Matching in Multi-arm Bandit with Collision

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper, we consider the matching of multi-agent multi-armed bandit problem, i.e., while agents prefer arms with higher expected reward, arms also have preferences on agents. In such case, agents pulling the same arm may encounter collisions, which leads to a reward of zero. For this problem, we design a specific communication protocol which uses deliberate collision to transmit information among agents, and propose a layer-based algorithm that helps establish optimal stable matching between agents and arms. With this subtle communication protocol, our algorithm achieves a state-of-the-art $O(\log T)$ regret in the decentralized matching market, and outperforms existing baselines in experimental results.

1 Introduction

Decentralized matching between two sides with preferences (e.g., supply and demand) is the core process in many online marketplaces, such as passengers and riders on ridesharing platforms (e.g., Uber), or customers and free-lancers in online labor markets (e.g., TaskRabbit), etc. Since in practical scenarios, participants in the online market obtain market information mainly through their own experiences, a crucial question arises on *how to achieve a (possibly optimal) stable matching among these competing agents, through learning from the iterative interactions with the other side, and, at what cost?*

Learning the uncertainty in two-sided markets has been studied from various angles (e.g., [1, 7]). Following a strand of recent literature (e.g., [8, 2]), we model the problem as a multi-agent multi-arm bandit problem. Specifically, in the decentralized matching market with multi-armed bandits, we consider a set $\mathcal{M}$ of M agents, and a set $\mathcal{A}$ of K arms, with $M \leq K$. Every agent knows M, K, and the time horizon T. Each arm has its fixed preference towards agents, which is unknown to all the agents. The notation $i \succ_k j$ means that arm k prefers agent i rather than agent j.

At each time t, every agent j pulls an arm $I_j(t) \in \mathcal{M}$. When multiple agents pull the same arm k at time t, a collision occurs on this arm. Only the agent that the arm prefers most, e.g., j, will win the competition. The winning agent does not encounter collision, and obtains a stochastic reward $R_j(t)$ from the arm, which is sampled from a unknown fixed distribution F_{jk} with mean u_{jk}. In the meantime, the other agents i that pull arm k will encounter collision and receive zero reward, i.e., $R_i(t) = 0$. Without loss of generality, we assume that the minimal gap of utility between two arms among all agents $\Delta \triangleq \min_{m\in\mathcal{M}} \min_{k,n\in\mathcal{A},k\neq n} |u_{mk} - u_{mn}|$ is positive. We let $C_j(t)$ represent the collision indicator for agent j at time t, i.e., $C_j(t) = 1$ denotes that agent j encounters collision at time step t and $C_j(t) = 0$ otherwise. In our model, each agent j is informed of his own collision signal $C_j(t)$ and reward $R_j(t)$ after his pull. At each time step t, every agent j pulls an arm $I_j(t)$ only according to his observed history $\{(I_j(\tau), R_j(\tau), C_j(\tau))\}_{\tau=1}^{t-1}$.

Now we introduce the optimal stable matching to give a suitable notion of regret ([6]). While arms have preferences towards agents, an agent j always prefers the arm with higher utility, i.e., the arm with higher u_{jk}. Thus, at a stable matching, within any two "agent, arm" pairs, we cannot rematch

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Table 1. Comparison between our work and prior results.

Algorithms	Model	Regret
UCB-D3 ([12])	globally ranked agent	$O(M(M-1)K\log T/\Delta^2)$
UCB-D4 ([2])	uniqueness consistency	$O(M(M-1)K\log T/\Delta^2)$
CA-UCB ([9])	partially decentralized	$O(\exp(M^4)M\log^2 T/\Delta^2)$ (pessimal regret)
Phased-ETC ([2])	general model	$O(MK\log^{1+\epsilon} T+\exp(1/\Delta^2))$
ML-ETC (our result)	general model	$O(MK\log T/\Delta^2)$

one agent with the other arm such that both are more satisfied. If there exists at least one stable matching where agent j is matched with arm k, we call arm k is an attainable arm to agent j. We denote the set of attainable arms of agent j by $\mathcal{A}_{stable}(j)$. The optimal stable arm for agent j is $a_j^* = \mathrm{argmax}_{k\in\mathcal{A}_{stable}(j)} u_{jk}$, i.e., the arm with highest mean to agent j in $\mathcal{A}_{stable}(j)$. The optimal regret is defined as the expected reward gap between the ideal reward according to the optimal stable matching and the achieved reward, i.e.,

$$Reg(T) = T\sum_{j=1}^{M} u_{ja_j^*} - \mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{M}(1-C_j(t))u_{jI_j(t)}\right].$$

1.1 Our Main Contribution

Following the explore-then-commit (ETC) framework [10, 8, 11], we propose an algorithm called ML-ETC. Specifically, all the agents will first explore arms and then commit to their optimal stable arms in terms of empirical means.

