Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

AN EFFICIENT DYNAMIC SAMPLING POLICY FOR MONTE CARLO TREE SEARCH

Gongbo Zhang
Yijie Peng

Guanghua School of Management
Peking University
5 Yiheyuan Road
Beijing 100871, P. R. CHINA

Yilong Xu

Department of Computer Science
Beijing Jiaotong University
3 Shangyuancun
Beijing 100044, P. R. CHINA

ABSTRACT

We consider the popular tree-based search strategy within the framework of reinforcement learning, the Monte Carlo Tree Search (MCTS), in the context of finite-horizon Markov decision process. We propose a dynamic sampling tree policy that efficiently allocates limited computational budget to maximize the probability of correct selection of the best action at the root node of the tree. Experimental results on Tic-Tac-Toe and Gomoku show that the proposed tree policy is more efficient than other competing methods.

1 INTRODUCTION

Monte Carlo Tree Search (MCTS) is a popular tree-based search strategy within the framework of reinforcement learning (RL), which estimates the optimal value of a state and action by building a tree with Monte Carlo simulation. It has been widely used in sequential decision makings, including scheduling problems, inventory, production management, and real-world games, such as Go, Chess, Tic-tac-toe and Chinese Checkers. See Browne et al. (2012), Fu (2018) and Świechowski et al. (2022) for thorough overviews. MCTS uses little or no domain knowledge and self learns by running more simulations. Many variations have been proposed for MCTS to improve its performance. In particular, deep neural networks are combined into MCTS to achieve a remarkable success in the game of Go (Silver et al. 2016; Silver et al. 2017).

A basic MCTS is to build a game tree from the root node in an incremental and asymmetric manner, where nodes correspond to states and edges correspond to possible state-action pairs. For each round of MCTS, a tree policy is used to find a node from which a roll-out (simulation) is then performed, and nodes in the collected search path is updated according to the received terminal reward. Moves are made during the roll-out by a default policy, which in the simplest case is to make uniform random moves. Different from depth-limited minimax search that needs to evaluate values of intermediate states, only the reward of the terminal state at the end of each roll-out is evaluated in MCTS, which greatly reduces the amount of domain knowledge required. The best action of the root node is selected based on the information collected from simulations after computational budget is exhausted. The tree policy plays a vital role in the success of MCTS since it determines how the tree is built and computational budget is allocated in simulations. The key issue is to balance the exploration of nodes that have not been well sampled yet and the exploitation of nodes that appear to be promising. In our work, we propose a new tree policy to improve the performance of MCTS.

One of the popular tree policies in MCTS is the Upper Confidence Bounds for Trees (UCT) algorithm, which is proposed by applying the Upper Confidence Bound (UCB1) algorithm (Auer et al. 2002)—originally designed for stochastic multi-armed bandit (MAB) problems—to each node of the tree (Kocsis and Szepesvári 2006; Kocsis et al. 2006). Stochastic MAB is a well-known sequential decision problem in which the goal is to maximize the expected total reward in finite rounds by choosing amongst finitely many actions (also known as arms of slot machines in the MAB literature) to sample. There are other variants of bandit-based methods developed for the tree policy. Auer et al. (2002) introduce UCB1-Tuned in order to tune the bounds of UCB1 more finely. Tesauro et al. (2010) suggest a Bayesian framework inspired by its more accurate estimation of values and uncertainties of nodes under limited computational budget. Teytaud and Flory (2011) employ the Exploration-Exploitation with Exponential weights in conjunction with UCT to deal with partially observable games with simultaneous moves. Mansley et al. (2011) combine the Hierarchical Optimistic Optimisation into the roll-out planning, overcoming the limitation of UCT for a continuous decision space. Teraoka et al. (2014) propose a tree policy by selecting the node with the largest confidence interval inspired by the Best Arm Identification (BAI) problem in the MAB literature (Bubeck et al. 2012), and Kaufmann and Koolen (2017) further extend their results to a tighter upper bound. However, both tree policies are pure exploration policies and only developed for the min-max game trees.

Although the goal in MCTS is very similar to the MAB problem, i.e., choosing an action at given state with the best average reward, their setups have many differences. Stochastic rewards are collected at all rounds in MAB, whereas in MCTS, the reward of the goal is collected only at the end of the algorithm. Most bandit-based methods assume that rewards are bounded and known—typically assumed to be $[0,1]$—however, a more general tree search problem has an unknown and unbounded range of values of nodes. A common objective function of bandit-based methods is the cumulative regret, i.e., the expected sum of difference between the performance of the best arm and that of the chosen arm for sampling. Li et al. (2021) show that the algorithms designed to minimize regret tend to discourage exploration. In addition to the differences mentioned above, most bandit-based tree policies only consider the average value and the number of visits of nodes, which do not utilize other available information such as variances. These findings lead us to formulate the tree policy as a statistical ranking and selection (R&S) problem (Chen and Lee 2011; Powell and Ryzhov 2012) that has been actively studied in simulation optimization. In statistical R&S, the goal is to efficiently allocate limited computational budget to finitely many actions (also known as alternatives in the R&S literature) so that the probability of correct selection (PCS) for the best action can be maximized. The samples for any action are usually assumed to be independent and identically Gaussian distributed with known variances, and a reward is collected after computational budget is exhausted. Despite the same goal of BAI and R&S, different assumptions on distributions of samples are made. In particular, the former assumes samples to be bounded or sub-Gaussian distributed.

In our work, we aim to maximize the PCS for the optimal action at the root node of the tree. We propose a dynamic sampling tree policy by applying the Asymptotically Optimal Allocation Policy (AOAP) algorithm (Peng et al. 2018), which is originally designed for statistical R&S problems. AOAP is a myopic sampling procedure that maximizes a value function approximation one-step look ahead. The closest work to our paper is Li et al. (2021), where they propose a tree policy by applying the Optimal Computing Budget Allocation (OCBA) algorithm (Chen et al. 2000; Chen and Lee 2011) to each node of the tree. The key algorithmic differences from ours lie in: OCBA is developed based on a static optimization problem and is designed to reach a good asymptotic behavior, whereas AOAP is derived in a stochastic dynamic programming framework that can capture the finite-sample behavior of a sampling policy. To implement OCBA in a fully sequential manner, they combine it with a "most starving" sequential rule. Our proposed tree policy removes the known and bounded assumption of the node value, and balances exploration and exploitation to efficiently identify the optimal action. We demonstrate the efficiency of our new tree policy through numerical experiments on Tic-tac-toe and Gomoku.

The rest of the paper is organised as follows. Section 2 formulates the proposed problem. The new tree policy and convergence results are proposed in Section 3. Section 4 provides numerical results. The last section concludes the paper.

