tag{3}$$

We also define k_{min} as the arm with the fewest observed samples among all arms k such that $\hat{c}_{k,e} > 0$ and $n_{min} = n_{k_{min},t_e}$ as its current number of samples. These quantities will be important in defining the length of our episode.

Following the calculation of $\hat{C}_e$, the algorithm relies on an offline planning sub-routine, `SPMatching`, defined in Section III-B to transition the system of agents from the current state to the desired one. Following this transition phase, the algorithm exploits the desired state by repeatedly sampling the (potentially suboptimal) allocation until the number of samples of the arm with the fewest prior samples, i.e. k_{min}, doubles; that is until $n_{k_{min},t} = 2n_{min}$. The doubling scheme is a well-known technique in reinforcement learning, e.g. [28]. We would like to remark that our specific choice of doubling scheme, doubling the least-sampled arm, is crucial in the regret analysis. Other schemes such as doubling the most-sampled arm would lengthen the exploitation phase upsetting the exploration/exploitation balance of the algorithm. This conclusion is further supported by the numerical results in Section V when we compare our proposed doubling scheme with two others. The pseudocode for `Multi-G-UCB` is given by Algorithm 1.

Algorithm 1 `Multi-G-UCB`:

Input: The initial nodes for each agent, $\{k_{i,0}\}_{i\in[N]}$. The offline planning algorithm `SPMatching` (see Section III-B) that computes the optimal policy given the graph G and a set of computed UCB values, $\{U_{k,t_e}\}_{k\in[K]}$, for each node.

1: $e \leftarrow 0$
2: Run the initialization algorithm to visit each vertex in G. See Section III-C.
3: **while** coordinator has not received a stopping signal **do**
4: $\quad e \leftarrow e+1$
5: $\quad$ Calculate the UCB values $\{U_{k,t_e}\}_{k\in[K]}$.
6: $\quad$ Solve the optimization problem in (3) for the desired allocation $\hat{C}_e$.
7: $\quad$ Denote by k_{min} the arm with fewest observed samples of all arms k such that $\hat{c}_{k,e} > 0$ and denote by n_{min} its current number of samples ($n_{min} = n_{k_{min},t_e}$).
8: $\quad$ Follow the offline planning policy `SPMatching` until agents are distributed according to $\hat{C}_e$.
9: $\quad$ Have each agent i continue to collect rewards at its current node until $n_{k_{min},t} = 2n_{min}$.
10: **end while**

B. Offline Planning

The goal of the offline planning subroutine is to transition from the current state to one in which the agents are distributed according to $\hat{C}_e$ while incurring the least regret possible. In the single-agent setting, the "allocation" of agents will always consist of just a single destination node, and thus this transition-of-least-regret is equivalent to a shortest path problem where the weight of each edge $(k',k) \in \mathcal{E}$ is the estimated regret of arm k. More formally, these paths—which we call *regret shortest paths*—are shortest paths on a graph $G' = ([K], \mathcal{E}, W)$ where the weights in W are individually defined as

$$w(k',k) = \max_{v\in[K]} (U_{v,t_e} - U_{k,t_e}). \tag{4}$$

Importantly, we only allow paths of maximum length D, noting that each pair of vertices is assured to have such a path by definition of our graph diameter.

In our multi-agent setting, the solution is more complicated. We propose a polynomial-time pseudo-solution, which we call `SPMatching`. In particular, we define $\mathcal{S}_e = ([K], \hat{C}_e)$ as a multiset of arms such that the multiplicity of each arm $k \in [K]$ is equal to $\hat{c}_{k,e}$. We then create a complete bipartite graph $G_B = ([N], \mathcal{S}_e, \mathcal{E}_b)$ where the weight of each $(i,k) \in \mathcal{E}_b$ is the length of the regret shortest path between the current location of agent i and arm k. We can then calculate the minimum weighted perfect matching of G_B to assign agents a corresponding arm in $\mathcal{S}_e$ and instruct each agent to follow the shortest path towards that arm [29]. The pseudocode for `SPMatching` is given in Algorithm 2.

