Counterfactual Online Learning for Open-Loop Monte-Carlo Planning

Thomy Phan[1], **Shao-Hung Chan**[1], **Sven Koenig**[2]

[1]University of Southern California
[2]University of California, Irvine
{thomy.phan,shaohung}@usc.edu, sven.koenig@uci.edu

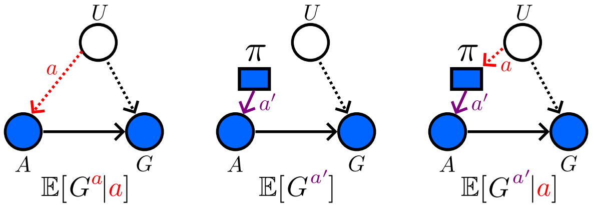
Abstract

Monte-Carlo Tree Search (MCTS) is a popular approach to online planning under uncertainty. While MCTS uses statistical sampling via multi-armed bandits to avoid exhaustive search in complex domains, common *closed-loop approaches* typically construct enormous search trees to consider a large number of potential observations and actions. On the other hand, *open-loop approaches* offer better memory efficiency by ignoring observations but are generally not competitive with closed-loop MCTS in terms of performance – even with commonly integrated human knowledge. In this paper, we propose *Counterfactual Open-loop Reasoning with Ad hoc Learning (CORAL)* for open-loop MCTS, using a causal *multi-armed bandit approach with unobserved confounders (MABUC)*. CORAL consists of two online learning phases that are conducted during the open-loop search. In the first phase, *observational values* are learned based on preferred actions. In the second phase, *counterfactual values* are learned with MABUCs to make a decision via an intent policy obtained from the observational values. We evaluate CORAL in four POMDP benchmark scenarios and compare it with closed-loop and open-loop alternatives. In contrast to standard open-loop MCTS, CORAL achieves competitive performance compared with closed-loop algorithms while constructing significantly smaller search trees.

1 Introduction

Many real-world problems can be modeled as *Partially Observable Markov Decision Process (POMDP)*, where the true state is unknown to the agent due to limited and noisy sensors, to model planning and reinforcement learning problems (Ross et al. 2008). However, solving POMDPs exactly is intractable for domains with enormous state spaces and long planning horizons due to the *curse of dimensionality* (Kaelbling, Littman, and Cassandra 1998) and the *curse of history*, respectively (Pineau, Gordon, and Thrun 2006).

Monte-Carlo Tree Search (MCTS) is a popular approach to online planning in POMDPs (Świechowski et al. 2023). MCTS breaks both curses with statistical sampling via *multi-armed bandits* and simulation. *Closed-loop MCTS* constructs sparse trees over actions and observations and represents the state-of-the-art for efficient planning in large POMDPs (Silver and Veness 2010; Somani et al. 2013; Bai et al. 2014). However, the constructed closed-loop trees can still become arbitrarily large for highly complex domains, therefore requiring considerable memory resources (Phan et al. 2019a; Powley, Cowling, and Whitehouse 2017).



(a) observational (b) interventional (c) counterfactual

Figure 1: Different decision-making models and expected returns with unobserved confounder U, decision A (i.e., an action), and payoff G (i.e., the return) based on [Pearl 2010]. In each model, blue nodes denote observable variables. Dashed lines illustrate the influence of unobserved variables, and solid lines the influence of the agent. **(a)** Natural or observational decision-making through an *intent* a, solely influenced by U. **(b)** Classic or interventional decision-making via policy π that ignores the existence of U. **(c)** Counterfactual decision-making that conditions on the intent of (a).

Open-loop MCTS offers better memory efficiency by ignoring observations, resulting in significantly smaller search trees (Lecarpentier et al. 2018; Perez Liebana et al. 2015; Phan et al. 2019a). However, open-loop MCTS is typically not competitive with closed-loop MCTS in terms of performance – even with commonly integrated human knowledge (Silver and Veness 2010). Due to the lack of context information, open-loop MCTS only converges to decisions that are suitable "on average" for unobserved situations (Weinstein and Littman 2012). Considering the existence of unknown structural properties of the underlying problem, i.e., *unobserved confounders*, may be helpful to improve the performance of memory-efficient open-loop planning, though (Bareinboim, Forney, and Pearl 2015).

