2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Optimal Cooperative Multiplayer Learning Bandits with Noisy Rewards and No Communication

William Chang[1], Yuanhao Lu[2],
[1]University of California, Los Angeles
[2]Princeton University

Abstract— **We consider a cooperative multiplayer bandit learning problem where the players are only allowed to agree on a strategy beforehand, but cannot communicate during the learning process. In this problem, each player simultaneously selects an action. Based on the actions selected by all players, the team of players receives a reward. The actions of all the players are commonly observed. However, each player receives a noisy version of the reward which cannot be shared with other players. Since players receive potentially different rewards, there is an asymmetry in the information used to select their actions. In this paper, we provide an algorithm based on upper and lower confidence bounds that the players can use to select their optimal actions despite the asymmetry in the reward information. We show that this algorithm is asymptotically optimal in T. We also show that it performs empirically better than the current state-of-the-art algorithm for this environment.**

I. Introduction

The field of stochastic Multi-armed Bandit (MAB) contains some of the most well-studied problems in reinforcement learning. These online learning algorithms are designed to understand how to implement exploration-exploitation tradeoffs to achieve the most reward. The classical version of MAB consists of a single agent with a set of $[m] := \{1, \ldots, m\}$ actions to choose from. Each action is associated with an unknown reward distribution that is sub-Gaussian. A *round* in a MAB environment is defined as one iteration where the player selects an action and obtains a reward. At every such round, the agent's goal is to select the arm with the highest expected reward. [1] One can measure the success of a policy for arm selection using the notion of *regret*, which measures how often a suboptimal arm is chosen (pulled). The goal is to minimize the regret for large horizons. In the classical single-agent setting, [19] showed that every policy will not be able to perform better than $O(\log T)$ gap-dependent regret. This lower bound on regret was first attained by the UCB algorithm.

Single-player MABs, however, inadequately model the complexities of real-world applications involving multiple interacting entities. This gap has sparked a recent growing interest in cooperative multiplayer MAB problems, where multiple agents are maximizing their collective expected rewards. For example, [12], [21], [1], [14], [27] introduces and applies multiplayer MAB models for spectrum sharing in wireless networks. These papers assume each player's reward is independent of the other's actions (i.e. joint actions are not considered).

While the aforementioned works successfully extend the classical MAB problem to include multiple players, they remain restrictive. Specifically, they fail to adequately model scenarios in which the decisions of one agent might influence the rewards of others, a common interaction arising in real-world settings. For instance, in shared network settings, the bandwidth an agent consumes directly affects the network's capacity for other users. Similarly, in financial markets, buying or selling decisions made by one trader can influence the stock price and, consequently, the rewards for other traders. To provide a framework to model these problems, we consider multiagent settings where each agent has their own (marginal) set of actions to choose from. Subsequently, all agents' collective actions form a joint action across all players. Note that this is different than leader-follower games [36], where the leader selects an action first, which the followers observe before they select their actions.

Furthermore, our paper considers the multiagent setting where *information asymmetry* is preset. By information asymmetry, we mean there exists some information that is not shared among all players. Specifically, we analyze the abovementioned multiplayer setting where *reward asymmetry* is present; that is, agents do not observe the rewards obtained by other agents. Information asymmetry naturally occurs when communication between agents is restricted. Thus, effectively, we do not allow for any form of communication between agents, although they are allowed to agree on a policy before the learning phase begins and know the number of actions the other players have. This setting is rich in real-world motivation, where agents collaboratively pursue a common objective despite the absence of direct inter-agent communications. For example, in decentralized traffic control systems where individual agents — traffic lights in disparate regions — endeavor to optimize the average vehicular travel time. In this application, the actions of individual agents depend on each other, and local traffic conditions are not observed by traffic lights in remote areas, necessitating the use of joint actions and reward asymmetry as investigated in this study.

Our Contributions We propose merry go around variant of the UCB algorithm we call `mUCB-Intervals`. The novelty of this algorithm is combining an interval method for best arm selection [3], applying it to regret minimization tasks, and

[1]We term 'arm' and 'action' interchangeably

including an aspect of coordination for the multiplayer setting. More explicitly, the players will decide on an ordering of the arms prior to the learning (they can use the ordering given in [10]), and during the process, they will pull each arm in order. They will also maintain a *desired set* in which all the arms will be in this set at the beginning. This desired set will remain the same for all players for each round. Based on the UCB "error" intervals each player will determine if the next arm should be in the set, and if not, they will communicate this to the other players by not pulling what should have been the next arm. Note that there is no explicit communication in the environment but the players know when they eliminate the next arm from the desired set, and thus can maintain the same desired set. This is similar to other bandit works in collision sensing [8] but their communication scheme is much simpler in that they simply have to observe when other bandits have pulled the same arm.

We show that our algorithm achieves $O(\frac{\log T}{\Delta_a})$ gap dependent regret or $O(\sqrt{T \log T})$ gap independent bound. Whereas [10] was able to achieve *almost* optimal regret for this setting, `mUCB-Intervals` is the first algorithm to achieve optimal regret for a reward asymmetric setting. It's easy to implement and understand and performs better empirically against `mDSEE` from [10], the current state of the art algorithm for this environment.

Related Work.: The literature on multi-armed bandits is overviewed in [31], [13], [20]. Some classical papers worth mentioning are [19], [2], [4]. Interest in multi-player MAB models was triggered by the problem of opportunistic spectrum sharing and some early papers were [12], [21], [1]. Other papers motivated by similar problems in communications and networks are [22], [18], [29], [9]. These papers were either for the centralized case, or considered the symmetric user case, i.e., all users have the same reward distributions. Moreover, if two or more users choose the same arm, there is a "collision", and neither of them get a positive reward. The first paper to solve this matching problem in a general setting was [14] which obtained log-squared regret. It was then improved to log regret in [27] by employing a posterior sampling approach. These algorithms required implicit (and costly) communication between the players. Thus, there were attempts to design algorithms without it [5], [28], [7], [11], [8]. Other recent papers on decentralized learning for multiplayer matching MAB models are [34], [30].

In the realm of multiplayer stochastic bandits, many works allow for limited communication such as those in [25], [26], [32], [17], [33]. An exception is [6] where all the players select from the same set of arms and their goal is to avoid a collision, that is, they do not want to select the same arm as another player. Another work that doesn't allow for communication [35] where they developed online learning algorithms that enable agents to cooperatively learn how to maximize reward with noisy global feedback without exchanging information. Recently, in light of the work from [10], there have been other works which have studied information asymmetric multiplayer bandits [24], [16], [23], [15].