The key point in ML-ETC is that we allow agents to communicate through deliberately designed collisions, so that they can switch from exploration to exploitation timely. Previous works in multi-agent multi-arm bandit (e.g., [4, 14, 10, 13, 3]) have discussed the use of deliberate collisions among agents for communication, but they fail in the case of matching market with multi-armed bandit. This is because when arms have preferences on agents, it may be impossible for every agent to communicate with each other. For example, consider an extreme case where agent j is ranked first on all arms. In this case, agent j will never encounter collision, let alone receiving information from others through deliberate collisions. To solve this problem, we show that under such circumstance, the agent who cannot transmit information to agent j will never be able to influence the stable matching of agent j, which means that there is nearly no difference whether they exist from agent j's point of view. Therefore, an agent could end the exploration once all the agents who can communicate with him have high confidence of their preference about arms. After that, we can apply GS ([6]) to make sure that each agent obtains his optimal stable arm within limited time steps and then commits to that optimal stable arm.

Based on these properties, we show that our subtle multi-layered algorithm ML-ETC achieves a state-of-the-art regret upper bound of $O(\log T)$ in the decentralized setting, which is better than the existing results, including the $O(\log^{1+\epsilon} T)$ regret bound for Phased-ETC [2] and the $O(\log^2 T)$ regret bound for CA-UCB in [9]. Please see Table 1 for the summary of comparison with previous results, and Section 1.2 for detailed discussions.

1.2 Related Work

The study of matching in multi-armed bandit is initiated by the recent work [8]. In this work, the authors introduce the model and design both a centralized ETC policy and a decentralized ETC policy. The centralized ETC has good performance but requires a central organizer to proceed matching, which may not exist in many real-world applications. The decentralized ETC, on the other hand, may cause very high regret when the minimal gap Δ is unknown to the agents, which is also a common case in reality.

A strand of literature studies the special case where the arms' preferences satisfy some special conditions. For example, the work of [12] assumes that the agents are globally ranked, which means all arms have the same preference toward agents, and proposes the UCB-D3 algorithm. Also, [2]

designs the UCB-D4 algorithm based on the assumption of uniqueness consistency on agents and arms. The assumption of uniqueness consistency requires that there exists a unique stable matching for all agents and arms, and the matching remains to be the unique stable matching when arbitrary matched pairs leave the system. In the meantime, [5] analyzes the stability, and fairness in the centralized matching market with bandit learners by introducing costs and transfers in agents' reward.

The other strand of literature focuses on the general case of unrestricted arm preferences. In [9], the authors design the CA-UCB algorithm, based on extra information that all agents can observe the winning agent for each arm at the end of each round. CA-UCB achieves an $O(\log^2 T)$ pessimal regret (a less strict notion of regret). Within this strand of work, the work [2] is the most related to our work, which also analyzes the optimal regret in the general market. Beyond the aforementioned UCB-D4, the authors also design a phased based algorithm Phased-ETC (for general arm preference) and achieves the regret bound of $O(\log^{1+\epsilon} T)$, with $\epsilon > 0$. However, in their algorithm, agents have no communication at all. Therefore, agents keeps exploring all the arms and have no idea of others' estimations of the arms. In addition to the $O(\log^{1+\epsilon} T)$ factor, Phased-ETC also has a huge regret term (an exponential term of $\exp(\frac{1}{\Delta^2})$) because of the possibility of the exploration failure, and this will be explained in detail in our experimental section. Compared to [2], we design a subtle communication protocol to allow agents to end the exploration timely while they have high confidence of the arms' order according to the empirical means. This allows us to obtain a state-of-the-art $O(\log T)$ regret upper bound without any potential exponential terms.

2 The ML-ETC Algorithm

In this section, we introduce our Multi-Layer Explore Then Commit (ML-ETC) algorithm, which is described in detail in Algorithm 1 and achieves an asymptotic regret of $O(\log T)$.

2.1 Brief Introduction

The algorithm can be briefly divided into 3 phases: the *initialization* phase, the *multi-layered exploration and communication* phase, and the *exploitation* phase.

In the initialization phase, all agents will be assigned an index and receive some information of the arm preferences. After the initialization phase, agents form an underlying hierarchical structure with respect to their communication capability through collision. Then, the algorithm proceeds to the multi-layered exploration and communication phase to match agents of each layer with arms. At each layer, the algorithm keeps iterating between exploration and communication. During exploration, every agent will explore the arms orthogonally (according to his index in order to avoid collision) and will estimate the expected rewards of all the remaining arms. During communication, every agent checks whether his estimations are accurate enough, and transmit this information to all the agents that can hear from him. Only when some of the agents are able to successfully find their empirical optimal stable arms, will they leave the multi-layered exploration and communication phase, and start their exploitation phase. The remaining agents will enter the next layer, until all the agents find out their empirical optimal stable arms. In the exploitation phase, every agent continuously pulls his empirical optimal stable arm.

2.2 Communication Protocol

In this subsection, we will introduce the core of ML-ETC, the communication protocol used in both the initialization phase and the multi-layered exploration and communication phase.