2 PROBLEM FORMULATION

We consider the setup of a finite-horizon discrete-time Markov decision process (MDP). An MDP is described by a four-tuple $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R})$ with a horizon length H, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{P}\equiv\mathcal{P}(s'|s,a)$ is the Markovian transition kernel, $\mathcal{R}\equiv\mathcal{R}(s,a)$ is a random bounded reward function. The random reward can be discrete (win/draw/loss), continuous or a vector of reward values relative to each agent for more complex multi-agent domains. We assume that $\mathcal{A}$ and $\mathcal{S}$ are finite sets and $\mathcal{P}$ is deterministic, i.e., $\mathcal{P}\equiv\mathcal{P}(s'|s,a)\in\{0,1\}$, $\forall s,s'\in\mathcal{S}$, $a\in\mathcal{A}$. The assumption of deterministic transition is reasonable since traditional MCTS is introduced in the context of deterministic games with a tree representation. At each stage, the system is in state $s\in\mathcal{S}$. After taking an action $a\in\mathcal{A}$, the state transits to next state $s'\in\mathcal{S}$ and an immediate reward is generated according to $\mathcal{R}(s,a)$. A stationary policy $\pi(a|s)$ specifies the probability of performing action $a\in\mathcal{A}$ given current state $s\in\mathcal{S}$. The value function for each state $s\in\mathcal{S}$ under policy π is defined as $V^{\pi}(s)=\mathbb{E}_{\pi}\left[\sum_{t=0}^{H-1}\mathcal{R}(s_t,a_t)\middle|s_0=s\right]$. The state-action value function is defined as $Q^{\pi}(s,a)=\mathbb{E}_{\pi}\left[\sum_{t=0}^{H-1}\mathcal{R}(s_t,a_t)\middle|s_0=s,a_0=a\right]$. The optimal value function under the optimal policy π^* is defined as $V^*(s)=V^{\pi^*}(s)=\sup_{\pi}V^{\pi}(s)$, $\forall s\in\mathcal{S}$. The following Bellman equation holds: $V^*(s)=\max_{a\in\mathcal{A}}[\mathbb{E}[\mathcal{R}(s,a)]+V^*(s')]=\max_{a\in\mathcal{A}}Q(s,a)$, where s' is the next state reached by applying action a on state s.

For the tree search problem, let $\mathbf{s}_i\overset{\Delta}{=}(s,i)\in\mathcal{S}$ and $\mathbf{a}_i\overset{\Delta}{=}(a,i)\in\mathcal{A}$ be a state $s\in\mathcal{S}$ and an action $a\in\mathcal{A}$ at depth i, $0\le i<H$, respectively. We model the best action identification for every explored state node in the tree policy of MCTS as separate R&S problems. All actions of current state node $\mathbf{s}_i$ are treated as alternatives. The optimal value of state node $V_i^*(\mathbf{s}_i)$ is unknown. Each state-action pair has an unknown value $Q_i(\mathbf{s}_i,\mathbf{a}_i)$, $0\le i<H-1$, which is estimated by random samples $(\widehat{Q}_i^1(\mathbf{s}_i,\mathbf{a}_i),\widehat{Q}_i^2(\mathbf{s}_i,\mathbf{a}_i),\cdots,\widehat{Q}_i^{N_{i+1}(\mathbf{s}_{i+1})}(\mathbf{s}_i,\mathbf{a}_i))$, where $N_{i+1}(\mathbf{s}_{i+1})=\sum_{\ell=1}^{t}\mathbb{1}\left(\mathbf{s}_{i+1}\in\mathcal{P}^{\ell}\right)\le T$ is the number of visits to the next state $\mathbf{s}_{i+1}$ after taking action $\mathbf{a}_i$ at state $\mathbf{s}_i$ in $1\le t\le T$ roll-outs, $\mathcal{P}^{\ell}$ is the search path collected at the ℓ-th roll-out, T is total roll-outs or simulations (also known as the number of total simulation budget in the R&S literature), and $\mathbb{1}(\cdot)$ is an indicator function that equals to 1 when the event in the bracket is true and equals to 0 otherwise. We assume that $\widehat{Q}_i^{\ell}$, $0\le i<H-1$, $1\le\ell\le T$ are independent and identically distributed normal random variables, i.e., $\widehat{Q}_i^{\ell}\sim\mathcal{N}(Q_i,\sigma_i^2)$ with a known state-action variance σ_i^2, where we suppress $(\mathbf{s}_i,\mathbf{a}_i)$ for simplicity of notation. The sample variance is used as a plug-in for σ_i^2 in practice, i.e., $\widehat{\sigma}_i^2=\frac{1}{N_{i+1}(\mathbf{s}_{i+1})-1}\sum_{\ell=1}^{N_{i+1}(\mathbf{s}_{i+1})}(\widehat{Q}_i^{\ell}-\bar{Q}_i)^2$, where $\bar{Q}_i=\frac{1}{N_{i+1}(\mathbf{s}_{i+1})}\sum_{\ell=1}^{N_{i+1}(\mathbf{s}_{i+1})}\widehat{Q}_i^{\ell}$ is sample mean. Under a Bayesian framework, we assume the prior distribution of Q_i is a conjugate prior of the sampling distribution of $\widehat{Q}_i^{\ell}$, which is also a normal distribution $\mathcal{N}(Q_i^{(0)},(\sigma_i^{(0)})^2)$. Then the posterior distribution of Q_i is $\mathcal{N}(Q_i^{(t)},(\sigma_i^{(t)})^2)$, $0\le i<H-1$, $1\le t\le T$, with posterior state-action variance

$$\left(\sigma_i^{(t)}(\mathbf{s}_i,\mathbf{a}_i)\right)^2=\left(\frac{1}{\left(\sigma_i^{(0)}(\mathbf{s}_i,\mathbf{a}_i)\right)^2}+\frac{N_{i+1}(\mathbf{s}_{i+1})}{\sigma_i^2(\mathbf{s}_i,\mathbf{a}_i)}\right)^{-1}, \tag{1}$$

and posterior state-action mean

$$Q_i^{(t)}(\mathbf{s}_i,\mathbf{a}_i)=\left(\sigma_i^{(t)}(\mathbf{s}_i,\mathbf{a}_i)\right)^2\left(\frac{Q_i^{(0)}(\mathbf{s}_i,\mathbf{a}_i)}{\left(\sigma_i^{(0)}(\mathbf{s}_i,\mathbf{a}_i)\right)^2}+\frac{N_{i+1}(\mathbf{s}_{i+1})\bar{Q}_i(\mathbf{s}_i,\mathbf{a}_i)}{\sigma_i^2(\mathbf{s}_i,\mathbf{a}_i)}\right)^{-1}. \tag{2}$$

Note that if $\sigma_i^{(0)}(\mathbf{s}_i,\mathbf{a}_i)\to\infty$, then $Q_i^{(t)}(\mathbf{s}_i,\mathbf{a}_i)=\bar{Q}_i(\mathbf{s}_i,\mathbf{a}_i)$ and $(\sigma_i^{(t)}(\mathbf{s}_i,\mathbf{a}_i))^2=\sigma_i^2(\mathbf{s}_i,\mathbf{a}_i)/N_{i+1}(\mathbf{s}_{i+1})$ and such a case is called uninformative. We aim to identify the best action that achieves the highest state-action value at the initial state $\mathbf{s}_0$, i.e., finding $\mathbf{a}_0^* = \arg\max_{\mathbf{a}_0 \in \mathcal{A}_{\mathbf{s}_0}} Q_0(\mathbf{s}_0, \mathbf{a}_0)$, where $\mathcal{A}_{\mathbf{s}_0}$ is the set of actions at state $\mathbf{s}_0$. A correct selection of the best action occurs when $(\mathbf{a}_0^{(T)})^* = \mathbf{a}_0^*$, where $(\mathbf{a}_0^{(T)})^* = \arg\max_{\mathbf{a}_0 \in \mathcal{A}_{\mathbf{s}_0}} Q_0^{(T)}(\mathbf{s}_0, \mathbf{a}_0)$ is the estimated best action that achieves the highest posterior mean at the initial state $\mathbf{s}_0$ after T roll-outs. The PCS for selecting $\mathbf{a}_0^*$ can be expressed as

$$\mathrm{PCS} = \Pr\left\{ \bigcap_{a \in \mathcal{A}_{\mathbf{s}_0},\, a \neq (\mathbf{a}_0^{(T)})^*} Q_0^{(T)}(\mathbf{s}_0, (\mathbf{a}_0^{(T)})^*) > Q_0^{(T)}(\mathbf{s}_0, a) \right\} .$$