II. Preliminary

We follow the formulation of *information asymmetry* only in rewards given in [10] which we reiterate here for completeness:

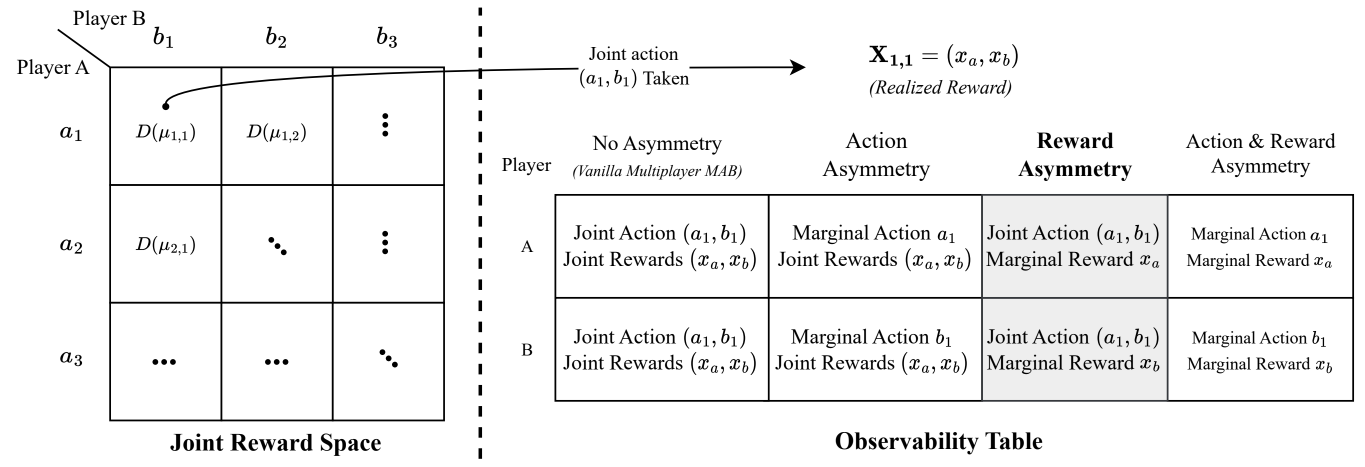
Consider a set of M players $P_1, \cdots, P_M$, in which player P_i has a set $\mathcal{K}_i$ of K_i arms to pick from. At each time instant, each player picks an arm independently and simultaneously from other players from their set $\mathcal{K}_i$. The joint action can be interpreted as an M-tuple of arms picked denoted by $\boldsymbol{a} = (a_1, \cdots, a_M)$. We shall use bold font to denote vectors. For simplicity, we shall assume each player has N actions to pick from which gives a total of $K = N^M$ joint actions. This generates M independent and identically distributed (iid) random rewards $X^i_{\boldsymbol{a}} \in [0, 1]$ with $i \in [M]$ from a 1-subgaussion distribution $F_{\boldsymbol{a}}$ with mean $\mu_{\boldsymbol{a}}$. Each player P_i can only observe their rewards $X^i_{\boldsymbol{a}}$ and are oblivious to all the other rewards (this is the reward asymmetry). However, they can observe the actions of all other players at all times after the joint action has been taken. Note that this is different from leader-follower games in that all players select their actions at the same time. All the players know the joint action space beforehand and they can decide on a strategy a priori. However, during learning they are not allowed to communicate in any way.

Denote $\Delta_{\boldsymbol{a}} = \mu^* - \mu_{\boldsymbol{a}}$ where μ^* is the highest reward mean among all arm tuples, and we shall call the corresponding arm the *optimal arm*. The players share a goal and that is to pull the optimal arm as often as possible. Let $a_t[i] \in \{1, \cdots, K_i\}$ be the arm chosen by player i at time t, and denote $\boldsymbol{a_t} = (a_t[1], \cdots, a_t[m])$. A high-level objective is for the players to collectively identify the best set of arms a^* corresponding to mean reward μ^*. But the players do not know the means $\mu_{\boldsymbol{a}}$, nor the distributions $F_{\boldsymbol{a}}$. They must learn by playing and exploring. Thus, we can capture learning efficiency of an algorithm via the notion of *expected regret*,

$$R_T = \mathbb{E}\left[T\mu^* - \sum_{t=1}^{T} X_{\boldsymbol{a(t)}}(t)\right] \tag{1}$$

where T is the number of learning instances and $X_{\boldsymbol{a(t)}}(t)$ is the random reward if arm-tuples $a(t)$ are pulled. We let $n_{\boldsymbol{a}}(t)$ be the number of times joint arm $\boldsymbol{a}$ has been pulled up to(not including) time t. Furthermore let $\widehat{\mu}^i_{\boldsymbol{a}}(n_{\boldsymbol{a}}(t))$ be the empirical mean of arm $\boldsymbol{a}$ for player i at time t after $n_{\boldsymbol{a}}(t)$ pulls of the joint arm $\boldsymbol{a}$. Note the empirical mean is indexed by the player since each player gets their own copy of the IID reward. However, as the rewards for each joint arm have the same mean across all players, the expected regret R_T is the same for each player. On the other hand, $n_{\boldsymbol{a}}$ is not indexed by the player i since each player is contributing to the same joint action $\boldsymbol{a}$ at every round.

Note that although each player will likely get different rewards, the rewards are iid so that in expectation the regret for all the players are the same. Note that fundamental results for single-player MAB problems [19] suggest a $O(\log T)$-regret lower bound for the multi-player MAB problem as well. If we can design a multi-player decentralized learning



Player	No Asymmetry *(Vanilla Multiplayer MAB)*	Action Asymmetry	**Reward Asymmetry**	Action & Reward Asymmetry
A	Joint Action (a_1, b_1) Joint Rewards (x_a, x_b)	Marginal Action a_1 Joint Rewards (x_a, x_b)	Joint Action (a_1, b_1) Marginal Reward x_a	Marginal Action a_1 Marginal Reward x_a
B	Joint Action (a_1, b_1) Joint Rewards (x_a, x_b)	Marginal Action b_1 Joint Rewards (x_a, x_b)	Joint Action (a_1, b_1) Marginal Reward x_b	Marginal Action b_1 Marginal Reward x_b

Fig. 1. A two-player information asymmetric in reward setting. The grid on the left is a visualization of the joint action space, where the rows correspond to the actions available to Player A while the columns correspond to the actions available to player B. Each entry in this grid has a subgaussian reward distribution $D(\mu)$ for some mean μ. The table on the right lists what is observed by each player for the different types of information asymmetry. In our setting in gray, for each player, the joint actions are observed but only their copy of the IID reward is observed.

algorithm with such a regret order, then it would imply that such a lower bound is tight for this setting as well.

III. Main Results

To maximize the cumulative rewards, the players will define the UCB (Upper Confidence Bound) index for each joint action (not just their own action set), and try to pull arms with high UCB indices as often as possible. This index for player i is given by

$$\eta^i_{\boldsymbol{a}}(t) = \begin{cases} \infty, & \text{if } n_{\boldsymbol{a}}(t) = 0, \\ \hat{\mu}^i_{\boldsymbol{a}}(n_{\boldsymbol{a}}(t)) + \sqrt{\frac{2\log(1/\delta)}{n_{\boldsymbol{a}}(t)}}, & \text{otherwise.} \end{cases} \tag{2}$$

Note that $\eta^i_{\boldsymbol{a}}$ is indexed by the player because each player observes a different empirical reward mean $\hat{\mu}^i_{\boldsymbol{a}}(n_{\boldsymbol{a}}(t))$.

