Partner-Aware Algorithms in Decentralized Cooperative Bandit Teams

Erdem Bıyık[1], Anusha Lalitha[1], Rajarshi Saha[1], Andrea Goldsmith[1,2], Dorsa Sadigh[1,3]

[1] Department of Electrical Engineering, Stanford University
[2] Department of Electrical and Computer Engineering, Princeton University
[3] Department of Computer Science, Stanford University
{ebiyik, alalitha, rajsaha, dorsa}@stanford.edu, goldsmith@princeton.edu

Abstract

When humans collaborate with each other, they often make decisions by observing others and considering the consequences that their actions may have on the entire team, instead of greedily doing what is best for just themselves. We would like our AI agents to effectively collaborate in a similar way by capturing a model of their partners. In this work, we propose and analyze a decentralized Multi-Armed Bandit (MAB) problem with coupled rewards as an abstraction of more general multi-agent collaboration. We demonstrate that naïve extensions of single-agent optimal MAB algorithms fail when applied for decentralized bandit teams. Instead, we propose a *Partner-Aware* strategy for joint sequential decision-making that extends the well-known single-agent Upper Confidence Bound algorithm. We analytically show that our proposed strategy achieves logarithmic regret, and provide extensive experiments involving human-AI and human-robot collaboration to validate our theoretical findings. Our results show that the proposed partner-aware strategy outperforms other known methods, and our human subject studies suggest humans prefer to collaborate with AI agents implementing our partner-aware strategy.

1 Introduction

One of the key characteristics of human-human interaction is people's ability to seamlessly anticipate and take complementary actions when working with others. For example, the moment our partner reaches for a box of cereal, we automatically walk to the fridge to grab milk. The success of multi-agent systems or human-AI teams usually depends on not only each agent's actions, but also how well they model other agents' policies and the interplay between them. As another example, consider a semi-autonomous car where both the actions of the driver, e.g., keeping or changing lanes, and assistive guidance, e.g., corrections that keep the car inside the lanes, determine the control of the vehicle. Here, we expect the guidance to predict the driver's intent and augment their actions to enhance safety and comfort.

Decentralized learning is particularly challenging when some agents have limited information of the outcomes of the actions taken by the team. In the car example, even though both human and assistive guidance share the same goal of safety and comfort, the guidance may not fully observe the driver's internal objective of, for instance, changing lanes to exit and get gas at a cheaper station. Although explicit communication can alleviate some of these challenges, it is often impractical or expensive: we cannot expect the guidance system to ask for and expect feedback after every decision of the driver. On the other hand, humans rely on implicit communication for coordination in many interactive settings (Breazeal et al. 2005; Che, Okamura, and Sadigh 2020; Losey et al. 2020). They generally make effective inferences by simply observing and reasoning over their partner's actions. Hence, we question if AI agents can accurately model others' policies without explicit communication to effectively coordinate and cooperate.

Previous works such as theory of mind (Simon 1995; Baker et al. 2017; Brooks and Szafir 2019; Lee, Sha, and Breazeal 2019) and opponent modeling in multi-agent learning (Foerster et al. 2018; Shih et al. 2021; Xie et al. 2020) showed the performance of human-AI and multi-agent teams may significantly increase if the agents accurately model each other's policies. However, most of these approaches require recursive belief modeling or rely on learned partner representations, which can often be complex and computationally intractable.

Our goal is to develop a simple and tractable approach for modeling partners in decentralized multi-agent teams that is guaranteed to improve performance. We specifically focus on decentralized Multi-Armed Bandit (MAB) problems, which extend the stochastic MAB, a fundamental model for sequential decision-making to explore an agent's environment efficiently. Our decentralized MAB formulation captures the essential elements of multi-agent collaborative learning. First, we model the team reward to be dictated by the actions of all agents, which is common in many realistic collaborations, e.g., the safety and comfort of a semi-autonomous vehicle depend on both the driver's and the guidance system's actions. Second, we model the heterogeneity in the information available to each agent by introducing partial observability over rewards, e.g., the vehicle does not always accurately observe whether the human is looking for the fastest route or the cheapest gas station. Hence, our formulation requires collaboration among agents to accomplish the task of learning the optimal team action while only observing each others' actions.

One might hope that naïve extensions of well-known bandit algorithms such as *Thompson Sampling* and *Upper Con-*

fidence Bound (UCB) would be sufficient for effective collaboration in these settings. However, we demonstrate these extensions fail to provide logarithmic regret. Our key insight is to leverage the simplicity of these well-known algorithms while predicting our partner's actions — *make the agent with lower observability of rewards follow the agent with the higher observability.* Specifically, we propose a computationally simple partner-aware bandit learning algorithm where the follower learns to predict its partner's actions while choosing its own action. We analytically show that this algorithm incurs regret logarithmic in time horizon.

Our main contributions are:

- We propose a computationally efficient partner-aware bandit algorithm, which anticipates the partner's action and effectively coordinates with the partner.
- We analytically prove our proposed algorithm significantly improves the team performance and provides logarithmic regret.
- Finally, we conduct extensive simulations and an in-lab collaborative robot experiment shown in Fig. 1. Our results suggest our algorithm significantly improves the team performance and is preferred by the users.

2 Related Work

Multi-Agent Multi-Armed Bandits. Existing decentralized cooperative MAB algorithms make one or more of the following assumptions: (i) agents independently interact with the same MAB (Lupu, Durand, and Precup 2019), (ii) they use sophisticated communication protocols to exchange information about rewards and the number of times actions were played (Landgren, Srivastava, and Leonard 2016; Martínez-Rubio, Kanade, and Rebeschini 2019; Sankararaman, Ganesh, and Shakkottai 2019; Shahrampour, Rakhlin, and Jadbabaie 2017; Barrett et al. 2014), (iii) when sophisticated communication is not possible, agents share their latest action and reward (Madhushani and Leonard 2019).