We aim to find an efficient dynamic sampling tree policy such that the PCS can be maximized. Compared with minimizing the expected cumulative regret in the canonical MAB problem, maximizing PCS results in an allocation of limited computational budget in a way that optimally balances exploration and exploitation. Based on the information collected from simulations, sampling policy $\mathbf{A}_T(\cdot) \triangleq (\mathbb{A}_1(\cdot), \mathbb{A}_2(\cdot), \cdots, \mathbb{A}_T(\cdot))$ is a sequence of mappings, where $\mathbb{A}_t(\cdot) \in \{\mathbf{a}_0^1, \mathbf{a}_0^2, \cdots, \mathbf{a}_0^{|\mathcal{A}_{\mathbf{s}_0}|}\}$ allocates the t-th computational budget to an action of the initial state based on the information $\mathcal{E}_{t-1}$ collected throughout the first $(t-1)$ roll-outs, where $|\cdot|$ denotes the cardinality of a set. The expected payoff for a dynamic sampling tree policy can be recursively defined in a stochastic dynamic programming problem by

$$\mathcal{V}_T(\mathcal{E}_T; \mathbf{A}_T) \triangleq \Pr\left\{ \bigcap_{a \in \mathcal{A}_{\mathbf{s}_0},\, a \neq (\mathbf{a}_0^{(T)})^*} Q_0^{(T)}(\mathbf{s}_0, (\mathbf{a}_0^{(T)})^*) > Q_0^{(T)}(\mathbf{s}_0, a) \middle| \mathcal{E}_T \right\} ,$$

and for $0 \le t < T$,

$$\mathcal{V}_t(\mathcal{E}_t; \mathbf{A}_T) \triangleq \mathbb{E}\left[\mathcal{V}_{t+1}\left(\mathcal{E}_t \cup \left\{ \widehat{Q}_0^{(N_1(\mathbf{s}_1^i)+1)}(\mathbf{s}_0, \mathbf{a}_0^i) \right\}; \mathcal{A}_T(\cdot) \right) \middle| \mathcal{E}_t \right] \Bigg|_{\mathbf{a}_0^i = \mathbb{A}_{t+1}(\mathcal{E}_t)} ,$$

where $\widehat{Q}_0^{(N_1(\mathbf{s}_1^i)+1)}(\mathbf{s}_0, \mathbf{a}_0^i)$ is the $(N_1(\mathbf{s}_1^i)+1)$-th sample for allocated action $\mathbf{a}_0^i \in \mathcal{A}_{\mathbf{s}_0}$. Then an optimal dynamic sampling tree policy can be defined as the solution of the stochastic dynamic programming problem: $\mathbf{A}_T^* \triangleq \arg\max_{\mathbf{A}_T} \mathcal{V}_0(\mathcal{E}_0; \mathbf{A}_T)$, where $\mathcal{E}_0$ contains the prior information. Such a stochastic dynamic programming problem can be viewed as a MDP, and then the optimality condition of a dynamic sampling tree policy is governed by the Bellman equation of the MDP. However, solving such a MDP typically suffers from curse-of-dimensionality. In the R&S literature, Peng et al. (2018) find a suitable value function approximation (VFA) for the Bellman equations and use a further approximation for the VFA, which leads to the so-called AOAP algorithm that maximizes a VFA one-step look ahead. Inspired by their work, we propose a tree policy by applying the AOAP algorithm to each node of the tree, leading to a dynamic sampling tree policy for MCTS.

3 A NEW TREE POLICY

In this section, we first briefly describe the AOAP algorithm under the tree search setup. Then we propose a new tree policy for MCTS that finds the best action at each state node.

In the tree policy of MCTS, for each visited state node $\mathbf{s}_i$ in the search path at the t-th roll-out, the AOAP algorithm first identifies the action with the largest posterior state-action mean $(\mathbf{a}_i^{(t)})^* = \arg\max_{\mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}} Q_i^{(t)}(\mathbf{s}_i, \mathbf{a}_i)$, and then calculates the following equations:

$$\widetilde{\mathcal{V}}_t\left(\mathbf{s}_i, (\mathbf{a}_i^{(t)})^*\right) = \min_{\mathbf{a}_i \neq (\mathbf{a}_i^{(t)})^*} \frac{\left(Q_i^{(t)}\left(\mathbf{s}_i, (\mathbf{a}_i^{(t)})^*\right) - Q_i^{(t)}(\mathbf{s}_i, \mathbf{a}_i)\right)^2}{\left(\sigma_i^{(t+1)}\left(\mathbf{s}_i, (\mathbf{a}_i^{(t)})^*\right)\right)^2 + \left(\sigma_i^{(t)}(\mathbf{s}_i, \mathbf{a}_i)\right)^2} , \tag{3}$$

and for $\mathbf{a}_i, \widetilde{\mathbf{a}}_i \neq (\mathbf{a}_i^{(t)})^*$, $\mathbf{a}_i, \widetilde{\mathbf{a}}_i \in \mathcal{A}_{\mathbf{s}_i}$,

$$\widetilde{\mathcal{V}}_t(\mathbf{s}_i, \mathbf{a}_i) = \min\left\{ \frac{\left(Q_i^{(t)}\left(\mathbf{s}_i, (\mathbf{a}_i^{(t)})^*\right) - Q_i^{(t)}(\mathbf{s}_i, \mathbf{a}_i)\right)^2}{\left(\sigma_i^{(t)}\left(\mathbf{s}_i, (\mathbf{a}_i^{(t)})^*\right)\right)^2 + \left(\sigma_i^{(t+1)}(\mathbf{s}_i, \mathbf{a}_i)\right)^2}, \min_{\widetilde{\mathbf{a}}_i \neq \mathbf{a}_i, (\mathbf{a}_i^{(t)})^*} \frac{\left(Q_i^{(t)}\left(\mathbf{s}_i, (\mathbf{a}_i^{(t)})^*\right) - Q_i^{(t)}(\mathbf{s}_i, \widetilde{\mathbf{a}}_i)\right)^2}{\left(\sigma_i^{(t)}\left(\mathbf{s}_i, (\mathbf{a}_i^{(t)})^*\right)\right)^2 + \left(\sigma_i^{(t)}(\mathbf{s}_i, \widetilde{\mathbf{a}}_i)\right)^2} \right\}, \tag{4}$$

where

$$\left(\sigma_i^{(t+1)}(\mathbf{s}_i, \mathbf{a}_i)\right)^2 = \left(\frac{1}{\left(\sigma_i^{(0)}(\mathbf{s}_i, \mathbf{a}_i)\right)^2} + \frac{N_{i+1}(\mathbf{s}_{i+1}) + 1}{\sigma_i^2(\mathbf{s}_i, \mathbf{a}_i)} \right)^{-1}.$$