Let $\epsilon_{\boldsymbol{a}}(\cdots)$ be the constant added to the empirical mean for arm $\boldsymbol{a}$ in calculating the UCB index. This constant is the same for every player which is why is not indexed by i. Since the intervals used in our algorithm 1 will be of length $2\epsilon_{\boldsymbol{a}}(\cdots)$, we add a hyperparameter γ to tune this interval length. More explicitly, it is defined as

$$\epsilon_{\boldsymbol{a}}(n_{\boldsymbol{a}}, \delta, \gamma) := \gamma\sqrt{\frac{\log(1/\delta)}{n_{\boldsymbol{a}}}} \tag{3}$$

where t is the round number, $n_{\boldsymbol{a}}$ is the number of times arm $\boldsymbol{a}$ has been fulled, δ a constant that is set to $\frac{1}{T^2}$, and γ scales the length of our interval. We know from Hoeffding's bound that for subgaussian variables, the true mean $\mu_{\boldsymbol{a}}$ is within the interval

$$I^i_{\boldsymbol{a}} = (\hat{\mu}^i_{\boldsymbol{a}} - \epsilon_{\boldsymbol{a}}, \hat{\mu}^i_{\boldsymbol{a}} + \epsilon_{\boldsymbol{a}}) \tag{4}$$

with high probability, where we have omitted the arguments as they are clear from the context. If an arm has not been pulled yet, we will give it infinite interval namely, $(-\infty, \infty)$. This guarantees that every arm is pulled at least once. Using this, at each round we can create a set of arms that are likely to be optimal and make our selection from this set. Note that each player i has a different interval for joint action $\boldsymbol{a}$ due to the empirical means $\hat{\mu}^i_{\boldsymbol{a}}$ being different. However, for each joint action $\boldsymbol{a}$, every player has intervals of the same length for this action. Thus $\epsilon_{\boldsymbol{a}}$ is not indexed by the player i since it is the same for each player.

We now propose `mUCB-Intervals` in Algorithm 1 used to deal with reward asymmetry. In this algorithm, all the players will maintain a *desired set* which contains the joint arms that are candidates for the optimal arm. Initially, all K joint actions are in this desired set. To ensure coordination, all the players will agree before the learning process on an order for the set of joint actions in this desired set. This is similar to how [10] dealt with asymmetry in actions. Furthermore, by observing the actions of the other players, they will be able to maintain the same desired sets at each round. At a particular round, a joint action is called *considered* if it's the arm in the desired set that was supposed to be pulled that round in accordance with the order that was agreed upon by all the players. We will use $\boldsymbol{c}$ to denote a joint action in the desired set. If there are ℓ joint actions $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_\ell$, then the ordering of the desired set can be viewed using the following flow chart.

$$\begin{tikzcd} \boldsymbol{c}_1 \arrow[r] & \boldsymbol{c}_2 \arrow[r] & \cdots \arrow[r] & \boldsymbol{c}_\ell \arrow[lll, bend left] \end{tikzcd} \tag{5}$$

A joint arm is eliminated from this desired set, if at a particular round t, a player i observes that the considered joint arm (call it $\boldsymbol{a}_{k_t}$ in the flow diagram above) has a UCB interval that is below and disjoint from another arm. Then that player i will effectively 'communicate' this elimination to the other players by not pulling $\boldsymbol{c}_t[i]$. Other players will observe that $\boldsymbol{c}_t[i]$ was not pulled by player i and eliminate it from their desired sets as well whilst maintaining the same ordering of the remaining arms. In other words, pulling a joint arm that isn't considered is the same as removing the considered arm from the desired set. The new flow diagram

Algorithm 1: `mUCB-Intervals`

```
1  Each player $P_i$ has all the joint arms in their *desired sets*. All the players will agree on the ordering of the joint arms.
2  Initialize each UCB interval to be $I_{\boldsymbol{a}} = (-\infty, \infty)$
3  for $t = 1, \dots, T$ do
4      Each player $i$ identifies the next arm considered $\boldsymbol{c}_t$ in the desired set based on the arm pulled in the previous round (see flowchart in (5)).
5      if *exists player $i$ and joint action $\boldsymbol{a'}$ such that $I^i_{\boldsymbol{a'}}$ is above and disjoint from $I^i_{\boldsymbol{c}_t}$* then
6          Player $i$ will not pull $\boldsymbol{c}_t[i]$ to inform the other players that he will remove $\boldsymbol{c}_t$ from his desired set.
7      else
8          Each player $i$ pull $\boldsymbol{c}_t[i]$.
9      end
10     Each player $i$ observes the actions from other players to determine the joint action taken $\boldsymbol{a}_t$ at that step. They observe their own i.i.d. reward, and update their $I^i_{\boldsymbol{a}_t}$.
11     if $\boldsymbol{a}_t \neq \boldsymbol{c}_t$ then
12         All players eliminate $\boldsymbol{c}_t$ from their desired set whilst maintaining the same ordering of the remaining arms (see flowchart in (6))
13     end
14 end
```

after the elimination is,

$$\begin{tikzcd} \boldsymbol{c}_1 \arrow[r] & \boldsymbol{c}_2 \arrow[r] & \cdots \boldsymbol{c}_{t-1} \arrow[r] & \boldsymbol{c}_{t+1} \arrow[r] & \cdots \arrow[r] & \boldsymbol{c}_\ell \arrow[lllll, bend left] \end{tikzcd} \tag{6}$$

and the arm that is considered in the next round $t+1$ is the next arm in the ordering prior to the elimination. In the flowchart (6) it is $\boldsymbol{a}_{k_t+1}$. On the other hand, if the arm that was considered is the same as the one that was pulled, then the flowchart in (5) remains unchanged.

