**Submitted to *INFORMS Journal on Computing***
**manuscript (Please, provide the manuscript number!)**

# An Efficient Node Selection Policy for Monte Carlo Tree Search with Neural Networks

Xiaotian Liu
PKU-WUHAN Institute for Artificial Intelligence, Guanghua School of Management, Peking University, Beijing 100871, China; Xiangjiang Laboratory, Changsha, 410000, China, xiaotianliu01@gmail.com

Yijie Peng
PKU-WUHAN Institute for Artificial Intelligence, Guanghua School of Management, Peking University, Beijing 100871, China; Xiangjiang Laboratory, Changsha, 410000, China, pengyijie@pku.edu.cn

Gongbo Zhang
PKU-WUHAN Institute for Artificial Intelligence, Guanghua School of Management, Peking University, Beijing 100871, China; Xiangjiang Laboratory, Changsha, 410000, China, gongbozhang@pku.edu.cn

Ruihan Zhou
PKU-WUHAN Institute for Artificial Intelligence, Guanghua School of Management, Peking University, Beijing 100871, China; Xiangjiang Laboratory, Changsha, 410000, China, rhzhou@stu.pku.edu.cn

Monte Carlo Tree Search (MCTS) has been gaining increasing popularity, and the success of AlphaGo has prompted a new trend of incorporating a value network and a policy network constructed with neural networks into MCTS, namely NN-MCTS. In this work, motivated by the shortcomings of the widely used Upper Confidence Bounds applied to Trees (UCT) policy, we formulate the node selection problem in NN-MCTS as a multi-stage Ranking and Selection (R&S) problem and propose a node selection policy that efficiently allocates a limited search budget to maximize the probability of correctly selecting the best action at the root state. The value network and policy network in NN-MCTS further improve the performance of the proposed node selection policy by providing prior knowledge and guiding the selection of the final action, respectively. Numerical experiments on two board games and an OpenAI task demonstrate that the proposed method outperforms the UCT policy used in AlphaGo Zero and MuZero, implying the potential of constructing node selection policies in NN-MCTS with R&S procedures.

*Key words*: Monte Carlo Tree Search, Node Selection Policy, Neural Networks, Ranking and Selection

## 1. Introduction

Recent years have witnessed the emergence of large-scale decision-making problems in various domains, such as supply chain management, manufacturing, and robotic control. Typically, the challenge of these problems lies in identifying the optimal action in a large action space, which is computationally intractable for most traditional search-based methods. As a simulation-based algorithm, Monte Carlo Tree Search (MCTS) has been gaining increasing popularity for its effectiveness in determining desirable actions with controllable computation costs.

In general, MCTS works by simulating different actions numerous times and determining the optimal action based on the simulation results. The objective of MCTS is to identify the best action to take at the root state within a limited search budget. A node selection policy in MCTS determines the allocation of the search budget and controls the growth of the search tree, which is crucial to the overall performance and efficiency of MCTS. One of the most popular node selection policies is Upper Confidence Bounds applied to Trees (UCT) (Kocsis and Szepesvári 2006), which selects actions based on the Upper Confidence Bounds (UCB) formula. UCT is derived from the stochastic multi-armed bandit (MAB) problem, a sequential decision-making problem aimed at maximizing the expected total reward (or minimizing the cumulative regret) within a finite number of rounds. However, the setup of the MAB problem differs from that of MCTS. In the MAB problem, rewards occur at each round and are assumed to be bounded and known, typically within the range $[0, 1]$. In contrast, the reward in MCTS is either 0 or 1, determined by whether the optimal action is identified when all simulation resources are exhausted, and samples in a rollout of MCTS could be unknown and unbounded. Recently, researchers from the operations research area have considered node selection as a multi-stage ranking and selection (R&S) problem and have extended the Optimal Computing Budget Allocation (OCBA) algorithm (Chen et al. 2000) to a node selection policy called OCBA-MCTS (Li et al. 2021), whose performance is demonstrated to be superior to UCT. The goal of the R&S problem is to minimize a loss function of incorrectly identifying the optimal action. The setup of the R&S problem aligns with that of MCTS. The samples in R&S problems are commonly assumed to be independently and identically Gaussian distributed with known variances, making them more suitable for the general decision-making problems solved by MCTS. A node selection algorithm based on sampling policies developed for R&S problems tends to explore different actions more than UCT and utilizes more sample information, such as sample variances.

The recent development of Deep Learning (DL) and Neural Networks (NNs) has prompted a new trend of incorporating NNs into MCTS to reduce simulation costs and improve performance by using value networks to estimate state payoffs, policy networks to guide action selection, and by focusing the search on high-potential moves, thus improving efficiency and decision-making. The value networks and policy networks are iteratively trained using a DL scheme. We refer to this class of MCTS as NN-MCTS. Two of the most famous examples of NN-MCTS are AlphaGo Zero (Silver et al. 2017) and MuZero (Schrittwieser et al. 2020), which achieve superhuman performance in the board game Go and general challenging decision problems, respectively. Though NN-MCTS has been widely applied to tasks in various domains such as physics (Loeffler et al. 2021) and video delivery (Liu et al. 2023), the node selection policies used in those works, including AlphaGo Zero and MuZero, are all based on UCT—either original UCT or UCT with minor modifications. There is a lack of other well-performing node selection policies in NN-MCTS, and there is also a lack of studies addressing the node selection problem in NN-MCTS using R&S schemes. To fill the above research gaps, our work makes the following contributions.

- We formulate the node selection problem in MCTS as a multi-stage R&S problem, assuming Gaussian and Bernoulli distributions for the state value samples, respectively.
- We extend the Asymptotically Optimal Allocation Procedure (AOAP) (Peng et al. 2018), originally developed for a one-stage R&S problem, to a node selection policy in MCTS, which we call the Asymptotically Optimal Allocation Procedure for Trees (AOAT). The proposed AOAT is proved to be consistent, ensuring that the optimal action for all possible states at each stage can be identified as the search budget goes to infinity.
- We show that both the value network and policy network in NN-MCTS can significantly benefit AOAT. In particular, the value network provides prior knowledge for AOAT, whereas the policy network assists in selecting the final action after all rollouts of NN-MCTS are completed.
- Numerical experiments on two board games with sparse rewards, including Tic-Tac-Toe and Five-In-A-Row (also called Gomoku), as well as on the CartPole-v0 problem with non-sparse rewards provided by OpenAI Gym, demonstrate the superior performance of the proposed method compared to the modified UCT used in AlphaGo Zero and MuZero. In addition, we compare the proposed method with a node selection policy derived from another R&S algorithm, OCBA, and highlight the value of the information provided by the value network and policy network when implementing AOAT, further showcasing the robustness of the proposed method.

The remainder of the paper is organized as follows. Section 2 reviews the relevant literature. In Section 3, we formulate the node selection problem in MCTS as a multi-stage R&S problem and introduce the AOAT algorithm, as well as its integration with both the value network and policy network in NN-MCTS. We present the results of numerical experiments on two board games in Section 4 and on the CartPole-v0 problem in Section 5. The last section concludes the paper.

## 2. Literature Review

In this section, we provide a brief review on two key elements in this work, i.e., MCTS and node selection policies in MCTS.

### 2.1. Monte Carlo Tree Search

MCTS is designed to find optimal or near-optimal actions in decision processes through stochastic simulations. The concept of MCTS, which approximates the optimal state values in Markov Decision Processes (MDPs) with adaptive sampling, is first introduced by Chang et al. (2005). MCTS works iteratively, with each iteration known as a rollout. The total number of rollouts is referred to as the search budget, which controls the total computation costs for conducting MCTS. MCTS constructs a search tree that records all the simulation results obtained from the rollouts. This search tree then serves as the reference for determining the final actions. One rollout of MCTS typically consists of the following four steps.

**Selection:** Starting from the current system state, a node selection policy determines the direction to move down the search tree until a new state that has never been visited before is found.

**Expansion:** The new state is added to the search tree as a new leaf node.

**Simulation:** The decision process starting from the new state is simulated to obtain a payoff for the new state.

**Backpropagation:** The payoff obtained from the simulation step is passed along the tree to update the statistics of nodes on the search path.

Two factors that determine the effectiveness of MCTS are the node selection policy and the simulation method. We review existing node selection policies in the next sub-section. For the simulation method, the most classic approach is to simulate the decision process starting from the new state for numerous steps until it reaches the end, then take the simulation results as the final payoff. Random play or certain heuristics are used to make decisions during the simulation step. The recent development of NNs and DL has contributed to a new method for the simulation step, where a value network is used to estimate the value of the new state, which is taken as the payoff. Simulating with a value network not only saves simulation resources but also improves performance, as a well-trained value network provides a highly accurate approximation of the expected state value. Besides the value network, the policy network is also commonly used to improve MCTS. The policy network is trained to estimate the optimal decision policy and assists in the selection of actions in MCTS. Two of the most famous examples of NN-MCTS, AlphaGo Zero (Silver et al. 2017) and MuZero (Schrittwieser et al. 2020), train Convolution Neural Networks (CNNs) to construct the value network and the policy network. Based on MuZero, Yin et al. (2022) incorporate behavioral cloning and adversarial imitation learning into NN-MCTS and propose a planning-based framework to efficiently imitate expert demonstrations. In this work, we leverage the information provided by the value network and the policy network in the proposed AOAT policy.

## 2.2. Node Selection Policies in MCTS

The node selection policy affects the efficiency of MCTS by controlling the growth direction of the search tree and making the trade-off between exploring potential actions and exploiting intensively visited actions. UCT, the primary approach for node selection in MCTS, is an extension of a particular bandit algorithm, UCB1 (Auer et al. 2002). UCT balances exploration and exploitation of action selection by maximizing artificially specified upper confidence bounds, which combine the action's historical average payoff with an exploration term negatively correlated with the action's visit times. There are also some variants that adapt how UCB1 is applied to MCTS. For this purpose, Auer et al. (2002) introduce UCB1-Tuned to refine the bounds of UCT more precisely to improve exploration. A Bayesian version of UCT developed by Tesauro et al. (2012) provides a better estimation of node values and uncertainties; however, it is computation-intensive. Mansley et al. (2011) incorporate hierarchical optimistic optimization into the rollout of MCTS to extend UCT to a continuous decision space. Teraoka et al. (2014) propose a tree policy that selects the node with the most extensive confidence interval, whereas Kaufmann and Koolen (2017) extend their results to provide a tighter upper bound.

Other than considering the node selection problem from the perspective of MAB, a recent direction is to solve it under the framework of R&S. In MAB problems, rewards are collected at each step, and each reward depends on the probability distribution specific to the pulled arm, whereas in R&S problems, the reward is collected at the end and depends on the underlying probability distribution of all alternatives. The R&S problem shares many similarities with the best-arm identification (BAI) problem proposed in recent years in the MAB literature. The differences between BAI and R&S are that the sampling distributions in BAI are commonly assumed to be bounded or have known variance bounds, and sampling policies derived for BAI focus on designing policies that can achieve some worst-case or problem-specific bounds. In contrast, R&S problems commonly utilize the Gaussian sampling distribution assumption, and sampling policies derived for R&S have some asymptotic guarantees. The commonly used Gaussian assumption on samples in R&S problems supports scenarios where MCTS is applied to solve general decision-making problems. For example, MCTS can be employed to search for an optimal path that minimizes the time taken for an agent to navigate through a grid-based region to reach a destination while avoiding obstacles. In this context, each sample in a rollout of MCTS represents the time taken, which is unknown and unbounded. In addition, MAB methods, such as Thompson sampling, exhibit poor asymptotic performance for the BAI problem since they tend to exclusively pull an arm in almost all rounds once it is estimated to be the best with a high probability (Russo et al. 2018, Russo 2020).

For the literature on applying R&S methods to the node selection problem in MCTS, Li et al. (2021) introduce a node selection policy based on the OCBA algorithm, OCBA-MCTS, which purportedly achieves a more balanced tradeoff between exploration and exploitation, outperforming UCT. However, recent studies in R&S highlight the limitations of the OCBA algorithm in its derivation, asymptotic optimality, and finite-sample performances (Peng et al. 2018, Russo 2020, Zhang et al. 2023b). Specifically, the derivation of OCBA is developed based on solutions to a static constrained optimization problem solved using some approximations, neglecting both the uncertainty in unknown parameters in sampling distributions and the finite-sample performance of a sequential sampling policy. OCBA does not have asymptotic optimality as defined in Glynn and Juneja (2004), and existing empirical evidence demonstrates its poor finite-sample performance in certain scenarios where different actions are difficult to compare. In this work, we extend AOAP, which is also a sampling policy for the classical R&S problem but can compensate for the mentioned shortages of OCBA, to a node selection policy. Compared with OCBA, AOAP is derived based on a stochastic dynamic programming problem under the Bayesian framework and has been demonstrated to have both theoretical guarantees, including consistency and asymptotic optimality, and superior finite-sample performance over OCBA (Peng et al. 2018, Peng and Zhang 2022). Recent advances in AOAP include addressing various assumptions on sampling distributions, such as exponential sampling distributions (Zhang et al. 2020), extending to different optimization goals, such as subset selection (Zhang et al. 2023a,b), and context-dependent simulation budget allocation (Li et al. 2024, Zhang et al. 2024a), as well as applications in warehouse management (Zhang et al. 2024b). In this work, we also show the benefits of integrating the AOAT with both value network and policy network in NN-MCTS, a topic seldom discussed in the existing literature on using R&S schemes in MCTS.

In the context of NN-MCTS, previous work (Loeffler et al. 2021, Liu et al. 2023) mainly puts their emphasis on designs related to NNs while neglecting the importance of the selection policy in MCTS. The node selection policies used in these works are either the original UCT or UCT with NN-related modifications such as the modified UCT selection policies in AlphaGo Zero and MuZero. We are not aware of any selection policy other than UCT that has been demonstrated to be effective in NN-MCTS. There also lacks of study on how R&S methods can be effectively applied in NN-MCTS. This work fills these research gaps.

## 3. Methodology

In this section, we first introduce the MCTS. Then we formulate the node selection in MCTS as a multi-stage R&S problem. After that, we propose AOAT by extending AOAP to the multi-stage R&S problem in MCTS and prove the consistency of AOAT. Lastly, we discuss how the value network in NN-MCTS could provide prior knowledge that benefits AOAT and how the policy network in NN-MCTS could assist the selection of the final action in AOAT.

### 3.1. MCTS on MDPs

We consider the problem of using MCTS to make decisions in MDPs. Typically, an MDP contains five elements: $<\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R}>$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T}$ is the transition probability, $\gamma$ is the discount factor, and $\mathcal{R}$ is the reward function. Under the umbrella of AlphaGo Zero and MuZero, we consider a finite Markov Decision Process (MDP) with a maximum horizon $H$ and sparse rewards. In this setup, only one reward $\mathcal{R}(s_H, a_H)$ occurs when the MDP reaches its end, and the discount factor $\gamma$ is set to 1. Consequently, the agent must explore over a typically very large set of trajectories to find the best action. MDPs with sparse rewards are common in the formulation of various decision problems such as video games, robotic controls, and board games. The action space is state-dependent, where $\mathcal{A}_s \subseteq \mathcal{A}$ represents the action space under state $s \in \mathcal{S}$. The state transition is deterministic, i.e., $\mathcal{T}(s'|s,a) \in \{0,1\}$, $\forall\ s,s' \in \mathcal{S}$, $\forall\ a \in \mathcal{A}_s$. Given a specific state $s$, the choice of an action $a \in \mathcal{A}_s$ uniquely determines a state $\tilde{s}$ such that $\mathcal{T}(\tilde{s}|s,a) = 1$. The rationale for deterministic transitions is rooted in the context of MCTS being introduced for deterministic games with a tree representation. At each time step $t \in \{1,\dots,H\}$, the agent receives the current state $s_t \in \mathcal{S}$ and makes an action $a_t \in \mathcal{A}_{s_t}$ based on its policy $\pi$, i.e., $a_t \sim \pi(\cdot|s_t)$. The system then transfers to the next state $s_{t+1} \sim \mathcal{T}(\cdot|s_t,a_t)$. A state value function that measures the value of being at state $s_t$ is defined as $V^\pi(s_t) = \mathbb{E}_{s'_{t+1:H}\sim\mathcal{T}, a'_{t:H}\sim\pi}\left[\mathcal{R}(s'_H, a'_H)|\, s'_t = s_t\right]$, where $s'_{t+1:H} \sim \mathcal{T}$ represents $s'_{\tau+1} \sim \mathcal{T}(\cdot|a'_\tau, s'_\tau)$ for any $\tau \in \{t,\dots,H-1\}$, and $a'_{t:H} \sim \pi$ represents $a'_\tau \sim \pi(\cdot|s'_\tau)$ for any $\tau \in \{t,\dots,H\}$. A state-action value function that measures the value of taking action $a_t$ under state $s_t$ is defined as $Q^\pi(s_t,a_t) = \mathbb{E}_{s'_{t+1:H}\sim\mathcal{T}, a'_{t+1:H}\sim\pi}\left[\mathcal{R}(s'_H, a'_H)|\, s'_t = s_t, a'_t = a_t\right]$.

