Decentralized Learning for Multi-player Multi-armed Bandits

Dileep Kalathil, Naumaan Nayyar and Rahul Jain
Department of Electrical Engineering,
University of Southern California
{manisser, nnayyar, rahul.jain}@usc.edu

Abstract—We consider the problem of distributed online learning with multiple players in multi-armed bandit models. Each player can pick among multiple arms. As a player picks an arm, it gets a reward from an unknown distribution with an unknown mean. The arms give different rewards to different players. If two players pick the same arm, there is a “collision”, and neither of them get any reward. There is no dedicated control channel for coordination or communication among the players. Any other communication between the users is costly and will add to the regret. We propose an online index-based learning policy called dUCB$_4$ that trades off exploration v. exploitation in the right way, and achieves expected regret that grows at most near $O(\log^2 T)$. The motivation comes from opportunistic spectrum access by multiple secondary users in cognitive radio networks wherein they must pick among various wireless channels that look different to different users.

Index Terms—Distributed adaptive control, multi-armed bandits, online learning, multi-agent systems.

I. INTRODUCTION

In [1], Lai and Robbins introduced the classical non-Bayesian multi-armed bandit model. Such models capture the essence of the learning problem that players face in an unknown environment, where the players must not only explore to learn but also exploit in choosing the best arm. Specifically, suppose a player can choose between $N$ arms. Upon choosing an arm $i$, it gets a reward from a distribution with density $f(x, \theta_i)$. Time is slotted, and players do not know the distributions (nor any statistics about them). The problem is to find a learning policy that minimizes the expected regret over some time horizon $T$. It was shown by Lai and Robbins that there exists an index-type policy that achieves expected regret that grows asymptotically as $\log T$, and this is order-optimal, i.e., there exists no causal policy that can do better.

This was generalized by Anantharam, et al to the case of multiple plays, i.e., when the player can pick multiple arms at the same time [2]. In another paper, they also presented an extension to the case when the rewards are not i.i.d. but Markovian [3]. Since the mid-80s, many extensions and re-interpretations of Lai and Robbins index policy have been given, including by Agrawal [4] who presented a considerable simplification, and by Auer, et al [5], who presented many variations.

Recently, there is increasing interest in multi-armed bandit models, partly because of opportunistic spectrum access problems. Consider a user who must choose between $N$ wireless channels. Yet, it knows nothing about the channel statistics, i.e., has no idea of how good the channel it, and what rate it may expect to get on a particular channel. These have been formulated as multi-armed bandit problems, and index-type policies have been proposed for choosing spectrum channels. In many scenarios, there are multiple users accessing the channels at the same time. Each of these users must be matched to a different channel. These have been formulated as a combinatorial multi-armed bandit problem [6], and it was shown that an “index-matching” algorithm that at each instant determines a matching by solving a sum-index maximization problem achieves $O(\log T)$ regret, and this is indeed order-optimal.

In other settings, the users cannot coordinate, and the problem must be solved in a decentralized manner. Thus, settings where all channels (arms) are identical for all users have been considered, and index-type policies that can achieve coordination have been proposed that get $O(\log T)$ regret [7], [8], [9]. The regret scales only polynomially in the number of users and channels. Surprisingly, the lack of coordination between the players asymptotically imposes no additional cost or regret.

In this paper, we consider the decentralized multi-armed bandit problem with distinct arms for each players. All players together must discover the best arms to play as a team. However, since they are all trying to learn at the same time, they may collide when two or more pick the same arm. We propose an index-type policy dUCB$_4$ based on a variation of the UCB$_1$ index. At its’ heart is a distributed bipartite matching algorithm such as Bertsekas’ auction algorithm [10]. This algorithm operates in rounds, and in each round prices for various arms are determined based on bid-values. This imposes communication (and computation) cost on the algorithm that must be accounted for. Nevertheless, we show that the dUCB$_4$ algorithm that we introduce still achieves
(at most) near-$O(\log^2 T)$ growth in expected regret. A lower bound, however, is not known at this point.