These assumptions are often required to simplify the analysis, but are not realistic in most human-AI interactions, e.g., collaborative transport, assembly, cooking, or autonomous driving, where (i) agents' actions influence the outcome for the whole team, (ii) they do not have explicit communication channels or might have different state, action representations that are difficult to communicate, or (iii) they have different capabilities, e.g., noisier sensors.

In our work, we do not make any of these assumptions. We advance the current literature by analyzing a more realistic model suited for collaborative human-AI interaction, where we: (i) relax the assumption of independent agents through coupled rewards, (ii) allow only implicit communication, i.e., agents can only observe each other's actions and not the rewards, (iii) relax the assumption on homogeneity of agents, i.e., some agents may receive noisier rewards. Hence, existing algorithms in multi-agent multi-armed bandits are not applicable in our setting. Our contribution is a novel and computationally simple partner-aware algorithm for decentralized collaboration, and proving it incurs logarithmic regret for any finite number of arms.

Multi-Agent Learning. Recent works have shown the importance of partner modeling in multi-agent environments (Devin and Alami 2016; Zhu, Biyik, and Sadigh 2020). Foerster et al. (2018) proposed an algorithm that improves performance in repeated prisoner's dilemma using opponent modeling. Losey et al. (2020) showed that agents can implicitly communicate through their actions. Other works have learned partner representations for effective coordination (Shih et al. 2021; Xie et al. 2020; Grover et al. 2018). However, these approaches are either not guaranteed to effectively coordinate as they heavily rely on learned representations or can lead to suboptimal solutions in multi-agent MAB, where agents need to take the optimal action more frequently over time to avoid linearly growing regrets.

Humans in Multi-Armed Bandits. Zhang and Yu (2013) compared how various algorithms match with the actions of humans playing a stochastic MAB. While our algorithm does not specifically model the partner as an agent incorporating the imperfections humans have, we observe via our user studies that it can collaborate well not only in multi-AI teams but also with human partners.

3 Problem Setting

In this section, we present a decentralized MAB formulation that captures essential aspects of multi-agent decentralized collaborative learning.

Running Example. Consider a human-robot team tasked with stacking burgers in a fast-food restaurant, where they stack the ingredients together (see Fig. 1). Suppose the human is responsible for the patty and the cheese in the burger, whereas the robot stacks tomatoes and lettuce. As many people have strong opinions about in what order these ingredients should be stacked (Burge 2017), the robot should predict the human's actions to better coordinate on stacking the burger. If the robot only has partial information about whether a guest liked a burger, it might take suboptimal actions even though the human may have already discovered the optimal action and expected the robot to comply.

Formally, at every time instant t each agent i, where $i \in \{1, 2\}$,[1] chooses an action $a_t^{(i)} \in \mathcal{A}_i$ locally. The team action is defined as the union of both agents' actions, i.e., $a_t := (a_t^{(1)}, a_t^{(2)})$. The team action space is denoted by $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$. It is helpful to think of $\mathcal{A}$ as a team action matrix and each possible team action as a cell of the matrix. Thus, at any time instant t, agent 1 selects one row out of the $|\mathcal{A}_1|$ rows and agent 2 selects one column out of the $|\mathcal{A}_2|$ columns. We assume the agents select their local actions simultaneously, i.e., before observing their partner's current action.

For each team action $a \in \mathcal{A}$, the rewards $\{r_t^*(a)\}_{t \geq 1}$ are sampled independently from a Bernoulli distribution with unknown mean $\mu_a \in [0, 1]$. Note the reward is a function of both agents' actions. We refer to such scenarios as settings with *coupled rewards*, where the actions of all agents govern the reward received by each agent. This necessitates that each agent learns to account for others' actions instead of greedily optimizing its own rewards. In the burger example,

[1] We generalize to more agents in Section 5.

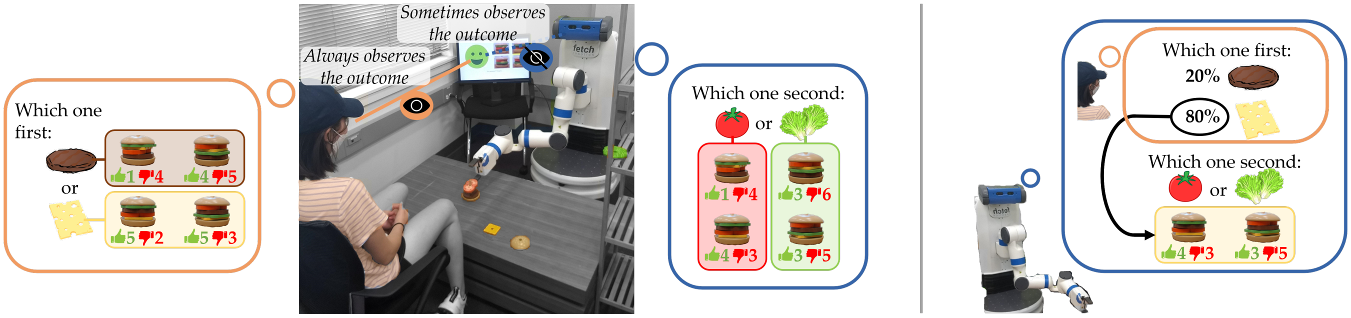

Figure 1: AI agents collaborating with humans should model their human partners. (Left) A robot and a human collaborate on a burger stacking task. (Right) The robot decides its actions by modeling the actions of the human.

the robot learns to choose actions to stack the ingredients in the right order while learning to predict the human's action.