The Bellman equation (Bellman 1954) gives that the optimal state value function satisfies $V^*(s_t) = \max_{a_t\in\mathcal{A}_{s_t}} Q^*(s_t,a_t)$, the optimal state-action value function satisfies $Q^*(s_t,a_t) = \mathbb{E}_{s_{t+1}\sim\mathcal{T}}\left[V^*(s_{t+1})\right]$, and the optimal action that yields the highest future expected rewards is $a^*_t = \arg\max_{a_t\in\mathcal{A}_{s_t}} Q^*(s_t,a_t)$.

For any time step $t_0 \in \{1,\dots,H\}$ in an MDP, we aim to find the optimal action $a^*_{t_0}$ under the root state $s_{t_0}$ by conducting MCTS. Specifically, MCTS iteratively estimates optimal state-action values $Q^*(s_{t_0}, a_{t_0})$. Each iteration during MCTS is called a rollout. Figure 1 shows a diagram of the $k$-th rollout of MCTS, which contains four steps: selection, expansion, simulation, and backpropagation.

**Selection:** On the search tree constructed during the previous $(k-1)$ rollouts, a path denoted by $\mathcal{P}^k$ (the yellow dotted arrow in Figure 1) is determined using a certain node selection policy. As a collection of experienced state-action pairs $\left\{(s_{t_0}, a^k_{t_0}),\dots,(s^k_t, a^k_t),\dots,(s^k_{t'-1}, a^k_{t'-1})\right\}$, $\mathcal{P}^k$ starts from the root state $s_{t_0}$ and ends with a new state $s^k_{t'}$ (the yellow dot in Figure 1) that has never been explored before, where the action at each state is determined by the node selection policy. In this work, we focus on the problem of designing an efficient node selection policy, which will be formulated in Section 3.2.

**Expansion:** By doing expansion, the new state $s^k_{t'}$ is added to the tree as a new leaf node.

**Simulation:** After the expansion step, a simulation is conducted on the new state $s^k_{t'}$ to obtain a sample $\hat{Q}^k$ that measures the value of state $s^k_{t'}$. The simulation is completed by adopting a given policy, such as random play or certain heuristics, to simulate the game to its end and recording the outcome of the game as the sample $\hat{Q}^k$. In NN-MCTS, we denote the value network that maps a state to a predicted state value by $f_\theta : \mathcal{S} \to \mathbb{R}$, where $\theta$ represents the parameters of the NN. As

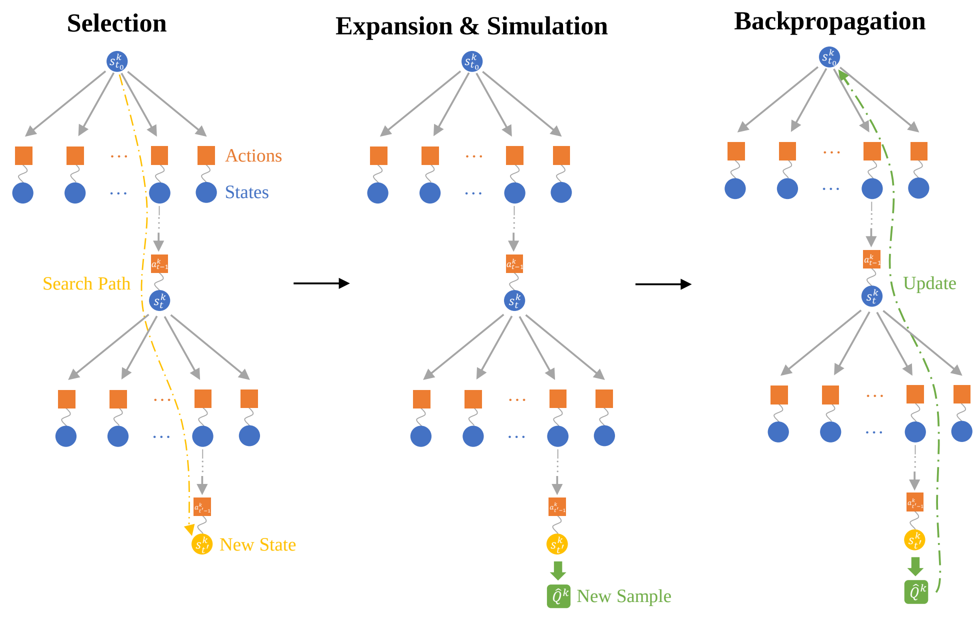


**Figure 1** **Diagram of the $k$-th rollout of NN-MCTS.**

mentioned before, a new state $s_{t'}^k$ is found after the selection step. When using the value network $f_\theta$, the new sample $\hat{Q}^k$ obtained from the $k$-th rollout is given by

$$\hat{Q}^k = f_\theta(s_{t'}^k) \ , \tag{1}$$

which is the estimated state value of the new state $s_{t'}^k$.

**Backpropagation:** Statistics about the state-action values of the state-action pairs on the search path $\mathcal{P}^k$ are updated based on the new sample $\hat{Q}^k$; see Section 3.2 for more details. These updated statistics will affect the path selection in the next $(k+1)$-th rollout.

After $K$ total rollouts, the final action $\hat{a}_{t_0}^K$ under state $s_{t_0}$ is determined based on the statistics collected during the $K$ rollouts. Ideally, we hope the optimal action could be selected, i.e., $\hat{a}_{t_0}^K = a_{t_0}^*$, with high probability and a low search budget. As mentioned, the effectiveness of MCTS is mainly affected by the node selection policy in the selection step, which controls the growth direction of the search tree and the quality of the statistics collected during MCTS. Here, we provide a new way of solving the node selection problem in MCTS using the R&S scheme. In the following, we show the formulation of the node selection problem in MCTS as a multi-stage R&S problem.

## 3.2. Formulation of Node Selection in MCTS as an R&S Problem

We consider a node selection situation in the $k$-th rollout, where we are at state $s_t^k, t \in \{t_0, \dots, H\}$, and need to determine the action $a_t^k$ that leads to the next state $s_{t+1}^k$. For any available state-action pair $(s_t^k, a_t)$, its related statistics collected during the previous $(k-1)$ rollouts are recorded in a statistical set $\mathcal{Q}^{k-1}(s_t^k, a_t)$, which is a set of numerous payoffs collected during the previous

$(k-1)$ rollouts. Specifically, for any $s<k$, if the state-action pair $(s_t^k, a_t)$ is in the path $\mathcal{P}^s$ of the $s$-th rollout, the payoff $\hat{Q}^s$ of the $s$-th rollout is added to the statistical set $\mathcal{Q}^{k-1}(s_t^k, a_t) = \left\{ \hat{Q}^i \middle| (s_t^k, a_t) \in \mathcal{P}^i, i \in \{1, \ldots, k-1\} \right\}$. The number of visit times of the state-action pair $(s_t^k, a_t)$ is given by $N^{k-1}(s_t^k, a_t) = \sum_{i=1}^{k-1} \mathbb{1}\big((s_t^k, a_t) \in \mathcal{P}^i\big)$, where $\mathbb{1}(\cdot)$ is an indicator function that equals 1 when the event in the bracket is true and 0 otherwise.

As mentioned before, each sample in the statistical set $\mathcal{Q}^{k-1}(s_t^k, a_t)$ is an estimated value of a future state that evolves from taking action $a_t$ under state $s_{t_0}^k$. Hence, the mean and variance of the statistical set $\mathcal{Q}^{k-1}(s_t^k, a_t)$ can serve as the sample mean $\bar{Q}^{k-1}(s_t^k, a_t)$ and sample variance $\bar{\sigma}^{k-1}(s_t^k, a_t)$ of the state-action value after $(k-1)$ rollouts, i.e.,

$$\bar{Q}^{k-1}(s_t^k, a_t) = \frac{1}{N^{k-1}(s_t^k, a_t)} \sum_{q \in \mathcal{Q}^{k-1}(s_t^k, a_t)} q \ , \tag{2}$$

$$\bar{\sigma}^{k-1}(s_t^k, a_t) = \frac{1}{N^{k-1}(s_t^k, a_t) - 1} \sum_{q \in \mathcal{Q}^{k-1}(s_t^k, a_t)} \left[ q - \bar{Q}^{k-1}(s_t^k, a_t) \right]^2 \ . \tag{3}$$

Under a Bayesian framework, we assume the samples in the statistical set $\mathcal{Q}^{k-1}(s_t^k, a_t)$ are independent and identically distributed (i.i.d.). The distribution followed by samples in $\mathcal{Q}^{k-1}(s_t^k, a_t)$ is called the sampling distribution. We also assume the prior distribution of the optimal state-action value $Q^*(s_t^k, a_t)$ is the conjugate prior of the sampling distribution. We then introduce the following two types of formulation: one assumes the sampling distribution follows a Gaussian distribution, and the other assumes the sampling distribution follows a Bernoulli distribution.

**Gaussian Assumption:** Under the Gaussian assumption, we assume that the sampling distribution follows a Gaussian distribution with known variance $\sigma(s_t^k, a_t)$, i.e., $\mathcal{N}\big(Q^*(s_t^k, a_t), \sigma(s_t^k, a_t)\big)$, which corresponds to general sparse-reward MDPs with reward $\mathcal{R}(s_H, a_H) \in \mathbb{R}$. As the conjugate prior of the sampling distribution, the prior distribution of $Q^*(s_t^k, a_t)$ is also a Gaussian distribution denoted by $\mathcal{N}\big(Q^0(s_t^k, a_t), \sigma^0(s_t^k, a_t)\big)$. The posterior distribution of $Q^*(s_t^k, a_t)$ after the previous $(k-1)$ rollouts is thus a Gaussian distribution denoted by $\mathcal{N}\big(Q^{k-1}(s_t^k, a_t), \sigma^{k-1}(s_t^k, a_t)\big)$. The posterior variance $\sigma^{k-1}(s_t^k, a_t)$ and posterior mean $Q^{k-1}(s_t^k, a_t)$ are given by

$$\sigma^{k-1}(s_t^k, a_t) = \left( \frac{1}{\sigma^0(s_t^k, a_t)} + \frac{N^{k-1}(s_t^k, a_t)}{\sigma(s_t^k, a_t)} \right)^{-1} \ , \tag{4}$$

and

$$Q^{k-1}(s_t^k, a_t) = \sigma^{k-1}(s_t^k, a_t) \left( \frac{Q^0(s_t^k, a_t)}{\sigma^0(s_t^k, a_t)} + \frac{N^{k-1}(s_t^k, a_t) \bar{Q}^{k-1}(s_t^k, a_t)}{\sigma(s_t^k, a_t)} \right) \ , \tag{5}$$

respectively. $Q^0(s_t^k, a_t)$ and $\sigma^0(s_t^k, a_t)$ are predetermined parameters that contain the prior knowledge of the distribution of $Q^*(s_t^k, a_t)$. In practice, the sample variance $\bar{\sigma}^{k-1}(s_t^k, a_t)$ is often used as a surrogate for $\sigma(s_t^k, a_t)$. When $\sigma^0(s_t^k, a_t) \to \infty$, we get $Q^{k-1}(s_t^k, a_t) = \bar{Q}^{k-1}(s_t^k, a_t)$, which is called an uninformative setting because no prior knowledge is involved. The i.i.d. Gaussian distribution with known variance is a commonly used assumption in R&S literature. This assumption can be justified through batch sampling, leveraging the central limit theorem.

**Bernoulli Assumption:** Under the Bernoulli assumption, we assume the sampling distribution follows $Ber\big(Q^*(s_t^k, a_t)\big)$. This formulation corresponds to MDPs of various games where the final reward $\mathcal{R}(s_H, a_H)$ only takes two possible values $\{0,1\}$, representing win and lose, respectively. As the conjugate prior of the sampling distribution, the prior distribution of $Q^*(s_t^k, a_t)$ is a Beta distribution denoted by $Beta\big(\alpha(s_t^k, a_t), \beta(s_t^k, a_t)\big)$. The posterior distribution of $Q^*(s_t^k, a_t)$ after the previous $(k-1)$ rollouts is thus also a Beta distribution denoted by $Beta\big(\alpha'(s_t^k, a_t), \beta'(s_t^k, a_t)\big)$, where $\alpha'(s_t^k, a_t) = \alpha(s_t^k, a_t) + N^{k-1}(s_t^k, a_t)\bar{Q}^{k-1}(s_t^k, a_t)$ and $\beta'(s_t^k, a_t) = \beta(s_t^k, a_t) + N^{k-1}(s_t^k, a_t) - N^{k-1}(s_t^k, a_t)\bar{Q}^{k-1}(s_t^k, a_t)$. $\alpha(s_t^k, a_t)$ and $\beta(s_t^k, a_t)$ are predetermined parameters that contain the prior knowledge of the distribution of $Q^*(s_t^k, a_t)$. When $\alpha(s_t^k, a_t) = \beta(s_t^k, a_t) = 1$, $Beta(1,1)$ is a Uniform distribution $U(0,1)$, which is the uninformative setting under the Bernoulli assumption.

Remark 1. We compare the differences in assumptions about the sampling distribution in MCTS rollouts between our work and UCT, which is derived under the assumption that the sampling distribution has a bounded support included in $[0,1]$, encompassing distributions with support contained in a known finite interval (allowing transformation into $[0,1]$), such as the Bernoulli distribution, Beta distribution, and uniform distribution. To address the lack of quantification of unknown and unbounded samples in UCT, we consider the assumption of a Gaussian distribution, which can be justified by the central limit theorem. Furthermore, the Gaussian and Bernoulli distribution assumptions in our work are suitable for RL tasks where each action yields a continuous reward and a binary reward (success or failure) with unknown probabilities, respectively. Both distributions are commonly used in reward distribution assumptions when formulating an MDP with uncertainty. Moreover, the Bayesian framework utilized in our work quantifies uncertain parameters in the sampling distribution, enabling the incorporation of prior knowledge into the prior distribution as a warm-up method for MCTS. In Section 5, we empirically demonstrate that AOAT outperforms UCT even when the sampling distribution does not follow a Gaussian or Bernoulli distribution but still satisfies the distribution assumption of UCT. This showcases the robustness of AOAT in more general scenarios, not solely relying on specific distribution assumptions.

With the posterior statistics obtained by making one of the above two assumptions, an R&S algorithm is adopted to determine the next action $a_t^k$ that leads to the next state $s_{t+1}^k$. This node selection process is repeated numerous times to move down the tree until a new state $s_{t'}^k$ is found, marking the end of the selection step in MCTS. The $k$-th rollout then proceeds to the expansion and simulation steps on state $s_{t'}^k$.

After reaching the search budget $K$, rollouts stop, and the action $\hat{a}_{t_0}^K$ with the highest posterior mean is selected as the final decision, given by

$$\hat{a}_{t_0}^K = \arg\max_{a\in\mathcal{A}_{s_{t_0}}} Q^K(s_{t_0}, a) \ , \tag{6}$$

and the probability of correct selection (PCS) of the optimal action is given by

$$\text{PCS} = \Pr\left\{ \bigcap\nolimits_{a\in\mathcal{A}_{s_{t_0}}, a\neq \hat{a}_{t_0}^K} \left( Q^*(s_{t_0}, \hat{a}_{t_0}^K) > Q^*(s_{t_0}, a) \right) \right\} . \tag{7}$$

The objective of the node selection problem in MCTS, as formulated in the R&S problem, is to maximize (7) with a given search budget $K$. For the $k$-th rollout, statistical information collected from the previous $(k-1)$ rollouts is denoted by $\varepsilon_{k-1}$, and the allocation policy of an R&S algorithm is denoted by $\mathbb{A}_k(\cdot)$. Based on the statistical information $\varepsilon_{k-1}$, the allocation policy determines the search path $\mathcal{P}^k$ for the $k$-th rollout, i.e., $\mathbb{A}_k(\varepsilon_{k-1}) = \mathcal{P}^k$. Following Peng et al. (2018), the expected payoff of the allocation policy $\mathbb{A} \triangleq (\mathbb{A}_1(\cdot), \ldots, \mathbb{A}_K(\cdot))$ can be recursively defined in a stochastic dynamic programming problem by considering the situation after reaching the search budget $K$ and observing statistical information $\varepsilon_K$:

$$\nu_K(\varepsilon_K; \mathbb{A}) = \Pr\left\{ \bigcap\nolimits_{a\in\mathcal{A}_{s_{t_0}}, a\neq \hat{a}_{t_0}^K} \left( Q^*\left(s_{t_0}, \hat{a}_{t_0}^K\right) > Q^*(s_{t_0}, a) \right) \middle| \varepsilon_K \right\} , \tag{8}$$

and for $k \in \{0, \ldots, K-1\}$,

$$\nu_k(\varepsilon_k; \mathbb{A}) = \mathbb{E}\left[ \nu_{k+1}(\varepsilon_{k+1}; \mathbb{A}) \middle| \varepsilon_k \right] \ . \tag{9}$$

The optimal allocation policy $\mathbb{A}^*$ can then be defined as the solution to the stochastic dynamic programming problem, i.e., $\mathbb{A}^* = \arg\max_{\mathbb{A}} \nu_0(\varepsilon_0; \mathbb{A})$, where $\varepsilon_0$ contains all the prior information.