We call `SPMatching` a pseudo-solution because there are no theoretical guarantees on its minimization of regret during the transition phase. Our restriction on the physical path length of any regret shortest path means that our paths may be suboptimal, but even without this constraint the regret shortest paths do not account for the actions of the other agents. In particular, if the optimal paths of two agents intersect at some node, then the reward observed at that node is potentially less than the reward estimated by the shortest path calculations. This increases regret in a manner unanticipated by the offline planning algorithm. As we see in Section IV, the sub-optimality of our transition phase does not adversely affect our regret since its worst-case scenario is equivalent to the worst-case of an optimal solution.

Algorithm 2 `SPMatching`

Input: The current time t, the current position of the agents, $\{k_{i,t}\}_{i\in[N]}$, and a desired allocation of agents, $\hat{C}_e$.

1: Calculate the multiset of destination arms $\mathcal{S}_e = ([K], \hat{C}_e)$.
2: For each agent m calculate the regret shortest path between its current location and each of the destination arms.
3: Initialize a complete bipartite graph $G_B = ([N], \mathcal{S}_e, \mathcal{E}_B)$ where the weight of each edge $(i,k) \in \mathcal{E}_B$ is defined as the length of the regret shortest path between the current location of i and the destination arm k.
4: Calculate the minimum weight perfect matching of G_B.
5: Have each agent follow the regret shortest path to their assigned destination node.

Remark 1 (Algorithmic Extension to Weighted Graphs)**.** *In order to extend `Multi-G-UCB` to the case of weighted arm graphs G as discussed in Section II-B.2 we would need only to change the definition of the regret shortest paths. In particular, if the cost of transition from arm k' to arm k was $\alpha(k',k)$, then by redefining the weight of edge (k',k) in G' as $w(k',k) = \alpha(k',k) + \max_{v\in[K]}(U_{v,t_e} - U_{k,t_e})$ we fully account for these transition costs.*

C. Initialization

In order for the algorithm to begin in earnest, it needs to have observed at least one reward from each arm so that the upper confidence bounds are well-defined. Therefore, we begin the algorithm with an initialization episode in which each agent runs an independent graph exploration algorithm, Depth-First Search (DFS), until all vertices have been visited at least once [30]. Despite the potential redundancy, we will see in Section IV that the initialization phase does not contribute to the asymptotic regret of the system.

Remark 2 (Algorithmic Extension to Directed Graphs)**.** *We remark that the initialization phase is the only time that our algorithm uses the assumption that the original graph G is undirected. This assumption is necessary for the DFS algorithm, which relies on backtracking. Therefore, given a different initialization technique, `Multi-G-UCB` could also be applied to a multi-agent graph problem with a directed G which would reflect asymmetric transition constraints.*

IV. Main Results

Having defined our algorithm, we now turn to a theoretical analysis of its regret. We summarize our findings in the Theorem 1.

Theorem 1 (Regret of `Multi-G-UCB`)**.** *Let $T \geq 1$ be any positive integer. Given an instance of a multi-agent graph bandit problem with N agents, K arms each with weight function $f_k(\cdot)$ bounded such that $f_k(c) \leq \gamma c$, and a graph diameter of D, the expected system-wide regret of `Multi-G-UCB` after taking a total of T steps (including initialization) is bounded by*

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathcal{R}_t\right] \leq O\left(\gamma N \log(T)\left[\sqrt{KT} + DK\right]\right). \qquad (5)$$

Note that our result matches the regret bound for single-agent graph bandit [1] when $N = 1$ (with a subtle difference on the dependency on the $\log T$ factor). It is also worth noting that the regret in our multi-agent graph bandit formulation exhibits a linear growth with respect to the number of agents N, which stands in contrast to the results of other algorithms (e.g. [22]) that solve multi-agent MAB with regret $O(\sqrt{KNT\log T})$. This discrepancy arises primarily from our consideration of the weight functions f_k for the total objective which complicates the analysis, as opposed to the unweighted setting (i.e. $f_k(\cdot) = 1$) considered in [22]. Indeed, in the case of the unweighted setting, it can be shown that our algorithm also achieves regret proportional to $\sqrt{N}$. It is an interesting open question whether we can further tighten the bound on the dependency on N for general $f_k(\cdot)$.