II. MODEL AND PROBLEM FORMULATION

We consider an $N$-armed bandit with $M$ players. In a wireless cognitive radio setting, each arm could correspond to a channel, and each player to a user who wants to use a channel. Time is slotted, and at each instant each player picks an arm. There is no dedicated control channel for coordination among the players. So, potentially more than one players can pick the same arm at the same instant. We will regard that as a collision. Player $i$ playing arm $k$ at time $t$ yields i.i.d. reward $S_{ik}(t)$ with univariate density function $f(s, \theta_{ik})$, where $\theta_{ik}$ is a parameter in the set $\Theta_{ik}$. We will assume that the rewards are bounded, and without loss of generality lie in $[0, 1]$. Let $\mu_{i,k}$ denote the mean of $S_{ik}(t)$ w.r.t. the pdf $f(s, \theta_{ik})$. We assume that the parameter vector $\theta = (\theta_{ij}, 1 \leq i \leq M, 1 \leq j \leq N)$ is unknown to the players, i.e., the players have no information about the mean, the distributions or any other statistics about the rewards from various arms other than what they observe while playing. We also assume that each player can only observe the rewards that they get. When there is a collision, we will assume that all players that choose the arm on which there is a collision get reward zero. This can be relaxed to a setting where the players share the reward in some manner though the results do not change appreciably.

Let $X_{ij}(t)$ be the reward that player $i$ gets from arm $j$. Thus, if player $i$ plays arm $k$ at time $t$ (and there is no collision), $X_{ik}(t) = S_{ik}(t)$, and $X_{ij}(t) = 0, j \neq k$. Denote the action of player $i$ at time $t$ by $a_i(t) \in A := \{1, \ldots, N\}$. Then, the history seen by player $i$ at time $t$ is $H_i(t) = \{(a_1(1), X_{1,a_1(1)}(1)), \cdots, (a_i(t), X_{i,a_i(t)}(t))\}$ with $H_i(0) = \emptyset$. A policy $\pi_i = (\pi_i(t))_{t=1}^\infty$ for player $i$ is a sequence of maps $\pi_i(t) : H_i(t) \rightarrow A$ that specifies the arm to be played at time $t$ given the history seen by the player. Let $\mathcal{P}(N)$ be the set of vectors such that $\mathcal{P}(N) := \{a = (a_1, \ldots, a_M) : a_i \in A, a_i \neq a_j, \text{for } i \neq j\}$.

The players have a team objective, namely over a time horizon $T$, they

$$\max_{\pi} \mathbb{E} \left( \sum_{t=1}^T \sum_{j=1}^M X_{i,a_i(t)}(t) \right),$$

the expected sum of rewards over some time horizon $T$. If the parameters $\mu_{i,j}$ are known, this could easily be achieved by picking a bipartite matching

$$k^* = \arg \max_{k \in \mathcal{P}(N)} \sum_{i=1}^M \mu_{i,k_i},$$

i.e., the optimal bipartite matching with expected reward from each match. But the expected rewards are unknown. So, the players must pick learning policies that minimize the expected regret, defined for policies $\pi = (\pi_i, 1 \leq i \leq M)$ as

$$R_\pi(T) = T \sum_{i=1}^M \mu_{i,k_i^*} - \mathbb{E}_\pi \left[ \sum_{t=1}^T \sum_{i=1}^M X_{i,\pi_i(t)}(t) \right].$$

Our goal is to find a decentralized algorithm that players can use such that together they minimize the regret.

III. SOME VARIATIONS ON SINGLE PLAYER MULTI-ARMED BANDITS

We first present some variations on the single player non-Bayesian multi-armed bandit model. These will later prove useful for the multi-player problem though they are also of independent interest.

A. UCB$_1$ with index recomputation every $L$ slots

Consider the classical single player non-Bayesian $N$-armed bandit problem. At each time $t$, the player picks a particular arm, say $j$, and gets a random reward $X_j(t)$. The reward $X_j(t), 1 \leq t \leq T$ are independent and identically distributed according to some unknown probability measure with an unknown expectation $\mu_j$. Without loss of generality, assume that $\mu_1 > \mu_2 > \cdots > \mu_N$, for $i = 2, \cdots, N - 1$. Let $n_j(t)$ denote the number of times the arm $j$ has been played by time $t$. Denote $\Delta_j = \mu_1 - \mu_j$. Also, let $\Delta_{\min} = \min_{1 < j \neq 1} \Delta_j$ and $\Delta_{\max} = \max_j \Delta_j$. The regret for any policy $\pi$ is then given by

$$R_\pi(T) := \mu_1 T - \sum_{j=1}^K \mu_j E_\pi[n_j(T)].$$

Define an index

$$g_j(t) = X_j(t) + \sqrt{\frac{2 \log(t)}{n_j(t)}},$$

where $X_j(t)$ is the average reward obtained by playing arm $j$ by time $t$. It is defined as $X_j(t) = \frac{\sum_{r \leq t} r_j(r)}{n_j(t)}$, where $r_j(t)$ is the reward obtained from arm $j$ at time $t$. If the arm $j$ is played at time $t$ then $r_j(t) = X_j(t)$ and otherwise $r_j(t) = 0$. Now, an index-based policy called UCB$_1$ [5] is to pick the arm that has the highest index at each instant. It can be shown that this algorithm achieves regret that grows logarithmically in $T$. Asymptotic scaling was established by Lai and Robbins [1] and Agrawal [4], while non-asymptotic scaling was established by Auer, et al [3].