Suppose agent i (i.e., the transmitter) wants to convey a binary sequence s to agent j (i.e., the receiver). Let $|s|$ denote the length of the binary sequence, i.e., $s \in \{0,1\}^{|s|}$. We design a procedure $com(i,j,k,s)$ to allow such communication, where agent i and agent j will create deliberate collisions on arm k to deliver the bits. The communication procedure will last for $|s|$ time steps, during which other agents are serving as bystanders. We specify the actions of all agents in $com(i,j,k,s)$ as follows.

- Transmitter i: During the procedure $com(i,j,k,s)$, at the ℓ-th time step, if $s(\ell)$ (the ℓ-th digit in s) is 1, then the transmitter i will pull arm k. Otherwise, he pulls an arbitrary arm $k' \neq k$ at the time step.

- Receiver j: During $com(i, j, k, s)$, the receiver j deliberately pulls arm k for all $|s|$ time steps to receive information, and the resulting collision indicators $C_j(t)$ during these time steps are the digits of the binary sequence s.
- Bystanders: During $com(i, j, k, s)$, all agent $m \neq i, j$ are bystanders. Each bystander chooses an arbitrary arm $k' \neq k$ to pull for each time step within all $|s|$ time steps of the procedure, to avoid interfering the bit transmission between agent i and j on arm k.

In this paper, agents may transmit three types of information: the index of an agent, the index of an arm, a binary signal of whether the exploration is success or not. It is easy to check that the communications of these three types of information are with length $\lceil \log M \rceil$, $\lceil \log K \rceil$ and 1.

However, $com(i, j, k, s)$ does not always succeed. Only if $i \succ_k j$, i.e., when agent i wins against j during the competition on arm k, the collision indicators $C_j(t)$'s form the sequence s. Otherwise, $C_j(t)$ always equals to 0 and therefore $com(i, j, k, s)$ is not effective. Based on this observation, we define the communication graph as follows.

Definition 1. *A communication graph $G(\mathcal{M}, \mathcal{A})$, where $\mathcal{M}$ is the set of agents and $\mathcal{A}$ is the set of arms, is a directed graph with each agent in $\mathcal{M}$ as one vertex. There is a directed edge from vertex i to vertex j in G if and only if there exists at least one arm $k \in \mathcal{A}$ such that $i \succ_k j$.*

Based on Definition 1, if there exists at least one edge from i to j in graph $G(\mathcal{M}, \mathcal{A})$, it means that agent i can transmit information to agent j effectively (through the proper arm). The communication graph captures the communication availability among agents. We will further categorize agents with similar communication "capability" as follow.

Definition 2. *An outer closed circle is a non-empty subset of vertexes in a directed graph G, i.e., $M \subseteq V(G)$, which satisfies that: i) M is connected; ii) there is no entry edge from vertices in $V(G) \setminus M$ to any vertex in M.*

From Definition 2, we can easily see that the agents in an outer closed circle of the communication graph $G(\mathcal{M}, \mathcal{A})$ can exchange (transmit and receive) information among each other. However, agents in the outer closed circle can only transmit information to agents outside, but are unable to receive information from them. The following proposition further shows that there always exists exactly one outer closed circle in a communication graph $G(\mathcal{M}, \mathcal{A})$.

Proposition 1. *If there exists at least one edge between any two vertices in directed graph G, there exists and only exists one outer closed circle in G.*

Due to space limit, the proof of Proposition 1 is deferred to Appendix in the supplementary material.

Remark: Multi-layer separation of agents. The result of Proposition 1 is critical. With this result, we can uniquely determine a group of agents with bidirectional communication capability within the group, but unidirectional transmission to the outside. Thus, by iteratively extracting outer closed circle from the agent set $\mathcal{A}$, we are naturally constructing a hierarchical structure within agents. In such structure, agents can transmit information only to agents of the same level and below, while receiving information from agents of the same level and higher (see Section 2.4 for detailed discussions).

2.3 Initialization Phase

The initialization phase (lines 1-4 in Algorithm 1) consists of two procedures: $index_assignment$ and $information_access$. Through procedure $index_assignment$, each agent is assigned an index, and obtains some information about each arm's preferences. Then, the procedure $information_access$ allows agents to share their obtained preference rankings on all arms.

The procedure $index_assignment$ (see details in Appendix) consists of K rounds with M time steps each, lasting a total MK time steps. In round 1, every agent starts with pulling arm 1. If an agent does not encounter collision at time step $t = j_1$ ($j_1 \in [1, M]$), the agent is assigned with index j_1 and starts to pull arm 2. This means that the index of an agent is his ranking on arm 1. Similarly, in round $k = 2, 3, ..., K$, if an agent does not encounter collision on arm k at time step $t = (k-1)M + j_k$, he knows that his ranking on arm k is j_k, and starts to keep pulling arm $k + 1 \mod K$, so that the other agents can continue to confirm their rankings on arm k. In this way, after MK time steps, every agent knows his ranking on all the arms.

