Toward Capability-Aware Cooperation for Decentralized Planning

Charles Jin, Zhang-Wei Hong, Martin Rinard[1]

Abstract— **Many works cast the problem of decentralized multi-agent cooperation as fundamentally one of misspecified (or incorrectly inferred) goals. In this work, we show that challenges can also arise naturally due a difference in capabilities—even in an idealized setting of perfectly aligned agents with full information. In particular, we consider cooperative 2-player games, in which each agent employs a truncated version of Monte Carlo tree search (MCTS) for infinite-horizon interactions that limits their simulation to a maximum depth. We provide examples of games in which deviations in computational capability between agents can lead to arbitrarily poor outcomes. To address this challenge, we propose an algorithm that maintains a belief over the other agent's computational capacities and incorporates this belief into the MCTS. We experimentally validate that our capability-aware algorithm avoids the anti-cooperative behavior of the naïve approach in several toy settings. These results thus suggest that decentralized multi-agent settings requires further consideration of the challenges arising from differing computational (or cognitive) capabilities.**

I. INTRODUCTION

Consider a generic two-player cooperative game where an expert is paired with a novice. Intuitively, the expert should only play strategies which the novice can reasonably infer and react to. Otherwise, the expert may attempt a complex strategy which has a high reward assuming optimal play, but incurs a costly penalty if the novice missteps. This observation is supported by experimental studies in psychology and cognitive science in the context of general strategic games [1], [2] and chess [3], [4].

This work initiates a study of the challenges of cooperation between agents of different capabilities in an idealized setting with perfectly aligned agents and full information. Our contributions are as follows: first, we introduce a formal setting for studying the problem of cooperation between heterogeneous agents of differing computational capacities. In particular, each agent searches using the MCTS method, but may have different simulation depths. Second, we describe the Capability-Aware Monte Carlo tree search (CA-MCTS) algorithm, which explicitly maintains a belief over the other agent's search depth, and incorporates this information during the search process. Several experiments support the ability of CA-MCTS to overcome the difficulties of differing computational capacities in two-player cooperative navigation tasks, whereas vanilla MCTS leads to deadlocking or infinite loops of negative rewards.

II. RELATED WORKS

Our work is relevant to teammate/opponent modeling in cooperative multi-agent planning [5] tasks. Cooperative multi-agent planning aims to coordinate a team of agents to maximize a joint utility function. Classic works employ heuristic search algorithms on each agent to maximize the joint utility. For example, [6] coordinate collision-free paths for agents in a team using the rapidly-exploring random tree (RRT) algorithm. In addition to heuristic search, prior works [7], [8] in multi-agent reinforcement learning (RL) [9] jointly learn policies for each agent. Both lines of work assume that the policies of teammates are known and controllable, while we assume the teammates are uncontrollable and have unknown policies.

An application domain of particular interest is human-robot interaction, where the humans' policies are unknown and thus robots have to collaborate without full knowledge of their teammates. One approach is for the robot to apply inverse reinforcement learning to infer the value function of the human [10], [11], [12]. However, humans are well-known to exhibit sub-optimal behavior, which poses a challenge to works that model humans as perfectly rational. One way to model such behavior is called *bounded rationality*, wherein humans are assumed to be (approximately) optimal subject to certain constraints [13], [14]. For instance, several works consider agents which have limited capacity to reason recursively about other agents [15], [16], and prove convergence assuming the recursive depths in the population follows a Poisson distribution. [17] models human teammates as having bounded memory in the sense that they reason over only the most recent k interactions. However, all these works treat the amount of "boundedness" as a known quantity; in contrast, our work studies the problem of adapting to an unknown level of boundedness.

III. PROBLEM FORMULATION

We formulate the problem of decentralized cooperation as a two-player game, characterized as an infinite-horizon discrete-time decentralized Markov decision process (Dec-MDP) [18]. The Dec-MDP is given by a tuple $\langle \mathcal{S}, \mathcal{N}, \mathcal{A}, P, R, \rho_0, \gamma \rangle$, where $\mathcal{S}$ is the set of all states of the MDP, $\mathcal{N} = \{0, 1\}$ denotes the two players, $\mathcal{A}$ denotes the set of available actions, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S} \times \{0, 1\}$ is the state transition function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho : \mathcal{S} \times \{0, 1\} \to [0, 1]$ is an initial state distribution, and $\gamma \in [0, 1)$ is the time discounting factor. Each episode of the game starts from a state and player s_0, n_0 sampled from ρ. At each timestep t, player n_t selects an action a_t based on s_t; then the two players receive a reward $r_t = R(s_t, a_t)$ and transition to the next timestep $t+1$ with state $s_{t+1}, n_{t+1} = P(s_t, a_t)$. The goal of both players is to maximize the joint return $\sum_{t=0}^{\infty} \gamma^t r_t$.