An easy variation of the above algorithm which will be useful in our analysis of subsequent algorithms is the following. Suppose the index is re-computed only once every $L$ slots. In that case, it is easy to establish the following.
Theorem 1. Under the UCB₃ algorithm with recomputation of the index once every L slots, the expected regret by time T is given by

\[ \mathcal{R}_{\text{UCB}_3}(T) \leq \sum_{j=1}^{K} \frac{8L \log T}{\Delta_j} + L \left(1 + \frac{\pi^2}{3}\right) \sum_{j=1}^{K} \Delta_j. \] (3)

The proof follows [5] and taking into account the fact that every time a suboptimal arm is selected, it is played for the next L slots rather than just one time.

B. UCB₄ Algorithm when index computation is costly

Often, learning algorithms pay a penalty or cost for computation. This is particularly the case when the algorithms must solve combinatorial optimization problems that are NP-hard. Such costs also arise in decentralized settings wherein algorithms pay a communication cost for coordination between the decentralized players. This is indeed the case, as we shall see later when we present an algorithm to solve the decentralized multi-armed bandit problem. Here, however, we will just consider an "abstract" communication or computation cost. The problem we formulate below can be solved with better bounds that also help in decentralization.

Consider a computation cost every time the index is recomputed. Let the cost be \( C \). We define an arm \( j \) to be the best arm if \( j^*(t) \in \arg \max_{1 \leq i \leq K} g_i(t) \).

We will use the following concentration inequality.


Let \( X_1, \ldots, X_n \) be random variables with common range such that \( \mathbb{E}[X_i] = \mu \). Then, \( S_n = \sum_{i=1}^{n} X_i \). Then for all \( a \geq 0 \),

\[ \mathbb{P}[S_n \geq n\mu + a] \leq e^{-2a^2/n}, \]
\[ \mathbb{P}[S_n \leq n\mu - a] \leq e^{-2a^2/n}. \] (6)

Algorithm 1: UCB₄

Initialization: Select each arm \( j \) once for \( t \leq K \). Update the UCB₄ indices. Set \( \eta = 1 \).

\begin{algorithmic}
  \For {\( t \leq T \)}
    \If {\( \eta = 2^q, q = 0, 1, 2, \ldots \)}
      Update the index vector \( g(t) \);
      Compute the best arm \( j^*(t) \);
      \If {\( j^*(t) \neq j^*(t-1) \)}
        Reset \( \eta = 1 \);
      \EndIf
    \Else
      \State \( j^*(t) = j^*(t-1) \);
    \EndIf
    Play arm \( j^*(t) \);
    Increment counter \( \eta = \eta + 1 \); \( t = t + 1 \);
  \EndFor
\end{algorithmic}

Thus, \( \mathcal{R}_{\text{UCB}_4}(T) = O(\log^2 T) \).

The proof is long and due to page limitations is omitted. For the complete proof, see [12].

Remarks. 1. It is easy to show that the lower bound for the single player MAB problem with computation costs is \( \Omega(\log T) \). This can be achieved by the UCB₂ algorithm [5]. To see this, note that the number of times the player selects a suboptimal arm when using UCB₂ is \( O(\log T) \). Since \( \mathbb{E}[n_j(T)] = O(\log T) \), we get \( \mathbb{E}[\sum_{j=1}^{K} n_j(T)] = O(\log T) \), and also \( \mathbb{E}[n_m(T)] = O(\log T) \). Now, since the epochs are not getting reset after every switch and are exponentially spaced, the number of updates that result in the optimal allocation, \( m_1(T) \leq \log T \). These together, then yield

\[ \mathcal{R}_{\text{UCB}_2}(T) \leq \sum_{j=1}^{K} \mathbb{E}[n_j(T)] \Delta_j + C \mathbb{E}[m(T)] = O(\log T). \]

2. Variations of the UCB₂ algorithm that use a deterministic schedule can also be used. But none of these algorithms can be used in solving the decentralized multi-armed bandit problem that we introduce in the next section. This is the main reason for introducing the UCB₄ algorithm.