In this paper, we propose *Counterfactual Open-loop Reasoning with Ad hoc Learning (CORAL)* for open-loop MCTS, using a causal *multi-armed bandit approach with un-*

observed confounders (MABUC) (Bareinboim, Forney, and Pearl 2015). Unlike standard multi-armed bandits, as illustrated in Fig. 1b, MABUCs select actions based on some "natural" *intent*, i.e., an intuitive action, depending on structural (but unknown) properties of the underlying problem, as illustrated in Fig. 1c. CORAL consists of two online learning phases that are conducted during the open-loop search. In the first phase, *observational values* are learned based on preferred actions (Silver and Veness 2010). In the second phase, *counterfactual values* are learned with MABUCs to make a decision via an intent policy obtained from the observational values. Our contributions are as follows:

- We adjust MABUCs for the heuristic search setting, where an intent policy needs to be obtained from prior human knowledge to conduct counterfactual learning.
- We use the modified MABUCs to formulate CORAL, an open-loop MCTS algorithm that learns observational and counterfactual values on the fly during the search.
- We evaluate CORAL in four POMDP benchmark scenarios and compare it with MCTS alternatives. In contrast to standard open-loop MCTS, CORAL achieves competitive performance compared with closed-loop algorithms while constructing significantly smaller search trees.

2 Background

2.1 Problem Setting

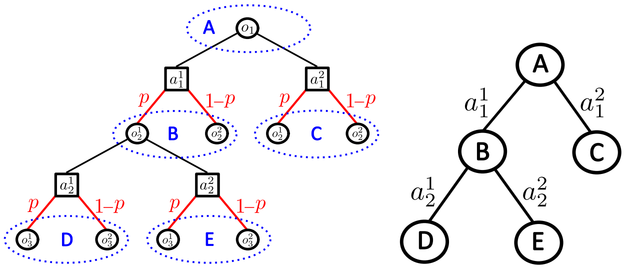
A POMDP is defined by $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \Omega, b_0 \rangle$, where $\mathcal{S}$ is a (finite) set of states, $\mathcal{A}$ is the (finite) set of actions, $\mathcal{P}(s_{t+1}|s_t, a_t)$ is the transition probability, $\mathcal{R}(s_t, a_t) = r_t \in \mathbb{R}$ is the reward, $\mathcal{O}$ is a (finite) set of observations, $\Omega(o_{t+1}|s_{t+1}, a_t)$ is the observation probability, and b_0 is a probability distribution over initial states $s_0 \in \mathcal{S}$ (Kaelbling, Littman, and Cassandra 1998). We define $\mathcal{A}_{legal}(s_t) \subseteq \mathcal{A}$ as the set of *executable actions* at s_t.

The agent maintains a *history* $h_t = \langle a_0, o_1, ..., a_{t-1}, o_t \rangle$ of past actions and observations. A *belief state* $b(s_t|h_t)$ is a sufficient statistic for history h_t and defines a distribution over states s_t. $b(s_t|h_t)$ can be updated by Bayes rule with b_0 representing the prior distribution (Kaelbling, Littman, and Cassandra 1998). The goal is to find a *policy* $\pi(h_t) \in \mathcal{A}$, which maximizes the expectation of *return* $G_t = \sum_{k=0}^{T-1} \gamma^k r_{t+k}$ for each state $s_t \in \mathcal{S}$ for a *horizon* or *search depth* T, where $\gamma \in [0, 1]$ is the discount factor.