We assume the agents observe the rewards with a fixed probability p_i, i.e., $r_t^{(i)}(a) = r_t^*(a)$ with probability p_i, and the agent incorrectly assumes $r_t^{(i)}(a) = 0$ with probability $1 - p_i$.[2] We refer to such scenarios as settings with *partial reward observability*. Going back to our burger stacking example, humans may have better reward observability—they could better sense when a guest has been happy about the burger—whereas the robot needs an explicit feedback.

We assume agents observe each other's actions but not the rewards: at any time t they have the knowledge of all past team actions $\{a_\tau\}_{\tau=1}^{t-1}$ and only their own local rewards.

To summarize, our setting considers truly realistic coordination scenarios where the agent's actions influence the outcome for the whole team. In addition, the agents only observe each other's actions and do not have access to direct communication channels, which covers the difficult case where agents are heterogeneous and might have different modes of communication—a human can easily use language, but that might not be as easy for an AI agent to interpret or use. Finally, as most realistic teams, we assume agents have different capabilities (e.g. sensing capabilities) leading to different observations corresponding to their own local reward observations.

The goal of the team at every time step t is to select an action a_t that maximizes the average team reward $r_t(a_t) = \frac{1}{2} r_t^{(1)}(a_t) + \frac{1}{2} r_t^{(2)}(a_t)$.[3] Let a_* denote the optimal team action, i.e., $a_* = \arg\max_{a \in \mathcal{A}} \mathbb{E}[r_t(a)]$. Thus, agent 1 seeks to identify the *optimal row* $(a_*^{(1)}, \cdot)$ and agent 2 tries to identify the *optimal column* $(\cdot, a_*^{(2)})$. Their intersection is the optimal cell $a_* = (a_*^{(1)}, a_*^{(2)})$. Alternatively, the team aims to minimize the cumulative regret defined as $\mathsf{R}(T) := \mathbb{E}\left[\sum_{t=1}^{T} (r_t(a_*) - r_t(a_t))\right]$. In this work, we aim to design decentralized team action strategies for each agent i that select $a_t^{(i)}$ as a function of past local rewards $r_1^{(i)}(a_1), \ldots, r_{t-1}^{(i)}(a_{t-1})$ and the team actions $a_1, \ldots, a_{t-1}$. We are interested in cases where at least one agent has partial reward observability, i.e, $p_i \neq 1$ for some $i \in \{1, 2\}$.

4 Partner-Aware Bandit Learning

We now present a learning algorithm for the decentralized collaborative MAB when the agents have partial reward observability, and their rewards are coupled. Different local observations due to partial reward observability can lead to the agents wanting to select different team actions. Since the agents' rewards are coupled, such a mismatch in the agents' action-choices can cause them to explore their action space inefficiently as a team. To successfully collaborate, the agents need to learn to predict their partners' actions correctly. Modeling partner's belief states and action-strategy has been well-studied in the theory of mind literature; however, such recursive belief modeling techniques can get computationally prohibitive and do not scale well with the number of agents (Hellström and Bensch 2018). Instead, we introduce a computationally simple way of predicting the partner's actions in the collaborative multi-armed bandit domain—which is a useful abstraction that enables theoretically analyzing multi-agent interactions. The core ideas of our approach, though simple, lead to an analytical algorithm with logarithmic regret, and can provide insight for partner modeling beyond multi-armed bandits.

Let $p_{\max} := \max\{p_1, p_2\}$ and $p_{\min} := \min\{p_1, p_2\}$. We refer to the agent with higher reward observability ($p_{\max}$) as the *leader* and the other agent as the *follower*.[4] In our approach, the follower learns to predict the leader's actions. It chooses its local action assuming the leader's current action will match its prediction. As its predictions become more accurate, the leader leads the follower to explore the optimal row in the action matrix $\mathcal{A}$. Since the leader has higher reward observability, the team can efficiently explore the action matrix. We rewrite the team action based on leader and follower assignment as: $a_t = (a_t^{(L)}, a_t^{(F)}) \in \mathcal{A} := \mathcal{A}_L \times \mathcal{A}_F$.

[2] Agents do not know when they failed to observe the reward. Knowing it is a simpler setting, where agents update their local reward statistics only when a reward is observed.

[3] Since the rewards are coupled, our analysis and Theorem 1 will extend to the case where the team reward is any linear combination of agents' rewards.

[4] Our algorithm extends to the case where p_1 and p_2 are unknown, in which case leader and follower roles are assigned randomly and the $p_{\max}$ terms in the denominators of Theorem 1 will be replaced with $p_{\min}$.

Algorithm 1: Partner-Aware UCB: Follower

Input: $\delta > 0$, $W \geq 1$, exploration constant: $c^{(F)} > 0$

1 **Definition:** Denote empirical mean
$\hat{\mu}_a^{(F)}(t) = \frac{\sum_{\tau=1}^{t} r_\tau^{(F)}(a_\tau)\mathbf{1}\{a_\tau = a\}}{n_a(t)} \quad \forall a \in \mathcal{A}$

2 Denote upper confidence bound
$f_a^{(F)}(t,\delta) = \hat{\mu}_a^{(F)}(t-1) + \sqrt{\frac{c^{(F)} \log 1/\delta}{n_a(t-1)}} \quad \forall a \in \mathcal{A}$

3 **Initialize:** $n_a(0) = 0, \hat{\mu}_a^{(F)}(0) = 0, f_a^{(F)}(1,\delta) = \infty$ for all $a \in \mathcal{A}$, set $\rho_t^{(L)}(a) = \frac{1}{|\mathcal{A}_L|}\ \forall a \in \mathcal{A}_L$