C. Algorithms with finite precision indices

Often, there might be a cost to compute the indices to a particular precision. In that case, indices may be known up to some \( \epsilon \) precision, and it may not possible to tell which of two indices is greater if they are within \( \epsilon \) of
each other. The question then is how is the performance of various index-based policies such as UCB₁, UCB₄, etc. affected if there are limits on index resolution, and only an arm with an ϵ-highest index can be picked. We first show that if Δₘᵢₙ is known, we can fix a precision ϵ < Δₘᵢₙ, so that UCB₄ algorithm will achieve order log-squared regret growth with T. If Δₘᵢₙ is not known, we can pick a sequence {εₜ} such that εₜ → 0, as t → ∞. Denote the cost of computation for ϵ-precision be C(ϵ). We assume that C(ϵ) → ∞ monotonically as ϵ → 0.

**Theorem 3.** (i) If Δₘᵢₙ is known, choose an ϵ < Δₘᵢₙ. Then, the expected regret of the UCB₄ algorithm with ϵ-precise computations is given by

\[
\hat{R}_{\text{UCB₄}}(T) \leq (\Delta_{\text{max}} + C(\epsilon))(1 + \log T) \cdot \left(\sum_{j=1}^{K} \frac{16 \log T}{(\Delta_j - \epsilon)^2} + 2K\right).
\]

Thus, \(\hat{R}_{\text{UCB₄}}(T) = O(\log^2 T)\).

(ii) If Δₘᵢₙ is unknown, denote \(\epsilon_{\text{min}} = \Delta_{\text{min}}/2\) and choose a sequence \{εₜ\} such that εₜ → 0 as t → ∞. Then, there exists a t₀ > 0 such that for all T > t₀,

\[
\hat{R}_{\text{UCB₄}}(T) \leq (\Delta_{\text{max}} + C(\epsilon_{\text{min}})) t₀ + (\Delta_{\text{max}} + C(\epsilon_T))(1 + \log T) \cdot \left(\sum_{j=1}^{K} \frac{16 \log T}{(\Delta_j - \epsilon_{\text{min}})^2} + 2K\right).
\]

where \(t₀\) is such that \(\epsilon_{t₀} = \epsilon_{\text{min}}\).

For \(C(\epsilon) = \log 1/\epsilon \) and \(\epsilonₜ = 1/\log k\) t (log iterated k times), we get

\[
\hat{R}_{\text{UCB₄}}(T) = O(\log^{k+1} T \cdot \log^2 T).
\]

The proof is omitted due to page limitations. For the complete proof, see [12].

**Remarks.** We can make the asymptotic order of regret arbitrarily close to \(O(\log^2 T)\) by choosing a sequence \{εₜ\} that decreases arbitrarily slowly to zero. This, however cause \(t₀\) to be arbitrarily large. For example, if \(C(\epsilon) = \log 1/\epsilon \) and \(\epsilonₜ = 1/\log \log t\), the expected regret is bounded by \(O(\log \log \log T \log^2 T)\), but \(t₀ = 2^{2^{k/\epsilon_{\text{min}}}}\).

**IV. THE DECENTRALIZED MAB PROBLEM**

We now consider the decentralized multi-armed bandit problem wherein multiple players are playing at the same time. Players have no information about means or distributions of rewards from various arms. There are no dedicated control channels for coordination or communication between the players. If two or more players pick the same arm, we assume that neither gets any reward. Thus, this is an online learning problem of distributed bipartite matching.

Distributed algorithms for bipartite matching algorithms are known [13, 14] which determine an ϵ-optimal matching with a ‘minimum’ amount of information exchange and computation. However, every run of this distributed bipartite matching algorithm incurs a cost, which is a combination of computation cost and communication cost to exchange information necessary for decentralization. Let \(C\) be the cost per run, and \(m(t)\) denotes the number of times the distributed bipartite matching algorithm is run by time t. Then, under policy \(π\), the expected regret is given by

\[
\hat{R}_π(T) = T \sum_{i=1}^{M} \mu_i, k_i^{∗∗} - \mathbb{E}_π \left[ \sum_{t=1}^{T} \sum_{i=1}^{M} X_{i, \pi(t)}(t) \right] + C \mathbb{E}[m(t)].
\]

**Temporal Structure.** We will divide time into frames. Each frame is one of two kinds: a decision frame, and an exploitation frame. In the decision frame, the index will be recomputed, and the distributed bipartite matching algorithm run again to determine the new matching. The length of such a frame can be seen as cost of the algorithm. In the exploitation frame, the current matching is exploited without updating the indices. When a sub-optimal matching is played in such frames, it contributes to regret. Later, we will allow length of the frames to increase with time.