The *value function* $V^\pi(h_t) = \mathbb{E}_\pi[G_t|h_t]$ is the expected return conditioned on history h_t given a policy π. An optimal policy π^* has a value function $V^{\pi^*} = V^*$ with $V^*(h_t) \geq V^{\pi'}(h_t)$ for all $h_t \in (\mathcal{A} \times \mathcal{O})^T$ and $\pi' \neq \pi^*$.

2.2 Monte-Carlo Planning in POMDPs

We focus on *Monte-Carlo planning*, where $\hat{M} \approx M$ is a generative model that can be used as a black box simulator (Silver and Veness 2010; Weinstein and Littman 2013). Given s_t and a_t, the simulator $\hat{M}$ provides a sample $\langle s_{t+1}, o_{t+1}, r_t \rangle \sim \hat{M}(s_t, a_t)$. Monte-Carlo planning approximates π^* and V^* by iteratively simulating and evaluating action sequences without requiring explicit probability distributions of the POMDP M. We focus on *online planning* where search and execution are interleaved, using a fixed *simulation budget* n_b, i.e., number of iterations, per time step t (Weinstein and Littman 2013).



(a) Closed-loop tree

(b) Open-loop tree

Figure 2: Closed- and open-loop planning schemes. **(a)** Closed-loop tree with observations (circular nodes) and actions (rectangular nodes). Red links indicate stochastic observations. **(b)** Open-loop tree with links as actions and summarized histories, according to the blue dotted ellipses in (a).

Planning can be closed- or open-loop. *Closed-loop planning* conditions the action selection on histories h_t of actions and observations. *Open-loop planning* only conditions the action selection on previous sequences of actions $\langle a_0, ..., a_{T-1} \rangle$ and summarized statistics about histories (Bubeck and Munos 2010; Weinstein and Littman 2013). An example is shown in Fig. 2. A closed-loop tree for a stochastic domain is shown in Fig. 2a, while Fig. 2b shows the corresponding open-loop tree, which summarizes the histories of Fig. 2a within the blue dotted ellipses.

Closed-Loop Monte-Carlo Planning *Partially Observable Monte-Carlo Planning (POMCP)* is a closed-loop approach based on *Monte-Carlo Tree Search (MCTS)* and the state-of-the-art for online planning in large POMDPs (Silver and Veness 2010). POMCP uses a search tree of histories with *o-nodes* representing observations and *a-nodes* representing actions of a history (Fig. 2a). Each *o-node* maintains a *multi-armed bandit* representing the *tree policy* π_{tree}, and each *a-node* represents an *arm* with a selection count n^a, a value estimate $\overline{G^a} \approx V(h_t)$, and a value variance estimate $(\sigma^a)^2$ for history h_t. A simulation starts at the root *o-node*, also representing the current belief state $b(s_t|h_t)$, by initially sampling a state $s_t \sim b(\cdot|h_t)$. For tree traversal, the policy π_{tree} selects *a-nodes*, whose represented actions are simulated with the generative model $\hat{M}$ to determine the next *o-node* to visit. After reaching a leaf *o-node*, the node is expanded, and a *rollout policy* $\pi_{rollout}$ is used to simulate sampled actions until a terminal state is reached or a maximum search depth T is exceeded. The observed rewards r_{t+k} are accumulated to returns G_{t+k} and propagated back to update all *o-* and *a-nodes* at each tree level k in the search path.

Open-Loop Monte-Carlo Planning (Lecarpentier et al. 2018; Phan et al. 2019a) formulate *open-loop MCTS variants*, which can be applied to POMDPs by constructing search trees that summarize all *o-nodes* to history distribution nodes, as illustrated in Fig. 2b.

Open-loop planning generally converges to sub-optimal solutions in stochastic domains due to ignoring past observations to optimize the node values $\hat{V}(N_t) \approx \sum_{s_t \in \mathcal{S}} b(s_t|h_t)V(h_t)$ as weighted averages *over all history values* $V(h_t)$ (Fig. 2b) (Lecarpentier et al. 2018). However, if the problem is too large to provide a sufficient simulation budget n_b or memory, then open-loop planning can be useful due to exploring a much smaller search space for reasonable decision-making (Weinstein and Littman 2013).