4 **for** $t = 1, \ldots, T$ **do**

5 Predict leader's action by sampling $\tilde{a}_t^{(L)} \sim \rho_t^{(L)}$

6 Select $a_t^{(F)} \leftarrow \arg\max_{a \in \mathcal{A}_F} f^{(F)}_{(\tilde{a}_t^{(L)}, a)}(t,\delta)$

7 Perform $a_t^{(F)}$

8 Observe partner's action $a_t^{(L)}$ and reward $r_t^{(F)}(a_t)$

9 Update $n_{a_t}(t) \leftarrow n_{a_t}(t-1) + 1$

10 Update $\hat{\mu}_{a_t}^{(F)}(t)$ and $f_{a_t}^{(F)}(t,\delta)$

11 Update $\rho_{t+1}^{(L)}(a) \leftarrow \frac{\sum_{\tau=\max\{1,t-W+1\}}^{t} \mathbf{1}\{a_\tau^{(L)} = a\}}{\min\{t,W\}}\ \forall a \in \mathcal{A}_L$

Algorithm 2: Partner-Aware UCB: Leader

Input: $\delta > 0$, $L \geq 1$, exploration constant: $c^{(L)} > 0$

1 **Definition:** Denote empirical mean
$\hat{\mu}_a^{(L)}(t) = \frac{\sum_{\tau=1}^{t} r_\tau^{(L)}(a_\tau)\mathbf{1}\{a_\tau = a\}}{n_a(t)} \quad \forall a \in \mathcal{A}$

2 Denote upper confidence bound
$f_a^{(L)}(t,\delta) = \hat{\mu}_a^{(L)}(t-1) + \sqrt{\frac{c^{(L)} \log 1/\delta}{n_a(t-1)}} \quad \forall a \in \mathcal{A}$

3 **Initialize:** $n_a(0) = 0, \hat{\mu}_a^{(L)}(0) = 0, f_a^{(L)}(1,\delta) = \infty$ for all $a \in \mathcal{A}$

4 **for** $t = 1, \ldots, T$ **do**

5 **if** $t \bmod L = 1$ **then**

6 Select $\left(a_t^{(L)}, \cdot\right) \leftarrow \arg\max_{a \in \mathcal{A}} f_a^{(L)}(t,\delta)$

7 **else**

8 $a_t^{(L)} \leftarrow a_{t-1}^{(L)}$

9 Perform $a_t^{(L)}$

10 Observe partner's action $a_t^{(F)}$ and reward $r_t^{(L)}(a_t)$

11 Update $n_{a_t}(t) \leftarrow n_{a_t}(t-1) + 1$

12 Update $\hat{\mu}_{a_t}^{(L)}(t)$ and $f_{a_t}^{(L)}(t,\delta)$

We denote the optimal team action as $a_* = (a_*^{(L)}, a_*^{(F)})$. Similarly, $r_t^{(L)}$ and $r_t^{(F)}$ denote the observed rewards.

Partner-Aware UCB: Follower. We provide the pseudocode in Algorithm 1. At every time step t, the follower predicts the leader's current action by sampling from a distribution $\tilde{\rho}_t^{(L)}$ over leader's action space $\mathcal{A}_L$ (line 5), which is obtained by normalizing the histogram computed from the leader's past W actions. Intuitively, $\tilde{\rho}_t^{(L)}$ serves an approximation of the leader's action selection strategy. As the leader becomes more confident about the optimal action and starts to exploit, the distribution $\tilde{\rho}_t^{(L)}$ concentrates over the optimal action. Hence, the follower gets more accurate in its predictions of the leader's actions. At every time step, the follower uses its prediction of leader's action $\tilde{a}_t^{(L)}$ to fix a row in the action matrix $\mathcal{A}$ and choose one of the $|\mathcal{A}_F|$ columns. To do so, it computes an upper confidence bound on the mean value for the actions in the row $\tilde{a}_t^{(L)}$, and chooses the action maximizing the upper confidence bound (line 6):

$$a_t^{(F)} := \arg\max_{a^{(F)} \in \mathcal{A}_F} \hat{\mu}^{(F)}_{(\tilde{a}_t^{(L)}, a^{(F)})}(t-1) + \sqrt{\frac{c^{(F)} \log 1/\delta}{n_{(\tilde{a}_t^{(L)}, a^{(F)})}(t-1)}},$$

where $\hat{\mu}_a^{(F)}$ denotes the empirical mean of the follower's local rewards, n_a denotes the action count for any team action a, and $c^{(F)}, \delta > 0$ are exploration parameters.

In short, the follower predicts the leader's action by looking at its past W actions. If the leader takes some actions more frequently, then the follower predicts those actions with high probability and aids the leader in exploring them.

Partner-Aware UCB: Leader. Our partner-aware UCB algorithm for the leader is an extension of the well-known UCB algorithm. We provide the pseudocode in Algorithm 2. For each team action, the leader computes an upper confidence bound on its mean value using the local observations. The leader then selects a team action that maximizes the upper confidence bound (line 6), similar to the follower's selection criterion. The leader then plays its own coordinate of the team action it selected, and it repeats every action it selects for L consecutive time steps (line 8).

As the follower predicts the leader's action based on each action's frequency in the past W time steps, the leader repeating its actions more than once ($L > 1$) ensures the follower's prediction matches the leader's action with a high probability. We use this for our analysis, but employ $L = 1$ in practice to avoid potential losses due to repetitive actions.