Proof Ideas. The detailed proof can be found in the full version of the paper, posted to arXiv [31]. Here we outline the main ideas. In order to prove Theorem 1, we analyze the regret of the system by episode, noting that since our algorithm doubles the samples of at least one arm each episode we can bound the number of episodes as $K\log(T)$. For each episode, we define the "good" event as

$$\mathcal{G}_e = \{\forall k \in [K], \mu_k \in [\hat{\mu}_{k,t_e} - b_{k,t_e}, \hat{\mu}_{k,t_e} + b_{k,t_e}]\}. \qquad (6)$$

This is the event where the true mean of each arm is within the arm's confidence radius of its estimated mean. The "bad" event is thus the probability that at least one of our confidence bounds fails. Following this decomposition,

our regret proof consists of four steps: (1) bounding the regret contribution of the bad episodes, (2) bounding the transition phase of the good episodes, (3) bounding the exploitation phase of the good episodes, and (4) bounding the regret of initialization.

Step 1: Regret Contribution of Bad Events. We use Hoeffding's inequality to bound the regret of the bad episodes by showing that for each episode e the probability that any single confidence radius fails is $2t_e^{-3}$. We then note that for any single time step we can incur regret no more than γN since our rewards are bounded in $[0,1]$ and the weight functions linear in the counts, so using a union bound over the arms and summing over all episodes gives a final regret contribution of

$$O(\gamma N(\log(K)+\log\log(T)). \tag{7}$$

Step 2: Regret of Transition Phase. As discussed, we divide the good episodes into two phases. The first phase, the transition phase, is when the agents move along the graph to their assigned destination nodes. Since this takes at most D steps—recall our restriction on the regret shortest paths—and at each time step we incur a regret of no more than γN, we can bound the total transition cost after summing over all episodes by

$$O(\gamma N\log(T)DK). \tag{8}$$

Note that since we assume the worst possible matching, the fact that our offline planning sub-routine `SPMatching` is only a pseudo-solution does not affect our theoretical regret analysis.

Step 3: Regret of Exploitation Phase. The second phase of the good episodes is the exploitation phase, when the system repeatedly samples from the estimated optimal arm allocation represented by $\hat{C}_e$. In analyzing this phase, we make use of our definitions for b_{k,t_e} and U_{k,t_e} to simplify the regret at each time step as

$$\begin{aligned}\sum_{k\in[K]}\left[f_k(c_k^*)-f_k(\hat{c}_{k,e})\right]\mu_k &\\ \leq 2\sum_{k\in[K]} f_k(\hat{c}_{k,e})b_{k,t_e}+\sum_{k\in[K]}\left[f_k(c_k^*)-f_k(\hat{c}_{k,e})\right]U_{k,t_e} &\\ \leq 2\sum_{k\in[K]} f_k(\hat{c}_{k,e})b_{k,t_e}, \quad (9)&\end{aligned}$$

where the last inequality follows from definition of $\hat{C}_e$ as the count vector that maximizes $\sum_{k\in K} f_k(c_k)U_{k,t_e}$. Then, we simplify this sum by extracting a $\sqrt{2\log(t_e)}\leq\sqrt{2\log(T)}$ from each confidence radius and showing

$$\sum_{k\in[K]}\frac{f_k(\hat{c}_{k,e})}{\sqrt{n_{k,t_e}}}\leq\sum_{k\in[K]}\frac{\gamma\hat{c}_{k,e}}{\sqrt{n_{k,t_e}}}\leq\frac{\gamma N}{\sqrt{n_{min}}} \tag{10}$$

by noting that for each destination arm k, $n_{k,t_e}\geq n_{min}$ and that $\sum_{k\in[K]}\hat{c}_{k,e}\leq N$. A final bound of

$$O(\gamma N\log(T)\sqrt{NK}) \tag{11}$$

follows when we sum over all episodes since for each episode e the exploitation phase cannot exceed n_{min} steps.

Step 4: Initialization Cost. Finally, the initialization cost contributes just $O(\gamma NK)$ to the regret since each of N agents runs DFS until the K vertices have been visited.

Combining these four steps gives the desired regret upper bound. Note that Steps 1 and 4 are asymptotically dominated by Steps 2 and 3 so that our final regret bound is the sum of the transition phase and exploitation phase regret.