After calculating values of $\widetilde{\mathcal{V}}_t(\mathbf{s}_i, \mathbf{a}_i)$, $\forall\, \mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}$, the AOAP algorithm selects the action with the largest $\widetilde{\mathcal{V}}_t(\mathbf{s}_i, \mathbf{a}_i)$, i.e., sample

$$\widehat{\mathbf{a}}_i^{(t)} = \arg\max\nolimits_{\mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}} \widetilde{\mathcal{V}}_t(\mathbf{s}_i, \mathbf{a}_i). \tag{5}$$

The MCTS algorithm using the AOAP as a tree policy is named as AOAP-MCTS. Compared with UCT, the tree policy based on AOAP utilizes posterior means and posterior variances, which incorporate average value, variances and the number of visits of nodes. The proposed tree policy attempts to balance the exploration of nodes with high variances and exploitation of nodes with high state-action values. In implementation, if more than one action has the same maximal posterior state-action mean or has the same value of $\widetilde{\mathcal{V}}_t(\mathbf{s}_i, \mathbf{a}_i)$, the tie can be broken by choosing randomly or the action with the highest $(\sigma_i^{(t)}(\mathbf{s}_i, \mathbf{a}_i))^2 / N_{i+1}(\mathbf{s}_{i+1})$, that is, choosing the action with low frequency of visits and large posterior state-action variance. In addition, notice that we use variance information of a node as a denominator in calculation of both posterior state-action mean and variance, and in order to ensure the variance is positive, a small positive real number ε can be introduced when $\widehat{\sigma}_i^2(\mathbf{s}_i, \mathbf{a}_i) = 0$.

We highlight some major modifications to the canonical MCTS when using AOAP as the tree policy. First, $\bar{Q}_i(\mathbf{s}_i, \mathbf{a}_i)$, $\widehat{\sigma}_i^2(\mathbf{s}_i, \mathbf{a}_i)$ and $N_{i+1}(\mathbf{s}_{i+1})$, $0 \leq i < H-1$ are required to store for each node in the tree. The prior information $Q_i^{(0)}(\mathbf{s}_i, \mathbf{a}_i)$ and $\sigma_i^{(0)}(\mathbf{s}_i, \mathbf{a}_i)$ can be specified and adjusted in implementation. Second, in order to calculate $\widehat{\sigma}_i^2(\mathbf{s}_i, \mathbf{a}_i)$ for each state-action node, each state node $\mathbf{s}_i$ is required to be well-expanded when it is visited, and each state-action pair $(\mathbf{s}_i, \mathbf{a}_i)$, $\forall\, \mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}$ and its corresponding child state-action node $\mathbf{s}_{i+1}$ is required to be added to the search path. Each state-action pair is required to be sampled $n_0 > 1$ times, i.e., a state node is expandable when one of its child nodes is visited less than n_0 times. Third, after receiving the terminal reward Δ_t of the collected search path $\mathcal{P}^t$ at t-th roll-out, all values of nodes in the collected search path are updated in reversed order through: for $0 \leq i < \ell$ and let $\mathbf{s}_\ell$ be the leaf node in $\mathcal{P}^t$,

$$N_{i+1}(\mathbf{s}_{i+1}) \leftarrow N_{i+1}(\mathbf{s}_{i+1}) + 1, \tag{6}$$

$$\widehat{V}_\ell^*(\mathbf{s}_\ell) \leftarrow \frac{N_\ell(\mathbf{s}_\ell) - 1}{N_\ell(\mathbf{s}_\ell)} \hat{V}_\ell^*(\mathbf{s}_\ell) + \frac{1}{N_\ell(\mathbf{s}_\ell)} \Delta_t, \tag{7}$$

$$\widehat{Q}_i^{N_{i+1}(\mathbf{s}_{i+1})}(\mathbf{s}_i, \mathbf{a}_i) = R(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}_{i+1}^*(\mathbf{s}_{i+1}), \tag{8}$$

$$\bar{\mu}_i(\mathbf{s}_i, \mathbf{a}_i) = \bar{Q}_i(\mathbf{s}_i, \mathbf{a}_i), \tag{9}$$

$$\bar{Q}_i(\mathbf{s}_i, \mathbf{a}_i) \leftarrow \frac{N_{i+1}(\mathbf{s}_{i+1}) - 1}{N_{i+1}(\mathbf{s}_{i+1})} \bar{Q}_i(\mathbf{s}_i, \mathbf{a}_i) + \frac{1}{N_{i+1}(\mathbf{s}_{i+1})} \Delta_t, \tag{10}$$

$$\widehat{\sigma}_i^2(\mathbf{s}_i, \mathbf{a}_i) \leftarrow \frac{N_{i+1}(\mathbf{s}_{i+1}) - 1}{N_{i+1}(\mathbf{s}_{i+1})} \widehat{\sigma}_i^2(\mathbf{s}_i, \mathbf{a}_i) + \frac{1}{N_{i+1}(\mathbf{s}_{i+1})} \left(\Delta_t - \bar{Q}_i(\mathbf{s}_i, \mathbf{a}_i)\right)\left(\Delta_t - \bar{\mu}_i(\mathbf{s}_i, \mathbf{a}_i)\right), \tag{11}$$

$$\widehat{V}_i^*(\mathbf{s}_i) \leftarrow \max\nolimits_{\mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}} \bar{Q}(\mathbf{s}_i, \mathbf{a}_i). \tag{12}$$

Algorithm 1 shows the pseudocode of the AOAP-MCTS algorithm. The AOAP-MCTS algorithm is run with T roll-outs from the root state node $\mathbf{s}_0$, after which a game tree is built and the estimated optimal action $(\mathbf{a}_0^{(T)})^*$ is found corresponding to an action of the root node with the highest posterior state-action mean. Notice that since we consider deterministic transitions, the tree is fixed once the root node is chosen. When a node in the tree is visited, the tree policy first determines whether the node is expandable. If there are state-action pairs that are not yet part of the tree, one of those is chosen randomly and added to the tree, and if there are state-action pairs are visited less than n_0 times, one of those is chosen randomly. If all state-action pairs are well-expanded, AOAP is used to find the allocated one. $\mathbf{s}_\ell$ is the node reached during the tree policy stage corresponding to state s at depth ℓ. A simulation is run from $\mathbf{s}_\ell$ according to a default policy, until a terminal node has been reached. The reward Δ_t of the terminal state is then backpropagated to all nodes collected in the search path during this round to update the node statistics.