Algorithm 1 ML-ETC Algorithm

Require: $K, M, T, p = 1$

1: $\triangleleft$ Initialization phase:
2: $(j, s_j) \leftarrow index_assignment(K, M)$ #assign index j to "this agent"
3: $(s_j, \mathcal{M}_p) \leftarrow information_access(K, M, j, s_j)$
4: #s_j is the arm preference known to agent j, $\mathcal{M}_p$ is the set of layer leaders in layer p
5: $\triangleleft$ Multi-layered exploration and communication phase:
6: $E(j) \leftarrow 0$ # $E(j)$ denotes whether agent j enter the exploitation phase
7: **while** $E(j) == 0$ **do**
8: $\quad\triangleleft$ Exploration:
9: $\quad$Let r be the order of agent j in remaining agent set $\mathcal{R}_p$, $i \leftarrow r$
10: $\quad$**for** $|\mathcal{A}_p|\lceil \log(T) \rceil$ time steps **do** #$\mathcal{A}_p$ is the set of remaining arms in layer p
11: $\quad\quad\pi \leftarrow \mathcal{A}_p[i]$
12: $\quad\quad i \leftarrow i + 1 \mod |\mathcal{A}_p|$
13: $\quad\quad$Pull arm π, update the empirical means $\hat{\boldsymbol{u}}(j) = \{\hat{u}_{j1}(t), \hat{u}_{j2}(t), ...\}$ and matching
14: $\quad\quad$times $\boldsymbol{N}(j) = \{N_{j1}(t), N_{j2}(t), ...\}$
15: $\quad$**end for**
16: $\quad\triangleleft$ Communication:
17: $\quad$**if** $j \in \mathcal{M}_p$ **then** #Agent j is a layer leader
18: $\quad\quad whether_success \leftarrow whether_succ(\hat{\boldsymbol{u}}(j), N(j), \mathcal{A}_p)$
19: $\quad\quad$#whether j achieves an accurate estimation for arms in $\mathcal{A}_p$
20: $\quad\quad(opt_arm, E(j)) \leftarrow layer_leader(j, s_j, \mathcal{A}_p, \mathcal{R}_p, \mathcal{M}_p, whether_success, M)$
21: $\quad\quad$#if all the layer leaders succeed, then $E(j) \leftarrow 1$ and j will enter the exploitation phase
22: $\quad\quad$#opt_arm is the empirical optimal stable arm of agent j
23: $\quad$**end if**
24: $\quad$**if** $j \notin \mathcal{M}_p$ **then** #Agent j is a layer follower
25: $\quad\quad(p, \mathcal{A}_p, \mathcal{M}_p, \mathcal{R}_p) \leftarrow layer_follower(j, s_j, \mathcal{A}_p, \mathcal{R}_p, \mathcal{M}_p, M, p)$
26: $\quad\quad$#if all the layer leaders succeed, then $p \leftarrow p + 1$ and agent j will enter the next layer
27: $\quad$**end if**
28: **end while**
29: $\triangleleft$ Exploitation phase:
30: Pull opt_arm until T

Then in the procedure of $information_access$ (given in Appendix), there will be M rounds of information exchange. In each round, we list all the "agent, agent, arm" tuples, i.e., $\{(i, j, k)\}, \forall i, j \in \mathcal{M}, k \in \mathcal{A}$, and conduct communication procedure $com(i, j, k, s_i)$ on all these tuples one by one according any publicly designated order, no matter whether they are effective or not. Here s_i represents agent i's updated knowledge about all rankings, e.g., $s_i \in \{0, 1, ..., M\}^{K \times M}$, where the matrix element $s_i[n][m]$ is the index of the agent that has the m-th ranking on arm n. If the index is unknown to agent i, $s_i[n][m]$ is 0. Each matrix element $s_i[n][m]$ is encoded in a binary sequence with length $\lceil \log M \rceil$ for transmission.

After the initialization phase, every agent i will receive an index and get part of the information about agents' rankings on each arm. Specifically, agent i receives information of the rankings of all the agents j such that there exists a path from j to i in $G(\mathcal{M}, \mathcal{A})$, i.e., agents of the same level as i or higher.

2.4 Multi-layered exploration and communication phase

The multi-layered exploration and communication phase (lines 5-28 in Algorithm 1) is driven by the layer categorization of agents. Recall Proposition 1, we can first identify a unique subset of agents from $\mathcal{M}$, called an outer closed circle, in which agents can communicate with each other. At this stage, we call the system is at "layer 1". Agents in the outer closed circle are called layer leaders of layer 1, the set of which is denoted as $\mathcal{M}_1$. Our algorithm will finish this stage by assigning each agent in $\mathcal{M}_1$ his empirical optimal stable matching arm. Thus, agents in $\mathcal{M}_1$ will leave this stage and directly jump to the exploitation phase. Those remaining agents and arms will enter the stage of layer 2 and start the identification of a new outer closed circle. The above process will keep repeating until all agents enter the exploitation phase.