We now present the dUCB₄ algorithm, a decentralized version of UCB₄. For each player \(i\) and each arm \(j\), we define a dUCB₄ index at the end of frame \(t\) as

\[
g_{i,j}(t) = \bar{X}_{i,j}(t) + \sqrt{\frac{(M + 2) \log n_i(t)}{n_i(t)}}.
\]

where \(n_i(t)\) is the number of successful plays (without collisions) of player \(i\) by frame \(t\), \(n_{i,j}(t)\) is the number of times player \(i\) picks arm \(j\) successfully by frame \(t\). \(\bar{X}_{i,j}(t)\) is the sample mean of rewards from arm \(j\) for player \(i\) from \(n_{i,j}(t)\) samples. Let \(g(t)\) denote the vector \((g_{i,j}(t))\), \(i = 1 : M, j = 1 : N\). Note that \(g\) is computed only in the decision frames using the information available upto that time. Each player now uses the dUCB₄ algorithm. We will refer to an ϵ-optimal distributed bipartite matching algorithm as dM₄(g(t)) that yields a solution \(k^{∗}(t) := (k_{i}^{∗}(t), \ldots, k_{M}^{∗}(t)) \in \mathcal{P}(N)\) such that

\[
\sum_{i=1}^{M} g_{i,k_i^{∗}}(t) \geq \sum_{i=1}^{M} g_{i,k_i}(t) - \epsilon, \forall k \in \mathcal{P}(N), k \neq k^{∗}.
\]

Let \(k^{∗∗} \in \mathcal{P}(N)\) be such that

\[
k^{∗∗} \in \arg\max_{k \in \mathcal{P}(N)} \sum_{i=1}^{M} \mu_i, k_i^{∗}.
\]

i.e., an optimal bipartite matching with expected rewards
The cost $C$ by the players to recompute the matching. Every time $k$ accrues either when the matching $\Delta$ happens for example when indices must be communicated $\epsilon$ when such an algorithm is run only for a finite number of $\Delta_{\text{min}}$ is not known, we can pick a sequence $\{\epsilon_t\}$ such that $\epsilon_t \to 0$, as $t \to \infty$. In a decentralized bipartite matching algorithm, the precision $\epsilon$ will depend on the amount of information exchanged in the decision frames. It, thus, is some monotonically decreasing function $\epsilon = f(L)$ of their length $L$ such that $\epsilon \to 0$ as $L \to \infty$. Thus, we must pick a sequence $\{L_t\}$ such that $L_t \to \infty$. Clearly, $C(f(L_t)) \to \infty$ as $t \to \infty$. This can happen arbitrarily slowly.

**Theorem 4.** (i) Let $\epsilon$ be the precision of the bipartite matching algorithm and the precision of the index representation. If $\Delta_{\text{min}}$ is unknown, choose $\epsilon$ such that $\epsilon < \Delta_{\text{min}}/(M+1)$. Let $L$ be the length of a frame. Then, the expected regret of the $\text{dUCB}_4$ algorithm is $\hat{R}_{\text{dUCB}_4}(T)$

$$
\leq \left(L\Delta_{\text{max}} + C(f(L))(1 + \log T)\right) \cdot 
\left(4M^3(M+2)N \log T \over (\Delta_{\text{min}} - (M+1)\epsilon)^2 + NM(2M+1)\right).
$$

Thus, $\hat{R}_{\text{dUCB}_4}(T) = O(\log^2 T)$.

(ii) When $\Delta_{\text{min}}$ is unknown, denote $\epsilon_{\text{min}} = \Delta_{\text{min}}/(2(M+1))$ and let $L_t \to \infty$ as $t \to \infty$. Then, there exists a $t_0 > 0$ such that for all $T > t_0$, $\hat{R}_{\text{dUCB}_4}(T) \leq \left(L_t \Delta_{\text{max}} + C(f(L_t))t_0 + (L_t \Delta_{\text{max}} + C(f(L_t)))(1 + \log T)\right) \cdot 
\left(4M^3(M+2)N \log T \over (\Delta_{\text{min}} - \epsilon_{\text{min}})^2 + NM(2M+1)\right),
$$

where $t_0$ is such that $\epsilon_{\text{min}} = f(L_t)$. For $\epsilon_t = 2^{-L_t}$, $C(\epsilon_t) = \log 1/\epsilon_t$ and $L_t = \log \log t$, we get

$$
\hat{R}_{\text{dUCB}_4}(T) \leq O(\log \log T \cdot \log^2 T).
$$

**Proof:** (i) First, we obtain a bound for $L = 1$. Then, appeal to a result like Theorem 1 to obtain the result for general $L$. The implicit dependence between $\epsilon$ and $L$ through the function $f(\cdot)$ does not affect this part of the analysis. Details are omitted due to page limitations. For the most part, our proof follows the main result in [6].