Algorithm 1 The AOAP-MCTS algoritm

Input: root state node $\mathbf{s}_0$, number of roll-outs T, and algorithmic constants: n_0, ε, $Q_i^{(0)}(\mathbf{s}_i,\mathbf{a}_i)$ and $\sigma_i^{(0)}(\mathbf{s}_i,\mathbf{a}_i)$.
Output: $(\mathbf{a}_0^{(T)})^*$

```
function AOAP-MCTS(s_0, T, n_0, ε, Q_i^(0)(s_i,a_i), σ_i^(0)(s_i,a_i))
    t ← 0
    while t < T do
        s_ℓ ← TREEPOLICY(s_0, n_0, ε, Q_i^(0)(s_i,a_i), σ_i^(0)(s_i,a_i))
        Δ_t ← DEFAULTPOLICY(s_ℓ)
        BACKPROPAGATE(s_ℓ, Δ_t)
        t ← t+1
    end while
    return (a_0^(T))*
end function

function TREEPOLICY(s_i)
    while s_i is nonterminal do
        if s_i is expandable then
            return EXPAND(s_i)
        else
            s_{i+1} ← AOAP(s_i, n_0, ε, Q_i^(0)(s_i,a_i), σ_i^(0)(s_i,a_i))
        end if
    end while
    return s_{i+1}
end function

function EXPAND(s_i)
   choose a_i ∈ untried actions from A_{s_i}
   append a new child s_{i+1} to s_i
    return s_{i+1}
end function

function AOAP(s_i, n_0, ε, Q_i^(0)(s_i,a_i), σ_i^(0)(s_i,a_i))
   by solving equations (1), (2), (3), (4) and (5)
    return allocated action â_i^(t)
end function
```

function DEFAULTPOLICY($\mathbf{s}_\ell$)
    while $\mathbf{s}_j,\ j \geq \ell+1$ is nonterminal **do**
        choose $\mathbf{a}_j \in \mathcal{A}(\mathbf{s}_j)$ uniformly at random
        append a new child $\mathbf{s}_{j+1}$ to $\mathbf{s}_j$
    end while
    return reward Δ_t of the collected search path
end function

function BACKPROPAGATE($\mathbf{s}_\ell$,Δ_t)
    while $\mathbf{s}_i,\ 0 \leq i \leq \ell$ is not null **do**
        update node values through equations (6), (7), (8), (9), (10), (11) and (12)
    end while
end function

We show theoretical results regarding AOAP-MCTS in Proposition 1.

Proposition 1. *The proposed AOAP-MCTS is consistent, i.e.,* $\lim_{T\to\infty}(\mathbf{a}_i^{(T)})^* = a_i^*$, $\lim_{T\to\infty}\widehat{V}_i^*(\mathbf{s}_i) = V_i^*(\mathbf{s}_i)$, $0 \leq i < H$.

At each explored state node in the tree policy, the best action is identified by the AOAP algorithm. As shown in Peng et al. (2018), AOAP is consistent, i.e., every alternative will be sampled infinitely often almost surely as the number of computational budget goes to infinity, so that the best alternative can be definitely selected. Following their analysis, Proposition 1 can be proved by induction. We leave the proof to future work.

4 NUMERICAL EXPERIMENTS

In this section, we conduct numerical experiments to test the performances of different tree policies for MCTS. We apply our proposed algorithm to the games of Tic-tac-toe and Gomoku. The proposed AOAP-MCTS is compared with UCT in Kocsis and Szepesvári (2006), OCBA-MCTS in Li et al. (2021) and TTTS-MCTS, which runs tree policy by the Top-Two Thompson Sampling (TTTS) in Russo (2020). We describe the three tree policies as follows:

- UCT: The policy selects the action with the highest upper confidence bound, i.e.,

$$\widehat{\mathbf{a}}_i^{(t)} = \arg\max_{\mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}} \left\{ \bar{Q}_i(\mathbf{s}_i,\mathbf{a}_i) + C_p\sqrt{2\log N_i(\mathbf{s}_i)/N_{i+1}(\mathbf{s}_{i+1})} \right\} ,$$

  where C_p is the exploration weight. We choose $C_p = 1$ in implementation.
- OCBA-MCTS: The policy solves a set of equations and selects the action that is the most starving, i.e., $\forall \mathbf{a}_i, \widetilde{\mathbf{a}}_i \neq \widehat{\mathbf{a}}_i^*, \mathbf{a}_i, \widetilde{\mathbf{a}}_i \in \mathcal{A}_{\mathbf{s}_i}$, let $\widehat{\mathbf{a}}_i^* = \arg\max_{\mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}} \bar{Q}_i(\mathbf{s}_i,\mathbf{a}_i)$ and $\delta_i(\widehat{\mathbf{a}}_i^*,\mathbf{a}_i) = \bar{Q}_i(\mathbf{s}_i,\widehat{\mathbf{a}}_i^*) - \bar{Q}_i(\mathbf{s}_i,\mathbf{a}_i)$, $\forall \mathbf{a}_i \neq \widehat{\mathbf{a}}_i^*$,

$$\frac{\widetilde{N}_{i+1}(\mathbf{s}_{i+1})}{\widetilde{N}_{i+1}(\widetilde{\mathbf{s}}_{i+1})} = \left(\frac{\sigma_i(\mathbf{s}_i,\mathbf{a}_i)/\delta_i(\widehat{\mathbf{a}}_i^*,\mathbf{a}_i)}{\sigma_i(\mathbf{s}_i,\widetilde{\mathbf{a}}_i)/\delta_i(\widehat{\mathbf{a}}_i^*,\widetilde{\mathbf{a}}_i)} \right)^2 ,$$

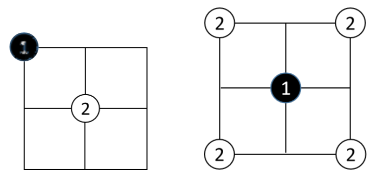
$$\widetilde{N}_{i+1}(\widetilde{\mathbf{s}}_{i+1}^*) = \sigma_i(\mathbf{s}_i,\widehat{\mathbf{a}}_i^*) \sqrt{\sum_{\mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}, \mathbf{a}_i \neq \widehat{\mathbf{a}}_i^*} \frac{\left(\widetilde{N}_{i+1}(\mathbf{s}_{i+1})\right)^2}{\sigma_i^2(\mathbf{s}_i,\mathbf{a}_i)}} ,$$

$$\widehat{\mathbf{a}}_i^{(t)} = \arg\max_{\mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}} \left(\widetilde{N}_{i+1}(\mathbf{s}_{i+1}) - N_{i+1}(\mathbf{s}_{i+1}) \right) .$$

- TTTS-MCTS: The policy first samples $\widehat{Q}_i^1(\mathbf{s}_i,\mathbf{a}_i)$, $\forall \mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}$ from $\mathcal{N}(Q_i^{(t)}(\mathbf{s}_i,\mathbf{a}_i), (\sigma_i^{(t)}(\mathbf{s}_i,\mathbf{a}_i))^2)$, and finds $\widehat{\mathbf{a}}_{i1}^{(t)} = \arg\max_{\mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}} \widehat{Q}_i^1(\mathbf{s}_i,\mathbf{a}_i)$. Then the policy samples $\widehat{Q}_i^2(\mathbf{s}_i,\mathbf{a}_i)$, $\forall \mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}$ from the same distribution until $\widehat{\mathbf{a}}_{i1}^{(t)} \neq \widehat{\mathbf{a}}_{i2}^{(t)}$, where $\widehat{\mathbf{a}}_{i2}^{(t)} = \arg\max_{\mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}} \hat{Q}_i^2(\mathbf{s}_i, \mathbf{a}_i)$. The allocated action is determined by randomly choosing from $\widehat{\mathbf{a}}_{i1}^{(t)}$ and $\widehat{\mathbf{a}}_{i2}^{(t)}$. Since the second stage of the policy can be time-consuming when the action space is large, we truncate it with 10 rounds in implementation, i.e., if $\widehat{\mathbf{a}}_{i2}^{(t)}$ can not be found in 10 rounds, we determine $\widehat{\mathbf{a}}_{i2}^{(t)}$ by the second largest value of $\widehat{Q}_i^1(\mathbf{s}_i, \mathbf{a}_i)$.