In general, for each layer $p = 1, 2, 3, ...$ the system is in, we let $\mathcal{R}_p$ denote the set of all remaining agents, and $\mathcal{A}_p$ denote the set of all remaining arms (we have $\mathcal{R}_1 = \mathcal{M}$ and $\mathcal{A}_1 = \mathcal{A}$). Then according to Proposition 1, there exist a group of agents $\mathcal{M}_p$, which are the outer closed circle of the communication graph $G(\mathcal{R}_p, \mathcal{A}_p)$. We call the agents in $\mathcal{M}_p$ layer leaders in this layer, while the remaining agents in $\mathcal{R}_p \setminus \mathcal{M}_p$ are called layer followers. Based on the properties of the outer closed circle, all the layer leaders can i) exchange information among each other, and ii) transmit information to all layer followers but not be able to receive information from them (via the remaining arms in $\mathcal{A}_p$). Then, the goal of this layer is to let all the layer leaders find out their empirical optimal stable arm. The reason that we only care about layer leaders is because the communication from layer followers to layer leaders are not effective, and layer followers can not influence layer leaders (including their pull decisions, results and optimal stable arms).

To achieve this goal, at any layer, all remaining agents will conduct exploration and then communication. They will first conduct exploration for $|\mathcal{A}_p|\lceil \log T \rceil$ time steps, and then communicate with each other to check whether all the layer leaders have obtained an accurate estimation on the remaining arms. Specifically, for a layer leader i, if for any two remaining arms m, n, either the lower confidence bound of u_{jn} is larger than the upper confidence bound of u_{jm} or the lower confidence bound of u_{jm} is larger than the upper confidence bound of u_{jn} (i.e., with high probability, the order of empirical means is the same as the order of real means), then he can confidently differentiate these two arms. Thus, the agent has successfully obtained an estimation (or, a sort) on all arms that is accurate enough for him to enter the exploitation phase. In the communication, we let layer leaders reach a consensus on a global signal of whether all layer leaders have achieved a successful estimation on every remaining arm, and transmit this signal to all layer followers, while layer followers do not transmit anything. Since all the layer leaders can hear from each other, after the communication, they know the updated status of all layer leaders. If all layer leaders successfully sort the arms with confidence, they start to find out their empirical optimal stable-matching arms, and then jump to the exploitation phase. Otherwise, all agents will go back to another round of exploration, so that the agents with unsuccessful signal of exploration have the chance to achieve an accurate estimation successfully before the next communication. When all the layer leaders jump to the exploitation phase, all the remaining agents (the layer followers in this layer) start to enter the next layer and learn the remaining arm set.

Note that at the beginning of layer p, a remaining agent i is able to identify all layer leaders and layer followers by his own, based on the knowledge of $\mathcal{R}_p$, $\mathcal{A}_p$ and his available information about the rankings from the initialization phase (See Appendix for detailed proof). Therefore, the above procedure can be done in a distributed manner.

Next, we further specify the exploration process and the communication process, respectively.

2.4.1 Exploration

Within one round of exploration at layer p that lasts for $|\mathcal{A}_p|\lceil \log T \rceil$ time steps, every agent in $\mathcal{R}_p$ will explore every remaining arm in $\mathcal{A}_p$ orthogonally according to their index to avoid collision. Specifically, at the τ-th time step in this round of exploration, the j-th agent in $\mathcal{R}_p$ will pull the $(j + \tau \mod |\mathcal{A}_p|)$-th arm in $\mathcal{A}_p$. Since $|\mathcal{A}_p| \geq |\mathcal{R}_p|$, every agent will pull different arms through the exploration and avoid collision with each other.

The empirical mean of arm k estimated by agent j during explorations after time step t is denoted by $\hat{u}_{jk}(t)$, the number of time steps that arm m is pulled by agent j in explorations at the end of time step t is denoted by $N_{jm}(t)$. We say agent j achieves an accurate estimation for the remaining arms in $\mathcal{A}_p$ successfully at time step t (i.e., he will transmit a success signal of exploration) only if for any pair of arms (m, n) in $\mathcal{A}_p$, either $\hat{u}_{jm}(t) - \sqrt{\frac{2\log(T)}{N_{jm}(t)}} > \hat{u}_{jn}(t) + \sqrt{\frac{2\log(T)}{N_{jn}(t)}}$ or $\hat{u}_{jn}(t) - \sqrt{\frac{2\log(T)}{N_{jn}(t)}} > \hat{u}_{jm}(t) + \sqrt{\frac{2\log(T)}{N_{jm}(t)}}$ is satisfied.

2.4.2 Communication

Layer leaders. For layer leaders at layer p, their communication procedure is described in Algorithm 2. From lines 1-12, they exchange their information about whether all the layer leaders achieve an accurate estimation successfully and transmit this information to the layer followers at the mean time.