We first upper bound the number of sub-optimal plays. We define $\tilde{n}_{i,j}(t), 1 \leq i \leq M, 1 \leq j \leq N$. Then, $\tilde{n}(T)$ denote the total number of suboptimal plays. Then, clearly,
\[ \hat{n}(T) = \sum_{i=1}^{M} \sum_{j=1}^{N} \hat{n}_{i,j}(T). \] Let \( \hat{I}_{i,j}(t) \) be the indicator function which is equal to 1 if \( \hat{n}_{i,j}(t) \) is incremented by one, at time \( t \). When \( \hat{I}_{i,j}(t) = 1 \), there will be a corresponding matching \( k(t) \neq k^* \) such that \( k_i(t) = j \). In the following, we denote it as \( k_i \), omitting the time index. Now, we can upper bound

\[
\hat{n}_{i,j}(T) \leq l + \sum_{m=1}^{T} \sum_{p=0}^{\infty} 2^{p} I \left\{ \sum_{i=1}^{M} g_{i,k^*_i} \right\} (m + 2^p - 1)
= (M + 1) \epsilon + \sum_{i=1}^{M} \hat{X}_{i,k_i} (m + 2^p - 1),
\]

If each index has an error of at most \( \epsilon \), the sum of \( M \) terms may introduce an error of at most \( M \epsilon \). In addition, the distributed bipartite matching algorithm \( \text{dBM} \) itself yields only an \( \epsilon \)-optimal matching. This accounts for the term \( (M + 1) \epsilon \) above. Thus, \( \hat{n}_{i,j}(T) \)

\[
\leq l + \sum_{m=1}^{T} \sum_{p=0}^{\infty} 2^{p} \sum_{i=1}^{M} \left( X_{i,k^*_i} (m + 2^p - 1)
\right.
+ c_{m+2p-1,n_i,k_i} (m + 2^p - 1)
\left. - l \right) \sum_{m=1}^{M} \left( X_{i,k_i} (m + 2^p - 1)
\right)
\leq (M + 1) \epsilon + \sum_{i=1}^{M} \hat{X}_{i,k_i} (m + 2^p - 1)
\]
\[
= (M + 1) \epsilon + \sum_{i=1}^{M} \hat{X}_{i,k_i} (m + 2^p - 1)
\]

Now, it is easy to observe that the event

\[
I \left\{ \sum_{i=1}^{M} \left( X_{i,k^*_i} (m + 2^p) + c_{m+2p,n_i,k_i} \right) \} \leq (M + 1) \epsilon + \sum_{i=1}^{M} \left( X_{i,k_i} (m + 2^p) + c_{m+2p,n_i,k_i} \right) \}
\]

implies at least one of the following events:

\[
A_i := \left\{ X_{i,k^*_i} (m + 2^p) \leq \mu_{i,k^*_i} - c_{m+2p,n_i,k^*_i} \right\}, \quad 1 \leq i \leq M,
\]

\[
B_i := \left\{ X_{i,k_i} (m + 2^p) \geq \mu_{i,k_i} + c_{m+2p,n_i,k_i} \right\}, \quad 1 \leq i \leq M.
\]

Using the Chernoff-Hoeffding inequality, we get

\[
P[A_i] \leq (m + 2^p)^{-2(M+2)}, \quad 1 \leq i \leq M,
\]

\[
P[B_i] \leq (m + 2^p)^{-2(M+2)}, \quad 1 \leq i \leq M.
\]

For \( \sum_{i=1}^{M} \mu_{i,k^*_i} - \sum_{i=1}^{M} \mu_{i,k_i} - (M + 1) \epsilon \)

\[
\geq \sum_{i=1}^{M} \mu_{i,k^*_i} - \sum_{i=1}^{M} \mu_{i,k_i} - (M + 1) \epsilon
\]

\[
- (\Delta_{\min} - (M + 1) \epsilon) \geq 0
\]