## 3.3. AOAP as a Node Selection Policy

Solving the dynamic programming problem in (8) and (9) typically faces the curse of dimensionality. In practice, determining the optimal allocation policy $\mathbb{A}^*$ becomes intractable. Peng et al. (2018) formulate the sampling decisions in R&S as a stochastic dynamic programming problem, establishing the Bellman equation under a Bayesian framework. For the Gaussian sampling distribution, a sequential sampling procedure, AOAP, is developed by maximizing a value function approximation (VFA) of a posterior PCS looking one-step ahead, and it is proved to converge to the static asymptotically optimal sampling ratios that maximize the large deviations rate (LDR) of the probability of false selection (PFS) of the best action (Glynn and Juneja 2004). The AOAP, designed for Gaussian sampling distribution, shows robust performances for non-Gaussian sampling distributions (Peng and Zhang 2022, Zhang et al. 2023b). We extend the VFA used in AOAP to the multi-stage node selection problem in MCTS, leading to a node selection policy named AOAT. We then provide the exact form of the AOAT policy for our node selection problem.

Consider the node selection situation in the $k$-th rollout, where we are at state $s_t^k$, $t \in \{t_0, \dots, H\}$, and need to determine the next action $a_t^k$. In the proposed AOAT method, for the action with the highest posterior mean, denoted as $(a_t^k)^* = \arg\max_{a_t \in \mathcal{A}_{s_t}} Q^{k-1}(s_t^k, a_t)$, the value of sampling $(a_t^k)^*$ is given by, for $a_t \in \mathcal{A}_{s_t^k}$,

$$\mathcal{V}\big(s_t^k, (a_t^k)^*\big) = \min_{a_t \neq (a_t^k)^*} \frac{\left(Q^{k-1}\left(s_t^k, (a_t^k)^*\right) - Q^{k-1}\left(s_t^k, a_t\right)\right)^2}{\tilde{\sigma}^{k-1}\left(s_t^k, (a_t^k)^*\right) + \sigma^{k-1}\left(s_t^k, a_t\right)} \ , \tag{10}$$

whereas the value of sampling an action $a_t \in \mathcal{A}_{s_t^k}, a_t \neq (a_t^k)^*$ is given by, for $\tilde{a}_t \in \mathcal{A}_{s_t^k}$,

$$\mathcal{V}(s_t^k, a_t) = \min\left\{ \frac{\left(Q^{k-1}\left(s_t^k, (a_t^k)^*\right) - Q^{k-1}\left(s_t^k, a_t\right)\right)^2}{\sigma^{k-1}\left(s_t^k, (a_t^k)^*\right) + \tilde{\sigma}^{k-1}\left(s_t^k, a_t\right)}, \min_{\tilde{a}_t \neq a_t, (a_t^k)^*} \frac{\left(Q^{k-1}\left(s_t^k, (a_t^k)^*\right) - Q^{k-1}\left(s_t^k, \tilde{a}_t\right)\right)^2}{\sigma^{k-1}\left(s_t^k, (a_t^k)^*\right) + \sigma^{k-1}\left(s_t^k, \tilde{a}_t\right)} \right\} . \tag{11}$$

Then the selected action $a_t^k$ under the AOAT is determined by maximizing the values obtained from (10) and (11), as follows:

$$a_t^k = \arg\max_{a_t \in \mathcal{A}_{s_t}} \mathcal{V}(s_t^k, a_t) \ . \tag{12}$$

Remark 2. Equations (10) and (11) serve as a VFA of (8) looking one-step ahead under the Gaussian sampling distribution. This approximation is justified by showing that the largest maximum inscribed sphere in the integral region defined by the posterior PCS captures the integral region, so the radius of a sphere is used as a feature, and the approximation error reduces as the search budget increases. More details can be found in Peng et al. (2018) and Zhang et al. (2023b). The differences between AOAP and AOAT are that the former focuses on the Gaussian sampling distribution and only has one stage, whereas the latter considers both Gaussian and Bernoulli sampling distributions and maintains multi-stages in a tree structure.

In the following, we provide specific forms of the AOAT for the Gaussian sampling distribution with known variance and Bernoulli sampling distribution, respectively. Under the Gaussian assumption, $\tilde{\sigma}^{k-1}(s_t^k, a_t)$, $a_t \in \mathcal{A}_{s_t^k}$ in (10) and (11) is given by

$$\tilde{\sigma}^{k-1}(s_t^k, a_t) = \left(\frac{1}{\sigma^0(s_t^k, a_t)} + \frac{N^{k-1}(s_t^k, a_t) + 1}{\sigma(s_t^k, a_t)}\right)^{-1} .$$

For the Bernoulli sampling distribution, we use a Gaussian distribution $\mathcal{N}\big(Q^{k-1}(s_t^k, a_t), \sigma^{k-1}(s_t^k, a_t)\big)$ to approximate the posterior distribution $Beta\big(\alpha'(s_t^k, a_t), \beta'(s_t^k, a_t)\big)$. Under this approximation, we extend the AOAP, originally designed for the continuous Gaussian sampling distribution, to a sampling policy that can be applied to the discrete Bernoulli sampling distribution. This enables AOAT to maintain a similar form whether it is used under the Bernoulli sampling distribution or the Gaussian sampling distribution. In this case,

$$Q^{k-1}(s_t^k, a_t) = \frac{\alpha'(s_t^k, a_t)}{\alpha'(s_t^k, a_t) + \beta'(s_t^k, a_t)} = \frac{\alpha(s_t^k, a_t) + N^{k-1}(s_t^k, a_t)\bar{Q}^{k-1}(s_t^k, a_t)}{\alpha(s_t^k, a_t) + \beta(s_t^k, a_t) + N^{k-1}(s_t^k, a_t)} \ , \tag{13}$$

and

$$\begin{aligned}\sigma^{k-1}(s_t^k, a_t) &= \frac{\alpha'(s_t^k, a_t)\beta'(s_t^k, a_t)}{\left(\alpha'(s_t^k, a_t) + \beta'(s_t^k, a_t)\right)^2 \left(\alpha'(s_t^k, a_t) + \beta'(s_t^k, a_t) + 1\right)} \\ &= \frac{Q^{k-1}(s_t^k, a_t)(1 - Q^{k-1}(s_t^k, a_t))}{\alpha(s_t^k, a_t) + \beta(s_t^k, a_t) + N^{k-1}(s_t^k, a_t) + 1}\ ,\end{aligned} \tag{14}$$

respectively. To justify the rationality of the normal approximation, Proposition 1 shows that the approximation error goes to zero as $N^{k-1}(s_t^k, a_t) \to \infty$ by analyzing the Kullback-Leibler (KL) divergence (i.e., the statistical distance) between two distributions. This indicates that there is not much difference when using the Gaussian approximation-based VFA for the problem with a Bernoulli underlying distribution in an asymptotic regime.

Proposition 1. *The KL divergence between* $Beta\big(\alpha'(s_t^k, a_t), \beta'(s_t^k, a_t)\big)$ *and* $\mathcal{N}\big(Q^{k-1}(s_t^k, a_t), \sigma^{k-1}(s_t^k, a_t)\big)$ *satisfies* $D_{KL}(Beta||Gaussian) = O\big(\frac{1}{N^{k-1}(s_t^k, a_t)}\big)$, *and then* $\lim_{N^{k-1}(s_t^k, a_t) \to \infty} D_{KL}(Beta||Gaussian) = 0$.

Remark 3. The proof of Proposition 1 is relegated to Section A of the online appendix. Notice that Shin et al. (2018) demonstrate that for the classical R&S problem with non-Gaussian underlying distributions whose first two moments exist, the PFS can be approximated by the Gaussian approximation-based LDR when the difference between the best and the second-best means is sufficiently close to 0. Our work approximates the value function by using a normal approximation to the posterior distribution of the underlying distribution.

Under the Bernoulli assumption, $\tilde{\sigma}^{k-1}(s_t^k, a_t)$, $a_t \in \mathcal{A}_{s_t^k}$ in (10) and (11) is given by

$$\tilde{\sigma}^{k-1}(s_t^k, a_t) = \frac{Q^{k-1}(s_t^k, a_t)\,(1 - Q^{k-1}(s_t^k, a_t))}{\alpha(s_t^k, a_t) + \beta(s_t^k, a_t) + N^{k-1}(s_t^k, a_t) + 2}\ .$$

Considering that $Q^{k-1}\big(s_t^k, (a_t^k)^*\big) - Q^{k-1}(s_t^k, a_t) = 0$ for $a_t \in \mathcal{A}_{s_t^k}$, where $a_t \neq (a_t^k)^*$ could happen with a positive probability under the Bernoulli assumption, a small positive number (e.g., $10^{-5}$) can be added to the inside of the square term in the numerator of each term in (10) and (11) to avoid the numerator being 0 in implementation.

The proposed AOAT utilizes posterior means and posterior variances, which are determined by sample means, sample variances, and the visit times of actions. AOAT attempts to balance the exploration of nodes with high variances and the exploitation of nodes with high state-action values. The proposed AOAT is shown to be consistent in Proposition 2, i.e., as the number of rollouts $K$ allocated to the root state $s_{t_0}$ goes to infinity, under any possible state $s_t$ evolved from the root state $s_{t_0}$, the action $\hat{a}_t^K$ with the highest posterior mean converges to the optimal action $a_t^*$, the posterior mean $Q^K(s_t, a_t)$ is the optimal state-action value $Q^*(s_t, a_t)$, and an estimated state value $V^K(s_t) = \max_{a \in \mathcal{A}_{s_t}} Q^K(s_t, a)$ is the optimal state value $V^*(s_t)$.

Proposition 2. *Define $\mathcal{S}_t,\ t \in \{t_0, \dots, H-1\}$ as the set of all possible states at time step $t$ evolved from the root state $s_{t_0}$. Let $\hat{a}_t^K = \arg\max_{a \in \mathcal{A}_{s_t}} Q^K(s_t, a)$, and $V^K(s_t) = \max_{a \in \mathcal{A}_{s_t}} Q^K(s_t, a)$, $s_t \in \mathcal{S}_t$, $t \in \{t_0, \dots, H-1\}$. When applying AOAT, it holds that $\lim_{K \to \infty} \hat{a}_t^K = a_t^*$, $\lim_{K \to \infty} Q^K(s_t, a) = Q^*(s_t, a)$, $\forall a \in \mathcal{A}_{s_t}$, and $\lim_{K \to \infty} V^K(s_t) = V^*(s_t)$.*

The proof of Proposition 2 can be found in Section A of the online appendix. The consistency of AOAT serves as a performance guarantee for MCTS using AOAT as a node selection policy. Although in Section 3.1, we mention that this work considers an MDP with sparse rewards to leverage insights from AlphaGo Zero and MuZero on combining neural network information with the node selection policy, MCTS incorporated with AOAT can be applied to general MDPs with either dense or sparse rewards, and Proposition 2 holds regardless of whether the rewards are sparse or dense. The convergence rate of the bias of the optimal value function, as estimated by the approach developed in Chang et al. (2005), is analyzed under the MAB framework. In contrast, analyzing the convergence of the bias of the optimal value function under a finite search budget is challenging in our work, where the proposed method is developed based on the R&S framework.

The UCT algorithm proposed in Kocsis and Szepesvári (2006) selects actions based on the following formula: for $a, a' \in \mathcal{A}_{s_t^k}$,

$$a_t^k = \arg\max_a \left\{ \bar{Q}^{k-1}(s_t^k, a) + \sqrt{\frac{2 \ln \sum_{a'} N^{k-1}(s_t^k, a')}{N^{k-1}(s_t^k, a)}} \right\}, \tag{15}$$

where the first term in the bracket encourages the selection of actions that have performed well in the past, whereas the second term adds an exploration bonus to less frequently tried actions. Considering that sampling a suboptimal action during the search does not incur a loss to the objective of MCTS, one may conjecture that a node selection policy derived from R&S policies could outperform UCT derived from an MAB policy. To demonstrate the advantage of AOAT over UCT, we further compare the asymptotic performance of UCT (15) with that of AOAT in identifying the best action at the root node.

Proposition 3. *Under UCT (15), the sampling ratio of each action at the root node, as the search budget goes to infinity, satisfies*

$$\lim_{k \to \infty} r^k(s_{t_0}, a_{t_0}) = \begin{cases} 1 & a_{t_0} = a_{t_0}^* \\ 0 & \text{otherwise} \end{cases} \quad a.s.,$$

*where $\sum_{a_{t_0} \in \mathcal{A}_{s_{t_0}}} r^k(s_{t_0}, a_{t_0}) = 1$, $r^k(s_{t_0}, a_{t_0}) > 0$, and $r^k(s_{t_0}, a_{t_0}) = N^k(s_{t_0}, a_{t_0})/k$.*

Remark 4. The proof of Proposition 3 can be found in Section A of the online appendix. In the MAB literature, the optimality of the growth rate of the cumulative regret bound for a sampling policy has been extensively studied. While UCB, which achieves the optimal growth rate of the cumulative regret bound in MAB settings, has been shown to be not well-suited for the BAI problem (Bubeck et al. 2009, Lattimore and Szepesvári 2020). In the R&S literature, the optimality of the sampling ratios in the LDR of the PFS for a sampling policy has been studied frequently. Proposition 3 demonstrates that the asymptotic property of the sampling ratios of UCT is undesirable in terms of the LDR of PFS. Sampling policies for the BAI problem are commonly developed based on bounds on the PFS (Audibert and Bubeck 2010). In contrast, we focus on extending a sampling policy developed for the classical R&S problem to a node selection policy in MCTS in this work.

Peng et al. (2018) demonstrate that the asymptotic sampling ratios of the basic engine in the derivation of AOAT, i.e., the sampling ratios of AOAP, converge to the asymptotically optimal sampling ratios defined in Glynn and Juneja (2004), all of which are positive numbers. The PFS of the best action exponentially decays to 0 at an optimal rate under AOAP. In contrast, the asymptotic sampling ratios of UCT are suboptimal, indicating that in the limit, UCT is not as effective as AOAT in identifying the best action at the root node. In addition, Proposition 3 demonstrates that UCT tends to exclusively exploit the optimal action at the root node as the search budget goes to infinity. However, considering that very few of the possible actions lead to a win in an RL task with a large state and action space, prematurely exploiting an empirically good action may never lead to a win. This underscores the value of exploration in the action space of the root node when using MCTS for an RL task. Compared with UCT, AOAT achieves a more balanced exploration and exploitation for MCTS.

In NN-MCTS, both value networks and policy networks play significant roles in guiding the search process, leading to efficient navigation through the vast space of possible moves and states, particularly in algorithms like AlphaZero and AlphaGo. The value network assesses the potential value of a state, providing an estimate of the likelihood of winning from a given state. In contrast, the policy network aims to suggest the most promising moves to explore during the search process, evaluating the probability distribution over possible actions given the current state. In Section 3.4 and Section 3.5, we aim to enhance the performance of AOAT using the value network and policy network, respectively.