Experiment 1: Tic-tac-toe Tic-tac-toe is a game played on a three-by-three board by two players, who alternately place the marks 'X' and 'O' in one of the nine spaces in the board. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner. If both players act optimally, the game will always end in a draw.

Experiment 1.1: Precision In this experiment, we focus on the precision of MCTS in finding the optimal move under different tree policies. The effectiveness of a policy is measured by PCS. Given a place marked by Player 1, we apply different tree policies to identify the optimal move for Player 2. Figure 1 shows two board setups, where we use black and white to represent 'X' and 'O', respectively, for ease of presentation.



(a) Setup 1　　(b) Setup 2

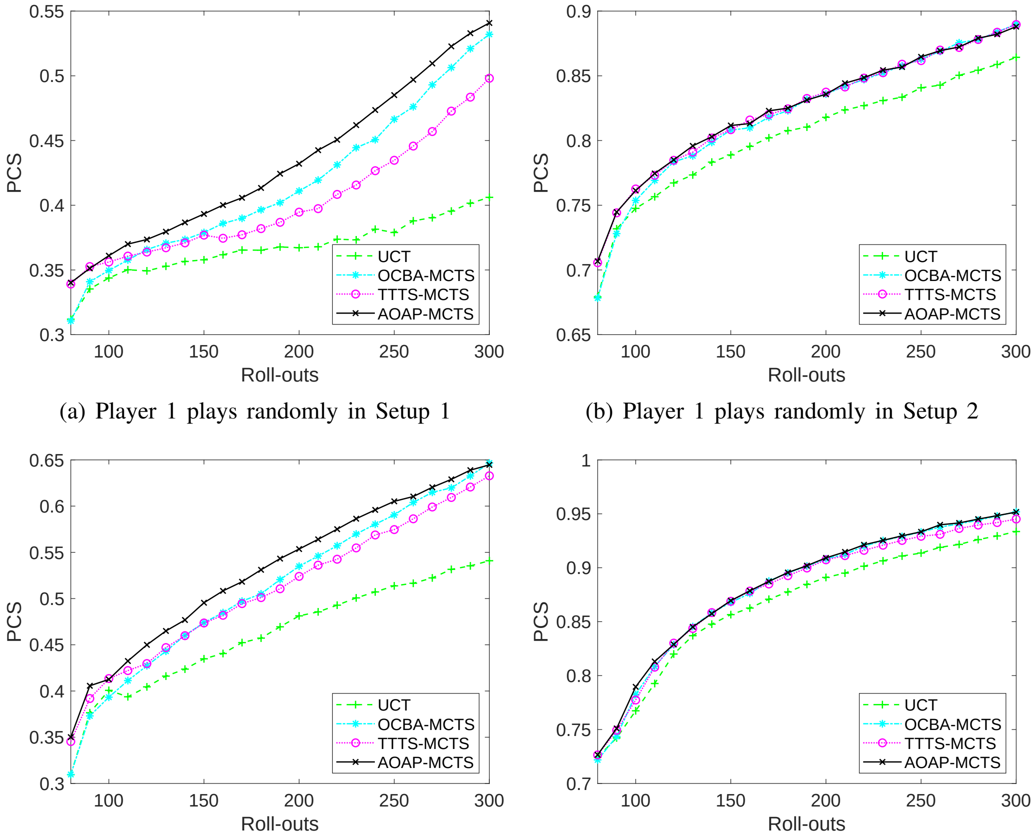
Figure 1: Tic-tac-toe board setup in Experiment 1.1.

The optimal move for Player 2 is unique in setup 1, whereas any of the four moves in the corner space is optimal for Player 2 in setup 2. The setup 2 is an easier setting since Player 2 has a 50% chance of choosing an optimal move even if choosing randomly. At the end of the game, if Player 2 wins, the reward of terminal state is 1, and if it leads to a draw, the reward is 0.5; otherwise, the reward is 0. We consider two policies for Player 1 under both setups: one is playing randomly, i.e., with equal probability to mark any feasible space, the other is playing UCT, which chooses the move that minimizes the lower confidence bound, i.e.,

$$\widehat{\mathbf{a}}_i^{(t)} = \arg\min_{\mathbf{a}_i \in \mathcal{A}_{\mathbf{s}_i}} \left\{ \bar{Q}_i(\mathbf{s}_i, \mathbf{a}_i) - C_p \sqrt{2 \log N_i(\mathbf{s}_i) / N_{i+1}(\mathbf{s}_{i+1})} \right\} ,$$

in order to minimize the reward of Player 2. We set $n_0 = 10$ for all policies, and set $\varepsilon = 10^{-5}$, $Q_i^{(0)}(\mathbf{s}_i, \mathbf{a}_i) = 0$, $\sigma_i^{(0)}(\mathbf{s}_i, \mathbf{a}_i) = 10$ for AOAP-MCTS. The PCS for the optimal move of Player 2 are estimated based on 100,000 independent macro experiments. We plot the PCS of all policies under each setup as a function of the number of roll-outs, ranging from 80 to 300. The results are shown in Figure 2. We can see that AOAP-MCTS performs the best among all tree policies and it has a better performance when the number of roll-outs is relatively low. The policies based on R&S (i.e., AOAP-MCTS and OCBA-MCTS) have better performances than the policies based on MAB (i.e., UCT and TTTS-MCTS) as the number of roll-outs increases. TTTS-MCTS has a better performance than OCBA-MCTS when the number of roll-outs is low. The performances of all policies become comparable as the number of roll-outs grows. AOAP-MCTS achieves 33.2%, 2.8%, 19.2% and 1.9% better than UCT in (a)-(d) settings, respectively. The gap of policies is smaller when Player 1 plays UCT, since Player 1 has a better chance to take optimal action in this case. Although the differences between all policies in Setup 2 are not as significant as that in Setup 1, AOAP still performs the best.