A. Example

Due to the novelty of this algorithm and setting, we will present an example of how this algorithm runs. Consider a two-player setting where each player has two arms. We will represent each joint arm as an entry in a matrix and what goes inside the matrix is the corresponding UCB interval defined in equation (4). The rows are numbered by player 1's actions while the columns are numbered by player 2's actions. We will write (a, b) to denote the a-th actions of player 1 and the b-th action of player 2 respectively. The number of rows would be the number of actions for player 1 and the number of columns would the be number of actions for player 2. Note this isn't important for this example, but if we had 3 players instead of 2, it will be a 3-dimensional matrix. Since for every joint action, the UCB intervals are infinite prior to the start of learning, our initial matrix for player 1 and player 2 respectively is

$$\underbrace{\begin{array}{c|cc} & 1 & 2 \\ \hline 1 & (-\infty,\infty) & (-\infty,\infty) \\ 2 & (-\infty,\infty) & (-\infty,\infty) \end{array}}_{\text{Player 1}}, \underbrace{\begin{array}{c|cc} & 1 & 2 \\ \hline 1 & (-\infty,\infty) & (-\infty,\infty) \\ 2 & (-\infty,\infty) & (-\infty,\infty) \end{array}}_{\text{Player 2}}$$

Suppose the players agreed prior to learning the ordering of the joint actions to be as follows

$$\begin{tikzcd} (1,1) \arrow[r] & (1,2) \arrow[r] & (2,1) \arrow[r] & (2,2) \arrow[lll, bend left] \end{tikzcd} \tag{7}$$

Note that any order will suffice as long as the players use the same ordering. Initially, all the UCB intervals are infinite and each arm belongs in the desired set so each arm gets pulled at least once in order. Suppose the matrices for both players after pulling these four joint actions from $t = 1$ to $t = 4$ are now [2]

$$\underbrace{\begin{array}{c|cc} & 1 & 2 \\ \hline 1 & (.2,.5) & (.3,.6) \\ 2 & (.55,.9) & (.65,.8) \end{array}}_{\text{Player 1}}, \underbrace{\begin{array}{c|cc} & 1 & 2 \\ \hline 1 & (.5,.8) & (.4,.7) \\ 2 & (.6,.95) & (.65,.8) \end{array}}_{\text{Player 2}} \tag{8}$$

In the following round $t = 5$, the next arm to pull in order is $(1, 1)$ because once you reach the end of the desired arm set you start over from the beginning. Thus the action that is *considered* at this round is $(1, 1)$. However, note that the interval for arm $(1, 1)$ for player 1 in red is now disjoint from the interval for arm $(2, 1)$ in blue. Therefore, player 1 knows with a high probability that this arm is not optimal and is thus ready to eliminate it. In order to communicate that to player 2, he will pull arm 2 instead of arm 1. Player 2 had no intention of eliminating arm $(1, 1)$ so he will pull arm 1. Thus the joint action that was taken at $t = 5$ is $(2, 1)$. Player 2 will observe that player 1 pulled a different arm than what was agreed upon, and understands that $(1, 1)$ needs to be eliminated. Note that player two did not observe the interval for action $(1, 1)$ being disjoint from action $(2, 1)$ but the arm gets eliminated anyways as long as *at least* one player observes this disjoint interval phenomenon.

Thus the desired set for both players now only contains,

$$\begin{tikzcd} (1,2) \arrow[r] & (2,1) \arrow[r] & (2,2) \arrow[ll, bend left] \end{tikzcd} \tag{9}$$

The next *considered* arm for $t = 6$ is $(1, 2)$ because that was the arm right after $(1, 1)$ prior to the elimination, and this process repeats itself until eventually there is only 1 arm left which will be the optimal arm with high probability.

[2]Note that for the same joint action, $\boldsymbol{a}$, the corresponding intervals between the two players are of the same length. For example for the joint action $(1, 2)$ the interval for player 1 is $(.3, .6)$ while the interval for player 2 is $(.4, 7)$. The centers of these intervals are different since they have different empirical means. However, the length of the interval for both is .3 because that only spends on the horizon and the number of times $(1, 2)$ has been pulled. Both of these quantities are the same for all players.

B. Discussion

We discuss why this algorithm performs so much better than `mDSEE` from [10] which was used in asymmetry in both rewards and actions. `mDSEE` explores all arms equally at exponentially increasing intervals, which incurs a lot of regret during the exploration process. In comparison, `mUCB-Intervals` only focuses on arms that can potentially be the optimal one, by eliminating arms that are clearly suboptimal immediately.

Furthermore, `mDSEE` requires a unique global optimum. This is because if there are two optimal actions, then at the committing phase, players will commit to different joint actions with constant probability. However, since the desired sets of all the players are the same, in this setting a unique global optimal action is not required.

This algorithm is similar to best arm identification algorithms such as those in [3], where they also study UCB intervals to determine which arm is optimal. However, in our case, the predefined ordering of the joint arms and maintenance of the same desired set for each player are what makes coordination possible.

C. Regret Bounds

We have the following regret bounds for `mUCB-Intervals`.

Theorem 1: For any choices of $\gamma > 0$, the gap dependent regret of algorithm `mUCB-Intervals` is

$$R_T = O\left(\left(\sum_{\boldsymbol{a}\in\mathcal{A}} \frac{1}{\Delta_{\boldsymbol{a}}}\right)\log T + MK^M\right) \tag{10}$$

Theorem 2: For any choices of $\gamma > 0$, the gap-independent regret bound of Algorithm 1 is

$$R_T = O\left(\sqrt{MK^M T\log(T)}\right). \tag{11}$$

IV. Proofs

In this section, we present the proofs to the regret bounds given in section III. The following property on subgaussian variables will be important.

Lemma 3: Assume that $X_i - \mu$ are independent, σ-subgaussian random variables. Then, for any $\epsilon \geq 0$,

$$\begin{aligned} &P\left(\hat\mu \geq \mu + \epsilon\right) \leq \exp\left(-\frac{T\epsilon^2}{2\sigma^2}\right) \\ \text{and}\quad &P\left(\hat\mu \leq \mu - \epsilon\right) \leq \exp\left(-\frac{T\epsilon^2}{2\sigma^2}\right) \end{aligned} \tag{12}$$

where $\hat\mu = \frac{1}{T}\sum_{t=1}^{T} X_t$.

Proof: The reader is encouraged to look at Corollary 5.5 of [20]. It relies on the observation that $\hat\mu - \mu$ is $\sigma/\sqrt{T}$-subgaussian. ■

The following regret decomposition is also going to be useful in our proofs.

Lemma 4: With regret defined in (1), we have the following regret decomposition for each player i:

$$R_T^i = \sum_{\boldsymbol{a}} \Delta_{\boldsymbol{a}} \mathbb{E}\left[n_{\boldsymbol{a}}(T)\right], \tag{13}$$

where $n_{\boldsymbol{a}}(T)$ is the number of times the arm $\boldsymbol{a}$ has been pulled up to round T.

Proof: The reader is encouraged to look at Lemma 4.5 of [20]. It follows from the fact that each round you pull arm $\boldsymbol{a}$, you incur (in expectation) $\Delta_{\boldsymbol{a}}$ regret. ■

We are now ready to present the proof of Theorem 1

Proof: We will write $\hat\mu^i_{\boldsymbol{a}}(n_{\boldsymbol{a}}(t))$ as $\hat\mu^i_{\boldsymbol{a}}$ when it is clear from the context what the argument should be.