## 3.4. Prior Knowledge from Value Network

In NN-MCTS, incorporating a value network provides a significant advantage when using AOAT for node selection, as the value network can offer prior knowledge to AOAT. As mentioned in Section 3.2, computing the posterior mean $Q^{k-1}(s_t^k, a_t)$ for any $a_t \in \mathcal{A}_{s_t^k}$ requires several predetermined parameters that contain prior knowledge of the distribution of the optimal state-action value $Q^*(s_t^k, a_t)$. These parameters largely determine the performance of the R&S methods by affecting the exploration of the node selection. This can be seen from the posterior mean and variance of a never-visited state-action pair. During the $k$-th rollout, suppose the state-action pair $(s_t^k, a_t)$ has never been visited during the previous $(k-1)$ rollouts, i.e., $N^{k-1}(s_t^k, a_t) = 0$. Under the Gaussian assumption, (4) and (5) give that the posterior mean $Q^{k-1}(s_t^k, a_t)$ and the variance $\sigma^{k-1}(s_t^k, a_t)$ of the state-action pair $(s_t^k, a_t)$ are

$$Q^{k-1}(s_t^k, a_t) = Q^0(s_t^k, a_t), \quad \sigma^{k-1}(s_t^k, a_t) = \sigma^0(s_t^k, a_t) \ . \tag{16}$$

Similarly, under the Bernoulli assumption, (13) and (14) give that the posterior mean $Q^{k-1}(s_t^k, a_t)$ and variance $\sigma^{k-1}(s_t^k, a_t)$ of state-action pair $(s_t^k, a_t)$ are

$$Q^{k-1}(s_t^k, a_t) = \frac{\alpha(s_t^k, a_t)}{\alpha(s_t^k, a_t) + \beta(s_t^k, a_t)}, \ \sigma^{k-1}(s_t^k, a_t) = \frac{\alpha(s_t^k, a_t)\beta(s_t^k, a_t)}{\left(\alpha(s_t^k, a_t) + \beta(s_t^k, a_t)\right)^2 \left(\alpha(s_t^k, a_t) + \beta(s_t^k, a_t) + 1\right)} \ . \tag{17}$$

Equations (16) and (17) show that for never-visited state-action pairs, their posterior means and variances are fully determined by predetermined parameters. Whether or not these new state-action pairs will be explored is thereby controlled by the predetermined parameters. Since these parameters are state-action dependent, manually setting them can be challenging and may lead to undesirable performance. Using a value network can largely alleviate the above-mentioned challenge. The rationale is that if taking action $a_t$ under state $s_t^k$ leads to state $s_{t+1}^k$, then the estimated state value $f_\theta(s_{t+1}^k)$ can be taken as the prior mean of $Q^*(s_t^k, a_t)$. Specifically, the prior mean of $Q^*(s_t^k, a_t)$ under the Gaussian assumption is $Q^0(s_t^k, a_t)$. With the value network $f_\theta$, the parameter $Q^0(s_t^k, a_t)$ can be directly determined as

$$Q^0(s_t^k, a_t) = f_\theta(s_{t+1}^k) \ . \tag{18}$$

Similarly, under the Bernoulli assumption, the prior mean of $Q^*(s_t^k, a_t)$ is $\frac{\alpha(s_t^k, a_t)}{\alpha(s_t^k, a_t) + \beta(s_t^k, a_t)}$. With the value network $f_\theta$, parameters $\alpha(s_t^k, a_t)$ and $\beta(s_t^k, a_t)$ satisfy

$$\frac{\alpha(s_t^k, a_t)}{\alpha(s_t^k, a_t) + \beta(s_t^k, a_t)} = f_\theta(s_{t+1}^k) \ . \tag{19}$$

With (18) and (19), only one parameter needs to be determined for each state-action pair $(s_t^k, a_t)$, i.e., $\sigma^0(s_t^k, a_t)$ under the Gaussian assumption and either $\alpha(s_t^k, a_t)$ or $\beta(s_t^k, a_t)$ under the Bernoulli assumption. The exploitation of prior knowledge from the value network makes the effective implementation of AOAT in NN-MCTS easier.

### 3.5. Using Policy Network to Assist Action Selection

Different from the value network $f_\theta$, the policy network $f_\pi$ outputs a prior probability distribution of actions, which is an approximation of the optimal policy. The prior probability of taking an action $a$ under the state $s$ given by the policy network is represented as $f_\pi(s, a)$. When conducting NN-MCTS, the policy network is typically used to assist in the selection of preferable actions because the action with a higher prior probability given by the policy network is more likely to be the optimal action. One of the most representative examples is UCT used in AlphaGo Zero, which tends to select the action with a high prior probability given by the policy network during each node selection. Specifically, to determine the next action $a_t^k$ at state $s_t^k$ in the $k$-th rollout, a modified UCT used in AlphaGo Zero (Silver et al. 2017) selects the action with the highest upper confidence bound as follows: for $a, a' \in \mathcal{A}_{s_t^k}$,

$$a_t^k = \arg\max_a \left\{ \bar{Q}^{k-1}(s_t^k, a) + c f_\pi(s_t^k, a) \frac{\sqrt{\sum_{a'} N^{k-1}(s_t^k, a')}}{1 + N^{k-1}(s_t^k, a)} \right\} , \tag{20}$$

where $c$ is a positive exogenous parameter that balances the exploration and exploitation during node selection. After all $K$ rollouts are conducted, AlphaGo Zero determines the final action $\hat{a}_{t_0}^K$ under the root state $s_{t_0}$ as the action with the highest number of visit times (Silver et al. 2017), i.e.,

$$\hat{a}_{t_0}^K = \arg \max_{a \in \mathcal{A}_{s_{t_0}}} N^K(s_{t_0}, a) . \tag{21}$$

We introduce the policy network into AOAT, leading to a variant of AOAT called “AOAT-Pi”. Unlike UCT used in AlphaGo Zero and MuZero, which refer to the policy network during the rollouts of MCTS, we only exploit the policy network to assist in determining the final action after completing all $K$ rollouts. Specifically, the node selection policy during rollouts in AOAT-Pi is the same as the original AOAT described in Section 3.3. After all $K$ rollouts are conducted, instead of following (6) and selecting the action with the highest posterior mean as the final action, AOAT-Pi selects the action that maximizes the product of the prior probability given by the policy network and the posterior mean, i.e.,

$$\hat{a}_{t_0}^K = \arg \max_{a \in \mathcal{A}_{s_{t_0}}} \left[ f_\pi(s_{t_0}, a) Q^K(s_{t_0}, a) \right] . \tag{22}$$

The intuition behind adopting the policy network as described in (22) is mentioned above; namely, the action with a higher prior probability given by the policy network is more likely to be the optimal action. Our numerical results in Section 4 show that adopting the prior probability given by the policy network can improve AOAT when dealing with difficult decision-making problems, and the product form in AOAT-Pi outperforms other possible ways of incorporating the policy network into AOAT. The pseudocode for implementing NN-MCTS using AOAT (i.e., AOAT incorporated with the value network) and AOAT-Pi (i.e., AOAT incorporated with both the value network and the policy network) as the node selection policy is provided in Algorithm 1.

**Algorithm 1** NN-MCTS with AOAT/AOAT-Pi

**Input:** Root state $s_{t_0}$, value network $f_\theta$, policy network $f_\pi$ (if using AOAT-Pi), and the search budget $K$.

**Output:** Final selected action $\hat{a}^K_{t_0}$.

```
1:  for $k = 1, \dots, K$ do
2:      Initialize the time step index $t$ as $t = t_0$, the root state $s^k_t$ for the selection step in the $k$-th
        rollout as $s^k_t = s_{t_0}$, and the path $\mathcal{P}^k$ as an empty set.
3:      while $s^k_t$ is visited in the previous $(k-1)$ rollouts do
4:          Determine the action $a^k_t$ under current state $s^k_t$ with the AOAT policy (10)–(12).
5:          Push the state-action pair $(s^k_t, a^k_t)$ into the path $\mathcal{P}^k$, take action $a^k_t$ in state $s^k_t$, and reach
            the next state $s^k_{t+1}$.
6:          Update the index $t$ as $t \leftarrow t+1$.
7:      end while
8:      Record the time step of the selected unvisited state $s^k_t$ as $t'$, i.e., $t' = t$.
9:      Apply expansion on the unvisited state $s^k_{t'}$ using the value network $f_\theta$, and obtain the new
        sample $\hat{Q}^k$ by following (1).
10:     for any visited state-action pair $(s, a)$ in the previous $k$ rollouts do
11:         Compute the statistical set $\mathcal{Q}^k(s,a)$, the visit times $N^k(s,a)$, the sample mean $\bar{Q}^k(s,a)$, the
            sample variance $\bar{\sigma}^k(s,a)$, the posterior mean $Q^k(s,a)$, and the posterior variance $\sigma^k(s,a)$
            by using (2)-(5), (13), and (14).
12:     end for
13: end for
14: Determine the final action $\hat{a}^K_{t_0}$ with (6) if AOAT is used, or with (22) if AOAT-Pi is used.
```

## 4. Numerical Experiments on Board Games

In this section, the effectiveness of the proposed AOAT and AOAT-Pi policies is verified on two board games. We first introduce our experiment setting and benchmark policy. Then, we provide details about the training of the value network and the policy network used in NN-MCTS. Finally, we present numerical results to demonstrate the superiority of AOAT and AOAT-Pi as the node selection policy. The codes for the numerical experiments in this paper are available in the IJOC GitHub repository (Liu et al. 2024).

### 4.1. Experiment Setting and Benchmark Policy

We verify the effectiveness of AOAT and AOAT-Pi using two board games: Tic-Tac-Toe and Five-In-A-Row. In Tic-Tac-Toe, two players take turns marking spaces on a 3×3 board, and the first player to place three marks in a line wins the game. Since Tic-Tac-Toe is a simple game with only 765 possible board configurations, it provides a cost-effective way to evaluate the performance of node selection policies in NN-MCTS with a low search budget. Five-In-A-Row is a more complex version of Tic-Tac-Toe, where two players take turns placing stones on a 10×10 board, and the first player to place five stones in a line wins the game. Here, we reduce the standard board size in Five-In-A-Row, which is 15×15, due to limit on computational power. However, strategies and patterns learned on a smaller board are often transferable, making insights gained applicable to standard-sized games. A draw can occur in both Tic-Tac-Toe and Five-In-A-Row if all spaces are marked but neither player achieves the required marks in a row. When formulating Tic-Tac-Toe and Five-In-A-Row as MDPs, the reward takes three possible values: $\{0, 0.5, 1\}$, corresponding to a loss, a draw, and a win, respectively.

For the benchmark policy, our work focuses on NN-MCTS, which has been less studied in the R&S literature. Hence, we compare the performance of AOAT and AOAT-Pi against UCT used in AlphaGo Zero described by (20) and (21), whose superb performance has been demonstrated by its success of beating human players in the game Go.

## 4.2. Training of NNs in NN-MCTS

We represent a state $s_t$ with a state feature $F(s_t) = [C_t, B_{t-2}, B_{t-1}, B_t]$. For Tic-Tac-Toe, $C_t$ is a 3×3 matrix where all elements equal 1 if the "×" player is to make the next move, and $-1$ if the "○" player is to make the next move. $B_{t-2}$, $B_{t-1}$, and $B_t$ are representations of the 3×3 boards at time steps $t-2$, $t-1$, and $t$, respectively, where each element equals 1 if the corresponding space on the board is marked with "×", $-1$ if it is marked with "○", and 0 if the corresponding space on the board is not marked yet. For Five-In-A-Row, $C_t$ is a 10×10 matrix where all elements equal 1 if the black player is to make the next move, and $-1$ if the white player is to make the next move. $B_{t-2}$, $B_{t-1}$, and $B_t$ are representations of the 10×10 boards at time steps $t-2$, $t-1$, and $t$, respectively, where each element equals 1 if the corresponding space on the board is marked by the black player, $-1$ if it is marked by the white player, and 0 if the corresponding space on the board is not marked yet.

Taking the state feature $F(s_t)$ as input, the value network is constructed with CNNs. The structure of the value network can be found in Section B.1 of the online appendix. The value network follows the classical encoder-decoder structure, with two residual blocks connecting the encoder and decoder. The policy network has the same structure as the value network, except that the width of the fully connected layer in the head decoder is 9 for Tic-Tac-Toe and 100 for Five-In-A-Row. The activation layer in the head decoder is SoftMax instead of Tanh, and the final normalization layer is omitted. The training procedure for the value network $f_\theta$ and the policy network $f_\pi$ is shown in Figure 2.

In each iteration, board games are simulated on $N$ CPUs in parallel to collect training data. At time step $t$ of a simulated game, NN-MCTS using the latest value network $f_\theta$ and policy network

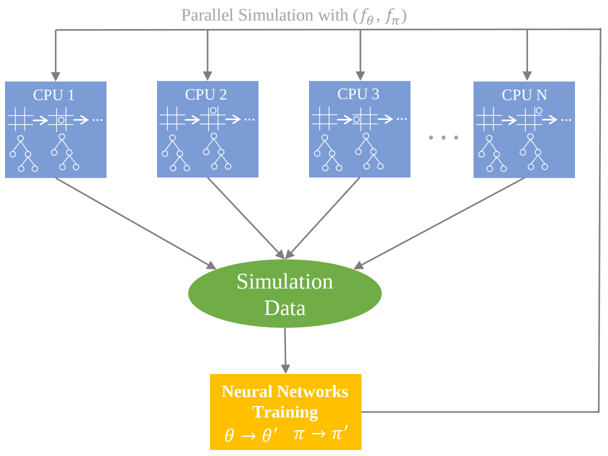

Parallel Simulation with ($f_\theta$, $f_\pi$)
CPU 1
CPU 2
CPU 3
CPU N
Simulation Data
Neural Networks Training
$\theta \to \theta'$ $\pi \to \pi'$

**Figure 2** **Training Procedure of the Value Network and the Policy Network.**

$f_\pi$ is conducted to make decisions. To be consistent with AlphaGo Zero, UCT is used as node selection policy during game simulations, and a probability distribution $P_t$ of actions is computed after all $K$ rollouts as follows:

$$P_t(a) = \frac{\sqrt{N^K(s_t,a)}}{\sum_{a'\in\mathcal{A}_{s_t}} \sqrt{N^K(s_t,a')}}, \quad \forall a \in \mathcal{A}_{s_t} \ . \tag{23}$$

The outcome of a simulated game is $z \in \{0, 0.5, 1\}$. The training data collected at time step $t$ is stored as $(s_t, P_t, z)$. The loss functions $L_\theta$ and $L_\pi$ for the value network $f_\theta$ and the policy network $f_\pi$ are $L_\theta = \left[f_\theta(s_t) - z\right]^2$ and $L_\pi = -\sum_{a\in\mathcal{A}} \left[P_t(a) f_\pi(a)\right]$, respectively. The gradients of $L_\theta$ and $L_\pi$ are calculated to update the value network and policy network using the Adam optimizer (Kingma and Ba 2014). After training with collected data for $E$ epochs, the parameters $\theta$ of the value network and $\pi$ of the policy network are updated to $\theta'$ and $\pi'$, which are used in the simulation step of the next iteration. Detailed exogenous parameters related to the above training procedure for Tic-Tac-Toe and Five-In-A-Row are provided in Section B.2 of the online appendix. The training losses $L_\theta$ and $L_\pi$ during the entire model training process for Tic-Tac-Toe are shown in Section B.3 of the online appendix.

## 4.3. Performance Comparison of Node Selection Policies

In Section 4.3.1, we compare AOAP-MCTS (AOAT without using a value network), AOAT, AOAT-Pi, and OCBA-MCTS to empirically demonstrate the superior performance of a node selection policy derived based on AOAP over that based on OCBA. Furthermore, under the NN-MCTS setting, we compare the performance of AOAT and AOAT-Pi against UCT in Tic-Tac-Toe and Five-In-A-Row games, in Sections 4.3.2 and 4.3.3, respectively.

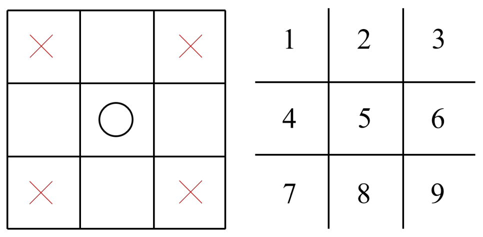
**4.3.1. Comparisons between AOAT and OCBA-MCTS.** In the $k$-th rollout, OCBA-MCTS selects an allocated action $a_t^k$ for the state $s_t^k$, $t \in \{t_0, \cdots, H\}$ according to $a_t^k = \arg\max_{a_t \in \mathcal{A}_{s_t^k}} \left(\widetilde{N}^{k-1}\left(s_t^k, a_t\right) - N^{k-1}\left(s_t^k, a_t\right)\right)$, where

$$\frac{\widetilde{N}^{k-1}\left(s_t^k, a_t\right)}{\widetilde{N}^{k-1}\left(s_t^k, \widetilde{a}_t\right)} = \left(\frac{\sigma\left(s_t^k, a_t\right)/\delta\left(\left(a_t^k\right)^*, a_t\right)}{\sigma\left(s_t^k, \widetilde{a}_t\right)/\delta\left(\left(a_t^k\right)^*, \widetilde{a}_t\right)}\right)^2,$$

$$\widetilde{N}^{k-1}\left(s_t^k, \left(a_t^k\right)^*\right) = \sigma\left(s_t^k, \left(a_t^k\right)^*\right)\sqrt{\sum\nolimits_{a_t \in \mathcal{A}_{s_t^k}, a_t \neq \left(a_t^k\right)^*} \frac{\widetilde{N}^{k-1}\left(s_t^k, a_t\right)}{\sigma^2\left(s_t^k, a_t\right)}},$$

$\delta\left(\left(a_t^k\right)^*, a_t\right) = \bar{Q}^{k-1}\left(s_t^k, \left(a_t^k\right)^*\right) - \bar{Q}^{k-1}\left(s_t^k, a_t\right)$, and $\left(a_t^k\right)^* = \arg\max_{\mathbf{a}_t \in \mathcal{A}_{s_t^k}} \bar{Q}^{k-1}\left(s_t^k, a_t\right)$.

Notice that the method for incorporating neural network information into OCBA-MCTS remains an open question. Therefore, we do not compare our node selection policies with OCBA-MCTS combined with neural networks. For a fair comparison, we first evaluate AOAP-MCTS, OCBA-MCTS, and UCT, all without utilizing information from neural networks in NN-MCTS, in a Tic-Tac-Toe game. The objective of the experiment is to find the best move for the "✕" player based on the move made by the "◯" player, as shown in Figure 3, where the best move is any of the four corners of the board.

