Poster: Multi-agent Combinatorial Bandits with Moving Arms

Zhiming Huang, Bingshan Hu, Jianping Pan
Department of Computer Science, University of Victoria, Victoria, BC V8P 5C2, Canada

***Abstract*—In this paper, we study a distributed stochastic multi-armed bandit problem that can address many real-world problems such as task assignment for multiple crowdsourcing platforms, traffic scheduling in wireless networks with multiple access points and caching at cellular network edge. We propose an efficient algorithm called *multi-agent combinatorial upper confidence bound (MACUCB)* with provable performance guarantees and low communication overhead. Furthermore, we perform extensive experiments to show the effectiveness of the proposed algorithm.**

***Index Terms*—Online learning, multi-armed bandits, multi-agent combinatorial bandits, upper confidence bound**

I. Introduction

We study an innovative distributed multi-armed bandit problem called *multi-agent combinatorial bandits with moving arms (MACMAB-M)* involving an agent set $\mathbb{K} := \{1, 2, \ldots, K\}$ and an arm set $\mathbb{N} := \{1, 2, \ldots, N\}$. Given a time horizon of T rounds, each arm can randomly move across each agent over the time, resulting in the variance of the available arms $A_{t,k} \subseteq \mathbb{N}$ to agent $k \in \mathbb{K}$ in each round t, and, apparently, we have $\bigcup_{k \in \mathbb{K}} A_{t,k} = \mathbb{N}$. Each agent k needs to play a *solution* $S_{t,k} \subseteq A_{t,k}$ (i.e., a combination of arms subject to certain constraints defined by the real-world problems) and observes the random reward of each played arm $\theta_t(n), \forall n \in S_{t,k}$ in each round t. For each arm n, $\theta_t(n)$ is drawn from a fixed but unknown distribution, and is *independent and identically distributed (i.i.d)* over the time. The objective of each agent is to accumulate as many rewards as possible. As the reward distribution for each arm is unknown a priori, each agent faces a dilemma between playing the empirically best arms to achieve higher rewards (i.e., exploitation) and playing other arms to explore more about their underlying reward distribution (i.e., exploration).

Many real-world multi-agent problems can be formulated as MACMAB-M. For example, in the problem of task assignment for multiple crowdsourcing platforms such as Amazon Mechanical Turk, Amazon Flex and Testlio, the tasks are usually large projects, which can be divided into multiple subtasks for different workers. Thus, each platform can be regarded as an agent that needs to assign an incoming task to a group of workers with unknown skill levels. The rewards are based on the quality of the completed task, which is a random variable depending on the skill level of workers as also assumed in [1]. The workers can move across the platforms and thus the available workers vary for each platform (i.e., the moving arm setting). Another example is the traffic scheduling problem in wireless networks with K *access points (APs)* and N clients moving across APs. In each scheduling cycle, a client can only connect to one AP, and each AP can schedule a subset of connected clients to transmit multiple packets simultaneously by using the multiplexing technologies. A delivered packet generates a reward that is dependent on the value of information (e.g., age of information) the packet contains.

Our contributions are summarized as follows. First, to the best of our knowledge, this is the first work that studies the MACMAB-M problem, which integrates important factors such as multiple agents, combinatorial arms, and moving arms based on a bandit model. Second, we design an efficient multi-agent learning algorithm called *multi-agent combinatorial upper confidence bound (MACUCB)* with provable performance guarantees and low communication overhead. Furthermore, we conduct extensive experiments to show the effectiveness and efficiency of the proposed algorithm.

II. The MACUCB Algorithm

We adopt the idea of *upper confidence bound (UCB)* [2] as the basis of the proposed algorithm, which is to play a solution with the highest UCB estimates. The UCB estimate by agent k for arm n in round t is defined as follows:

$$U_{t,k}(n) = \hat{\theta}_{t-1}(n) + \sqrt{\frac{1.5 \ln(t-1)}{O_{t-1}(n)}}, \forall n \in A_{t,k}. \quad (1)$$

where $\hat{\theta}_t(n)$ is the empirical mean (or sample-mean) reward and $O_t(n)$ is the total number of times that arm n has been played, by the end of round t. As we can see, when an arm is not played for many times, its UCB estimate will become large so the algorithm will play this arm to explore more about its true mean reward. When am arm is played sufficiently, it can be proved that its UCB estimate is very close to its true mean reward. Thus, UCB can help us balance the tradeoff between exploration and exploitation.

On the other hand, to reduce the communication overhead, we can just transmit $\hat{\theta}_{t-1}(n)$ and $O_{t-1}(n)$ for a moving arm n instead of all its past playing history. The information can be transmitted through pairwise communication between agents or can be directly carried by the arm itself (e.g., a worker can carry her curriculum vitae in the aforementioned crowdsourcing application). Thus, the communication overhead is at most $O(N)$ in each round for at most N arms moving.

The MACUCB algorithm is described as follows. In each round t, each agent k first observes the set of available arms

$A_{t,k}$, and calculates the UCB estimate defined in (1) for each arm $n \in A_{t,k}$. Each agent then plays a solution $S_{t,k}$ with the highest sum of the UCB estimates. After observing the reward for each played arm, each agent updates $\hat{\theta}_t(n)$ and $O_t(n)$ for all arms in $A_{t,k}$. By the end of round t, each agent can transmit the information about the moving arms by pairwise communication (or the information is carried by each moving arm itself).