2.3 Bandits with Unobserved Confounders

Multi-armed Bandits (MAB) *MABs or bandits* are fundamental decision-making problems with a single state s, a set of actions $a \in \mathcal{A}$, and a stochastic payoff function $\mathcal{X}(s, a) := G^a$, where G^a is a random variable with an unknown distribution. The goal is to determine an action a', which maximizes the expected payoff $\mathbb{E}\big[G^{a'}\big]$, as illustrated in Fig. 1b. The agent has to balance between sufficiently trying out actions to estimate their expected payoff accurately and exploiting its current knowledge by greedily selecting the action with the currently highest estimated payoff. This is known as the *exploration-exploitation dilemma*, where exploration can find actions with better payoffs but requires time for trying them out, while exploitation can lead to fast convergence but possibly gets stuck in a local optimum. Our paper focuses on *Thompson Sampling* (Thompson 1933).

MABs with Unobserved Confounders (MABUC) We assume a *Structural Causal Model (SCM)*, as illustrated in Fig. 1 (Pearl 2010). Each SCM is associated with a directed acyclic graph $\mathcal{G}$, a set of *endogenous* (observed) variables W, and a set of *exogenous* (unobserved) variables U. Edges in $\mathcal{G}$ correspond to functional relationships between endogenous variables $w_i \in W$, i.e., $w_i \leftarrow f_i(W \setminus \{w_i\}, u_i)$, where $u_i \subseteq U$ with probability $P(u_i)$. Given two endogenous variables A and G, we define the *counterfactual* expression $g = G^a$ meaning "G would be g, if A had been a" (Forney, Pearl, and Bareinboim 2017).

A MABUC is a bandit model, where for each trial, some unobserved variable or confounder $u \subseteq U$ is emitted that triggers an *intent* $a \in \mathcal{A}$ via $a = f(u)$, as shown in Fig. 1a. A *counterfactual policy* π determines the agent's decision via $a' = \pi(a) = \pi(f(u))$, as illustrated in Fig. 1c.

The goal is to determine an action a', which maximizes the *regret decision criterion (RDC)* or the *expected counterfactual value* $\mathbb{E}\big[G^{a'}|a\big]$ given an intent a that was caused by a variable $u \in U$ (Bareinboim, Forney, and Pearl 2015).

MABUC is a generalization of the standard MAB, which would completely ignore the existence of any unobserved variable u and therefore optimize the weighted average of RDCs $\mathbb{E}\big[G^{a'}\big] = \sum_{a \in \mathcal{A}} P(a)\mathbb{E}\big[G^{a'}|a\big]$ instead (Fig. 1b).

MABUCs are intriguing for open-loop planning in POMDPs to make memory-efficient decisions under the uncertainty of states s_t and observations o_t.

3 Related Work

Open-loop planning, as shown in Fig. 2b, is an efficient approach to online planning in complex domains with restricted resources (Bubeck and Munos 2010; Perez Liebana et al. 2015; Lecarpentier et al. 2018). However, open-loop planning is generally inferior to closed-loop planning in terms of performance due to the lack of context information (Barenboim and Indelman 2023; Wu et al. 2021). While human knowledge, e.g., heuristic functions or preferred action sets, is commonly used for enhancement, open-loop planning is generally not competitive in terms of performance (Perez Liebana et al. 2015; Silver and Veness 2010). We propose *counterfactual online learning* to effectively leverage human knowledge in open-loop MCTS to overcome the original performance limitation and compete with closed-loop MCTS in a *more memory-efficient* way.

Stack-based planning is a memory-bounded approach to open-loop planning. A sequence or stack of T MABs is maintained to sample open-loop plans with high expected return (Weinstein and Littman 2013; Belzner and Gabor 2017; Phan et al. 2019a,b). Despite the fixed memory usage of at most T MABs, the performance of stack-based planning is not competitive with tree search algorithms.