Theorem 1. *For any horizon T, if $\delta = \frac{1}{T^2}$, $L = 2$ and $W = 1$, the cumulative regret of partner-aware bandit learning algorithm, as defined in Algorithms 1 and 2, is logarithmic in the horizon T. Specifically, the cumulative regret $\mathsf{R}(T)$ can be upper bounded by*

$$(p_{max} + p_{min})\Delta_{max}\left[\sum_{i \neq a_*^{(L)}} \frac{16}{p_{max}^2 \Delta^2_{(i,j^*(i))}} \log T + \sum_{i \in \mathcal{A}_L} \sum_{j \neq j^*(i)} \frac{16}{p_{max}^2 \tilde{\Delta}^2_{(i,j)}} \log T + \frac{3|\mathcal{A}_L||\mathcal{A}_F|}{2}\right],$$

where $\Delta_{max} = \max_{a \in \mathcal{A}} \Delta_a$, $j^*(i) := \arg\max_{j \in \mathcal{A}_F} \mu_{(i,j)}$, $\tilde{\Delta}_{(i,j)} = \mu_{(i,j^*(i))} - \mu_{(i,j)}$ for $i \in \mathcal{A}_L$ and $j \neq j^*(i)$, and $\Delta_{(i,j^*(i))} = \mu_{a^*} - \mu_{(i,j^*(i))}$ for $i \neq a_*^{(L)}$.

This theorem analyzes a special case, $L = 2$ and $W = 1$, where the follower predicts the leader will take the same action it took in the last time step. As the leader repeats its actions twice, these predictions are correct for at least half of all time steps.[5] Thus, the agents jointly explore the row

[5] Similarly, the proof can be generalized to any $\lfloor W/2+1 \rfloor < L$.

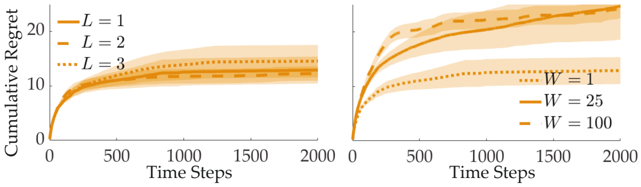

Figure 2: Cumulative regret values over 100 runs for varying (left) L and (right) W. Shaded regions show standard error.

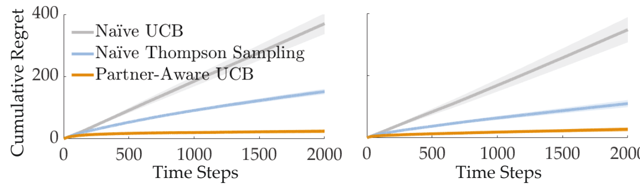

Figure 3: Cumulative regrets over 100 runs for different algorithms with (left) fixed and (right) random reward means.

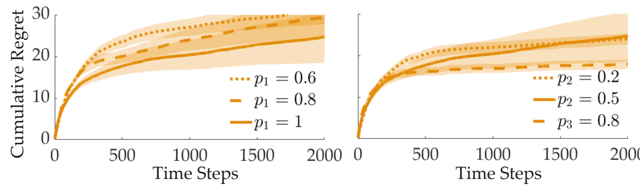

Figure 4: Cumulative regret values over 100 runs under different (left) leader and (right) follower observabilities.

which the leader intends to explore for at least half of the time steps. After the leader converges to the optimal row $a_*^{(L)}$, the agents jointly explore the optimal column to learn the optimal cell $(a_*^{(L)}, a_*^{(F)})$. The proof uses this intuition (see Appendix B in the extended paper (Bıyık et al. 2021)).

5 Simulations

We assess the performance of Partner-Aware UCB through a set of simulations.[6] Unless otherwise noted $|\mathcal{A}_1| = |\mathcal{A}_2| = 2$, $p_1 = 1$, $p_2 = 0.5$, $c^{(L)} = c^{(F)} = 0.025$ in these simulations.

Validation of Theoretical Results. We start with validating the theoretical results we established in Theorem 1. For this, we ran a simulation with fixed reward means.[7]

Figure 2 (left) shows the results that validate the theorem. It also provides a comparison between different L values. Having seen that the algorithm performs comparably with no significant difference with varying L, we use $L = 1$ for the rest of the simulations and experiments, because it reduces the leader to a standard UCB agent and relaxes the assumption that the leader repeats its actions, which is particularly desirable in human-AI interaction with the human acting in the leader role.

We compare different W in Fig. 2 (right). Here, $W = 1$ outperforms larger window widths. However, this assumes the follower is paired with a UCB leader, who selects the next action based on the entire history of actions and local rewards. This is unrealistic when interacting with a human leader. Humans are often bounded rational and make decisions only based on the most recent information (Zhang and Yu 2013; Simon 1995). We thus will use higher values of W in practice to increase the follower's horizon to the past, which can potentially improve robustness (see Appendix C).

Effect of Partner-Awareness. When established MAB algorithms, such as UCB and Thompson sampling, are naïvely used in the multi-agent case, each agent attempts to solve a single-agent MAB problem in the team action space. While they are known to produce logarithmic regret in the single-agent case, the collaborative problem is much more challenging, since agents can only decide their part of the team action. Hence, we empirically show that simply pairing such standard algorithms in the multi-agent setting fails whereas our Partner-Aware UCB achieves sublinear regret.

For this, we ran two simulations: one with fixed reward means (the same as before), and one where the reward means are generated randomly from a Unif$[0, 1]$ prior. For Partner-Aware UCB, we set $L = 1$, $W = 25$. Figure 3 shows the results. While Naïve UCB and Naïve Thompson Sampling result in linear regrets, our algorithm achieves sublinear regret in both cases. This result provides strong empirical evidence for our claim and demonstrates the importance of partner-awareness and partner-modeling. Additional simulations are presented in Appendix D.

Varying Other Conditions. Having demonstrated the success of Partner-Aware UCB, we investigate its performance under varying conditions. Specifically, we check the effects of observability.

For this, we ran simulations with fixed reward means (the same as before) and vary $p_1 \in \{0.6, 0.8, 1.0\}$, $p_2 \in \{0.2, 0.5, 0.8\}$. We set $L = 1$, and $W = 25$.