So, we get,

\[
E[\hat{n}_{i,j}(T)] \leq \sum_{i=1}^{M} \sum_{j=1}^{N} E[\hat{n}_{i,j}(T)] \leq 4M^2(M + 2) \frac{\log T}{(\Delta_{\min} - (M + 1) \epsilon)^2} + 2M \sum_{m=1}^{\infty} \sum_{p=0}^{\infty} \frac{m + 2^p}{(m + 2^p)^{-2(M+2)}},
\]

\[
\leq \frac{4M^4(M + 2) \log T}{(\Delta_{\min} - (M + 1) \epsilon)^2} + 2M \left( \sum_{m=1}^{\infty} \sum_{p=0}^{\infty} \frac{m + 2^p}{(m + 2^p)^{-2(M+2)}} \right)^{-1}
\]

\[
\leq \frac{4M^2(M + 2) \log T}{(\Delta_{\min} - (M + 1) \epsilon)^2} + (2M + 1) M N.
\]

Now, by proof of the Theorem \( \square \)
We can now bound the regret,
\[ R_{\text{dUCB}_k}(T) = \sum_{k \in \mathcal{P}(N), k \neq k^*} \Delta_k \mathbb{E}[\hat{n}_{i,k}(T)] + C\mathbb{E}[m(T)] \]
\[ \leq \Delta_{\text{max}} \sum_{k \in \mathcal{P}(N), k \neq k^*} \mathbb{E}[\hat{n}_{i,k}(T)] + C\mathbb{E}[m(T)] \]
\[ = \Delta_{\text{max}} \mathbb{E}[\hat{n}(T)] + C\mathbb{E}[m(t)]. \]
For a general \( L \), by Theorem 1
\[ \hat{R}_{\text{dUCB}_k}(T) \leq L \Delta_{\text{max}} \mathbb{E}[\hat{n}(T)] + C(f(L))\mathbb{E}[m(T)] \]
\[ \leq (L\Delta_{\text{max}} + C(f(L))(1 + \log T))\mathbb{E}[\hat{n}(T)]. \]
Now, using the bound (8), we get the desired upper bound on the expected regret.

(ii) Since \( \epsilon_t = f(L_t) \) is a monotonically decreasing function of \( L_t \) such that \( \epsilon_t \to 0 \) as \( L_t \to \infty \), there exists a \( t_0 \) such that for \( t > t_0 \), \( \epsilon_t < \epsilon_{\text{min}} \). We may get a linear regret up to time \( t_0 \) but after that by the analysis of Theorem 1 regret grows only sub-linearly. Since \( C(\cdot) \) is monotonically increasing, \( C(f(L_t)) \geq C(f(L_{t_0})) \), \( \forall t \leq T \), and we get the desired result. The last part is illustrative and can be trivially established using the obtained bound on the regret in (ii).

Remark. The UCB2 algorithm described in [5] performs computations only at exponentially spaced time epochs. So, it is natural to imagine that a decentralized algorithm based on it could be developed, and get a better regret bound. Unfortunately, the single player UCB2 algorithm has an obvious weakness: regret is linear in the number of arms. Thus, the decentralized/combinatorial extension of UCB2 would yield regret growing exponentially in the number of players and arms. We use a similar index but a different scheme, allowing us to achieve poly-log regret growth and a linear memory requirement for each player.

V. DISTRIBUTED BIPARTITE MATCHING: ALGORITHM AND IMPLEMENTATION

In the previous section, we referred to an unspecified distributed algorithm for bipartite matching dBM, that is used by the dUCB4 algorithm. We now present one such algorithm, namely, Bertsekas’ auction algorithm [10], and its distributed implementation. We note that the presented algorithm is not the only one that can be used. The dUCB4 algorithm will work with a distributed implementation of any bipartite matching algorithm. For example, the algorithms given in [14] can be used.

Consider a bipartite graph with \( M \) players on one side, and \( N \) arms on the other, and \( M \leq N \). Each player \( i \) has a value \( \mu_{i,j} \) for each arm \( j \). Each player knows only his own values.

Let us denote by \( k^* \), a matching that maximizes the matching surplus \( \sum_{i,j} \mu_{i,j} x_{i,j} \), where the variable \( x_{i,j} \)
is 1 if \( i \) is matched with \( j \), and 0 otherwise. Note that \( \sum_i x_{i,j} \leq 1, \forall j \), and \( \sum_j x_{i,j} \leq 1, \forall i \).

Our goal is to find an \( \epsilon \)-optimal matching. We call any matching \( k^* \) to be \( \epsilon \)-optimal if \( \sum_i \mu_{i,k^*(i)} - \sum_i \mu_{i,k^*(i)} \leq \epsilon \).