Remark 3 (Regret Consideration of Weighted Graphs)**.** *The consequences of a weighted G to the theoretical analysis would manifest in Steps 1, 3, and 4. In particular, the assumption that each time step incurs a worst-case regret of N would need to be updated to reflect the transition costs potentially incurred by the agents. So long as these costs are bounded by some constant B, we conclude that at each time step the regret incurred is no more than* $(1+B)N$ *which does not effect our asymptotic regret bound.*

V. Numerical Simulations

To test the performance of our model experimentally, we construct a synthetic multi-agent graph bandit problem with $K=300$ arms and $N=20$ agents. The graph, which we display in Figure 1, is initialized as an Erdos-Renyi graph with probability parameter 0.05. Each agent is assigned a random start vertex. The reward distribution for each arm is distributed as $\mathcal{N}(\mu,\sigma^2)$ with μ chosen uniformly from the interval $(0.25,0.75)$ and $\sigma^2=0.06$. Motivated by the case of decreasing marginal rewards, for each arm $k\in[K]$, we also define the concave function

$$f_k(x)=\frac{\log_{2+k}(\frac{x}{20}+\frac{1}{2+k})+1}{\log_{2+k}(\frac{1}{20}+\frac{1}{2+k})+1}.$$

We use a state-of-the-art optimization software, Gurobi, to solve the non-convex optimization problem given by (3) [32].

In addition to `Multi-G-UCB`, we define three benchmark algorithms. `Multi-G-UCB-median` is a variation where for each episode we sample the estimated optimal arm allocation until the arm with the median number of prior samples has its sample count doubled (rather than the minimum). `Multi-G-UCB-max` is defined similarly but uses the arm with the maximum number of prior samples as the baseline for each episode. Lastly, `Indv-G-UCB` is derived by having each agent individually perform the G-UCB algorithm of [1] without communication and coordination. Figure 2 displays the cumulative regret of each algorithm as a function of time averaged over 10 independent trials each with time horizon $T=1.5\cdot10^5$.

As expected, all three of the cooperative algorithms considerably outperform `Indv-G-UCB`. `Multi-G-UCB` also outperforms its two variations both of whom over-prioritize exploitation and under-prioritize exploration. For `Multi-G-UCB-max` in particular, once one arm has been shown to be part of the optimal set it becomes increasingly harder to find the other arms since the episode lengths continue to increase even when including relatively unknown arms in the destination allocation.

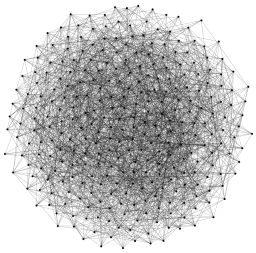

Fig. 1. The graph G.

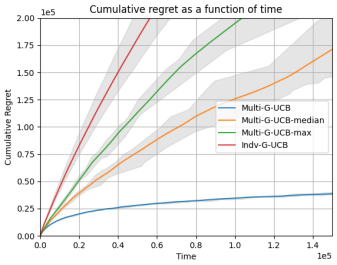

Fig. 2. The average cumulative regret of each algorithm across 10 trials.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we define the multi-agent graph bandit problem, propose a learning algorithm, `Multi-G-UCB`, provide a theoretical analysis of its regret and present experimental results that validate its performance. We believe that the formulation multi-agent graph bandit problem opens up many interesting future directions, such as extending our algorithm to accommodate decentralized learning scenarios, generalizing to more complicated case where rewards of arms are correlated across time and/or across arms, as well as performing more comprehensive theoretical studies such as analyzing instance-dependent regret.

REFERENCES

[1] T. Zhang, K. Johansson, and N. Li, "Multi-armed bandit learning on a graph," in *2023 57th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2023, pp. 1–6.

[2] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[3] A. Slivkins *et al.*, "Introduction to multi-armed bandits," *Foundations and Trends® in Machine Learning*, vol. 12, no. 1-2, pp. 1–286, 2019.

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

[5] S. Bubeck, N. Cesa-Bianchi, *et al.*, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[6] J. Zhu, R. Sandhu, and J. Liu, "A distributed algorithm for sequential decision making in multi-armed bandit with homogeneous rewards," in *2020 59th IEEE Conference on Decision and Control (CDC)*. Jeju, Korea (South): IEEE, Dec 2020, p. 3078–3083. [Online]. Available: https://ieeexplore.ieee.org/document/9303836/

[7] M. Chakraborty, K. Y. P. Chua, S. Das, and B. Juba, "Coordinated versus decentralized exploration in multi-agent multi-armed bandits," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization, 2017, p. 164–170. [Online]. Available: https://www.ijcai.org/proceedings/2017/24

[8] D. Martínez-Rubio, V. Kanade, and P. Rebeschini, "Decentralized cooperative stochastic bandits," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[9] P.-A. Wang, A. Proutiere, K. Ariu, Y. Jedra, and A. Russo, "Optimal algorithms for multiplayer multi-armed bandits," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 4120–4129.