There are closed-loop approaches to regularizing the tree size of classic MCTS or POMCP in complex domains. AR-DESPOT is an anytime algorithm that constructs and optimizes sparse search trees from a fixed set of scenarios (Somani et al. 2013). POMCPOW uses progressive widening to cope with large or continuous state, action, and observation spaces (Sunberg and Kochenderfer 2018). Despite the progress of regularizing closed-loop MCTS for smaller trees, we demonstrate that our *open-loop approach* constructs *significantly smaller trees* while achieving *competitive performance* in a variety of benchmark scenarios.

Causal models have been used to provide more accurate dynamics models $\hat{M}$ for planning and reinforcement learning (Wang et al. 2022; Cannizzaro and Kunze 2023). Our work focuses on MABUCs to directly enhance the *decision-making algorithm*, i.e., open-loop MCTS, instead of the dynamics model $\hat{M}$, therefore being orthogonal to these works.

4 Counterfactual Online Learning for MCTS

4.1 Thompson Sampling

Thompson Sampling is a Bayesian MAB approach (Thompson 1933). We assume that G^a follows a Normal distribution $\mathcal{N}(\mu, \frac{1}{\tau})$ with unknown mean μ and precision $\tau = \frac{1}{\sigma^2}$, where σ^2 is the variance (Bai, Wu, and Chen 2013; Bai et al. 2014). $\langle \mu, \tau \rangle$ follows a Normal Gamma distribution $\mathcal{NG}(\mu_0, \lambda, \alpha, \beta)$ with $\lambda > 0$, $\alpha \geq 1$, and $\beta \geq 0$. The distribution over τ is a Gamma distribution $\tau \sim \textit{Gamma}(\alpha, \beta)$ and the conditional distribution over μ given τ is a Normal distribution $\mu \sim \mathcal{N}(\mu_0, \frac{1}{\lambda\tau})$.

Given a *prior distribution* $P(\theta^a) = \mathcal{NG}(\mu_0, \lambda_0, \alpha_0, \beta_0)$ and n observations $D^a = \{G^a_1, ..., G^a_n\}$, the *posterior distribution* is defined by $P(\theta^a|D^a) = \mathcal{NG}(\mu_1, \lambda_1, \alpha_1, \beta_1)$, where $\mu_1 = \frac{\lambda_0\mu_0 + n\overline{G^a}}{\lambda_0 + n}$, $\lambda_1 = \lambda_0 + n$, $\alpha_1 = \alpha_0 + \frac{n}{2}$, and $\beta_1 = \beta_0 + \frac{1}{2}(n\sigma^2 + \frac{\lambda_0 n(\overline{G^a} - \mu_0)^2}{\lambda_0 + n})$. $\overline{G^a}$ is the mean of all values in D^a and $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(G^a_i - \overline{G^a})^2$ is the variance.

The posterior $P(\theta^a|D^a)$ is inferred for each action $a \in \mathcal{A}$ to sample an estimate μ^a for the expected return. The action with the highest estimate is selected. The complete formulation is given in Algorithm 1, where $\mathcal{A}$ is the set of executable actions, $N_t = \langle \overline{G^a}, (\sigma^a)^2, n^a \rangle$ are the MAB statistics, and *greedy* indicates, if the MAB shall select greedily w.r.t. $\overline{G^a}$.