Figure 4 shows the results for both varying p_1 (left), and p_2 (right) experiments. In all cases, Partner-Aware UCB incurs only sublinear regret.[8]

We also experiment with varying number of available actions to the agents. Figure 5 (left) shows the results for $|\mathcal{A}_1| = |\mathcal{A}_2| \in \{2, 3, 4\}$ averaged over 100 runs.[9] While regret naturally increases with more actions, Partner-Aware UCB achieves sublinear regret in all cases. Due to different scale, we present the results with 30 actions on a separate plot in Fig. 5 (right) under the random reward setting.

Generalization to More Agents. We now generalize our algorithm to more than two agents, a useful formalism for applications in human-robot teams. For this, we first note there is a leader and a follower in the original algorithm, and only one of them models the other. The primary motivation for

[6] Code at: https://sites.google.com/view/partner-aware-ucb

[7] We set $\mathcal{A}_1 = \mathcal{A}_2 = \{0, 1\}$ and $\mu_{(0,0)} = 0.8$, $\mu_{(0,1)} = 0.4$, $\mu_{(1,0)} = 0.2$, $\mu_{(1,1)} = 0.6$. This is a difficult setting, as agents may easily converge to the local optimum $a = (1, 1)$.

[8] As the cumulative regret takes partial observability into account, it does not necessarily increase with lower observability.

[9] Similar to the fixed reward values as in the two-action case, we designed the rewards such that there are $|\mathcal{A}_1| = |\mathcal{A}_2|$ local optima.

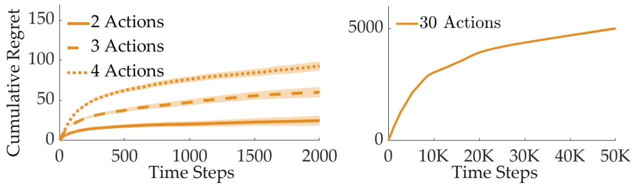

Figure 5: Average regret values over 100 runs with varying number of available actions for both agents.

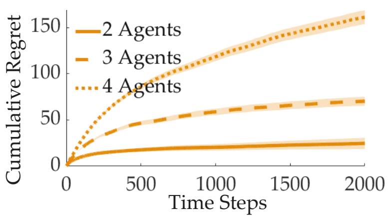

Figure 6: Average regret values over 100 runs with varying number of agents with the extended algorithm.

this is to avoid deadlock situations where agents oscillate between actions to "catch" the other agent's behavior.

While agents should not model each other, it would also not be enough if they modeled only the agent with the highest reward observability. Even if they could accurately predict that agent's actions, they would still need to solve a decentralized MAB among themselves.

This informs us about the following recursive approach. Suppose there are N agents. As in the original Partner-Aware UCB, the agent with the highest observability does not model the others and optimizes its own action as if the others will comply. All other agents model and attempt to predict this leader agent. They now have to deal with an $(N-1)$-agent problem. Hence, the agent with the second highest observability does not model the remaining $N-2$ agents who, on the other hand, model this "second leader". This hierarchy we impose based on observability continues until the problem reduces to single-agent for the last agent.

To test if the extended algorithm achieves sublinear regret, we ran simulations with varying number of agents from $N \in \{2, 3, 4\}$, and $|A_i| = 2$ for all agents and $p_i = i/N$. The reward means were fixed[10], and we set $c^{(i)} = 0.025$, $L = 1$, and $W = 25$ for all agents. Fig. 6 shows the results averaged over 100 runs. The extended Partner-Aware UCB achieves sublinear regret in all cases.

Generalization to Other Bandits. The reason why Partner-Aware UCB performs successfully is that it allows the follower agent to learn its best individual action conditioned on the leader's action, which allows the agents to discover the team-optimal action. We hypothesize this idea could solve a broader class of decentralized cooperative bandit problems.

To test this, we simulated two settings: (i) a flipped setting where the agents get a reward of 1 instead of 0 with probability of p_i (and still get $r_t^*(a)$ with probability $1-p_i$),

[10]Similar to the other experiments, we designed the reward values such that there are $|A_1| = \cdots = |A_N| = 2$ local optima.

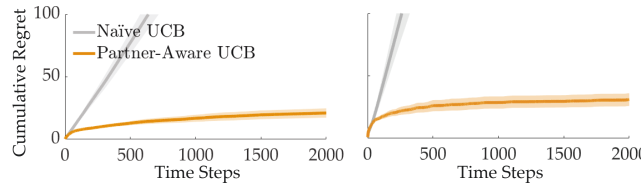

Figure 7: Average regret values over 100 runs in the (left) flipped and (right) Gaussian settings.

(ii) a Gaussian bandits setting where $r_t^*(a)$ comes from an action-dependent stationary Gaussian distribution with unknown mean and variance, and agents get reward $r_t^*(a) + \nu_i$ where $\nu_i \sim \mathcal{N}(0, \sigma_i)$ for different σ_i (we set the agent with higher σ_i to be the follower as it has noisier rewards).

As it can be seen in Fig. 7, where we ran simulations with random reward means (and random std for $r_t^* \in [0.1, 0.5]$ in Gaussian bandits with $\sigma_1 = 0.1$ and $\sigma_2 = 0.5$), Partner-Aware UCB outperforms Naïve UCB and achieves a sublinear regret even in these modified settings.

6 Experiments

We now empirically analyze our algorithm through an in-lab human-subject study where the participants collaborate with a robot arm to stack burgers. While this experiment involves a short horizon, we also present an online human-subject study where the participants collaborate with a robot for long horizons to maximize their profit on a grid of slot machines in Appendix F. Our user studies have been approved by the local research compliance office. Subjects were compensated with \$15/hour for their participation.