[1]CSAIL, MIT, Cambridge, MA, USA. Corresponding author contact: ccj@csail.mit.edu.

A. *Monte Carlo tree search*

We will assume that the two agents in the Dec-MDP play using the Monte Carlo tree search (MCTS) algorithm [19], a common approach for sequential decision making tasks; MCTS has seen a surge in popularity for games and planning in recent years since forming an integral part of AlphaGo, the super-human computer program for the game of Go [20]. Here, we give a basic description of the MCTS algorithm to motivate our setting; for a more complete overview, we refer the reader to [21].

Given a state $s_0 \in \mathcal{S}$, we can formally construct a game tree $\mathcal{T}$ as follows: the root of the tree is the initial state s_0, and two nodes s and s' in the tree have a parent-child relationship with directed edge $a : s \to s'$ if and only if we have a transition $P(s, a) = s'$; in this case we also label the child node s' with the reward $R(s, a)$. Finally, we say a node with state s has **value** $v(s)$ if the optimal directed path starting from s yields total (time-discounted) rewards of $v(s)$. Clearly, the value of a node can be computed recursively given the value (and rewards) of its children.

The planning problem can thus be formalized as computing the value of all children of the root node. Of course, in general this may require searching the game tree exhaustively; hence Monte Carlo tree search provides a method of constructing incremental approximations $\widetilde{\mathcal{T}}$ of the game tree with *estimated* values.

We provide a brief overview of the main steps in MCTS. The search is initialized with the single root node $\widetilde{\mathcal{T}} = s_0$. At each step, we **Select** a path to a leaf node s, **Expand** the leaf node by adding all legal transitions as children, **Simulate** an initial value estimate of s, then **Backpropogate** the estimated value of s to the root, using the recursive formula to update the value of its predecessors. In particular, **Simulate** performs m simulations of random actions down to a maximum depth of d (or until the game terminates), and takes the average cumulative reward to be the initial value estimate. **Select** is subject to the usual explore-exploit tradeoff (exploration prefers nodes which have fewer visits; exploitation prefers nodes with higher estimated values); the popular Upper Confidence Trees (UCT) variant [22] treats each choice as a multi-armed bandit problem, using the celebrated UCB1 method [23] to achieve good theoretical guarantees (and excellent performance in practice).

IV. CAPABILITY-AWARE COOPERATION

In this section, we describe an modification of the MCTS method for cooperative games which explicitly adjusts for the capability of the other player. In particular, each player maintains a prior over the depth bound of the other agent, which is updated each time the player observes an action from the other player. The player then adapts its play to incorporate its belief about the capability of the other player.

For the remainder of this section, we will fix the discussion from the perspective of an arbitrary player i. Each player in the game independently updates their own prior about the other player, and runs their own modified search in a decentralized manner. We begin by formally defining the notion of a depth prior:

Definition 1: Given a player $j \neq i$, the **depth prior** p^i_j is a probability distribution over the search depth of the player j, i.e., $p^i_j(d)$ is (the belief of player i about) the probability that the player j has maximum search depth d.

We also introduce the corresponding cumulative distribution, which will be useful our later development:

Definition 2: The **cumulative depth prior** is defined as $P^i_j(d) = \sum_{k=d}^{\infty} p^i_j(k)$, i.e., the probability that the player j has maximum search depth at least d.

We now show how to condition a game tree on a depth prior. In what follows, we define the multiplication of a tree $\mathcal{T}$ by a scalar s as the same tree but with rewards multiplied by s; and the addition of two trees $\mathcal{T}_1$ and $\mathcal{T}_2$ (with the same structure) as the tree with rewards at all nodes equal to the sum of the corresponding nodes in $\mathcal{T}_1$ and $\mathcal{T}_2$.

Definition 3: Given a game tree $\mathcal{T}$, we define the **truncation at depth** d, denoted $\mathcal{T}^d$, to be the same tree but with a reward of 0 for any node at depth greater than d.

Definition 4: Given a game tree $\mathcal{T}$ and a depth prior p, we define the **game tree conditioned on the depth prior** to be $\mathcal{T}|p = \sum_{d=0}^{\infty} p(d)\mathcal{T}^d$.