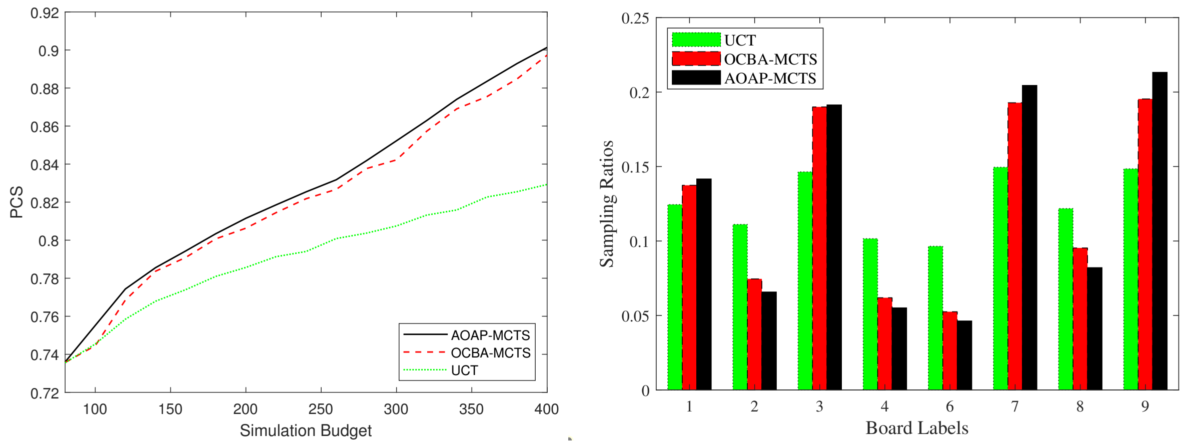


(a) Board setup (b) Board labels

**Figure 3 The Tic-Tac-Toe board setup, where the best move for the "✕" player, based on the "◯" player selecting the middle move (board label 5), is any one marked in red (board labels 1,3,7, and 9).**

We test the PCS of the best move for each node selection policy, under the assumption that the "◯" player randomly selects a move in the game. We set $Q^0(s_t^k, a_t) = 0$ and $\sigma^0(s_t^k, a_t) = 10$ for AOAT. The number of total rollouts is set as $K = 400$. Figure 4 (a) and (b) show the PCS of the best moves and the sampling ratio of each move after exhausting all total rollouts for all policies, respectively, each estimated by 100,000 independent macro experiments.

From Figure 4 (a), we observe that AOAP-MCTS outperforms OCBA-MCTS and UCT. This observation aligns with the results reported in Li et al. (2021), where OCBA-MCTS is shown to perform better than UCT. Furthermore, the slight edge of AOAP-MCTS over OCBA-MCTS empirically supports the argument that designing a node selection policy based on AOAP could



(a) PCS (b) Sampling Ratios

**Figure 4 Comparisons of (a) PCS of the best moves (board labels 1,3,7, and 9) and (b) sampling ratio of each move with $K = 400$ (all averaged over 100,000 runs) for three node selection policies.**

lead to better performance than one based on OCBA. As shown in Figure 4 (b), the averaged sampling ratios of OCBA-MCTS and AOAP-MCTS show a slight difference, which are significantly different from those of UCT. UCT tends to exploit the best moves more. Both OCBA-MCTS and AOAP-MCTS pay less attention to inferior actions (board labels 2,4,6, and 8) than UCT. The slight edge of AOAP-MCTS over OCBA-MCTS shown in Figure (a) can be attributed to that AOAP-MCTS spends slightly more effort on equally optimal actions compared to OCBA-MCTS.

Then, we demonstrate that the advantage of AOAP-MCTS over OCBA-MCTS becomes more significant when AOAP-MCTS is combined with a value network and a policy network. We assign two players to utilize different policies in Tic-Tac-Toe and Five-In-A-Row games. The compared policies include UCT, OCBA-MCTS, AOAT, and AOAT-Pi, where the latter two benefit from the information provided by NN-MCTS. Tables 1 and 2 display the average margin winning rates of Player 1 across 200 games of Tic-Tac-Toe and Five-In-A-Row, respectively. The average margin winning rate refers to the average difference in the number of wins between the winning and losing players across all games played. This metric is valuable for assessing the efficiency of different policies, as it not only measures how often a policy wins but also how decisively it wins, distinguishing between policies that may have similar win rates but differ significantly in their level of dominance. To avoid one player having the advantage of playing the first move, each player takes the first move in half of the 200 tested games of Tic-Tac-Toe and Five-In-A-Row.

From Tables 1 and 2, we observe that AOAT and AOAT-Pi outperform UCT and OCBA-MCTS, achieving higher average margin winning rates. The higher average margin winning rate of AOAT-Pi compared to AOAT demonstrates the benefit of the information provided by the policy network. The differences in performance among the compared policies are more pronounced in

**Table 1** **The average margin winning rate of Player 1 across 200 games of Tic-Tac-Toe.**

| Player1 \ Player2 | UCT | OCBA-MCTS | AOAT | AOAT-Pi |
|---|---|---|---|---|
| UCT | 0.01 | 0.01 | 0.01 | 0 |
| OCBA-MCTS | 0.06 | 0.01 | 0.01 | 0 |
| AOAT | 0.09 | 0.07 | 0.01 | 0.01 |
| AOAT-Pi | 0.12 | 0.11 | 0.09 | 0.01 |

**Table 2** **The average margin winning rate of Player 1 across 200 games of Five-In-A-Row.**

| Player1 \ Player2 | UCT | OCBA-MCTS | AOAT | AOAT-Pi |
|---|---|---|---|---|
| UCT | 0.05 | 0.04 | 0.03 | 0.01 |
| OCBA-MCTS | 0.07 | 0.04 | 0.03 | 0.01 |
| AOAT | 0.21 | 0.18 | 0.06 | 0.04 |
| AOAT-Pi | 0.25 | 0.23 | 0.11 | 0.05 |

the more complex game, Five-In-A-Row. The overall low average margin winning rates in both Tic-Tac-Toe and Five-In-A-Row is attributed to that all four compared node selection policies are highly competitive and they tie often against each other. Therefore, even a small margin can be meaningful, especially if one policy consistently performs better.

**4.3.2. Results of Playing Tic-Tac-Toe** We take AOAT/AOAT-Pi and UCT as the two opposing sides in Tic-Tac-Toe games to compare the winning rates of different policies. To initialize the games randomly, the first move of each player is not deterministic but is sampled from a probability distribution $P_t$ of actions. For NN-MCTS using UCT, the probability distribution $P_t$ is defined by (23), whereas for NN-MCTS using AOAT and AOAT-Pi, the probability distribution $P_t$ is defined by

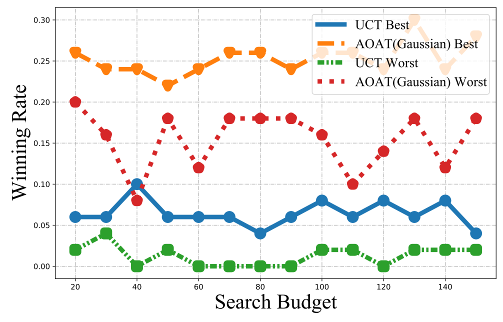
$$\begin{aligned} \text{AOAT: } P_t(a) &= \frac{\sqrt{Q^K(s_t,a)}}{\sum_{a'\in\mathcal{A}_{s_t}}\sqrt{Q^K(s_t,a')}}, \quad \forall a\in\mathcal{A}_{s_t}, \\ \text{AOAT-Pi: } P_t(a) &= \frac{\sqrt{f_\pi(s_t,a)Q^K(s_t,a)}}{\sum_{a'\in\mathcal{A}_{s_t}}\sqrt{f_\pi(s_t,a')Q^K(s_t,a')}}, \quad \forall a\in\mathcal{A}_{s_t}\ , \end{aligned} \tag{24}$$

where (24) is derived from (23) by replacing visiting times obtained after $K$ rollouts with the posterior means obtained after $K$ rollouts, considering that AOAT identifies actions based on their posterior means. Notice that $P_t(a)$ is only utilized to sample the first move of each policy for initialization, which could have little effect on the performance of the compared node selection policies.

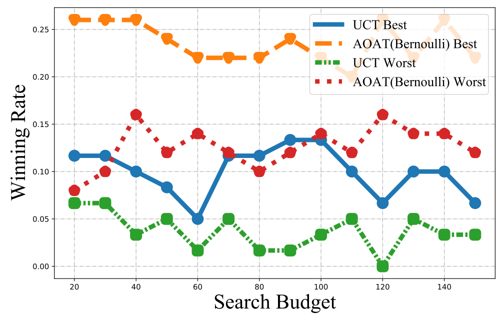
Other than the first move, move $a_t$ for any player is deterministic, following (21) for UCT, (6) for AOAT, and (22) for AOAT-Pi. For exogenous parameters, we consider possible values for $c$ in UCT as $\{0.2, 0.4, 0.6, 0.8, 1\}$, possible values for $\sigma^0(s_t,a_t)$ in AOAT and AOAT-Pi under the

Gaussian assumption as $\{0.05, 0.1, 0.15, 0.20, 0.25\}$, and possible values for $\beta(s_t, a_t)$ in AOAT and AOAT-Pi under the Bernoulli assumption as $\{1, 2, 3, 4, 5, 6\}$. Under each policy parameter setting, we randomly simulate 20 games. For each policy, we select the parameter that yields the highest winning rate and the parameter that yields the lowest winning rate as the best and worst case of its performance.

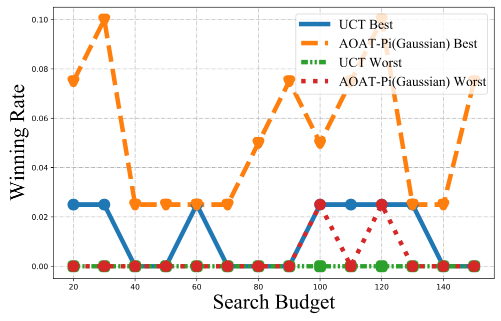
**Winning rates of AOAT/AOTP-Pi vs. UCT.** The results of AOAT/AOAT-Pi vs. UCT when playing Tic-Tac-Toe are shown in Figure 5. The winning rate does not improve with the increase in the search budget, which could be attributed to the reason that both AOAT and UCT improve as the search budget increases, indicating that the opponent of each policy also becomes better. Under both the Gaussian assumption and the Bernoulli assumption, AOAT and AOAT-Pi dominate UCT as the node selection policy in NN-MCTS in terms of winning rates. Specifically, the best cases of AOAT and AOAT-Pi achieve higher winning rates than the best case of UCT under all search budget settings. The worst cases of AOAT and AOAT-Pi are generally comparable to the best case of UCT. These results indicate the superiority of AOAT and AOAT-Pi over UCT in Tic-Tac-Toe.



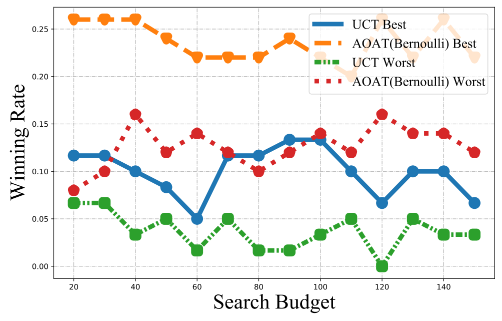
(a) AOAT(Gaussian) vs. UCT



(b) AOAT(Bernoulli) vs. UCT



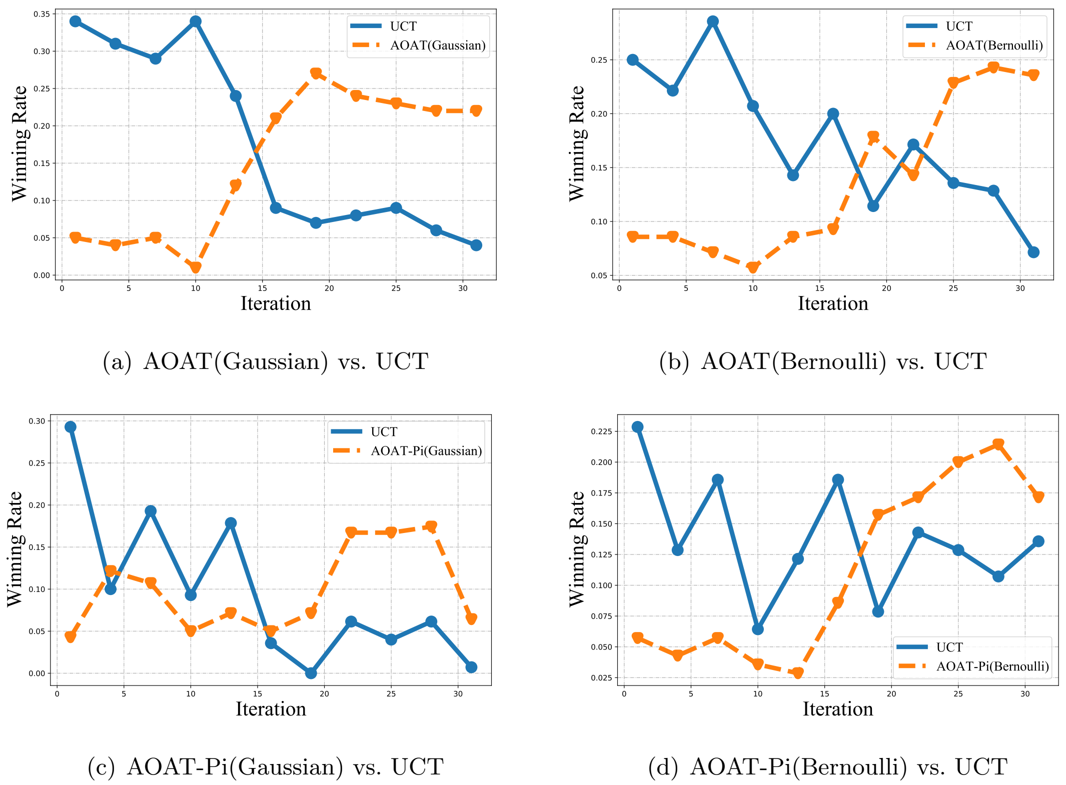
(c) AOAT-Pi(Gaussian) vs. UCT



(d) AOAT-Pi(Bernoulli) vs. UCT

**Figure 5 Winning Rates of AOAT/AOAT-Pi vs. UCT for Tic-Tac-Toe.**

**Winning rates of AOAT/AOAT-Pi vs. UCT during training.** Other than the policy competition using the well-trained value network and policy network, we also investigate policy performance during the training of NNs. As shown in Figure 6, both AOAT and AOAT-Pi are inferior to UCT at the beginning of the training and gradually gain advantages over UCT as the value network and policy network improve. The phenomenon could be attributed to several reasons. First, the value network tends to be under-trained in the initial iterations, leading to less accurate estimations of state values, which may impact the performance of AOAT more than that of UCT. Second, prior knowledge from the value network is incorporated to determine some of the exogenous parameters in AOAT and AOAT-Pi. Poor estimation of state-action values by the value network can result in improper settings of exogenous parameters, further compromising the performance of AOAT and AOAT-Pi.



(a) AOAT(Gaussian) vs. UCT

(b) AOAT(Bernoulli) vs. UCT

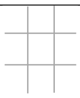
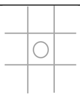
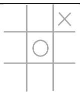
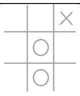
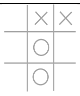
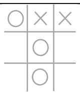
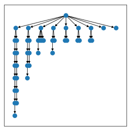
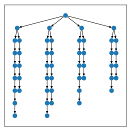
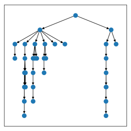
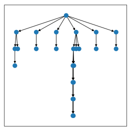
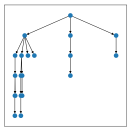
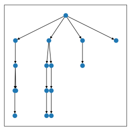
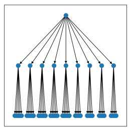
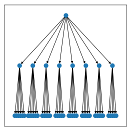
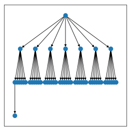
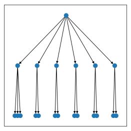
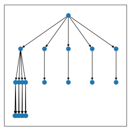
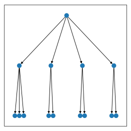
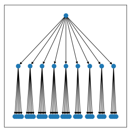
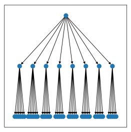
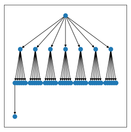
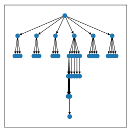
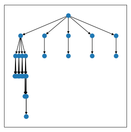
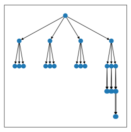
(c) AOAT-Pi(Gaussian) vs. UCT

(d) AOAT-Pi(Bernoulli) vs. UCT

**Figure 6 Winning Rates of AOAT/AOAT-Pi vs. UCT for Tic-Tac-Toe During Training.**

**Visualization of Search Trees.** To better understand how AOAT and AOAT-Pi outperform UCT in node selection, we visualize search trees constructed by NN-MCTS during a Tic-Tac-Toe game. As mentioned, AOAT-Pi adopts the original AOAT during rollouts of MCTS. Hence, the search trees constructed by AOAT-Pi are the same as the trees constructed by AOAT. As shown in Table 3, search trees constructed by AOAT under both the Gaussian assumption and the Bernoulli assumption are more balanced than trees constructed by UCT, which implies that AOAT makes more exploration during node selection than UCT.