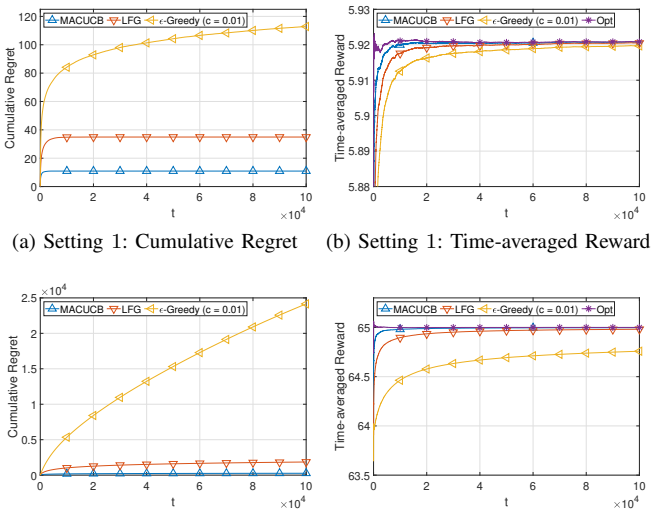
It can be proved that the regret of MACUCB (i.e., the performance gap with regard to the optimal algorithm that can only be obtained when the true mean reward of each arm is known a priori) is $O\left(\frac{NM^{\frac{4}{3}}\log T}{\Delta_{\min}}\right)$, where $\Delta_{\min}$ is the gap between the sum of expected rewards for the optimal and best suboptimal solutions, and M is the maximum number of arms in a solution played in each round.

III. Evaluations

We have conducted evaluations in MATLAB to compare the proposed MACUCB algorithm with the ϵ-Greedy algorithm [2] which is an oft-used algorithm for reinforcement learning, and the LFG algorithm [1] which can be regarded as the MACUCB algorithm without communication between agents. The basic idea of ϵ-Greedy is to randomly play a solution with probability ϵ (i.e., exploration), and play the currently known best solution with probability $1-\epsilon$ (i.e., exploitation), where $\epsilon := \min\left\{1, \frac{cN}{t}\right\}$ decreases with time. Note that c is a input parameter determining the initial exploration rate. We perform ϵ-Greedy with a full information setting, i.e., each agent will broadcast the information about the played arms in each round, and let $c = 0.01$ as we found with which ϵ-Greedy performs best in our evaluations.

The evaluations are performed on two bandit settings. In Setting 1, the total number of agents is $K = 3$, the total number of arms is $N = 10$ and the maximum number of arms can be played in each round is $M = 3$. The rewards of arms in each round are i.i.d. drawn from Bernoulli distributions with mean rewards $0.95, 0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6, 0.55, 0.5$. Setting 2 is more complex. We have $K = 10$, $N = 100$, and $M = 10$. The reward of each arm also follows a Bernoulli distribution with a randomly generated mean value. The reward distributions for both settings are not known a priori to the three compared algorithms (and we denote by Opt the optimal algorithm that knows the reward distributions a priori). The results are the average of 100 independent experiments. We do not plot the error bars as they are too small to be observed clearly.

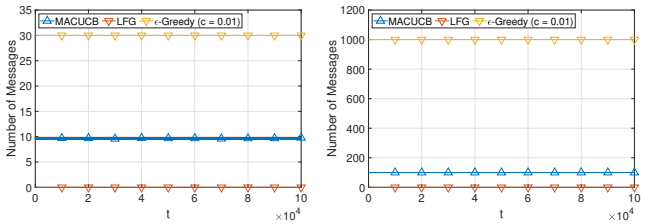
a) Cumulative regret and time-averaged reward: We can see that in both settings, the cumulative regret of MACUCB is the lowest compared to that of LFG and ϵ-Greedy ($c = 0.01$), as shown in Figs. 1a and 1c. Regarding the time-averaged reward, all considered algorithms can converge to the optimal solution, but MACUCB is the quickest for both settings, as shown in Fig. 1b and 1d. Note that in Fig. 1c, MACUCB still has a learning curve (i.e., the cumulative regret increases at the beginning) but is too small to be observed clearly when comparing with other algorithms.



(a) Setting 1: Cumulative Regret (b) Setting 1: Time-averaged Reward

(c) Setting 2: Cumulative Regret (d) Setting 2: Time-averaged Reward

Fig. 1: The evaluation results for both settings.



(a) Setting 1 (b) Setting 2

Fig. 2: The communication overhead for the algorithms.

b) Communication overhead and time complexity: The results of communication overhead for both settings are shown in Fig. 2, where LFG has 0 messages as it does not communicate, and we can see that MACUCB is efficient in terms of the number of messages, with 10 and 100 messages per round in Settings 1 and 2, respectively, which corresponds to the analyzed communication overhead $O(N)$.

TABLE I: The time consumption for each agent over T rounds

Algorithm	MACUCB	LFG	ϵ-Greedy
Setting 1	0.344 s	0.345 s	0.335 s
Setting 2	3.934 s	3.773 s	3.779 s

Although MACUCB needs to spend a bit more time to process incoming and outgoing arms, all the algorithms have the similar performance for both settings, as it can be proved that their time complexity is $O(N \log N)$ for each agent.

References

[1] F. Li, J. Liu, and B. Ji, "Combinatorial Sleeping Bandits with Fairness Constraints," *IEEE Transactions on Network Science and Engineering (TNSE)*, 2019.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multi-armed Bandit Problem," *Machine Learning*, vol. 47, no. 2–3, pp. 235–256, 2002.