Intuitively, given the ground-truth (possibly infinite-depth) game tree $\mathcal{T}$ and a depth prior p^i_j, instead of playing optimally using $\mathcal{T}$, player i should instead play according to $\mathcal{T}|p^i_j$, which incorporates its belief about how deep into the game tree player j is able to explore. Hence, to apply this insight to our setting, it remains to show two things: first, how player i should infer p^i_j from a history of interactions; and second, how to incorporate the belief p^i_j when player i is not given a ground truth game tree $\mathcal{T}$, but is rather computing an online approximation to the game tree using MCTS at every step.

A. *Capability-Aware MCTS*

In this section, we describe a method of modifying MCTS to account for the depth prior. Note that the naïve method—which simply performs MCTS as usual to recover an approximate game tree $\widetilde{\mathcal{T}}$, and then conditions the approximate tree on the depth prior $\widetilde{\mathcal{T}}|p^i_j$—does not account for the effects of capability *during the search procedure*.[1] In particular, since MCTS preferentially explores portions of the game tree with higher values, different search depths yield different approximations to the ground-truth game tree, as the search depth directly affects estimates of node values.

Instead, we propose to perform MCTS on $\mathcal{T}|p^i_j$, the true game tree *conditioned by the depth prior*. Clearly, as $\mathcal{T}$ is generally too large to represent in complete form, we also cannot compute $\mathcal{T}|p^i_j$ directly. However, our main insight is that we can efficiently simulate MCTS over $\mathcal{T}|p^i_j$ with negligible overhead. The key is that $\mathcal{T}|p^i_j$ differs from $\mathcal{T}$ only in the rewards, so given a node in $\mathcal{T}$ and p^i_j, we can lazily compute the corresponding node in $\mathcal{T}|p^i_j$. In particular,

[1]Another wrinkle is that action selection for MCTS is most often performed by taking the node with the most visits, rather than value; in this case, the naïve approach does not directly yield a useful algorithm.

we will condition all rewards (i.e., those collected during simulation and backpropagation) on the probability that the other player would have been able to observe the reward: given a node at depth d with reward r, we will instead collect a reward of $P^i_j(d) \cdot r$ after conditioning on the depth prior (and before applying the time-based discount factor). We call this version **Capability-Aware MCTS**, or CA-MCTS. The following theorem establishes the correctness of CA-MCTS:

Theorem 1: Capability-Aware Monte Carlo tree search over the game tree $\mathcal{T}$ with depth prior p^i_j is equivalent to Monte Carlo tree search over the game tree $\mathcal{T}|p^i_j$.

To the best of our knowledge, this relationship also holds for any variant of vanilla MCTS (e.g., UCT).

B. Updating the Depth Prior

We next describe how to update the depth prior given new evidence (i.e., an action by the other player). The key is to compute the set of depths at which the other player's action *would have been* rational. Repeated interactions yields the relative frequency that the other player's actions were rational at a given depth. These relative frequencies form the basis of the depth prior. We begin with the following definition:

Definition 5: An action a is **rational at depth** d if a achieves the maximum value over all possible actions at the root of $\mathcal{T}^d$, the truncation of the ground-truth game tree $\mathcal{T}$ at depth d.

The main challenge is finding the set of depths at which the other player's action a is rational. A naïve approach would be to simulate MCTS for all depths d (up to your own maximum depth), and compare a to the best possible action at each depth. Unfortunately, this is computationally prohibitive. Instead, given a single truncated game tree $\mathcal{T}^d$, we show how to compute the best possible action (and value) at all depths $d' \leq d$ in a single bottom-up pass over $\mathcal{T}^d$. This algorithm allows us to construct the set of rational depths with negligible overhead, since $\widetilde{\mathcal{T}}^d$ can be reused from the player's original MCTS computation.

To start, consider the usual bottom-up approach to computing the single best path to a leaf node in $\mathcal{T}^d$. Each node takes the best value over its children, adds its own reward, and passes the result to its parent. The root of the tree then contains the best value over all possible paths. A simple modification allows us to return the best value for all depths in the tree. In particular, each node now maintains a *list of the best value at all depths* in the subtree below it. In the bottom-up pass, leaf nodes just return their value as before, but a node with children computes an element-wise maximum over the lists of its children, adds its own reward to every entry in the result, then inserts its own reward as the list head.

Next, to compute the set of rational depths, player i first computes the best value at all depths for all possible actions that player j could have taken in the previous turn; comparing the value of action a at each depth with the best possible value at each depth yields the set of depths at which taking the action a would have been rational.