**Table 3 Visualizations of Search Trees in One Tic-Tac-Toe Game.**

| | | | | | | |
|---|---|---|---|---|---|---|
| Tic-Tac-Toe Board |  |  |  |  |  |  |
| UCT |  |  |  |  |  |  |
| AOAT (Gaussian) |  |  |  |  |  |  |
| AOAT (Bernoulli) |  |  |  |  |  |  |

**Effect of Value Network.** To validate whether incorporating a value network indeed improves node selection policies, we conduct an ablation study testing the performance of AOAT and UCT without the value network. Specifically, instead of using the value network to estimate the state value of the new leaf node $s^k_{t'}$ obtained from the $k$-th rollout, as shown in (1), we conduct random play starting from the new leaf node $s^k_{t'}$. In other words, each player randomly places its pieces by uniformly sampling actions from the feasible action set, and we take the sample $\hat{Q}^k$ as 1 if the current player wins, 0.5 if it is a draw, and 0 otherwise. We label AOAT and UCT implemented with random play as "AOAT-RP" and "UCT-RP", respectively. The results of AOAT vs. AOAT-RP and UCT vs. UCT-RP when playing Tic-Tac-Toe are shown in Tables 4 and 5, respectively. Both AOAT and UCT achieve higher winning rates when competing against AOAT-RP and UCT-RP, indicating that using the value network to estimate state values indeed improves node selection policies.

**Table 4 Winning rates of AOAT vs. AOAT-RP for Tic-Tac-Toe.**

| Search Budget | Winning Rate (%) of AOAT(Bernoulli) vs. AOAT-RP(Bernoulli) | | | | | | Winning Rate (%) of AOAT(Gaussian) vs. AOAT-RP(Gaussian) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\beta$=1 | $\beta$=2 | $\beta$=3 | $\beta$=4 | $\beta$=5 | $\beta$=6 | $\sigma$=0.05 | $\sigma$=0.1 | $\sigma$=0.15 | $\sigma$=0.2 | $\sigma$=0.25 |
| 30 | (55,30) | (65,30) | (60,30) | (70,30) | (55,45) | (55,45) | (60,25) | (50,35) | (35,35) | (40,25) | (45,40) |
| 60 | (60,30) | (65,25) | (40,25) | (60,20) | (45,40) | (50,50) | (55,20) | (55,25) | (40,40) | (50,25) | (45,30) |
| 90 | (55,35) | (50,50) | (65,15) | (65,20) | (35,30) | (55,20) | (45,30) | (55,25) | (30,20) | (40,35) | (35,15) |
| 120 | (55,15) | (60,15) | (55,30) | (35,25) | (50,35) | (35,25) | (70,10) | (35,20) | (40,25) | (45,25) | (30,30) |
| 150 | (50,25) | (40,35) | (30,25) | (55,35) | (45,30) | (70,15) | (60,25) | (50,20) | (50,30) | (30,30) | (55,40) |

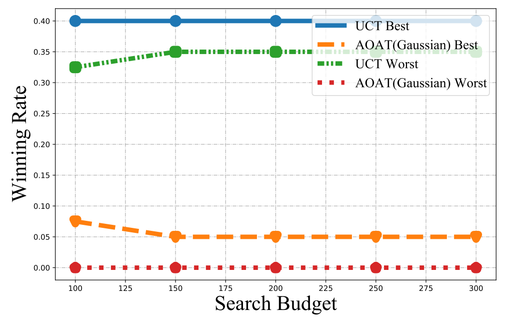
**Table 5 Winning rates of UCT vs. UCT-RP for Tic-Tac-Toe.**

| **Search Budget** | **Winning Rate (%) of UCT vs. UCT-RP** | | | | |
|---|---|---|---|---|---|
| | $c$=0.2 | $c$=0.4 | $c$=0.6 | $c$=0.8 | $c$=1 |
| 30 | (35,30) | (35,20) | (40,5) | (55,20) | (65,15) |
| 60 | (35,15) | (35,35) | (30,10) | (30,20) | (35,20) |
| 90 | (30,20) | (20,20) | (25,15) | (25,25) | (25,15) |
| 120 | (35,35) | (40,30) | (25,25) | (35,25) | (40,25) |
| 150 | (35,30) | (40,15) | (30,30) | (30,15) | (40,25) |

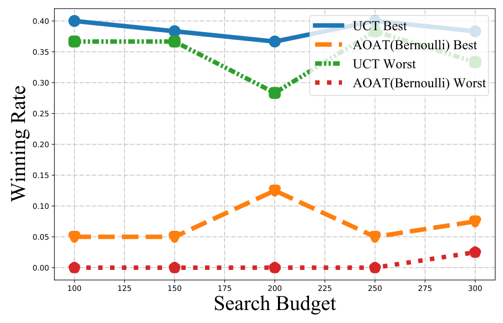
**4.3.3. Results of Five-In-A-Row** The experiment setting for Five-In-A-Row is the same as the setting for Tic-Tac-Toe. We conduct policy competitions between AOAT/AOAT-Pi and UCT under different combinations of policy parameters, i.e., $c \in \{0.2, 0.4, 0.6, 0.8, 1\}$, $\sigma^0(s_t, a_t) \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$, and $\beta(s_t, a_t) \in \{1, 2, 3, 4, 5, 6\}$. Under each policy parameter setting, we randomly simulate 20 games. The first move of each player is sampled from a probability distribution defined by (23) and (24), while subsequent moves are deterministic by following (6), (21), and (22).

**Winning rates of AOAT/AOAT-Pi vs. UCT.** The results of AOAT/AOAT-Pi vs. UCT when playing Five-In-A-Row are shown in Figure 7. Unlike the results for Tic-Tac-Toe, AOAT is inferior to UCT, and AOAT-Pi is superior to UCT when playing Five-In-A-Row. Specifically, the best case of AOAT achieves lower winning rates than the worst case of UCT under both the Gaussian and Bernoulli assumptions. On the contrary, the best case of AOAT-Pi achieves higher winning rates than the best case of UCT under both assumptions. The worst case of AOAT-Pi is better than the best case of UCT under the Gaussian assumption and is comparable to the worst case of UCT under the Bernoulli assumption. We visualize the complete stone placements on the board at the end of four games between the best case of AOAT-Pi against the best case of UCT in Figure 8, where AOAT-Pi wins all four games.

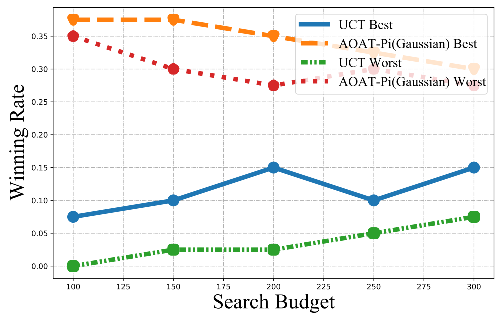
Notice that AOAT-Pi and UCT adopt the policy network, whereas AOAT does not. These results indicate that the incorporation of the policy network is necessary when playing difficult games such as Five-In-A-Row. For the simple game Tic-Tac-Toe, where only 765 possible board situations exist, the value network can be easily trained well and can be highly accurate in estimating the outcomes of the games. Also, rollouts can usually reach the end of the game and fully explore the action space even with a relatively low search budget. Hence, for simple games, the posterior mean $Q^K(s_{t_0}, a)$ for any action $a \in \mathcal{A}_{s_{t_0}}$ under the root state $s_{t_0}$, estimated by conducting NN-MCTS, is accurate enough to identify the optimal action, making the benefits of adopting the policy network not obvious. For more complex games such as Five-In-A-Row, the large number of board situations makes the training of the value network difficult. Also, even with a high search budget, the rollouts of NN-MCTS can only explore part of a large action space. These factors contribute to the relatively poor estimation of the posterior mean $Q^K(s_{t_0}, a)$, making the prior probability given by the policy network an important reference when determining the final action.



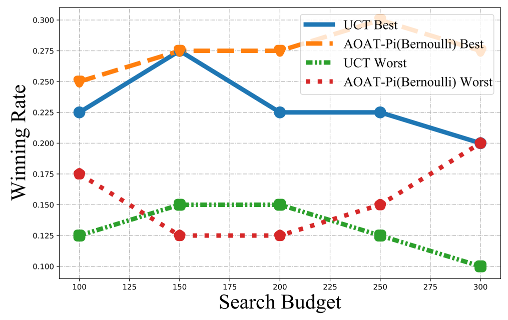
(a) AOAT(Gaussian) vs. UCT



(b) AOAT(Bernoulli) vs. UCT



(c) AOAT-Pi(Gaussian) vs. UCT



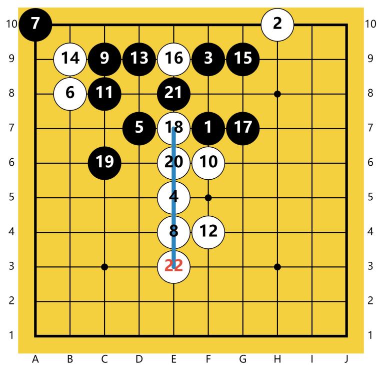
(d) AOAT-Pi(Bernoulli) vs. UCT

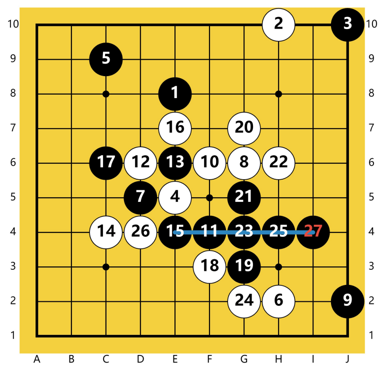
**Figure 7 Winning Rates of AOAT/AOAT-Pi vs. UCT for Five-In-A-Row.**

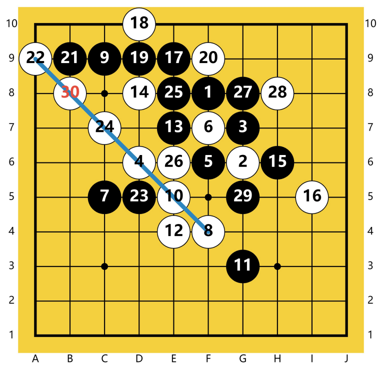
**Winning rates of AOAT/AOAT-Pi vs. UCT during training.** The winning rates of AOAT/AOAT-Pi vs. UCT during the training of NNs are shown in Figure 9. AOAT is inferior to UCT throughout the entire training process, further indicating the importance of adopting the policy network when playing Five-In-A-Row. AOAT-Pi is inferior to UCT at the beginning of the training and gradually gains an advantage over UCT as the training proceeds, which is consistent with the results of playing Tic-Tac-Toe.

**Different forms of incorporating policy network.** In AOAT-Pi, we incorporate the policy network by selecting the action that maximizes the product of the prior probability given by the policy network and the posterior mean, as shown in (22). In this set of experiments, we investigate two additional forms of incorporating the policy network: one in a sum form, labeled as "AOAT-Pi-Sum", and the other in an exponentiation form, labeled as "AOAT-Pi-Exp".

$$\begin{aligned}\text{AOAT-Pi-Sum: } \hat{a}_{t_0}^K &= \arg\max_{a\in\mathcal{A}_{s_{t_0}}} \left[f_\pi(s_{t_0}, a) + Q^K(s_{t_0}, a)\right], \\ \text{AOAT-Pi-Exp: } \hat{a}_{t_0}^K &= \arg\max_{a\in\mathcal{A}_{s_{t_0}}} \left[\left(Q^K(s_{t_0}, a)\right)^{1-f_\pi(s_{t_0}, a)}\right].\end{aligned} \tag{25}$$


(a) UCT vs. AOAT-Pi(Gaussian)


(b) AOAT-Pi(Gaussian) vs. UCT


(c) UCT vs. AOAT-Pi(Bernoulli)

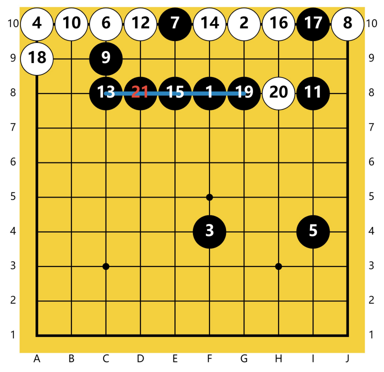
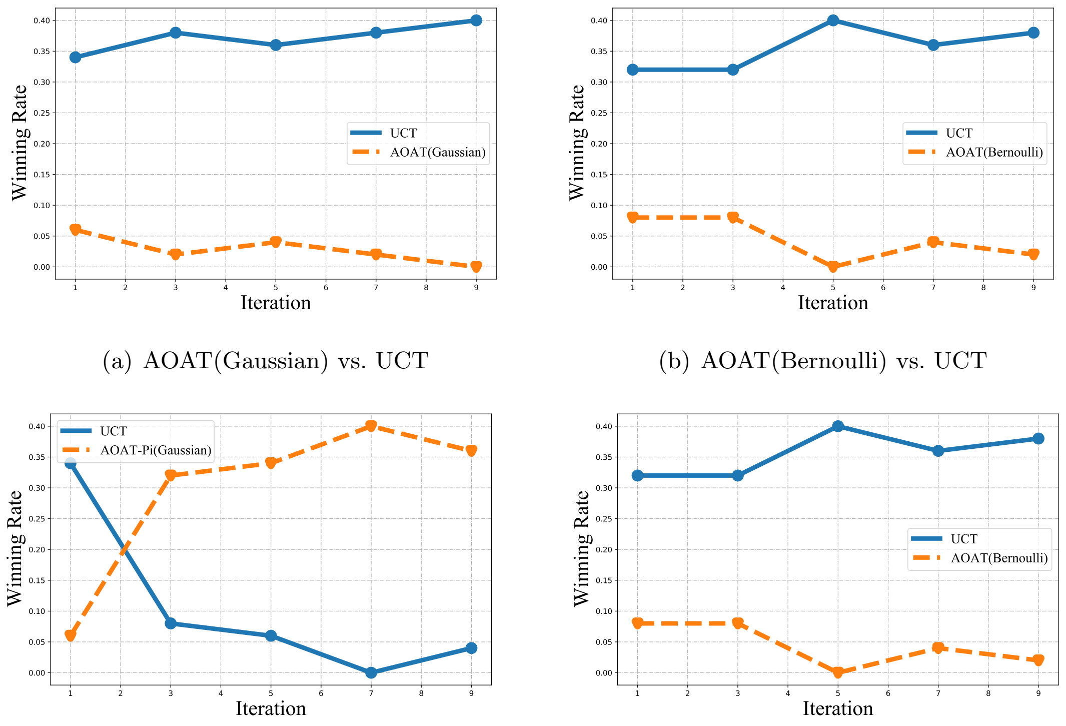

(d) AOAT-Pi(Bernoulli) vs. UCT

**Figure 8 Visualizations of Five-In-A-Row Games, where "A vs. B" in the label of each subfigure represents the black player versus the white player. The stone with the red number is the last move leading to the end of the game.**

The results of AOAT-Pi vs. AOAT-Pi-Sum/AOAT-Pi-Exp implemented under the Gaussian assumption when playing Five-In-A-Row are shown in Table 6. In general, AOAT-Pi with the product form outperforms both AOAT-Pi-Sum and AOAT-Pi-Exp by achieving higher winning rates under most of the parameter settings.

**Table 6 Winning rates of AOAT-Pi vs. AOAT-Pi-Sum/AOAT-Pi-Exp for Five-In-A-Row.**

| Search Budget | Winning Rate (%) of AOAT-Pi vs. AOAT-Pi-Exp | | | | | Winning Rate (%) of AOAT-Pi vs. AOAT-Pi-Sum | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\sigma$=0.05 | $\sigma$=0.1 | $\sigma$=0.15 | $\sigma$=0.2 | $\sigma$=0.25 | $\sigma$=0.05 | $\sigma$=0.1 | $\sigma$=0.15 | $\sigma$=0.2 | $\sigma$=0.25 |
| 100 | (50,50) | (50,50) | (50,50) | (70,30) | (50,50) | (50,50) | (70,30) | (60,40) | (70,30) | (50,50) |
| 150 | (60,40) | (50,50) | (60,40) | (60,40) | (60,40) | (80,20) | (40,60) | (50,50) | (30,70) | (70,30) |
| 200 | (70,30) | (80,20) | (60,40) | (60,40) | (40,60) | (60,40) | (50,50) | (50,50) | (60,40) | (60,40) |
| 250 | (50,50) | (70,30) | (80,20) | (80,20) | (60,40) | (80,20) | (70,30) | (50,50) | (60,40) | (60,40) |
| 300 | (30,70) | (60,40) | (50,50) | (50,50) | (60,40) | (50,50) | (60,40) | (40,60) | (80,20) | (80,20) |



(a) AOAT(Gaussian) vs. UCT

(b) AOAT(Bernoulli) vs. UCT

(c) AOAT-Pi(Gaussian) vs. UCT

(d) AOAT-Pi(Bernoulli) vs. UCT

**Figure 9 Winning Rates of AOAT/AOAT-Pi vs. UCT for Five-In-A-Row During Training.**

## 5. Numerical Experiments on a General RL Task

In this section, to demonstrate the general applicability of AOAT and AOAT-Pi in MCTS, we conduct numerical experiments to verify their effectiveness in a general decision problem where rewards are dense and can take various values.