Experiment 1.2: Win-draw-lose In this experiment, we focus on the number of win, draw and lose when Player 1 plays against with Player 2. Both players play randomly or one of the four tree policies. Since opponent's policies are unknown, the player's policy is trained against a random or UCT opponent. The algorithmic constants are the same as in Experiment 1, except $Q_i^{(0)}(\mathbf{s}_i, \mathbf{a}_i) = 1$. The number of roll-outs



(a) Player 1 plays randomly in Setup 1

(b) Player 1 plays randomly in Setup 2

(c) Player 1 plays UCT in Setup 1

(d) Player 1 plays UCT in Setup 2

Figure 2: Comparison of PCS of 4 tree policies in Experiment 1.1.

to determine a move at a state is set to 200. The number of win, draw and lose of Player 1 are estimated by 1000 independent rounds. The results are shown in Table 1 and Table 2, where the trivariate vector in each blank comprises of number of win, draw and lose, respectively. The last column of each Table shows the net win of a policy, calculated by the cumulative wins minus the cumulative loses of both players. From Tables 1 and 2, we can see that the net win of AOAP-MCTS is the highest among all policies. OCBA-MCTS has a better performance than TTTS-MCTS and UCT. The net wins of AOAP-MCTS and TTTS-MCTS trained against a UCT opponent are lower than that of AOAP-MCTS and TTTS-MCTS trained against a random opponent, showing that both policies are relatively conservative when the opponent has a better chance to take an optimal action.

Table 1: The number of win, draw and lose in Experiment 1.2, where each policy is trained against a random opponent.

Player 2 / Player 1	Random	UCT	OCBA-MCTS	TTTS-MCTS	AOAP-MCTS	Net Win
Random	(526,449,25)	(425,552,23)	(504,475,21)	(454,520,26)	(469,513,18)	-483
UCT	(604,391,5)	(446,545,9)	(404,588,8)	(319,671,10)	(408,586,6)	-142
OCBA-MCTS	(574,415,11)	(439,551,10)	(522,467,11)	(497,491,12)	(527,463,10)	156
TTTS-MCTS	(557,431,12)	(455,538,7)	(484,506,10)	(501,490,9)	(402,589,9)	135
AOAP-MCTS	(555,430,15)	(583,403,14)	(493,499,8)	(513,477,10)	(469,522,9)	334

Table 2: The number of win, draw and lose in Experiment 1.2, where each policy is trained against a UCT opponent.

Player 1 \ Player 2	Random	UCT	OCBA-MCTS	TTTS-MCTS	AOAP-MCTS	Net Win
Random	(434,545,21)	(307,658,35)	(291,672,37)	(288,689,23)	(288,693,19)	-633
UCT	(476,507,17)	(340,649,11)	(278,713,9)	(258,730,12)	(204,787,9)	-2
OCBA-MCTS	(415,579,6)	(314,682,4)	(286,703,11)	(312,678,10)	(270,721,9)	227
TTTS-MCTS	(416,572,12)	(297,696,7)	(279,710,11)	(293,692,15)	(277,715,8)	93
AOAP-MCTS	(435,551,14)	(307,685,8)	(278,708,14)	(341,643,16)	(279,704,17)	315

Experiment 2: Gomoku We consider a game played on a larger board, which is called Gomoku. It is played on a fifteen-by-fifteen board by two players. Players alternate turns to place a stone of their color on an empty intersection. Black plays first. The winner is the first player to form an unbroken chain of only five stones horizontally, vertically, or diagonally. We restrict the board size to eight-by-eight for ease of computation. We focus on the number of win, draw and lose when Player 1 plays against with Player 2. The setups of the experiment are the same as in Experiment 1.2. The algorithmic constants are the same as in Experiment 1.2, except $(\sigma_i^{(0)}(\mathbf{s}_i,\mathbf{a}_i))^2 = 36$. The number of roll-outs to determine a move at a state is set to 2000. The results are shown in Table 3 and Table 4. We can see that AOAP-MCTS performs the best among all policies, and OCBA-MCTS has a better performance than TTTS-MCTS and UCT. Compared with Experiment 1.2, the advantage of AOAP-MCTS is more significant in the larger board.

Table 3: The number of win, draw and lose in Experiment 2, where each policy is trained against a random opponent.

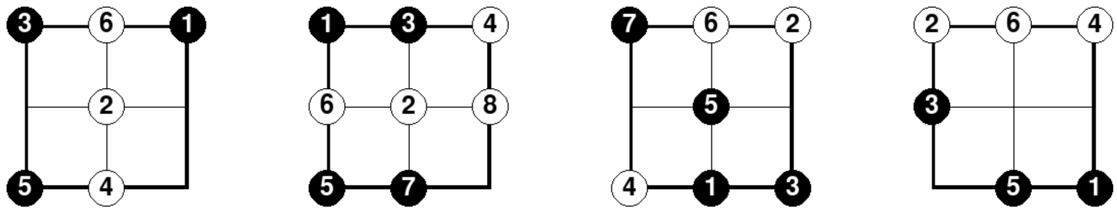
Player 1 \ Player 2	Random	UCT	OCBA-MCTS	TTTS-MCTS	AOAP-MCTS	Net Win
Random	(501,0,499)	(392,5,603)	(401,5,594)	(481,4,515)	(450,13,537)	-1549
UCT	(540,4,456)	(472,14,514)	(412,4,584)	(471,11,518)	(471,0,529)	-374
OCBA-MCTS	(660,7,333)	(580,10,610)	(560,5,435)	(521,12,467)	(432,14,554)	635
TTTS-MCTS	(660,11,329)	(531,5,464)	(480,14,506)	(511,0,489)	(542,1,457)	377
AOAP-MCTS	(641,0,359)	(571,13,416)	(591,3,406)	(552,3,445)	(551,2,447)	635

Table 4: The number of win, draw and lose in Experiment 2, where each policy is trained against a UCT opponent.

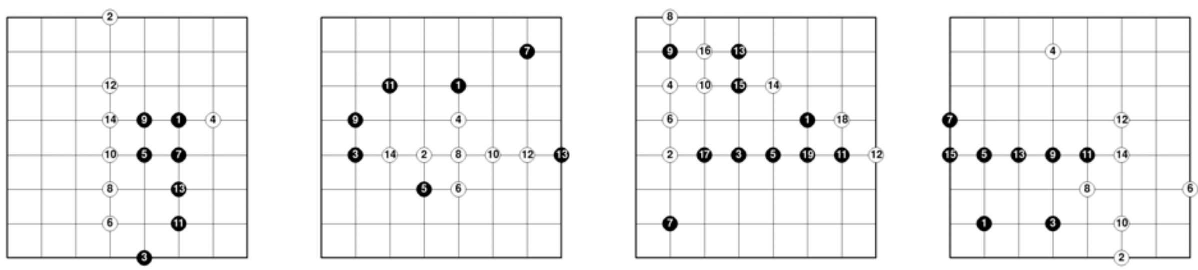
Player 1 \ Player 2	Random	UCT	OCBA-MCTS	TTTS-MCTS	AOAP-MCTS	Net Win
Random	(500,7,493)	(510,9,481)	(432,49,519)	(331,4,665)	(211,7,782)	-1814
UCT	(481,2,511)	(521,12,477)	(421,3,576)	(341,17,652)	(350,6,644)	-1619
OCBA-MCTS	(610,11,379)	(661,5,334)	(511,7,482)	(540,5,455)	(481,3,516)	901
TTTS-MCTS	(691,7,302)	(550,3,447)	(501,16,483)	(510,3,487)	(361,9,630)	717
AOAP-MCTS	(621,19,360)	(681,8,311)	(661,9,330)	(541,2,457)	(531,6,463)	2215

Experiment 3: Behaviors In this experiment, we analyze the behaviors of four tree policies by observing the boards at the terminal state in games of Tic-tac-toe and Gomoku. Some terminal boards are shown in Figure 3. We find that the performance of UCT does not vary much as the game goes on. The behavior of TTTS-MCTS is similar to UCT, but it has a better performance than UCT. OCBA-MCTS tends to have a better performance at the beginning of the game, but it sometimes fails to intercept the opponent's moves

in time, leading to a failure. AOAP-MCTS can aggressively intercept opponents' moves or greedily win adaptively. Although it sometimes do not choose the optimal action at the beginning of the game, its performance becomes better as the game goes on.