Experimental Setup. We designed a collaborative burger stacking experiment as shown in Fig. 1.[11] Subjects were told they work at a burger store with a robot to stack burgers. They are responsible for placing the patty and the cheese, whereas the robot is for the tomatoes and lettuce. They decide whether the patty or the cheese should go on top of the bottom bun, and the robot decides the second layer. Decisions are simultaneous without knowing each other's action.

The participants were told there is a fixed probability associated with whether a customer liked the burger. After stacking each burger, the robot and the human are informed about if a customer was satisfied. The robot has a sensor defect and observes only half of the satisfied customers. It senses the others as unsatisfied (human's observability is $p_1 = 1$ and robot's $p_2 = 0.5$). The goal of both the human and the robot is to maximize the number of satisfied customers.

Independent Variables. We varied the robot's algorithm: Naïve UCB and Partner-Aware UCB. We set, when relevant, $L = 1$, $W = 2$ and $c^{(L)} = c^{(F)} = 0.01$.

Procedure. We conducted a within-subjects study with a Fetch robot (Wise et al. 2016) for which we recruited 58 participants (22 female, 36 male, ages 18 – 69). Due to the pandemic conditions, the first five of the subjects participated the study with a real robot in the lab, and the rest participated remotely with an online interface. The participants

[11]Video at: https://sites.google.com/view/partner-aware-ucb

interacted with the robot to prepare 40 burgers together, 20 with each algorithm. The participants knew the number of burgers they are going to prepare in advance.

Initially, MAB requires significant exploration, so comparison between the two algorithms at early stages will not yield any meaningful results. However, evaluating later stages of collaboration would require many repeated long-term interactions with the robot, which is not feasible due to limitations on the duration of in-lab studies with a robot. Instead, we warm-start each algorithm by allowing them to collaborate with a simulated Naïve UCB agent for stacking 20 burgers to proceed forward in the exploration stage so that the robot's algorithm will be more critical for performance. After these 20 burgers, the simulated agent is replaced with the study participant for preparing 20 more burgers with each algorithm.

The user interface aided the participants by providing information about: the number of satisfied and unsatisfied customers for each burger configuration, the total number of burgers stacked, the configuration of the latest burger and whether it made the customer satisfied.

For a fair comparison, we randomized the reward means only between the users and not between the algorithms. We swapped the actions to prevent participants from realizing they are dealing with the same problem instance. Hence, between the two sets, for example, $\mu_{(0,1)}$ of the first set was equal to $\mu_{(1,0)}$ in the second. To further avoid any bias due to ordering, half of the participants first worked with Naïve UCB and the other half with the Partner-Aware UCB.

Dependent Measures. We measured cumulative regret and the number of satisfied customers. We excluded the first 20 simulated time steps for fairness. The participants took a 5-point rating scale survey (1-Strongly Disagree, 5-Strongly Agree) consisting of 5 questions for each algorithm: "I was usually able to stack the burger I wanted" (*Ability*), "The robot insisted on some suboptimal burgers" (*Insisting*), "The robot was easy to collaborate with" (*Easy*), "The robot was annoying" (*Annoying*), and "I could get more happy customers if I were stacking burgers alone" (*Alone*).

Hypotheses.

H1. *Users interacting with Partner-Aware UCB robot will incur smaller regret and keep the customers more satisfied.*
H2. *Users will subjectively perceive the Partner-Aware UCB robot as a better partner who can effectively collaborate with them.*

Results-Objective. Partner-Aware UCB achieves lower regret (2.7 ± 0.31) compared to Naïve UCB (3.6 ± 0.31) with statistical significance ($p < .005$). Fig. 8 (left) shows the cumulative regret incurred over time with both algorithms.

The significant difference in the cumulative regret is also reflected in the number of satisfied customers supporting **H1**: Partner-Aware UCB achieved significantly higher number of satisfied customers (13.5 ± 0.6) than Naïve UCB (12.7 ± 0.6), with $p < .05$.

Results-Subjective. The Naïve UCB robot often decides stacking an under-explored burger and insists on the same action until that burger is made. While this occasionally helps the humans to exploit a burger, it often causes deadlock situations where both agents are unable to stack their intended burgers. On the other hand, Partner-Aware UCB keeps a model of the human, and avoids such situations. We believe this explains the subjective preferences of the users.

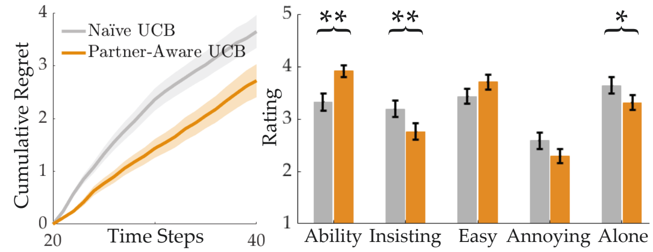

Figure 8: (Left) Average regret over time, (right) survey results for the burger-stacking robot experiment. Single and double asterisks indicate $p < .05$ and $p < .005$, respectively.

We plot the users' survey responses in Fig. 8 (right). The responses were reliable with Cronbach's alpha > 0.95. The users indicated they were able to stack the burger they wanted (*Ability*) more frequently with the Partner-Aware UCB ($p < .005$), and thought it was easier to collaborate with (*Easy*, $p \approx .07$), whereas found the Naïve UCB robot more annoying (*Annoying*, $p \approx .05$). They also indicated the Naïve UCB insisted more on the suboptimal burgers (*Insisting*, $p < .005$). Finally, the users think they could have more satisfied customers if they were stacking burgers alone with a higher confidence when partnered with the Naïve UCB robot (*Alone*, $p < .05$). These results strongly support **H2**. We further analyze and discuss how different populations of human users perform differently in Appendix E.