### 5.1. Experiment Setting and Benchmark Policy

For the experiment setting, we focus on a well-known learning environment provided by OpenAI Gym, namely "CartPole-v0", which is commonly used in the training and testing of different RL algorithms. In CartPol-v0, the player controls the movement of a cart with a pole standing on top. The cart needs to move left and right dynamically to keep the pole upright. The game ends when either the pole tilts more than 15° or the position of the cart exceeds a given range (2.4 units from the middle to either side). At each time step, the cart decides whether to move left or right, making the dimension of the action space 2. The state at each time step includes the position of the cart, the angle between the pole and the vertical line, the velocity of the cart, and the angular speed of the pole. The reward is fixed at 1 for each time step, and the objective is to play the game as long as possible to maximize accumulated rewards. More details about the CartPole-v0 environment can be found in OpenAI Gym (2021).

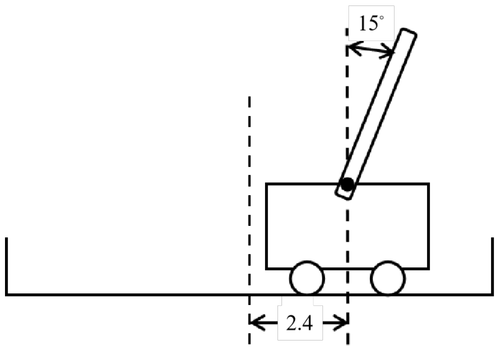


**Figure 10 Diagram of the CartPole-v0 environment.**

To apply AOAT and AOAT-Pi to the CartPole-v0 problem, we implement MuZero and use AOAT and AOAT-Pi as the node selection policies in MuZero. Based on NN-MCTS, MuZero extends AlphaGo Zero from board games to general decision problems with non-sparse rewards, such as Atari games. Specifically, MuZero comprises three NN models: a representation model, a dynamics model, and a prediction model. The representation model encodes observations into hidden states, which serve as the root nodes of NN-MCTS. The dynamics model predicts the next hidden state and future rewards based on the current hidden state and action. The prediction model serves as the value network $f_\theta$ and the policy network $f_\pi$ in AlphaGo Zero, estimating state values and providing prior distributions of actions, respectively. With these three models, MuZero first encodes observations into hidden states and then conducts NN-MCTS on the hidden states to determine actions. MuZero uses the same modified UCT selection policy as AlphGo Zero, described by (20), except that the exogenous parameter $c$ in MuZero is determined by

$$c = c_1 + \log\left(\frac{\sum_{a' \in \mathcal{A}_{s_t^k}} N^{k-1}(s_t^k, a') + c_2 + 1}{c_2}\right),$$

where $c_1 = 1.25$ and $c_2 = 19652$. By implementing MuZero, we simply replace the modified UCT policy in MuZero with our AOAT and AOAT-Pi to evaluate the CartPole-v0 problem. Since the rewards are non-sparse and do not follow the Bernoulli distribution, we apply AOAT and AOAT-Pi under the Gaussian assumption in MuZero. We take the modified UCT policy used in MuZero as the benchmark policy. For more technical details about MuZero, such as the training methods of the NN models, please refer to Schrittwieser et al. (2020). The structures of the NN models in our implementation of MuZero are provided in Section B.4 of the online appendix. The exogenous parameters related to training the NN models on the CartPole-v0 problem are provided in Section B.5 of the online appendix.

## 5.2. Numerical Results on CartPole

Notice that the CartPole-v0 problem has a maximum score (number of steps it can take) of 200, and UCT has already performed well in this problem. Thus, achieving a higher score under a proposed node selection policy is not easy. Table 7 shows the numerical results of the CartPole-v0 problem, where we exhibit the performances of AOAT and AOAT-Pi under the Gaussian assumption with $\sigma^0(s_t,a_t)=4$, leading to the best performance of these two policies when $\sigma^0(s_t,a_t)$ is set to $[1,2,\cdots,10]$. The search budget is fixed at 60. The plus-minus sign in Table 7 is used to present the statistical margin of error of a quantity.

**Table 7 Numerical Results of the CartPole-v0 problem.**

| Policies | UCT | AOAT | AOAT-Pi |
|---|---|---|---|
| Average Score after 500 Iterations | 177±3.3 | 185±4.5 | 191±2.7 |
| Average Number of Iterations Needed to Reach 200 Score | 689±11.2 | 631±16.7 | 605±15.4 |

From Table 7, we can observe that AOAT and AOAT-Pi perform comparably and outperform UCT in terms of the average score obtained after 500 iterations of training. In addition, AOAT and AOAT-Pi require fewer training iterations to reach a score of 200 compared to UCT. Table 8 presents a comparison of the average scores in 200 randomly played games with different search budgets. As the search budget increases, the scores of UCT, AOAT, and AOAT-Pi all exhibit an upward trend. Across all considered search budget levels, both AOAT and AOAT-Pi consistently outperform UCT, with AOAT-Pi showing a slight edge over AOAT. These results strongly indicate the superiority of the proposed AOAT and AOAT-Pi over UCT when applied in MuZero to solve general decision problems such as the CartPole-v0 problem.

**Table 8 Comparison of average scores with different search budgets.**

| Search budget / Policies | 10 | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|---|
| UCT | 159±8.6 | 161±8.5 | 165±8.3 | 168±8.1 | 173±7.9 | 176±7.9 |
| AOAT | 172±8.1 | 176±7.9 | 179±8.2 | 184±8.0 | 187±8.1 | 191±8.0 |
| AOAT-Pi | 174±8.7 | 177±8.4 | 182±8.2 | 186±8.1 | 188±8.1 | 193±8.2 |

## 6. Conclusion and Future Work

In this work, we formulate the node selection problem in MCTS as a multi-stage R&S problem. We then propose a new node selection policy called AOAT and prove its consistency when applied in MCTS. In addition, we incorporate both the value network and policy network in NN-MCTS into AOAT and propose the AOAT-Pi policy. Lastly, we conduct numerical experiments on two board games and a representative RL task to verify the superior performance of the proposed

AOAT and AOAT-Pi over the modified UCT policy used in AlphaGo Zero and MuZero, and also demonstrate the additional benefits to AOAT from the information provided by the value network and the policy network. How to develop node selection policies based on other R&S procedures or BAI procedures and incorporate neural networks into those policies deserves future study. The applications of MCTS with the proposed node selection policy to decision-making problems also deserve future work.

## Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 72325007, 72250065, and 72022001, and PKU-Boya Postdoctoral Fellowship 2406396158. An earlier version of this paper was presented in part at the Proceedings of 2022 Winter Simulation Conference (2022 WSC) (Zhang et al. 2022).

## References

Audibert JY, Bubeck S (2010) Best arm identification in multi-armed bandits. *COLT-23th Conference on learning theory-2010*, 13–p.

Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47:235–256.

Bellman R (1954) The theory of dynamic programming. *Bulletin of the American Mathematical Society* 60(6):503–515.

Bubeck S, Munos R, Stoltz G (2009) Pure exploration in multi-armed bandits problems. *Algorithmic Learning Theory: 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings 20*, 23–37 (Springer).

Chang HS, Fu MC, Hu J, Marcus SI (2005) An adaptive sampling algorithm for solving markov decision processes. *Operations Research* 53(1):126–139.

Chen CH, Lin J, Yücesan E, Chick SE (2000) Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems* 10:251–270.

Glynn P, Juneja S (2004) A large deviations perspective on ordinal optimization. *Proceedings of the 2004 Winter Simulation Conference (WSC)*, volume 1, 577–585 (IEEE).

Kaufmann E, Koolen WM (2017) Monte-carlo tree search by best arm identification. *Advances in Neural Information Processing Systems* 30:1–10.

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Kocsis L, Szepesvári C (2006) Bandit based monte-carlo planning. *European conference on machine learning*, 282–293 (Springer).

Lattimore T, Szepesvári C (2020) *Bandit algorithms* (Cambridge University Press).

Li H, Lam H, Peng Y (2024) Efficient learning for clustering and optimizing context-dependent designs. *Operations Research* 72(2):617–638.

Li Y, Fu MC, Xu J (2021) An optimal computing budget allocation tree policy for monte carlo tree search. *IEEE Transactions on Automatic Control* 67(6):2685–2699.

Liu X, Lam H, Peng Y (2023) Training deep q-network via monte carlo tree search for adaptive bitrate control in video delivery. *Available at SSRN 4757662* .

Liu X, Peng Y, Zhang G, Zhou R (2024) An efficient node selection policy for monte carlo tree search with neural networks URL http://dx.doi.org/10.1287/ijoc.2023.0307.cd, available for download at https://github.com/INFORMSJoC/2023.0307.

Loeffler TD, Banik S, Patra TK, Sternberg M, Sankaranarayanan SK (2021) Reinforcement learning in discrete action space applied to inverse defect design. *Journal of Physics Communications* 5(3):031001.

Mansley C, Weinstein A, Littman M (2011) Sample-based planning for continuous action markov decision processes. *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 21, 335–338.

OpenAI Gym (2021) Openai gym cartpole-v0. https://github.com/openai/gym/wiki/CartPole-v0, Last edited on 2021-09-29.

Peng Y, Chong EK, Chen CH, Fu MC (2018) Ranking and selection as stochastic control. *IEEE Transactions on Automatic Control* 63(8):2359–2373.

Peng Y, Zhang G (2022) Thompson sampling meets ranking and selection. *Proceedings of the 2022 Winter Simulation Conference (WSC)*, 3075–3086 (IEEE).

Russo D (2020) Simple bayesian algorithms for best-arm identification. *Operations Research* 68(6):1625–1647.

Russo DJ, Van Roy B, Kazerouni A, Osband I, Wen Z, et al. (2018) A tutorial on thompson sampling. *Foundations and Trends® in Machine Learning* 11(1):1–96.

Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, Guez A, Lockhart E, Hassabis D, Graepel T, Lillicrap T, Silver D (2020) Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature* 588(7839):604–609.

Shin D, Broadie M, Zeevi A (2018) Tractable sampling strategies for ordinal optimization. *Operations Research* 66(6):1693–1712.

Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, et al. (2017) Mastering the game of go without human knowledge. *Nature* 550(7676):354–359.

Teraoka K, Hatano K, Takimoto E (2014) Efficient sampling method for monte carlo tree search problem. *IEICE Transactions on Information and Systems* 97(3):392–398.

Tesauro G, Rajan V, Segal R (2012) Bayesian inference in monte-carlo tree search. *arXiv preprint arXiv:1203.3519* .

Yin ZH, Ye W, Chen Q, Gao Y (2022) Planning for sample efficient imitation learning. *Advances in Neural Information Processing Systems* 35:2577–2589.

Zhang G, Chen B, Jia QS, Peng Y (2023a) Efficient sampling policy for selecting a subset with the best. *IEEE Transactions on Automatic Control* 68(8):4904–4911.

Zhang G, Chen S, Huang K, Peng Y (2024a) Efficient learning for selecting top-$m$ context-dependent designs. *IEEE Transactions on Automation Science and Engineering, published online, DOI:10.1109/TASE.2024.3391020* .

Zhang G, Li H, Liu X, Peng Y (2024b) Simulation budget allocation for improving scheduling and routing of automated guided vehicles in warehouse management. *Journal of the Operations Research Society of China, published online, DOI:10.1007/s40305-024-00553-0* .

Zhang G, Li H, Peng Y (2020) Sequential sampling for a ranking and selection problem with exponential sampling distributions. *Proceedings of the 2020 Winter Simulation Conference (WSC)*, 2984–2995 (IEEE).

Zhang G, Peng Y, Xu Y (2022) An efficient dynamic sampling policy for monte carlo tree search. *Proceedings of the 2022 Winter Simulation Conference (WSC)*, 2760–2771 (IEEE).

Zhang G, Peng Y, Zhang J, Zhou E (2023b) Asymptotically optimal sampling policy for selecting top-m alternatives. *INFORMS Journal on Computing* 35(6):1261–1285.

# Online Appendix for "An Efficient Node Selection Policy for Monte Carlo Tree Search with Neural Networks"

Xiaotian Liu
Department of Management Science and Information Systems, Guanghua School of Management, Peking University, Beijing 100871, China, xiaotianliu01@gmail.com

Yijie Peng
Department of Management Science and Information Systems, Guanghua School of Management, Peking University, Beijing 100871, China, pengyijie@pku.edu.cn

Gongbo Zhang
Department of Management Science and Information Systems, Guanghua School of Management, Peking University, Beijing 100871, China, gongbozhang@pku.edu.cn

Ruihan Zhou
Department of Management Science and Information Systems, Guanghua School of Management, Peking University, Beijing 100871, China, rhzhou@stu.pku.edu.cn

## A. Proof of Propositions

*Proof of Proposition 1* Note that both Beta distribution and Gaussian distribution are continuous probability distributions. By the definition of the KL divergence, we have

$$\begin{aligned} D_{KL}(Beta||Gaussian) &= \int_0^1 f_B(x;\alpha',\beta') \ln \frac{f_B(x;\alpha',\beta')}{f_N(x;Q^{k-1},\sigma^{k-1})} dx \\ &= \int_0^1 f_B(x;\alpha',\beta') \ln f_B(x;\alpha',\beta') dx \\ &\quad - \int_0^1 f_B(x;\alpha',\beta') \ln f_N(x;Q^{k-1},\sigma^{k-1}) dx \ , \end{aligned} \tag{1}$$

where we suppress $(s_t^k, a_t)$ for simplicity of notation,

$$f_B(x;\alpha',\beta') = \frac{\Gamma(\alpha'+\beta')}{\Gamma(\alpha')\Gamma(\beta')} x^{\alpha'-1}(1-x)^{\beta'-1}, \quad 0 \le x \le 1 \ ,$$

is the probability density function (PDF) of the Beta distribution, $\Gamma(z) = \int_0^\infty t^{z-1}\exp(-t)dt$ is the gamma function, and

$$f_N(x;Q^{k-1},\sigma^{k-1}) = \left(\sqrt{2\pi\sigma^{k-1}}\right)^{-1} \exp\left( - \frac{(x-Q^{k-1})^2}{2\sigma^{k-1}} \right) \ ,$$

is the PDF of the Gaussian distribution. Following the definition of $\alpha'(s_t^k, a_t)$ and $\beta'(s_t^k, a_t)$, we have $\lim_{N^{k-1}(s_t^k,a_t)\to\infty} \alpha'(s_t^k, a_t) = \infty$, and $\lim_{N^{k-1}(s_t^k,a_t)\to\infty} \beta'(s_t^k, a_t) = \infty$. Then the first term in the second equality of (1) can be calculated as

$$\begin{aligned}
&\int_0^1 f_B(x;\alpha',\beta')\ln f_B(x;\alpha',\beta')dx = \mathbb{E}_{X\sim f_B(x;\alpha',\beta')}\left[\ln f_B(x;\alpha',\beta')\right]\\
=& -\ln\Gamma(\alpha') - \ln\Gamma(\beta') + \ln\Gamma(\alpha'+\beta') + (\alpha'-1)\left.\frac{d\ln\Gamma(y)}{dy}\right|_{y=\alpha'} + (\beta'-1)\left.\frac{d\ln\Gamma(y)}{dy}\right|_{y=\beta'}\\
&-(\alpha'+\beta'-2)\left.\frac{d\ln\Gamma(y)}{dy}\right|_{y=\alpha'+\beta'}\\
=& -\frac{1}{2}\left[1+\ln 2\pi - 3\ln(\alpha'+\beta') + \ln(\alpha'\beta')\right] + \frac{1}{3\alpha'} + \frac{1}{3\beta'} - \frac{5}{6(\alpha'+\beta')} + o\left(\frac{1}{N^{k-1}(s_t^k,a_t)}\right) ,
\end{aligned}$$

where the second equality holds since $\alpha'(s_t^k, a_t) + \beta'(s_t^k, a_t) = \alpha(s_t^k, a_t) + \beta(s_t^k, a_t) + N^{k-1}(s_t^k, a_t)$, and as $z\to\infty$,

$$\ln\Gamma(z) = (z-\frac{1}{2})\ln z - z + \frac{1}{2}\ln 2\pi + \frac{1}{12z} + o\left(\frac{1}{z}\right) ,$$

and

$$\frac{d\ln\Gamma(z)}{dz} = \ln z - \frac{1}{2z} - \frac{1}{12z^2} + o(x) ,$$

where $f(x) = o(x)$ if $f(x)/x \to 0$ as $x\to 0$.