We suppose that the first arm is the optimal one for each player, that is arm $\mathbf{1} = (1, \ldots, 1)$ has the highest average reward. We define the following "good" event G where all arms have their true means in the intervals at all times. Explicitly, this is written as

$$G = \bigcap_{i=1}^{M} \bigcap_{\boldsymbol{a}\in\mathcal{A}} \left\{|\hat\mu^i_{\boldsymbol{a}} - \mu_{\boldsymbol{a}}| < \epsilon_{\boldsymbol{a}}(t,\delta,\gamma) | \forall t \in [1, n]\right\} \tag{14}$$

We define another good event for each arm based on the observed means of arm $\boldsymbol{a}$ and arm 1.

$$\begin{aligned} G^{\text{cross}}_{\boldsymbol{a}} = \bigcap_{i=1}^{M} \{&\hat\mu^m_{\boldsymbol{a}}(u_{\boldsymbol{a}},\delta,\gamma) + \epsilon_{\boldsymbol{a}}(u_{\boldsymbol{a}},\delta,\gamma) \\ &< \mu_{\mathbf{1}} - 2\epsilon_{\mathbf{1}}(u_{\boldsymbol{a}}-1,\delta,\gamma)\} \end{aligned} \tag{15}$$

This set is the event that after $u_{\boldsymbol{a}}$ pulls, the UCB indices of all the arms $\boldsymbol{a}$ is strictly smaller than lower bound of UCB index of arm $\mathbf{1}$. The next lemma shows that when $G \cap G^{\text{cross}}_{\boldsymbol{a}}$ occurs, the number of pull is at most $u_{\boldsymbol{a}}$. Note that $u_{\boldsymbol{a}}$ is an abstract quantity that is only used in the proof and not in the algorithm.

Lemma 5: Under the event $G \cap G^{\text{cross}}_{\boldsymbol{a}}$, the number of pulls of arm $\boldsymbol{a}$ is at most $u_{\boldsymbol{a}}$.

Proof: Under the event G the optimal arm $\mathbf{1}$ is always in the desired set. As the arms are pulled one at at time in a predefined order in the desired set, if an arm a is at the desired set at time t then $n_{\mathbf{1}}(t) \geq n_{\boldsymbol{a}}(t) - 1$. It follows that at the time when $n_{\boldsymbol{a}}(t) = u_{\boldsymbol{a}}$, we have $n_{\mathbf{1}}(t) \geq u_{\boldsymbol{a}} - 1$. Furthermore, $\epsilon_{\mathbf{1}}(u_{\boldsymbol{a}}-1,\delta,\gamma)$ is decreasing as a function of the number of pulls so $\epsilon_{\mathbf{1}}(u_{\boldsymbol{a}}-1,\delta,\gamma) \geq \epsilon_{\mathbf{1}}(n_{\mathbf{1}}(t),\delta,\gamma)$

Suppose for the sake of contradiction that $n_{\boldsymbol{a}}(n) > u_{\boldsymbol{a}}$. Then there must exist a round t such that $n_{\boldsymbol{a}}(t-1) = u_{\boldsymbol{a}}$ and the action taken at step t was $\boldsymbol{a}$,

$$\begin{aligned} \eta^i_{\boldsymbol{a}}(t-1) &= \hat\mu^m_{\boldsymbol{a}}(\cdot, u_{\boldsymbol{a}}) + \gamma\sqrt{\frac{\log(1/\delta)}{u_{\boldsymbol{a}}}} \\ &< \mu_{\mathbf{1}} - 2\epsilon_{\mathbf{1}}(u_{\boldsymbol{a}}-1,\delta,\gamma) && \text{since } G^{\text{cross}}_{\boldsymbol{a}} \text{ occurs} \\ &\leq \mu_{\mathbf{1}} - 2\epsilon_{\mathbf{1}}(n_{\mathbf{1}}(t),\delta,\gamma) && \text{since } n_{\mathbf{1}}(t) \geq u_{\boldsymbol{a}} - 1 \\ &< \hat\mu_{\mathbf{1}} - \epsilon_{\mathbf{1}}(n_{\mathbf{1}}(t),\delta,\gamma) && \text{since } G \text{ occurs} \end{aligned}$$

Thus, for each player, it follows that the interval corresponding to arm $\boldsymbol{a}$ is disjoint and below the interval corresponding to arm 1. Thus, arm $\boldsymbol{a}$ should have been eliminated from desired set, by round t. This contradiction completes the proof. ■

Using the regret decomposition lemma, we can aim to bound $\mathbb{E}[n_{\boldsymbol{a}}(t)]$. We can decompose this quantity as follows,

$$\begin{aligned}\mathbb{E}[n_{\boldsymbol{a}}(T)] &= \mathbb{E}[n_{\boldsymbol{a}}(T)\mathbb{I}(G \cap G_{\boldsymbol{a}}^{\text{cross}})] \\ &\quad + \mathbb{E}[n_{\boldsymbol{a}}(T)\mathbb{I}((G \cap G_{\boldsymbol{a}}^{\text{cross}})^c)] \\ &= \mathbb{E}[n_{\boldsymbol{a}}(T)\mathbb{I}(G \cap G_{\boldsymbol{a}}^{\text{cross}})] \\ &\quad + \mathbb{E}[n_{\boldsymbol{a}}(T)\left(\mathbb{I}(G^c) + \mathbb{I}(G \cap (G_{\boldsymbol{a}}^{\text{cross}})^c)\right)] \\ &\leq u_{\boldsymbol{a}} + \left(P(G^c) + P(G \cap (G_{\boldsymbol{a}}^{\text{cross}})^c)\right) T\end{aligned}$$

We start with $P(G^c)$:

$$\begin{aligned}P(G^c) &= P\left(\bigcup_{i=1}^{M}\bigcup_{\boldsymbol{a}}\left(\{\exists t \in [1,n] : |\hat{\mu}_{\boldsymbol{a}}^i - \mu_{\boldsymbol{a}}| \geq \epsilon_{\boldsymbol{a}}(t,\delta,\gamma)\}\right)\right) \\ &= \sum_{i=1}^{M}\sum_{\boldsymbol{a}} P\{\exists t \in [1,n] : |\hat{\mu}_{\boldsymbol{a}}^i - \mu_{\boldsymbol{a}}| \geq \epsilon_{\boldsymbol{a}}(t,\delta,\gamma)\} \\ &= M\sum_{\boldsymbol{a}}\left(\sum_{t=1}^{T} P\{|\hat{\mu}_{\boldsymbol{a}}^i - \mu_{\boldsymbol{a}}| \geq \epsilon_{\boldsymbol{a}}(t,\delta,\gamma)\}\right) \\ &\leq M\sum_{\boldsymbol{a}}\left(\sum_{t=1}^{T} 2\exp\left\{-\frac{t\epsilon_{\boldsymbol{a}}(t,n_{\boldsymbol{a}},\delta,\gamma)^2}{2}\right\}\right) \\ &\leq M\sum_{\boldsymbol{a}}\left(\sum_{t=1}^{T} 2\delta^{\frac{t\gamma}{2n_{\boldsymbol{a}}}}\right) \\ &\leq MKT\delta^{\gamma}\end{aligned}$$