Algorithm 2 layer_leader

Require: $j, s_j, \mathcal{A}_p, \mathcal{R}_p, \mathcal{M}_p, whether_success, M$
Ensure: $opt_arm, E(j)$
$opt_arm \leftarrow 1,\ E(j) \leftarrow 0,\ all_leader_success \leftarrow whether_success$
for $m = 1, 2, ..., |\mathcal{M}_p|, i = 1, 2, ..., |\mathcal{A}_p|, i_1 = 1, 2, ..., M, i_2 = i_1 + 1, ..., M$ **do**
    if $s_j[i][i_1] == j$ **then**
        If $all_leader_success == 1$, agent j pulls arm i; otherwise pulls an arbitrary arm $i' \neq i$.
        # agent j is a transmitter
    else if $s_j[i][i_2] == j$ **then**
        Agent j pulls arm i, and only if $C_j(t) == 0$, he sets $all_leader_success = 0$.
        # agent j is a receiver
    else
        Agent j pulls an arbitrary arm $i' \neq i$. # agent j is a bystander
    end if
end for
if $all_leader_success == 1$ **then**
    $opt_arm \leftarrow GS_and_arm_information(j, \hat{\boldsymbol{u}}(j), \mathcal{A}_p)$
    $E(j) \leftarrow 1$
end if
return $opt_arm, E(j)$

If all layer leaders achieve an accurate estimation successfully, then they continue to execute $GS_and_arm_information$ and enter the exploitation phase. Otherwise, they return to exploration again according to ML-ETC. The $GS_and_arm_information$ procedure in Algorithm 2 lasts for $|\mathcal{M}_p|^2 + (|\mathcal{R}_p| - |\mathcal{M}_p|)|\mathcal{A}_p|$ time steps. In the first $|\mathcal{M}_p|^2$ time steps, layer leaders follow the standard Gale–Shapley algorithm [6], where each agent's preference towards arms is in the descending order of the empirical means. Thus, each layer leader proposes to the best unrejected arm iteratively (i.e., the empirical best arm on which agent hasn't encounter a collision yet). This makes sure that all the agents obtain their empirical optimal stable arms. In the rest $(|\mathcal{R}_p| - |\mathcal{M}_p|)|\mathcal{A}_p|$ time steps, they commit to their empirical optimal stable arms so that the layer followers have enough time to receive the information about these "leaving arms" (which is used to construct the remaining arm set in the next layer).

Layer followers. For a layer follower in layer p, their actions in the communication is described in Algorithm 3. From lines 2-8, the agent either stands by to wait for his turn or to receive the information whether all layer leaders achieve an accurate estimation successfully (denoted by $all_leader_success$ in line 4).

If there exist some layer leaders sending the unsuccess signal of exploration, all the layer followers switch back to a new round of exploration again. Otherwise, a layer follower continues to execute lines 10-25. Specifically, layer followers first wait for $|\mathcal{M}_p|^2$ time steps by choosing an arbitrary arm to pull. Note that based on the property of layer followers, their actions will not influence the GS algorithm among layer leaders. Then for each arm in $\mathcal{R}_p$, layer followers explore whether the arm is available (i.e., the remaining arm set $\mathcal{A}_{p+1}$ in the next layer) based on the order of the their indices one by one. For an agent i, if it is not his turn to explore an arm, e.g., k, he will always choose the next arm in $\mathcal{A}_p$ to pull. Note that in i's exploration of available arms, all the other layer followers $j \neq i$ are pulling another arm in $\mathcal{A}_p$. Therefore, those arms on which the agent i gets a collision will leave at the end of this layer.

2.5 Regret Analysis

Theorem 1. *If every agent runs Algorithm ML-ETC, then the optimal regret after T time steps is upper bounded by:*

$$Reg(T) \leq MK\lceil\frac{32}{\Delta^2}\rceil\lceil\log(T)\rceil + MC_1 + MC_2 + MC_3, \tag{1}$$

where $C_1 = MK + M^4K^2\lceil\log M\rceil$, $C_2 = (\lceil\frac{32}{\Delta^2}\rceil + M)(\frac{KM^2(M-1)}{2} + M^2 + KM)$, and $C_3 = 2MK\lceil\frac{32}{\Delta^2}\rceil$.