Finally, let player i's depth bound be d. Then player i maintains a vector of $d+1$ counts, one for each depth less than or equal to its own depth. This vector is initialized to all zeros, except for the player's own depth, which is initialized to a parameter $\alpha > 0$; this allows the player to play at least the first turn with its full capabilities, where α controls the strength of the player's belief in the initialization. To update the prior, denote the set of rational depths as $D = \{d_1, d_2, ..., d_n\}$. For each depth $d_i \in D$, we increment the entry at depth d_i by $1/n$. If no depths are found to be rational, then we increment the entry at the maximum depth d by 1; this reflects player i's belief that player j has *greater* capability (since by assumption, player j is playing rationally at some depth). The depth prior is then the normalized version of this vector.

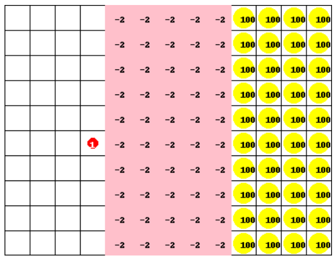
Fig. 1. The initial state for the Wall of Fire task.

V. Experimental Results

We implemented CA-MCTS with the popular UCT variant (henceforth, CA-UCT), and compared its performance to vanilla UCT on two cooperative navigation tasks. Each task consists of two players that alternate taking actions. One player is an *expert*, with a deeper search depth, and the other player is a *novice*, with a shallower search depth. The agents are otherwise identical (i.e., have the same action space, share the same reward function, use the same UCT hyperparameters, and have complete knowledge of the environment as well as the actions taken by the other agent). All results are taken from the median of 5 runs.

A. Wall of Fire

In the Wall of Fire task, the two players take turns controlling a single avatar on a two-dimensional board. The available actions at each step are to move the avatar in one of the four cardinal directions (up, down, left, right), subject to the boundaries of the board. After each move, the players collect a reward based on their position on the board: every time step ending on a red "fire" tile yields a penalty of -2, whereas every time step ending on a yellow "coin" tile yields a reward of +100. The coin tiles are also consumable, meaning that they can only be collected once; the fire tiles are permanent. We run each episode for 20 turns (10 per player). Figure 1 displays the initial state of the players and the board. Table I reports the performance of various team compositions on the task.

Clearly, the optimal strategy is to traverse the wall of fire in 5 steps, and spend the remaining steps collecting coins. Indeed, an expert with a search depth of 20 is able to reliably perform this task optimally (Expert + Expert, Table I). We

Team Composition	Reward
Novice + Novice	0
Expert + Expert	1490
Expert + Novice	-20
CA-Expert + Novice	-2

TABLE I

PERFORMANCE ON WALL OF FIRE TASK FOR DIFFERENT TEAMS.

Blue Avatar	Red Avatar	Reward
Novice	Novice	4
Expert	Expert	90
Novice	Expert	0
Novice	CA-Expert	4

TABLE II

PERFORMANCE ON NARROW TUNNEL TASK FOR DIFFERENT TEAMS.

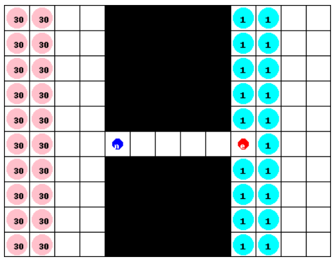

Fig. 2. The initial state for the Narrow Tunnel task. In this case, the novice controls the blue avatar and the expert controls the red avatar.

also introduce a novice player with a search depth of 2, which is insufficient to "see" past the wall of fire; the novice player thus spends its in entire time exploring the neutral tiles on the left side of the wall of fire, and ends every episode with a reward of 0 (Novice + Novice, Table I).

However, when we allow the expert and the novice to take turns controlling the avatar, we find that the expert (using vanilla UCT) takes a single step into the wall of fire (thus collecting a penalty of -2), then the novice takes a step back out; this loop often repeats for the entirety of the episode, leading to a large negative reward (Expert + Novice, Table I).

In contrast, an expert running the CA-UCT algorithm is able to infer from a single interaction that the novice is unable to see past the wall of fire, and only collects the penalty once before cooperating with the novice to "explore" the neutral left side of the board for the remainder of the episode (CA-Expert + Novice, Table I).

B. Narrow Tunnel

In the Narrow Tunnel task, two players each control an avatar on a two-dimensional board. The action space contains an additional choice to not move (up, down, left, right, no-op). The red coins are worth +30 if collected by the red avatar, and the blue coins are worth +1 if collected by by the blue avatar; the coins are worthless otherwise, and disappear regardless of which avatar collects them. We run each episode for 20 turns. Figure 2 displays the initial state of the players and the board. Table II reports the performance of various team compositions on the task.