The second term in the second equality of (1) can be calculated as

$$\begin{aligned}
&\int_0^1 f_B(x;\alpha',\beta')\ln f_N(x;Q^{k-1},\sigma^{k-1})dx\\
=&\int_0^1 f_B(x;\alpha',\beta')\left(-\ln(\sqrt{2\pi\sigma^{k-1}}) - \frac{(x-Q^{k-1})^2}{2\sigma^{k-1}}\right)dx\\
=&-\ln(\sqrt{2\pi\sigma^{k-1}}) - \frac{1}{2\sigma^{k-1}}\int_0^1 f_B(x;\alpha',\beta')(x-Q^{k-1})^2dx\\
=&-\ln(\sqrt{2\pi\sigma^{k-1}}) - \frac{(Q^{k-1}\beta')^2 + Q^{k-1}\beta'\left[2\alpha'(Q^{k-1}-1)+Q^{k-1}\right] + \alpha'(\alpha'+1)(Q^{k-1}-1)^2}{2\sigma^{k-1}(\alpha'+\beta'+1)(\alpha'+\beta')}\\
=&\ln(\alpha'+\beta') - \frac{1}{2}\left[1+\ln 2\pi + \ln(\alpha'\beta') - \ln(\alpha'+\beta'+1)\right] ,
\end{aligned}$$

where the third equality holds by substituting $Q^{k-1}(s_t^k, a_t)$ and $\sigma^{k-1}(s_t^k, a_t)$ into the second equality. Summarizing the above, we have as $N^{k-1}(s_t^k, a_t)\to\infty$,

$$\begin{aligned}
D_{KL}(Beta||Gaussian) =& \frac{\alpha'+\beta'}{3\alpha'\beta'} + \frac{1}{2}\ln\frac{\alpha'+\beta'}{\alpha'+\beta'+1} - \frac{5}{6(\alpha'+\beta')} + o\left(\frac{1}{N^{k-1}(s_t^k,a_t)}\right)\\
=&\frac{1}{3Q^{k-1}\cdot(1-Q^{k-1})} + \frac{1}{2}\ln\left(1-\frac{1}{\alpha'+\beta'+1}\right) - \frac{5}{6(\alpha'+\beta')} + o\left(\frac{1}{N^{k-1}(s_t^k,a_t)}\right)\\
=&\frac{1}{N^{k-1}(s_t^k,a_t)}\left(\frac{1}{3Q^*\cdot(1-Q^*)} - \frac{4}{3}\right) + o\left(\frac{1}{N^{k-1}(s_t^k,a_t)}\right),
\end{aligned}$$

where the third equality holds since $\lim_{N^{k-1}(s_t^k,a_t)\to\infty} Q^{k-1}(s_t^k, a_t) = Q^*(s_t, a_t)$ and $\ln(1-z) = -z + o(z), \forall z\in[-1,1)$. Therefore, we have $D_{KL}(Beta||Gaussian) = O\big(\frac{1}{N^{k-1}(s_t^k,a_t)}\big)$. With $N^{k-1}(s_t^k, a_t)\to\infty$, we have $\lim_{N^{k-1}(s_t^k,a_t)\to\infty} D_{KL}(Beta||Gaussian) = 0$. □

*Proof of Proposition 2* The process of finding the best action for each explored state is adapted from AOAP as proposed in Peng et al. (2018). With the proof of the consistency of AOAP in Theorem 3 in Peng et al. (2018), which shows that every action of a certain state will be sampled infinitely often almost surely as the number of rollouts goes to infinity, the consistency of AOAP can be proved by induction.

Suppose at step $\widetilde{k}$, all states $s_\ell \in \mathcal{S}_t$, $t_0 \le \ell \le H$ are expanded. $\forall\ k > \widetilde{k}$, running AOAP at each state $s_t \in \mathcal{S}_t$, $t_0 \le t < H$ reduces to a single-stage R&S problem. For the state $s_{t_0}$, all state-action pairs $(s_{t_0}, a_{t_0})$ will be sampled infinitely often almost surely, leading to $\lim_{K\to\infty} N^K(s_{t_0}, a_{t_0}) \to \infty$, *a.s.*. Suppose $\exists\ j,\ t_0 < j < H$ such that $\lim_{K\to\infty} N^K(s_h, a_h) \to \infty$, *a.s.*, $s_h \in \mathcal{S}_h,\ t_0 \le h < j$, and then following AOAP, $(s_j, a_j)$ will be sampled infinitely often almost surely, leading to $\lim_{K\to\infty} N^K(s_j, a_j) \to \infty$, *a.s.*. Summarizing the above, it follows that $\lim_{K\to\infty} N^K(s_t, a_t) \to \infty$, *a.s.*, $\forall\ s_t \in \mathcal{S}_t$ and $\lim_{K\to\infty} \hat{a}_t^K = a_t^*$, $t_0 \le t < H$ will follow by the Law of Large Numbers (LLN).

Then for the state $s_{H-1}$, with $\lim_{K\to\infty} N^K(s_{H-1}, a_{H-1}) \to \infty$, *a.s.* and the LLN, we have $\lim_{K\to\infty} Q^K(s_{H-1}, a_{H-1}) = Q^*(s_{H-1}, a_{H-1})$, and

$$\begin{aligned}\lim_{K\to\infty} \widehat{V}^K(s_{H-1}) &= \lim_{K\to\infty} \max_{a_{H-1}\in\mathcal{A}_{s_{H-1}}} Q^K(s_{H-1}, a_{H-1}) \\ &= \max_{a_{H-1}\in\mathcal{A}_{s_{H-1}}} \lim_{K\to\infty} Q^K(s_{H-1}, a_{H-1}) \\ &= V^*(s_{H-1})\ ,\end{aligned}$$

where the second equality holds since we assume the reward of the MDP is bounded. Suppose $\exists\ \widetilde{j},\ t_0 < \widetilde{j} < H$ such that $\lim_{K\to\infty} V^K(s_{\widetilde{h}}) = V^*(s_{\widetilde{h}})$, $s_{\widetilde{h}} \in \mathcal{S}_{\widetilde{h}}$, $\widetilde{j} \le \widetilde{h} < H$, and then with $\lim_{K\to\infty} N^K(s_{\widetilde{j}-1}, a_{\widetilde{j}-1}) \to \infty$, *a.s.* and the LLN, we have

$$\lim_{K\to\infty} Q^K(s_{\widetilde{j}-1}, a_{\widetilde{j}-1}) = \lim_{N^K(s_{\widetilde{j}-1}, a_{\widetilde{j}-1})\to\infty} Q^K(s_{\widetilde{j}-1}, a_{\widetilde{j}-1}) = \mathbb{E}[\mathcal{R}(s_{\widetilde{j}-1}, a_{\widetilde{j}-1})] + V^*(s_{\widetilde{j}}) = Q^*(s_{\widetilde{j}-1}, a_{\widetilde{j}-1})\ ,$$

and $\lim_{K\to\infty} V^K(s_{\widetilde{j}-1}) = V^*(s_{\widetilde{j}-1})$. Summarizing the above, the proposition is proved. $\square$

*Proof of Proposition 3* We first prove that each action at the root node will be sampled infinitely often, almost surely, following the UCT, i.e., $\lim\limits_{k\to\infty} N^k(s_{t_0}, a_{t_0}) = \infty$, $a_{t_0} \in \mathcal{A}_{s_{t_0}}$. Suppose that $a_{t_0} \in \mathcal{A}_{s_{t_0}}$ is sampled finitely often a.s., whereas $\widetilde{a}_{t_0} \in \mathcal{A}_{s_{t_0}}$, $\widetilde{a}_{t_0} \ne a_{t_0}$ is sampled infinitely often a.s., then sampling $a_{t_0}$ still leads to a higher UCT value, which contradicts with the sampling rule that the action with the highest UCT value is sampled in UCT. Therefore, $\lim\limits_{k\to\infty} N^k(s_{t_0}, a_{t_0}) = \infty$, and $\lim\limits_{k\to\infty} \bar{Q}^{k-1}(s_{t_0}, a_{t_0}) = Q^*(s_{t_0}, a_{t_0})$, $a_{t_0} \in \mathcal{A}_{s_{t_0}}$ following the law of large numbers.

Then we prove the almost surely convergence of the sampling ratios of UCT. We replace sample mean $\bar{Q}^{k-1}(s_{t_0}, a_{t_0})$ with $Q^*(s_{t_0}, a_{t_0})$ for simplicity of analysis. Notice that $\boldsymbol{r}^k \triangleq \left(r^k\left(s_{t_0}, a_{t_0}^*\right), r^k(s_{t_0}, a_{t_0}), \cdots\right)$, $\forall\ a_{t_0} \in \mathcal{A}_{s_{t_0}}$, $a_{t_0} \ne a_{t_0}^*$ is a bounded sequence. Following the Bolzano-Weierstrass theorem, there exists a subsequence of $\boldsymbol{r}^k$ converging to

$\widetilde{\boldsymbol{r}} \triangleq \left(\widetilde{r}\left(s_{t_0}, a_{t_0}^*\right), \widetilde{r}\left(s_{t_0}, a_{t_0}\right), \cdots\right)$, $\forall\, a_{t_0} \in \mathcal{A}_{s_{t_0}}$, $a_{t_0} \neq a_{t_0}^*$, where $\sum_{a_{t_0} \in \mathcal{A}_{s_{t_0}}} \widetilde{r}\left(s_{t_0}, a_{t_0}\right) = 1$, $\widetilde{r}\left(s_{t_0}, a_{t_0}\right) \geq 0$. Without loss of generality, we assume that $\boldsymbol{r}^k$ converges to $\widetilde{\boldsymbol{r}}$; otherwise, the following argument is made over a subsequence.

Since $\sum_{a_{t_0} \in \mathcal{A}_{s_{t_0}}} \widetilde{r}\left(s_{t_0}, a_{t_0}\right) = 1$, we have $\exists\, a_{t_0} \in \mathcal{A}_{s_{t_0}}$ such that $\widetilde{r}\left(s_{t_0}, a_{t_0}\right) > 0$. We then claim that $\widetilde{r}\left(s_{t_0}, a_{t_0}\right) = 0$, $\forall\, a_{t_0} \in \mathcal{A}_{s_{t_0}}$, $a_{t_0} \neq a_{t_0}^*$; otherwise, $\exists\, \widetilde{a}_{t_0} \in \mathcal{A}_{s_{t_0}}$, $\widetilde{a}_{t_0} \neq a_{t_0}^*$ such that $\widetilde{r}\left(s_{t_0}, \widetilde{a}_{t_0}\right) > 0$. Then we have $\exists\, T_0 > 0$ such that $\forall\, k > T_0$, action $a_{t_0}^*$ will be sampled and action $\widetilde{a}_{t_0}$ will stop being sampled since $Q^*\left(s_{t_0}, a_{t_0}^*\right) > Q^*\left(s_{t_0}, \widetilde{a}_{t_0}\right)$, and $\lim\limits_{k \to \infty} \sqrt{\frac{2 \ln k}{r^{(k-1)}\left(s_{t_0}, \widetilde{a}_{t_0}\right) \cdot (k-1)}} = 0$, which contradicts the assumption that $\boldsymbol{r}^k$ converges to $\widetilde{\boldsymbol{r}}$. Since $\sum_{a_{t_0} \in \mathcal{A}_{s_{t_0}}} \widetilde{r}\left(s_{t_0}, a_{t_0}\right) = 1$, then we have $\widetilde{r}\left(s_{t_0}, a_{t_0}^*\right) = 1$. Summarizing the above, the proposition is proved. $\square$

## B. Supplementary on Numerical Experiments

In this section, we provide additional discussions regarding the neural networks used in the numerical experiments described in Sections 4 and 5 of the main text.

### B.1 Structure of neural networks for board games

**Table 1 Structure of the Value Network.**

| Module | Layer | Parameter |
|---|---|---|
| Input Encoder × 1 | Convolution | Filters 16, Kernel Size 1×1, Stride 1 |
| | Batch Normalizer | - |
| | ReLU Activation | - |
| Residual Block × 2 | Convolution | Filters 16, Kernel Size 3×3, Stride 1 |
| | Batch Normalizer | - |
| | ReLU Activation | - |
| | Convolution | Filters 16, Kernel Size 3×3, Stride 1 |
| | Batch Normalizer | - |
| | Skip Connection from Input | - |
| | ReLU Activation | - |
| Head Decoder × 1 | Convolution | Filters 1, Kernel Size 1×1, Stride 1 |
| | Batch Normalizer | - |
| | Fully Connected | Width 1 |
| | Tanh Activation | - |
| | Normalizer | $\frac{x+1}{2}$ |

### B.2 Exogenous parameters for the training procedure of neural networks for board games

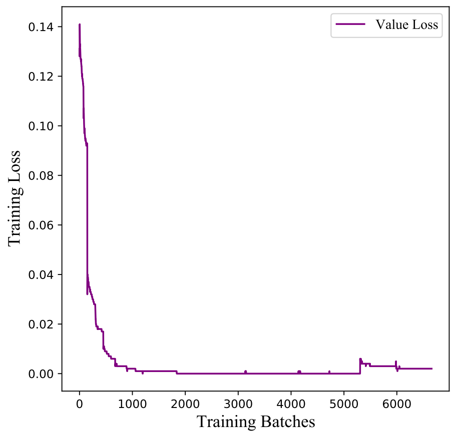
**Table 2** Exogenous Parameters in Training of NNs for Tic-Tac-Toe.

| **Parameters** | **Value** |
|---|---|
| CPU Number $N$ | 5 |
| Number of Games Simulated on 1 CPU for 1 Iteration | 20 |
| Number of Iterations $I$ | 35 |
| Parameter $c$ in UCT | 1 |
| Search Budget $K$ | 100 |
| Epochs $E$ | 2 |
| Batch Size | 64 |
| Learning Rate | 5e-4 |

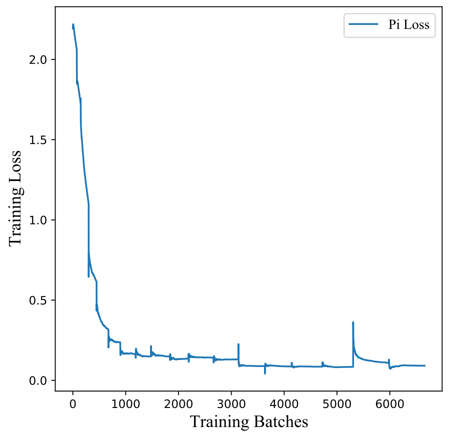
**Table 3** Exogenous Parameters in Training of NNs for Five-In-A-Row.

| **Parameters** | **Value** |
|---|---|
| CPU Number $N$ | 10 |
| Number of Games Simulated on 1 CPU for 1 Iteration | 80 |
| Number of Iterations $I$ | 10 |
| Parameter $c$ in UCT | 1 |
| Search Budget $K$ | 500 |
| Epochs $E$ | 2 |
| Batch Size | 128 |
| Learning Rate | 5e-4 |

## B.3 Training losses of the value network and policy network for Tic-Tac-Toe



(a) Training loss of the value network



(b) Training loss of the policy network

**Figure 1** Training losses of the value network and policy network for Tic-Tac-Toe.

## B.4 Structure of neural networks for the CartPole-v0 problem

**Table 4 Structure of NNs in MuZero for the CartPole-v0 problem.**

| Model | Layer | Size | Number |
|---|---|---|---|
| Representation Model | Fully Connected | (12, 50) | 1 |
| | Fully Connected<br>ReLU Activation | (50, 50)<br>- | 3 |
| | Fully Connected | (50, 50) | 1 |
| Dynamics Model | Fully Connected | (51, 50) | 1 |
| | Fully Connected<br>ReLU Activation | (50, 50)<br>- | 3 |
| | Fully Connected | (50, 51) | 1 |
| Prediction Model | Fully Connected<br>ReLU Activation | (50, 50)<br>- | 4 |
| | Fully Connected | (50, 3) | 1 |

## B.5 Exogenous parameters for the training procedure of neural networks for the CartPole-v0 problem

**Table 5 Exogenous Parameters in Training of NNs for the CartPole-v0 problem.**

| Parameters | Value |
|---|---|
| CPU Number $N$ | 1 |
| History Length | 3 |
| Number of Steps Rollout Forward | 6 |
| Number of Iterations $I$ | 500 |
| Search Budget $K$ | 60 |
| Epochs $E$ | 2 |
| Batch Size | 32 |
| Learning Rate | 1e-3 |

## References

Peng Y, Chong EK, Chen CH, Fu MC (2018) Ranking and selection as stochastic control. *IEEE Transactions on Automatic Control* 63(8):2359–2373.