Algorithm 1: Thompson Sampling

```
procedure ThompsonSampling($\mathcal{A}, N_t, greedy$)
    if greedy then
        return $argmax_{a \in \mathcal{A}} \overline{G^a}$
    end if
    for $a \in \mathcal{A}$ do
        Infer posterior parameters $\langle \mu_1, \lambda_1, \alpha_1, \beta_1 \rangle$ from
        $\overline{G^a}, (\sigma^a)^2, n^a, P(\theta^a)$
        $\mu^a, \tau^a \sim \mathcal{NG}(\mu_1, \lambda_1, \alpha_1, \beta_1)$
    end for
    return $argmax_{a \in \mathcal{A}} \mu^a$
end procedure

procedure UpdateBandit($N_t, G'$)
    $n^a \leftarrow n^a + 1$
    $\langle \overline{G^a_{old}}, \overline{G^a} \rangle \leftarrow \langle \overline{G^a}, (n^a \overline{G^a_{old}} + G')/(n^a + 1) \rangle$
    $(\sigma^a)^2 \leftarrow [(n^a - 1)(\sigma^a)^2 + (G' - \overline{G^a_{old}})(G' - \overline{G^a})]/n^a$
end procedure
```

The prior should ideally reflect knowledge about the underlying model, especially for initial turns, where only a small amount of data has been observed (Honda and Takemura 2014). If there is no knowledge, then *uninformative priors* should be used to sample all possibilities (almost) uniformly. This can be achieved by setting the priors such that the variance of the resulting Normal distribution $\mathcal{N}(\mu_0, \frac{1}{\lambda_0 \tau})$ becomes infinite ($\frac{1}{\lambda_0 \tau_0} \to \infty$ and $\lambda_0 \tau \to 0$). Since τ follows a Gamma distribution $Gamma(\alpha_0, \beta_0)$ with expectation $\mathbb{E}(\tau) = \frac{\alpha_0}{\beta_0}$, α_0 and β_0 should be chosen such that $\frac{\alpha_0}{\beta_0} \to 0$. Given the hyperparameter space $\lambda_0 > 0$, $\alpha_0 \geq 1$, and $\beta_0 \geq 0$, it is recommended to set $\alpha_0 = 1$ and $\mu_0 = 0$ to center the Normal distribution. λ_0 should be chosen small enough and β_0 should be sufficiently large (Bai et al. 2014).

4.2 MABUC with Human Knowledge

As explained in Section 2.3 and sketched in Fig. 1c, MABUCs require some "natural" intent a that depends on structural (but generally unknown) properties of the underlying problem (Bareinboim, Forney, and Pearl 2015).

In the context of search algorithms, a "natural" intent can be represented by heuristics, which encode human knowledge about some structural properties of the underlying problem (Perez Liebana et al. 2015). Many MCTS approaches use human knowledge in the form of *preferred actions* $\mathcal{A}_{pref}(s_t) \subseteq \mathcal{A}_{legal}(s_t)$ (Silver and Veness 2010).

Therefore, we propose *two online learning phases*: first, to obtain an *intent policy* π_{intent} based on $\mathcal{A}_{pref}(s_t)$ and then find a *counterfactual policy* $\pi_{counterfactual}$ based on $\mathcal{A}_{legal}(s_t)$.

Given a simulation budget n_b and a *training ratio* $\eta \in [0, 1]$, we train a Thompson Sampling bandit N to select actions $a \in \mathcal{A}_{pref}(s_t)$ for the first $\lfloor n_b \eta \rfloor$ trials to learn *observational values* $\mathbb{E}\big[G^a | a\big]$ according to Fig. 1a. After $\lfloor n_b \eta \rfloor$ trials, we select *intent actions* a via bandit N by maximizing $\mathbb{E}\big[G^a | a\big]$. The intent actions a are then used to query another Thompson Sampling bandit N^a per intent a to sample *interventional actions* $a' \in \mathcal{A}_{legal}(s_t)$ in order to learn *counterfactual values* $\mathbb{E}\big[G^{a'} | a\big]$, according to Fig. 1c.

Fig. 3 illustrates the counterfactual value table used for online learning. In the first phase, our MABUC learns the observational values $\mathbb{E}\big[G^a | a\big]$ on the diagonal for π_{intent}. In the second phase, a row is queried via $\pi_{intent}(\mathcal{A}_{pref}(s)) = argmax_{a \in \mathcal{A}_{pref}(s)} \mathbb{E}\big[G^a | a\big]$. The selected row is used to learn the counterfactual values $\mathbb{E}\big[G^{a'} | a\big]$ for $\pi_{counterfactual}$.