Algorithm 3 layer_follower

Require: $j, s_j, \mathcal{A}_p, \mathcal{R}_p, \mathcal{M}_p, M, p$
Ensure: $p, \mathcal{A}_p, \mathcal{M}_p, \mathcal{R}_p$
$\mathcal{A} \leftarrow \mathcal{A}_p,\ \mathcal{M} \leftarrow \mathcal{M}_p,\ \mathcal{R} \leftarrow \mathcal{R}_p$
for $m = 1, 2, ..., |\mathcal{M}_p|, i = 1, 2, ..., |\mathcal{A}_p|, i_1 = 1, 2, ..., M, i_2 = i_1 + 1, ..., M$ **do**
    if $s_j[i][i_2] == j$ **then**
        Agent j pulls arm i, and sets $all_leader_success = C_j(t)$. # agent j is a receiver
    else
        Agent j pulls an arbitrary arm $i' \neq i$. # agent j is a bystander
    end if
end for
if $all_leader_success == 1$ **then**
    Pull an arbitrary arm for $|\mathcal{M}_p|^2$ time steps.
    # wait for the layer leaders to look for their optimal stable matching arms
    for $\pi \in \mathcal{A}_p, \tau = 1, 2..., |\mathcal{R}_p| - |\mathcal{M}_p|$ **do**
        if $\tau == f$ (f is the order of agent j in layer follower set $\mathcal{R}_p \setminus \mathcal{M}_p$) **then**
            # it is agent j's turn to explore available arms
            Agent j pulls arm π
            if $C_j(t) == 1$ **then**
                $\mathcal{A} \leftarrow \mathcal{A} \setminus \pi$
            end if
        else
            Agent j pulls an arbitrary arm $\pi' \neq \pi$
        end if
    end for
    $\mathcal{M} \leftarrow occ(j, \mathcal{A}_p, \mathcal{R}_p, s_j), \mathcal{R} \leftarrow \mathcal{R}_p \setminus \mathcal{M}_p$ #$\mathcal{M}$ denotes the set of new layer leaders
    $p \leftarrow p + 1$ #agent j will later enter the next layer
end if
return $p, \mathcal{A}, \mathcal{M}, \mathcal{R}$

In Theorem 1, the regret term MC_1 is the communication cost induced by the initialization phase; the regret term MC_2 is the communication cost in the multi-layered exploration and communication phase; and the regret term MC_3 is the cost of wrong estimation under a low-probability event. The majority term of regret is $MK\lceil \frac{32}{\Delta^2} \rceil \lceil \log(T) \rceil$, which is an $O(\log T)$ term and better than all prior works [2, 9].

Proof sketch: The regret of our algorithm ML-ETC can be divided into three parts.

The regret caused by the initialization phase is upper bounded by MC_1, since the initialization phase lasts for C_1 time steps.

To bound the regret caused by the other two phases, we let $r_0 = r_1 = \lceil \frac{32}{\Delta^2} \rceil$, $r_2 = r_0 + M$, and define the following event.

$$\mathcal{E} = \left\{ \forall j \in \mathcal{M}, r \leq r_0, m \in \mathcal{A}, |u_{jm}(t) - \hat{u}_{jm}(t)| \leq \sqrt{\frac{2\log(T)}{N_{jm}(t)}} \textit{ holds after } r \textit{ rounds of exploration} \right\}.$$

We prove that $\Pr[\mathcal{E}] \geq 1 - \frac{2r_0MK}{T}$. Because of this, we know that the expected regret in the last two phases under $\neg\mathcal{E}$ is upper bounded by a constant MC_3.

Then we consider the expected regret in these two phases under $\mathcal{E}$. In this case, the length of the multi-layer exploration and communication phase can be upper bounded by $r_1K\lceil \log T \rceil + MC_2$, since the length of one round of exploration is $|A_p|\lceil \log T \rceil \leq K\lceil \log T \rceil$, the length of one round of communication can be upper bounded by $\frac{KM^2(M-1)}{2} + M^2 + KM$ and every agent will enter the exploitation phase after at most r_1 rounds of exploration and r_2 rounds of communication conditioning on $\mathcal{E}$.

On the other hand, under $\mathcal{E}$, all the agents will get the right estimation during the $GS_and_arm_information$ procedure. Therefore, every agent j must match with his optimal

stable arm a_j^* in the exploitation phase. This means that the expected regret in the exploitation phase under $\mathcal{E}$ is 0.

Summing over the expected regret in three phases, we finally get the regret upper bound in Theorem 1. Due to space limit, the details of the proof are deferred to Appendix. □

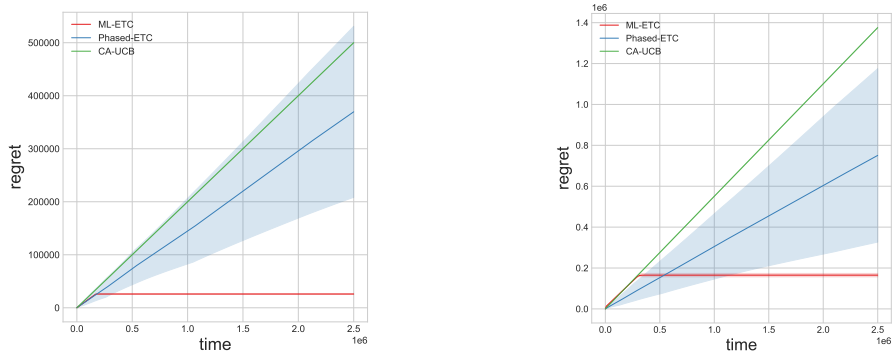
3 Simulation

Setup. We choose the time horizon to be $T = 2.5 \times 10^6$, arms' mean utilities within $[0.3, 0.6]$, and the minimal gap $\Delta = 0.05$. We have tested the following two cases: i) 3 agents and 3 arms; ii) 5 agents and 5 arms. To investigate the quality of the converging stable matching under different algorithms, we choose the arm preferences such that there exist multiple stable matches between agents and arms (see Appendix for the implementation detail).