Finally, we turn to $P(G \cap (G_{\boldsymbol{a}}^{\text{cross}})^c)$.

$$\begin{aligned}&P(G \cap (G_{\boldsymbol{a}}^{\text{cross}})^c) \\ &= P\left(\bigcup_{i=1}^{M}\{\hat{\mu}_{\boldsymbol{a}}^m(u_{\boldsymbol{a}},\delta,\gamma) + \epsilon_{\boldsymbol{a}}(u_{\boldsymbol{a}},\delta,\gamma) \geq \mu_{\mathbf{1}} - 2\epsilon_{\mathbf{1}}(u_{\boldsymbol{a}}-1,\delta,\gamma)\}\right) \\ &= MP\{\hat{\mu}_{\boldsymbol{a}}^m(u_{\boldsymbol{a}},\delta,\gamma) + \epsilon_{\boldsymbol{a}}(u_{\boldsymbol{a}},\delta,\gamma) \geq \mu_{\mathbf{1}} - 2\epsilon_{\mathbf{1}}(u_{\boldsymbol{a}}-1,\delta,\gamma)\} \\ &= MP\{\hat{\mu}_{\boldsymbol{a}}^m(u_{\boldsymbol{a}},\delta,\gamma) - \mu_{\boldsymbol{a}} \geq \Delta_{\boldsymbol{a}} - \epsilon_{\boldsymbol{a}}(u_{\boldsymbol{a}},\delta,\gamma) - 2\epsilon_{\mathbf{1}}(u_{\boldsymbol{a}}-1,\delta,\gamma)\} \\ &\leq M\exp\left(-\frac{u_{\boldsymbol{a}}(\Delta_{\boldsymbol{a}} - \epsilon_{\boldsymbol{a}}(u_{\boldsymbol{a}},\delta,\gamma) - 2\epsilon_{\mathbf{1}}(u_{\boldsymbol{a}}-1,\delta,\gamma))^2}{2}\right)\end{aligned}$$

Let us pick $u_{\boldsymbol{a}} = \left\lceil \frac{9\gamma^2\log(1/\delta)}{(1-c)^2\Delta_{\boldsymbol{a}}^2} \right\rceil$ for some $c \in (0,1)$, and this will satisfy,

$$\begin{aligned}\Delta_{\boldsymbol{a}} - \epsilon_{\boldsymbol{a}}(u_{\boldsymbol{a}},\delta,\gamma) - 2\epsilon_{\mathbf{1}}(u_{\boldsymbol{a}}-1,\delta,\gamma) &\geq \Delta_{\boldsymbol{a}} - 4\epsilon_{\boldsymbol{a}}(u_{\boldsymbol{a}},\delta,\gamma) \\ &\geq c\Delta_{\boldsymbol{a}}\end{aligned}$$

Combining everything together, we get

$$\begin{aligned}&P((G_{\boldsymbol{a}}^{\text{cross}} \cap G)^c) \\ &\leq MKT\delta^{\gamma} + M\exp\left(\frac{-u_{\boldsymbol{a}}c^2\Delta_{\boldsymbol{a}}^2}{2}\right) \\ &\leq MKT\delta^{\gamma} + M\delta^{\frac{\gamma^2\log(1/\delta)c^2}{(1-c)^2\Delta_{\boldsymbol{a}}^2}}\end{aligned} \tag{16}$$

Choosing $\delta = \frac{1}{T^{2/\gamma}}$, we obtain the following regret bound

$$\begin{aligned}\mathbb{E}[n_{\boldsymbol{a}}(T)] &\leq \left\lceil \frac{18\gamma^2\log(1/\delta)}{(1-c)^2\Delta_{\boldsymbol{a}}^2} \right\rceil \\ &\quad + \left(MKT\delta^{\gamma} + M\delta^{\frac{9\gamma^2\log(1/\delta)c^2}{(1-c)^2\Delta_{\boldsymbol{a}}^2}}\right)T \\ &= \left\lceil \frac{18\gamma\log(T)}{(1-c)^2\Delta_{\boldsymbol{a}}^2} \right\rceil + MK + MT^{-\frac{9\gamma c^2}{(1-c)^2\Delta_{\boldsymbol{a}}^2}+1}\end{aligned}$$

Our goal is now to select c so that the exponent of T in the expression above is negative. In order for $-\frac{16\gamma c^2}{(1-c)^2\Delta_{\boldsymbol{a}}^2} + 1 \leq 0$ to hold, it is sufficient that $c \geq \frac{\Delta_{\boldsymbol{a}}}{\Delta_{\boldsymbol{a}} + 4\sqrt{\gamma}}$. Furthermore, it's clear that $c \in (0,1)$ as originally stated. Thus, with this value of c, our bound above becomes

$$\mathbb{E}[n_{\boldsymbol{a}}(T)] \leq \left\lceil \frac{32\gamma\log(T)}{(1-c)^2\Delta_{\boldsymbol{a}}^2} \right\rceil + M(K^M + 1) + 1 \tag{17}$$

Plugging this into the regret decomposition shows $R_T = O(\log(T))$, completing the proof of theorem 1 ■

While our analysis seems to suggest that the smaller the $\gamma > 0$, the better the performance, however, note that when γ is too small, there is a higher probability that a good arm is eliminated. In the experiments, we show that when γ is sufficiently small, it can outperform even the mUCB algorithm from [10] which is a coordinated version of the UCB algorithm in single player setting.

Proof: [Proof of Theorem 2] We first take the regret decomposition given by equation (13) and partition the tuples $\boldsymbol{a}$ to those whose mean are at most ϵ away from the optimal and those whose means are more than ϵ. This gives us the following inequality:

$$R_T = \sum_{\boldsymbol{a}} \Delta_{\boldsymbol{a}}\mathbb{E}\left[n_{\boldsymbol{a}}(T)\right] \tag{18}$$

$$= \sum_{\Delta_{\boldsymbol{a}} > \epsilon} \Delta_{\boldsymbol{a}}\mathbb{E}\left[n_{\boldsymbol{a}}(T)\right] + \sum_{\Delta_{\boldsymbol{a}} \leq \epsilon} \Delta_{\boldsymbol{a}}\mathbb{E}\left[n_{\boldsymbol{a}}(T)\right] \tag{19}$$

$$\leq \sum_{\Delta_{\boldsymbol{a}} > \epsilon} \Delta_{\boldsymbol{a}}\mathbb{E}\left[n_{\boldsymbol{a}}(T)\right] + \epsilon T. \tag{20}$$

Equation (17) yields:

$$R_T \leq \sum_{\Delta_{\boldsymbol{a}} > \epsilon} O\left(\frac{\log(T)}{\Delta_{\boldsymbol{a}}}\right) + \epsilon T \tag{21}$$

$$\leq \sum_{\Delta_{\boldsymbol{a}} > \epsilon} O\left(\frac{\log(T)}{\epsilon}\right) + \epsilon T \tag{22}$$

Since the last inequality holds for all ϵ, we can pick $\epsilon = \sqrt{\frac{\log(T)}{T}}$ to obtain the result in Theorem 2.