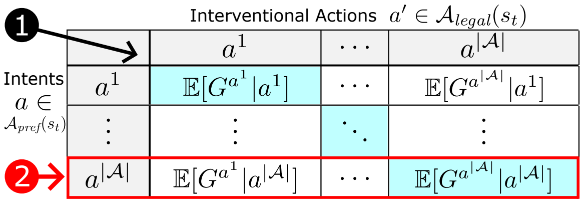
Algorithm 2: CORAL Planning

```
procedure CORAL($h_t, T, \eta$)
    Create $N_t$ for $h_t$
    $\tau \leftarrow \lceil n_b \eta \rceil$
    while $n_b > 0$ do
        $s_t \sim \hat{b}(\cdot | h_t)$ ▷ Sample $s_t$ for simulation (Fig. 4)
        CounterfactualSearch($N_t, s_t, T, 0, n_b < \tau$)
        $n_b \leftarrow n_b - 1$
    end while
    return $argmax_{a_t \in \mathcal{A}} (\overline{G_t^{a_t}})$
end procedure

procedure CounterfactualSearch($N_t, s_t, T, d, cf$)
    if $d \geq T$ or $s_t$ is a terminal state then
        return 0
    end if
    if $N_t$ is a leaf node then
        Expand $N_t$
        Perform rollout with $\pi_{rollout}$ to sample $G_t$
        return $G_t$
    end if
    $a \leftarrow$ ThompsonSampling($\mathcal{A}_{pref}(s_t), N_t, cf$)
    if $cf \wedge \mathcal{A}_{legal}(s_t) \neq \mathcal{A}_{pref}(s_t)$ then
        $a' \leftarrow$ ThompsonSampling($\mathcal{A}_{legal}(s_t), N_t^a, \neg cf$)
    else
        $a' \leftarrow a$
    end if
    $\langle s_{t+1}, r_t, o_{t+1} \rangle \sim \hat{M}(s_t, a')$      ▷ Simulate action $a'$
    $G_{t+1} \leftarrow$ CounterfactualSearch($N_{t+1}, s_{t+1}, T, d +
    1, cf$)
    $G_t \leftarrow r_t + \gamma G_{t+1}$
    if $a = a'$ then
        UpdateBandit($N_t, G_t$)                ▷ Update $\pi_{intent}$
    end if
    if $cf \wedge \mathcal{A}_{legal}(s_t) \neq \mathcal{A}_{pref}(s_t)$ then
        UpdateBandit($N_t^a, G_t$)      ▷ Update $\pi_{counterfactual}$
    end if
    return $G_t$
end procedure
```

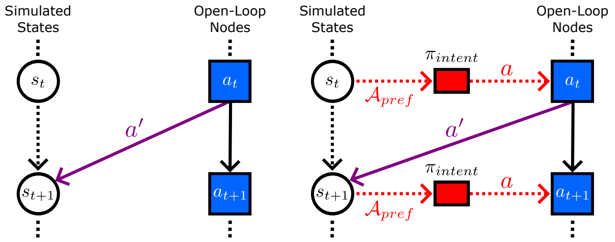
4.3 Open-Loop MCTS with MABUCs

We now define *Counterfactual Open-loop Reasoning with Ad hoc Learning (CORAL)* for open-loop MCTS, using MABUCs from Section 4.2. Each CORAL node represents