[10] P. Landgren, V. Srivastava, and N. E. Leonard, "Distributed cooperative decision-making in multiarmed bandits: Frequentist and bayesian algorithms," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 167–172.

[11] M. Agarwal, V. Aggarwal, and K. Azizzadenesheli, "Multi-agent multi-armed bandits with limited communication," *The Journal of Machine Learning Research*, vol. 23, no. 1, pp. 9529–9552, 2022.

[12] A. Sankararaman, A. Ganesh, and S. Shakkottai, "Social learning in multi agent multi armed bandits," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 1–35, 2019.

[13] R. Chawla, A. Sankararaman, A. Ganesh, and S. Shakkottai, "The gossiping insert-eliminate algorithm for multi-agent bandits," in *International conference on artificial intelligence and statistics*. PMLR, 2020, pp. 3471–3481.

[14] D. Kalathil, N. Nayyar, and R. Jain, "Decentralized learning for multi-player multi-armed bandits," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec 2012, p. 3960–3965, arXiv:1206.3582 [cs, math]. [Online]. Available: http://arxiv.org/abs/1206.3582

[15] K. Liu and Q. Zhao, "Distributed learning in multi-armed bandit with multiple players," *IEEE Transactions on Signal Processing*, vol. 58, no. 11, p. 5667–5681, Nov 2010.

[16] P.-A. Wang, A. Proutiere, K. Ariu, Y. Jedra, and A. Russo, "Optimal algorithms for multiplayer multi-armed bandits," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. PMLR, Jun 2020, p. 4120–4129. [Online]. Available: https://proceedings.mlr.press/v108/wang20m.html

[17] E. Boursier and V. Perchet, "Sic-mmab: Synchronisation involves communication in multiplayer multi-armed bandits," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[18] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proceedings of the 30th International Conference on Machine Learning*. PMLR, Feb 2013, p. 151–159. [Online]. Available: https://proceedings.mlr.press/v28/chen13a.html

[19] W. Chen, Y. Wang, Y. Yuan, and Q. Wang, "Combinatorial multi-armed bandit and its extension to probabilistically triggered arms," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1746–1778, 2016.

[20] S. Wang and W. Chen, "Thompson sampling for combinatorial semi-bandits," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, Jul 2018, p. 5114–5122. [Online]. Available: https://proceedings.mlr.press/v80/wang18a.html

[21] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, p. 1466–1478, Oct 2012.

[22] B. Kveton, Z. Wen, A. Ashkan, and C. Szepesvari, "Tight regret bounds for stochastic combinatorial semi-bandits," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. PMLR, Feb 2015, p. 535–543. [Online]. Available: https://proceedings.mlr.press/v38/kveton15.html

[23] J. Cortes, S. Martinez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[24] V. Ramaswamy and J. R. Marden, "A sensor coverage game with improved efficiency guarantees," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 6399–6404.

[25] X. Sun, C. G. Cassandras, and X. Meng, "A submodularity-based approach for multi-agent optimal coverage problems," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 4082–4087.

[26] M. Prajapat, M. Turchetta, M. Zeilinger, and A. Krause, "Near-optimal multi-agent learning for safe coverage control," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14 998–15 012, 2022.

[27] V. Ramaswamy, D. Paccagnan, and J. R. Marden, "Multiagent maximum coverage problems: The tradeoff between anarchy and stability," *IEEE Transactions on Automatic Control*, vol. 67, no. 4, pp. 1698–1712, 2021.

[28] 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

[29] Z. Galil, "Efficient algorithms for finding maximum matching in graphs," *ACM Comput. Surv.*, vol. 18, no. 1, p. 23–38, mar 1986. [Online]. Available: https://doi.org/10.1145/6462.6502

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[31] P. Paschalidis, R. Zhang, and N. Li, "Cooperative multi-agent graph bandits: Ucb algorithm and regret analysis," 2024.

[32] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com