■

V. Numerical Simulations

In this section, we run simulations to verify the empirical performance of our algorithm. In Figure 2, we plot the regret versus time of `mUCB-Intervals` analyzed in this paper in comparison with mDSEE and mUCB from [10]. It should be emphasized these algorithms assume different types of asymmetry. For mDSEE, both action[3] and reward asymmetry are assumed. mUCB assumes only action asymmetry, and `mUCB-Intervals` assume only reward asymmetry.

We perform these simulations with Gaussian rewards sampled from distributions with means sampled uniformly from 0 to 1 and standard deviations uniformly from 0 to

[3]Action asymmetry refers to the scenarios where agents do not observe the action taken by other agents, and thus the joint-action taken remains unknown. In general, action asymmetry is usually easier to solve than reward asymmetry because a proper coordination scheme would reduce action asymmetry to a single-player problem.

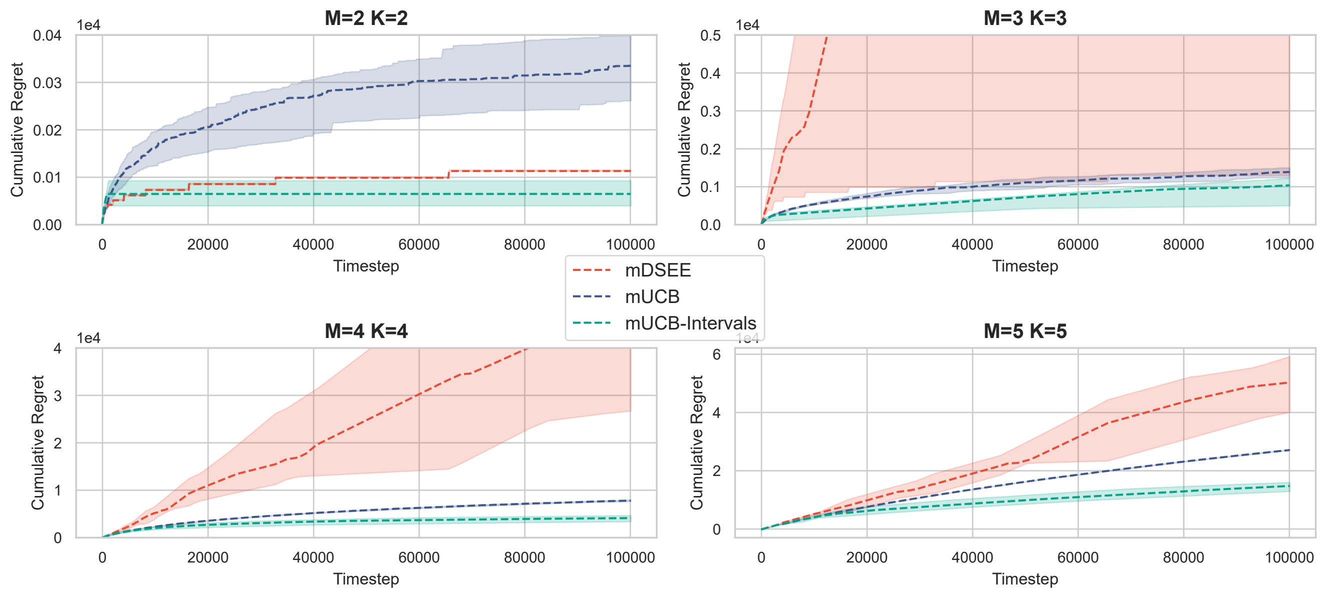


Fig. 2. Plots comparing the regret of mUCB [10], `mUCB-Intervals`, and mDSEE [10] under the same reward environment but under asymmetry in actions, asymmetry in rewards, and asymmetry in both respectively for horizon $T = 10^5$. The shaded regions are 95% confidence intervals. It's clear to see that the algorithm proposed in this paper `mUCB-Intervals` outperforms the SOTA algorithm for asymmetry in both (mDSEE) by a large margin. The superiority of `mUCB-Intervals` over `mDSEE` is more pronounced as the joint action space increases in size.

0.5. For each environment, we run the simulations for a total of $T = 100,000$ rounds and repeat the simulations for 10 times to plot both the median and the 95% confidence interval of regret. We set the hyperparameter $\gamma = 0.5$ for `mUCB-Intervals`. This choice is arbitrary and the performance of `mUCB-Intervals` is not significantly affected by the choice of γ for non-extreme values.

In all simulations, we observe log-like behaviors for the `mUCB-Intervals` algorithm. It is clear from the plots that `mUCB-Intervals` exhibits an absolute competitive advantage over other algorithms as their confidence intervals on regret become disjoint in longer horizons. For robustness, `mUCB-Intervals` also has a significant advantage over mDSEE, as demonstrated by the large red-shaded regions.

These discrepancies in performance can be explained by the differences in the type of asymmetries the algorithms consider. Since `mDSEE` plotted in red also deals with asymmetry in actions, when too many actions have the same empirical mean, it becomes easy for the players to mis-coordinate, hence the difficulties in convergence for this algorithm for large action spaces. On the other hand, `mUCB-Intervals` utilizes observation of actions from the other players to coordinate the actions, giving it a superior performance. On the other hand, we see that `mUCB-Intervals` outperforms the coordinated UCB algorithm mUCB from [10] as well. This is because for small values of γ, there is less exploration so that the `mUCB-Intervals` is able to eliminate suboptimal arms faster and commit to better arms more often. However, one must be careful to not choose a value of γ that is too small, as such a value would discourage sufficient exploration. This will cause god arms to be eliminated too quickly.

In more detail, in the first few rounds, the UCB intervals for `mUCB-Intervals` are large, and thus all of them are included in the set for all players. As the rounds progress, the intervals shrink as they tend towards the true means of the arms. Some sub-optimal arms then get eliminated from the set, giving a higher probability that a good arm is pulled. Eventually, with high probability, all the players will only have 1 arm in their set - the optimal one. From that point forward, the regret curve is horizontal because no additional suboptimal arms are pulled. It is clear from this plot that we have sub-log regret.