Due to the uneven rewards, the optimal strategy is for the blue avatar to yield the narrow tunnel to the red avatar, allowing it to collect the more valuable red coins—a strategy which the expert (with search depth 30) has no trouble following (Expert + Expert, Table II). However, because the red avatar is 8 steps away from the red coins and the blue avatar is only 6 steps away from the blue coins, the novice (with search depth 10) is unable to devise this strategy, and instead sends the blue avatar through the narrow tunnel (Novice + Novice, Table II).

We next consider a heterogeneous team of an expert playing the red avatar, and a novice playing the blue avatar. When the expert runs the vanilla UCT algorithm, the two players meet in a deadlock in the center of the tunnel, neither willing to yield to the other (Novice + Expert, Table II). However, an expert running the CA-UCT algorithm takes only 1 turn of deadlock to infer that the novice's depth is insufficient to use the optimal strategy, and hence yields the tunnel to the blue agent (Novice + CA-Expert, Table II).

C. Discussion

Our experimental results highlight several interesting phenomena. First, performance does not increase monotonically as agents get more powerful. In particular, we show that increasing the computational capacity of an agent can lead to worse behavior, e.g., in the Wall of Fire task, a team of novices collects no rewards, but *upgrading* one of the novices to an expert yields a strictly worse outcome of -20. Second, this behavior occurs even though the tasks are in a ideal setting of *full information* with *complete cooperation*. In the context of human-robot cooperation, many prior works in the robot learning literature identify challenges arising from value misalignment, where the robot needs to learn the true reward function of the human from interaction [12]. Conversely, this works suggests that true reward function of humans may be subject to additional constraints which cannot be captured by a reward function in the MDP, and attempting to learn under the Markov assumption can lead to poor outcomes.

VI. CONCLUSION

In this work, we study the task of cooperation between a heterogeneous team of agents with differing computational capacities. The key observation is that agents that do not account for the mismatch in computational capacities may arrive at conflicting plans. We study an instance of this problem in agents that use a truncated version of MCTS for infinite-horizon interactions, and show that empirically such conflicting behavior does occur despite having full information and sharing rewards. To address this problem, we introduce the Capacity-Aware Monte Carlo tree search (CA-MCTS) algorithm, which explicitly incorporates a belief over the other agent's depth bound during planning. Our experiments indicate that CA-MCTS leads to better cooperation in heterogeneous teams when compared to vanilla MCTS.

References

[1] C. F. Camerer, T.-H. Ho, and J.-K. Chong, "A cognitive hierarchy model of games," *The Quarterly Journal of Economics*, vol. 119, no. 3, pp. 861–898, 2004.

[2] G. Devetag and M. Warglien, "Games and phone numbers: Do short-term memory bounds affect strategic behavior?" *Journal of Economic Psychology*, vol. 24, no. 2, pp. 189–202, 2003, the Economic Psychology of Herbert A. Simon. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167487002002027

[3] G. Campitelli and F. Gobet, "Adaptive expert decision making: Skilled chess players search more and deeper," *ICGA Journal*, vol. 27, no. 4, pp. 209–216, 2004.

[4] P. Chassy and F. Gobet, "Measuring chess experts' single-use sequence knowledge: an archival study of departure from 'theoretical'openings," *PLoS One*, vol. 6, no. 11, p. e26692, 2011.

[5] A. Torreno, E. Onaindia, A. Komenda, and M. Štolba, "Cooperative multi-agent planning: A survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–32, 2017.

[6] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4956–4961.

[7] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[8] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[10] R. Gibbons, "Incentives in organizations," *Journal of economic perspectives*, vol. 12, no. 4, pp. 115–132, 1998.

[11] A. Y. Ng, S. Russell, *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[12] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," *Advances in neural information processing systems*, vol. 29, 2016.

[13] S. J. Russell and E. Wefald, *Do the right thing: studies in limited rationality*. MIT press, 1991.

[14] H. A. Simon, *Models of bounded rationality: Empirically grounded economic reason*. MIT press, 1997, vol. 3.

[15] F. Fotiadis and K. G. Vamvoudakis, "Recursive reasoning for bounded rationality in multi-agent non-equilibrium play learning systems," in *2021 IEEE Conference on Control Technology and Applications (CCTA)*, 2021, pp. 741–746.

[16] Y. Wen, Y. Yang, R. Luo, and J. Wang, "Modelling bounded rationality in multi-agent interactions by generalized recursive reasoning," *arXiv preprint arXiv:1901.09216*, 2019.

[17] S. Nikolaidis, A. Kuznetsov, D. Hsu, and S. Srinivasa, "Formalizing human-robot mutual adaptation: A bounded memory model," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2016, pp. 75–82.

[18] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[19] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *International conference on computers and games*. Springer, 2006, pp. 72–83.

[20] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[21] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[22] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[23] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256, 2002.