7 Conclusion

Summary. We studied multi-agent decentralized MAB, where the reward obtained by the team depends on all agents' actions. We showed naïve extensions of optimal single-agent MAB algorithms – where each agent disregarded others' actions – fail when rewards are coupled. We proposed a simple yet powerful algorithm for partners to model and coordinate with the partners who have higher observability over the task. Our algorithm only relies on the observation of partner's actions and accomplishes the coordination without explicit communication. We analytically showed it achieves logarithmic regret and tested our hypotheses through simulations and experiments.
Limitations and Future Work. The decentralized MAB is a useful abstraction for many real-world coordination tasks, and we are excited that our algorithm yet simple demonstrates significant improvements to enable seamless coordination. However, many applications require more complex formulations such as Markov Decision Processes. In the future, we plan to extend the intuitions gained by our algorithm and analysis to some of these more complex settings.

Another interesting direction is pairing the partner-aware strategy with algorithms other than UCB, e.g., Thompson sampling. Our preliminary results indicate it still gives significant improvements over the naïve counterparts.

Acknowledgments

The authors acknowledge funding from NSF Awards #1941722, #2006388, and #2125511, Office of Naval Research, and Air Force Office of Scientific Research.

References

Baker, C. L.; Jara-Ettinger, J.; Saxe, R.; and Tenenbaum, J. B. 2017. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, 1(4): 1–10.

Barrett, S.; Agmon, N.; Hazon, N.; Kraus, S.; and Stone, P. 2014. Communicating with Unknown Teammates. In *ECAI*, 45–50.

Bıyık, E.; Lalitha, A.; Saha, R.; Goldsmith, A.; and Sadigh, D. 2021. Partner-Aware Algorithms in Decentralized Cooperative Bandit Teams. *arXiv preprint arXiv:2110.00751*.

Breazeal, C.; Kidd, C. D.; Thomaz, A. L.; Hoffman, G.; and Berlin, M. 2005. Effects of nonverbal communication on efficiency and robustness in human-robot teamwork. In *2005 IEEE/RSJ international conference on intelligent robots and systems*, 708–713. IEEE.

Brooks, C.; and Szafir, D. 2019. Building Second-Order Mental Models for Human-Robot Interaction. *arXiv preprint arXiv:1909.06508*.

Burge, J. 2017. How Do You Like Your Burger Emoji? https://blog.emojipedia.org/how-do-you-like-your-burger-emoji/. Accessed: 2021-05-20.

Che, Y.; Okamura, A. M.; and Sadigh, D. 2020. Efficient and Trustworthy Social Navigation Via Explicit and Implicit Robot-Human Communication. *IEEE Transactions on Robotics (T-RO)*.

Devin, S.; and Alami, R. 2016. An implemented theory of mind to improve human-robot shared plans execution. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 319–326. IEEE.

Foerster, J.; Chen, R. Y.; Al-Shedivat, M.; Whiteson, S.; Abbeel, P.; and Mordatch, I. 2018. Learning with Opponent-Learning Awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 122–130.

Grover, A.; Al-Shedivat, M.; Gupta, J.; Burda, Y.; and Edwards, H. 2018. Learning policy representations in multiagent systems. In *International conference on machine learning*, 1802–1811. PMLR.

Hellström, T.; and Bensch, S. 2018. Understandable robots-what, why, and how. *Paladyn, Journal of Behavioral Robotics*, 9(1): 110–123.

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

Lee, J. J.; Sha, F.; and Breazeal, C. 2019. A Bayesian theory of mind approach to nonverbal communication. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 487–496. IEEE.

Losey, D. P.; Li, M.; Bohg, J.; and Sadigh, D. 2020. Learning from my partner's actions: Roles in decentralized robot teams. In *Conference on Robot Learning*, 752–765. PMLR.

Lupu, A.; Durand, A.; and Precup, D. 2019. Leveraging Observations in Bandits: Between Risks and Benefits. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, 6112–6119. AAAI Press.

Madhushani, U.; and Leonard, N. E. 2019. Heterogeneous stochastic interactions for multiple agents in a multi-armed bandit problem. In *2019 18th European Control Conference (ECC)*, 3502–3507. IEEE.

Martínez-Rubio, D.; Kanade, V.; and Rebeschini, P. 2019. Decentralized cooperative stochastic multi-armed bandits. *Advances in Neural Information Processing Systems*.

Sankararaman, A.; Ganesh, A.; and Shakkottai, S. 2019. Social learning in multi agent multi armed bandits. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(3): 1–35.

Shahrampour, S.; Rakhlin, A.; and Jadbabaie, A. 2017. Multi-armed bandits in multi-agent networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2786–2790. IEEE.

Shih, A.; Sawhney, A.; Kondic, J.; Ermon, S.; and Sadigh, D. 2021. On the Critical Role of Conventions in Adaptive Human-AI Collaboration. In *9th International Conference on Learning Representations (ICLR)*.

Simon, H. A. 1995. The information-processing theory of mind. *American Psychologist*, 50(7): 507.

Wise, M.; Ferguson, M.; King, D.; Diehr, E.; and Dymesich, D. 2016. Fetch and freight: Standard platforms for service robot applications. In *Workshop on autonomous mobile service robots*.

Xie, A.; Losey, D.; Tolsma, R.; Finn, C.; and Sadigh, D. 2020. Learning Latent Representations to Influence Multi-Agent Interaction. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*.

Zhang, S.; and Yu, A. 2013. Cheap but clever: Human active learning in a bandit setting. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 35.

Zhu, Z.; Biyik, E.; and Sadigh, D. 2020. Multi-Agent Safe Planning with Gaussian Processes. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.