(a) OCBA; **AOAP** (b) OCBA; **UCT** (c) **AOAP**; TTTS (d) UCT; **AOAP**



(e) Random; **AOAP** (f) TTTS; **AOAP** (g) **AOAP**; OCBA (h) **AOAP**; UCT

Figure 3: Boards of Tic-tac-toe and Gomoku at the terminal state. (a)-(d): boards of Tic-tac-toe; (e)-(h): boards of Gomoku. The name before and after the semicolon are policies taken by Player 1 and Player 2, respectively, where we omit the 'MCTS' in the name for ease of presentation. The name with the bold font is the winner.

5 CONCLUSION

The paper studies the tree policy for Monte Carlo Tree Search. We formulate the tree policy in MCTS as a ranking and selection problem, and propose an efficient dynamic sampling tree policy named as AOAP-MCTS, which maximizes the probability of correct selection of the best action at the root state. Numerical experiments demonstrate that AOAP-MCTS is more efficient than other tested tree policies.

A rigorous theoretical analysis of MCTS deserves future work. It is non-trivial due to the hierarchical and iterative structure of tree search, which induces complicated probabilistic dependency between nodes within a sub-tree. Future research also includes using AOAP-MCTS in sequential decision makings and large-scale real-world games. The normal assumption of samples for $Q(\mathbf{s},a)$ deserves verification. How to guarantee sampling precision under a limited computational budget could also be future work.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 71901003 and 72022001.

REFERENCES

Auer, P., N. Cesa-Bianchi, and P. Fischer. 2002. "Finite-time Analysis of the Multiarmed Bandit Problem". *Machine Learning* 47(2):235–256.

Browne, C. B., E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. 2012. "A Survey of Monte Carlo Tree Search Methods". *IEEE Transactions on Computational Intelligence and AI in games* 4(1):1–43.

Bubeck, S., N. Cesa-Bianchi et al. 2012. "Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems". *Foundations and Trends® in Machine Learning* 5(1):1–122.

Chen, C.-H., and L. H. Lee. 2011. *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*, Volume 1. Singapore: World Scientific.

Chen, C.-H., J. Lin, E. Yücesan, and S. E. Chick. 2000. "Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization". *Journal of Discrete Event Dynamic Systems* 10(3):251–270.

Fu, M. C. 2018. "Monte Carlo Tree Search: A tutorial". In *2018 Winter Simulation Conference (WSC)*, edited by M. Rabe, A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 222–236. Gothenburg, Sweden: Institute of Electrical and Electronics Engineers, Inc.

Kaufmann, E., and W. M. Koolen. 2017. "Monte-Carlo Tree Search by Best Arm Identification". In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, edited by U. v. Luxburg, I. Guyon, S. Bengio, H. Wallach, and R. Fergus, 4904–4913. Long Beach, California: Curran Associates, Inc.

Kocsis, L., and C. Szepesvári. 2006. "Bandit Based Monte-Carlo Planning". In *17th European Conference on Machine Learning*, edited by J. Frnkranz, T. Scheffer, and M. Spiliopoulou, 282–293. Berlin, Germany: Springer.

Kocsis, L., C. Szepesvári, and J. Willemson. 2006. *Improved Monte-Carlo Search*. Estonia: University of Tartu.

Li, Y., M. C. Fu, and J. Xu. 2021. "An Optimal Computing Budget Allocation Tree Policy for Monte Carlo Tree Search". *IEEE Transactions on Automatic Control* 67(6):2685–2699.

Mansley, C., A. Weinstein, and M. Littman. 2011. "Sample-based Planning for Continuous Action Markov Decision Processes". In *21st International Conference on Automated Planning and Scheduling*, edited by F. Bacchus, C. Domshlak, S. Edelkamp, and M. Helmert, 335–338. Freiburg, Germany: Association for the Advancement of Artificial Intelligence, Inc.

Peng, Y., E. K. Chong, C.-H. Chen, and M. C. Fu. 2018. "Ranking and Selection as Stochastic Control". *IEEE Transactions on Automatic Control* 63(8):2359–2373.

Powell, W. B., and I. O. Ryzhov. 2012. *"Ranking and Selection"*. In Chapter 4 in Optimal Learning, 71-88: New York: John Wiley and Sons.

Russo, D. 2020. "Simple Bayesian Algorithms for Best-Arm Identification". *Operations Research* 68(6):1625–1647.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al. 2016. "Mastering the Game of Go With Deep Neural Networks and Tree Search". *Nature* 529(7587):484–489.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton et al. 2017. "Mastering the Game of Go Without Human Knowledge". *Nature* 550(7676):354–359.

Świechowski, M., K. Godlewski, B. Sawicki, and J. Mańdziuk. 2022. "Monte Carlo Tree Search: A Review of Recent Modifications and Applications". *Artificial Intelligence Review* 2022:1–66.

Teraoka, K., K. Hatano, and E. Takimoto. 2014. "Efficient Sampling Method for Monte Carlo Tree Search Problem". *IEICE Transactions on Information and Systems* 97(3):392–398.

Tesauro, G., V. Rajan, and R. Segal. 2010. "Bayesian Inference in Monte-Carlo Tree Search". In *Proceedings of Conference on Uncertainty in Artificial Intelligence*, edited by P. Grunwald and P. Spirtes, 580–588. Catalina Island, California: Association for Uncertainty in Artificial Intelligence, Inc.

Teytaud, O., and S. Flory. 2011. "Upper Confidence Trees With Short Term Partial Information". In *Proceedings of the 2011 International Conference on Applications of Evolutionary Computation-Volume Part I*, edited by C. D. Chio, S. Cagnoni, C. Cotta, M. Ebner, and A. Ekárt, 153–162. Torino, Italy: Springer.

AUTHOR BIOGRAPHIES

GONGBO ZHANG is a Ph.D. candidate in the Department of Management Science and Information Systems in Guanghua School of Management at Peking University, Beijing, China. He received the B.S. degree in mathematics and applied mathematics from College of Sciences, Northeastern University, Shenyang, China, in 2018. His research interests include stochastic modeling and analysis, simulation optimization and reinforcement learning. His email address is gongbozhang@pku.edu.cn.

YIJIE PENG is an Associate Professor in Guanghua School of Management at Peking University. His research interests include stochastic modeling and analysis, simulation optimization, machine learning, data analytics, and healthcare. He is a member of INFORMS and IEEE, and serves as an Associate Editor of the Asia-Pacific Journal of Operational Research and the Conference Editorial Board of the IEEE Control Systems Society. His email address is pengyijie@pku.edu.cn.

YILONG XU is pursuing a bachelor degree in the Department of Computer Science at Beijing Jiaotong University. His research interests include reinforcement learning and artificial intelligence. His email address is 18281110@bjtu.edu.cn.