Baseline. *Phased-ETC* [2] is a phased based decentralized algorithm which theoretically obtains the $O(\log^{1+\epsilon} T + \exp(\frac{1}{\Delta^2}))$ regret. The duration of each exploration is determined by the parameter ϵ. When ϵ is smaller, the duration of each exploration is shorter. Same as the simulation in [2], we choose $\epsilon = 0.2$ in our simulation.

CA-UCB [9] is an algorithm which aims to avoid collision combined with the UCB method. It guarantees an $O(\log^2 T)$ pessimal regret (a less strict notion of regret). Same as the simulation in [9], we choose the parameter λ of delay probability to be $\lambda = 0.1$. Note that CA-UCB requires more information (i.e., all agents know the complete arm preference and can observe the winning agent for each arm at the end of each round), thus the comparison here is not entirely fair for our ML-ETC.

Result. Figure 1 shows the average regret and the standard deviation of regret over 50 independent runs. From Figure 1, we can find that ML-ETC outperforms both Phased-ETC and CA-UCB from an asymptotic view, which shows that our proposed algorithm has lower regret in general cases, but may encounter a cold start. The cold start means that when the time horizon T is small, ML-ETC may cause a relatively high regret. This is because ML-ETC has strict requirement of agents' estimation, only when the agents have very high confidence, will they enter the exploitation. As for Phased-ETC, the duration of the exploration is determined by the parameter ϵ. The agents may start the exploitation only with limited explorations, and in this case they are not able to distinguish the arms, especially when Δ is small. Though this leads to a smaller short-term expected regret (at the beginning, Phased-ETC outperforms our ML-ETC), for large T, this can cause a very high expected regret of $O(\exp(\frac{1}{\Delta^2}))$ (in our experiments with $\Delta = 0.05$, this is even larger than T, and therefore the regret of Phased-ETC increases linearly as T increases). In CA-UCB, there is possibility that an agent j converges to pulling a sub-optimal arm $k \neq a_j^*$ in $\mathcal{A}_{stable}(j)$, which also causes a linear optimal regret.



(i) Case of 3 agents and 3 arms

(ii) Case of 5 agents and 5 arms

Fig. 1. Comparison between ML-ETC and baselines.

References

[1] Itai Ashlagi, Anilesh K Krishnaswamy, Rahul Makhijani, Daniela Saban, and Kirankumar Shiragur. Assortment planning for two-sided sequential matching markets. In *Web and Internet Economics: 16th International Conference, WINE 2020, Beijing, China, December 7–11, 2020, Proceedings*, volume 12495, page 476. Springer Nature, 2020.

[2] Soumya Basu, Karthik Abinav Sankararaman, and Abishek Sankararaman. Beyond $log^2(t)$ regret for decentralized bandits in matching markets. In *International Conference on Machine Learning*, pages 705–715. PMLR, 2021.

[3] Ilai Bistritz, Tavor Baharav, Amir Leshem, and Nicholas Bambos. My fair bandit: Distributed learning of max-min fairness with multi-player bandits. In *International Conference on Machine Learning*, pages 930–940. PMLR, 2020.

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

[5] Sarah H Cen and Devavrat Shah. Regret, stability & fairness in matching markets with bandit learners. In *International Conference on Artificial Intelligence and Statistics*, pages 8938–8968. PMLR, 2022.

[6] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[7] Ramesh Johari, Vijay Kamble, and Yash Kanoria. Matching while learning. *Operations Research*, 69(2):655–681, 2021.

[8] Lydia T Liu, Horia Mania, and Michael Jordan. Competing bandits in matching markets. In *International Conference on Artificial Intelligence and Statistics*, pages 1618–1628. PMLR, 2020.

[9] Lydia T Liu, Feng Ruan, Horia Mania, and Michael I Jordan. Bandit learning in decentralized matching markets. *Journal of Machine Learning Research*, 22(211):1–34, 2021.

[10] Abbas Mehrabian, Etienne Boursier, Emilie Kaufmann, and Vianney Perchet. A practical algorithm for multiplayer bandits when arm means vary among players. In *International Conference on Artificial Intelligence and Statistics*, pages 1211–1221. PMLR, 2020.

[11] 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.

[12] Abishek Sankararaman, Soumya Basu, and Karthik Abinav Sankararaman. Dominate or delete: Decentralized competing bandits in serial dictatorship. In *International Conference on Artificial Intelligence and Statistics*, pages 1252–1260. PMLR, 2021.

[13] Chengshuai Shi and Cong Shen. On no-sensing adversarial multi-player multi-armed bandits with collision communications. *IEEE Journal on Selected Areas in Information Theory*, 2(2):515–533, 2021.

[14] 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 3
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] In Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]