Algorithm 3: dBM (Bertsekas Auction Algorithm)
1: All players \( i \) initialize price \( p_j = 0 \), \( \forall \) channels \( j \);
2: while (prices change) do
3: \hspace{1em} Player \( i \) communicates his preferred arm \( j_i^* \) and bid \( \hat{b}_i = \max_j (\mu_{i,j} - p_j) - 2\max_j (\mu_{i,j} - p_j) + \frac{\epsilon}{MT} \) to all other players.
4: \hspace{1em} Each player determines on his own if he is the winner \( i_j^* \) on arm \( j \);
5: \hspace{1em} All players set prices \( p_j = \mu_{i_j^*,j} \);
6: end while

Here, \( 2\max_j \) is the second highest maximum over all \( j \). The best arm for a player \( i \) is arm \( j_i^* = \arg \max_j (\mu_{i,j} - p_j) \). The winner \( i_j^* \) on an arm \( j \) is the player who submitted the highest bid on that arm.

The following lemma in [10] establishes that Bertsekas’ auction algorithm will find the \( \epsilon \)-optimal matching in a finite number of steps, with an upper bound that depends on problem primitives.

Lemma 1. [10] Given \( \epsilon > 0 \), Algorithm 3 with rewards \( \mu_{i,j} \), for player \( i \) playing the \( j \)th arm, converges to a matching \( k^* \) such that \( \sum_i \mu_{i,k^*(i)} - \sum_i \mu_{i,k^*(i)} \leq \epsilon \) where \( k^* \) is an optimal matching. Furthermore, this convergence occurs in less than \( \frac{M^2 \max_{i,j} (\mu_{i,j})}{\epsilon} \) iterations.

The temporal structure of the dUCB4 algorithm is such that time is divided into frames of length \( L \). Each frame is either a decision frame, or an exploitation frame. In the exploitation frame, each player plays the arm it was allocated in the last decision frame. The distributed bipartite matching algorithm (e.g. based on Algorithm 3), is run in the decision frame. The decision frame has an interrupt phase of length \( M \) and negotiation phase of length \( L - M \). We now describe an implementation structure for these phases in the decision frame.

Interrupt Phase: The interrupt phase can be implemented very easily. It has length \( M \) time slots. On a pre-determined channel, each player by turn transmits a ‘1’ if the arm with which it is now matched has changed, ‘0’ otherwise. If any user transmits a ‘1’, everyone knows that the matching has changed, and they reset their counter \( \eta = 1 \).

Negotiation Phase: The information needed to be exchanged to compute an \( \epsilon \)-optimal matching is done in the negotiation phase. We first provide a packetized implementation of the negotiation phase. The negotiation phase consists of \( J \) subframes of length \( M \) each (See
In each subframe, the users transmit a packet by turn. The packet contains bid information: (channel number, bid value). Since all users transmit by turn, all the users know the bid values by the end of the subframe, and can compute the new allocation, and the prices independently. The length of the subframe, $J$, determines the precision $\epsilon$ of the distributed bipartite matching algorithm. Note that in the packetized implementation, $\epsilon_1 = 0$, i.e., bid values can be computed exactly, and for a given $\epsilon_2$, we can determine $J$, the number of rounds the dB algorithm[3] runs for, and returns an $\epsilon_2$-optimal matching.

If a packetized implementation is not possible, we can give a physical implementation. Our only assumption here is going to be that each user can observe a channel, and determine if there was a successful transmission on it, a collision, or no transmission, in a given time slot. The whole negotiation phase is again divided into $J$ sub-frames. In each sub-frame, each user transmits by turn. It simply transmits $\lceil \log M \rceil$ bits to indicate a channel number, and then $\lceil \log 1/\epsilon_1 \rceil$ bits to indicate its bid value to precision $\epsilon_1$. The number of such sub-frames $J$ is again chosen so that the dB algorithm (based on Algorithm 3) returns an $\epsilon_2$-optimal matching.

![Fig. 1. Figure showing structure of decision frame.](image)

### VI. Simulations

In this section, we present simulation results to evaluate the performance of the dUCB4 algorithm. In simulations, all channel rewards were assumed to be i.i.d. and generated from Bernoulli distributions with means equal to the mean channel rewards, $\mu_{i,j}$. Figure 2 shows the the cumulative regret when the number of players $M = 3$ and the number of arms $N = 3$. The horizontal axis is the number of slots and the vertical axis is the accumulated regret.

### VII. Conclusions

We have considered the problem of decentralized multi-armed bandits with communication/computation cost. We have proposed an algorithm, dUCB4, which achieves a regret of $O(\log^2 T)$. Finding a lower bound is a work in progress.

![Fig. 2. Cumulative regret : $M = 3$, $N = 3$. Mean reward matrix = $[0.9, 0.7, 0.3; 0.6, 0.8, 0.1; 0.8, 0.6, 0.1]$](image)

### References