VI. Conclusions and Future Work

In this paper we considered a cooperative multiplayer bandit learning problem where the players are only allowed to agree on a strategy beforehand, but cannot communicate during the learning process. The actions of all the players are commonly observed. However, each player receives a noisy version of the reward which cannot be shared with other players. We provide an algorithm `mUCB-Intervals` based on upper and lower confidence bounds scaled by γ that the players can use to select their optimal actions despite the asymmetry in the reward information. For any choice of γ we show that this algorithm can achieve logarithmic (gap-dependent) regret as well as $O(\sqrt{T}\log T)$ gap-independent regret giving us asymptotically optimal regret for our problem. We ran numerical simulations on multiplayer bandit problem and compared it with `mDSEE` from [10], and saw that some choices of γ perform better while others don't. For future work, we can better understand what choices of γ lead to a better performance. We can remove the asymmetry in actions (i.e. consider a more general setting where the players cannot observe the other player's action either) and try to derive an algorithm that gives asymptotically optimal regret. We can also consider a bandit MDP setting, where each joint action now changes the environment for everyone. In this setting we can still consider, asymmetry in rewards, asymmetry in actions, or both.

REFERENCES

[1] Animashree Anandkumar, Nithin Michael, Ao Kevin Tang, and Ananthram Swami. Distributed algorithms for learning and cognitive medium access with logarithmic regret. *IEEE Journal on Selected Areas in Communications*, 29(4):731–745, 2011.

[2] V. Anantharam, P. Varaiya, and J. Walrand. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part ii: Markovian rewards. *IEEE Transactions on Automatic Control*, 32(11):977–982, 1987.

[3] Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *COLT*, pages 41–53, 2010.

[4] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[5] Orly Avner and Shie Mannor. Concurrent bandits and cognitive radio networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 66–81. Springer, 2014.

[6] Ilai Bistritz, Tavor Z Baharav, Amir Leshem, and Nicholas Bambos. One for all and all for one: Distributed learning of fair allocations with multi-player bandits. *IEEE Journal on Selected Areas in Information Theory*, 2(2):584–598, 2021.

[7] Ilai Bistritz and Amir Leshem. Distributed multi-player bandits-a game of thrones approach. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[8] Etienne Boursier and Vianney Perchet. Sic-mmab: Synchronisation involves communication in multiplayer multi-armed bandits. *Advances in Neural Information Processing Systems*, 32, 2019.

[9] Mithun Chakraborty, Kai Yee Phoebe Chua, Sanmay Das, and Brendan Juba. Coordinated versus decentralized exploration in multi-agent multi-armed bandits. In *IJCAI*, pages 164–170, 2017.

[10] William Chang, Mehdi Jafarnia-Jahromi, and Rahul Jain. Online learning for cooperative multi-player multi-armed bandits. *arXiv preprint arXiv:2109.03818*, 2021.

[11] Raphaël Féraud, Réda Alami, and Romain Laroche. Decentralized exploration in multi-armed bandits. In *International Conference on Machine Learning*, pages 1901–1909. PMLR, 2019.

[12] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking*, 20(5):1466–1478, 2012.

[13] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

[14] Dileep Kalathil, Naumaan Nayyar, and Rahul Jain. Decentralized learning for multiplayer multiarmed bandits. *IEEE Transactions on Information Theory*, 60(4):2331–2345, 2014.

[15] Hsu Kao. *Efficient Methods for Optimizing Decentralized Multi-Agent Systems*. PhD thesis, 2022.

[16] Hsu Kao, Chen-Yu Wei, and Vijay Subramanian. Decentralized cooperative reinforcement learning with hierarchical information structure. In *International Conference on Algorithmic Learning Theory*, pages 573–605. PMLR, 2022.

[17] Nikolai Karpov, Qin Zhang, and Yuan Zhou. Collaborative top distribution identifications with limited interaction. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 160–171. IEEE, 2020.

[18] Nathan Korda, Balazs Szorenyi, and Shuai Li. Distributed clustering of linear bandits in peer to peer networks. In *International conference on machine learning*, pages 1301–1309. PMLR, 2016.

[19] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[20] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[21] Keqin Liu and Qing Zhao. Distributed learning in multi-armed bandit with multiple players. *IEEE Transactions on Signal Processing*, 58(11):5667–5681, 2010.

[22] Setareh Maghsudi and Slawomir Stanczak. Channel selection for network-assisted d2d communication via no-regret bandit learning with calibrated forecasting. *IEEE Transactions on Wireless Communications*, 14(3):1309–1322, 2014.

[23] Weichao Mao, Tamer Basar, Lin F Yang, and Kaiqing Zhang. Decentralized cooperative multi-agent reinforcement learning with exploration. *arXiv preprint arXiv:2110.05707*, 2021.

[24] Weichao Mao, Lin Yang, Kaiqing Zhang, and Tamer Basar. On improving model-free algorithms for decentralized multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 15007–15049. PMLR, 2022.

[25] David Martínez-Rubio, Varun Kanade, and Patrick Rebeschini. Decentralized cooperative stochastic bandits. *arXiv preprint arXiv:1810.04468*, 2018.

[26] David Martínez-Rubio, Varun Kanade, and Patrick Rebeschini. Decentralized cooperative stochastic bandits. 2019.

[27] Naumaan Nayyar, Dileep Kalathil, and Rahul Jain. On regret-optimal learning in decentralized multiplayer multiarmed bandits. *IEEE Transactions on Control of Network Systems*, 5(1):597–606, 2018.

[28] Jonathan Rosenski, Ohad Shamir, and Liran Szlak. Multi-player bandits–a musical chairs approach. In *International Conference on Machine Learning*, pages 155–163. PMLR, 2016.

[29] Shahin Shahrampour, Alexander Rakhlin, and Ali Jadbabaie. Multi-armed bandits in multi-agent networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2786–2790. IEEE, 2017.

[30] Chengshuai Shi, Wei Xiong, Cong Shen, and Jing Yang. Decentralized multi-player multi-armed bandits with no collision information. In *International Conference on Artificial Intelligence and Statistics*, pages 1519–1528. PMLR, 2020.

[31] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[32] Balazs Szorenyi, Róbert Busa-Fekete, István Hegedus, Róbert Ormándi, Márk Jelasity, and Balázs Kégl. Gossip-based distributed stochastic bandit algorithms. In *International Conference on Machine Learning*, pages 19–27. PMLR, 2013.

[33] Chao Tao, Qin Zhang, and Yuan Zhou. Collaborative learning with limited interaction: Tight bounds for distributed exploration in multi-armed bandits. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 126–146. IEEE, 2019.

[34] Po-An Wang, Alexandre Proutiere, Kaito Ariu, Yassir Jedra, and Alessio Russo. Optimal algorithms for multiplayer multi-armed bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 4120–4129. PMLR, 2020.

[35] Jie Xu, Cem Tekin, Simpson Zhang, and Mihaela Van Der Schaar. Distributed multi-agent online learning based on global feedback. *IEEE Transactions on Signal Processing*, 63(9):2225–2238, 2015.

[36] Yaolong Yu, Haifeng Xu, and Haipeng Chen. Learning correlated stackelberg equilibrium in general-sum multi-leader-single-follower games. *arXiv preprint arXiv:2210.12470*, 2022.