Interventional Actions $a' \in \mathcal{A}_{legal}(s_t)$
Intents $a \in \mathcal{A}_{pref}(s_t)$
1
2

	a^1	$\cdots$	$a^{\vert\mathcal{A}\vert}$
a^1	$\mathbb{E}[G^{a^1}\vert a^1]$	$\cdots$	$\mathbb{E}[G^{a^{\vert\mathcal{A}\vert}}\vert a^1]$
$\vdots$	$\vdots$	$\ddots$	$\vdots$
$a^{\vert\mathcal{A}\vert}$	$\mathbb{E}[G^{a^1}\vert a^{\vert\mathcal{A}\vert}]$	$\cdots$	$\mathbb{E}[G^{a^{\vert\mathcal{A}\vert}}\vert a^{\vert\mathcal{A}\vert}]$

Figure 3: Counterfactual value table as a cross of intent and interventional actions a and a', respectively. Each cell stores an estimate of the counterfactual value $\mathbb{E}\left[G^{a'}|a\right]$. The MABUC **(1)** queries the intent a by maximizing the observational values on the diagonal (cyan) before **(2)** making a counterfactual decision a' based on the selected row (red).



(a) Standard open-loop (b) Counterfactual open-loop

Figure 4: Open-loop variants based on interventional and counterfactual decision-making (Fig. 1). States are simulated but not stored explicitly, according to Fig. 2 and Algorithm 2, Line 26. **(a)** *Standard open-loop* completely ignores states for tree traversal. **(b)** *Counterfactual open-loop* considers states indirectly through the intent policy π_{intent}.

a MABUC with one Thompson Sampling bandit N_t to obtain the intent policy π_{intent} and at least another bandit N_t^a for the counterfactual policy $\pi_{counterfactual}$ for each intent action $a \leftarrow \pi_{intent}(\mathcal{A}_{pref}(s_t))$. We propose a *lazy initialization scheme*, where a *counterfactual bandit* N_t^a is only allocated when intent a is actually selected by π_{intent} in the counterfactual phase. Therefore, we can save some row allocations.

A simulation starts at a state s_t, sampled from the current belief state $\hat{b}(s_t|h_t) \approx b(s_t|h_t)$, which is approximated by a particle filter, as described in (Silver and Veness 2010). An open-loop tree (Fig. 2b) is iteratively constructed using Thompson Sampling for tree traversal. In the first $\lfloor n_b\eta \rfloor$ iterations, π_{intent} is obtained via Thompson Sampling. Afterward, $\pi_{counterfactual}$ is obtained via Thompson Sampling and π_{intent} (Table 1). When a leaf node is reached, it is expanded by a child node, and a rollout is performed with a policy $\pi_{rollout}$ until a terminal state is reached or a maximum search depth T is exceeded. The observed rewards are accumulated to returns G_t and propagated back to update the corresponding MABUC of every node in the simulated path. When the simulation budget n_b has run out, the action a_t with the highest average counterfactual return $\overline{G^{a_t}}$ is selected for execution. The complete formulation is given in Algorithm 2, where h_t is the current history, T is the maximum search depth, and η is the training ratio.

Table 1: Tree traversal policies according to Fig. 1.

Policy	**Definition**
π_{intent}	$argmax_{a \in \mathcal{A}_{pref}(s)} \mathbb{E}\left[G^a|a\right]$
π_{tree}	$argmax_{a' \in \mathcal{A}_{legal}(s)} \sum_a P(a)\mathbb{E}\left[G^{a'}|a\right]$
$\pi_{counterfactual}$	$argmax_{a' \in \mathcal{A}_{legal}(s)} \mathbb{E}\left[G^{a'}|\pi_{intent}(\mathcal{A}_{pref}(s))\right]$

4.4 Conceptual Discussion

CORAL represents a *strict generalization* of the standard open-loop MCTS: If $\eta = 0$, then CORAL reduces to open-loop MCTS without any human knowledge, that only selects actions from $\mathcal{A}_{legal}(s_t)$. If $\eta = 1$, then CORAL completely depends on human knowledge, thus only selecting actions from $\mathcal{A}_{pref}(s_t)$, effectively representing human intuition.

The MABUCs can converge to optimal actions a' if $\mathcal{A}_{pref}(s_t)$ sufficiently captures the structural properties of the POMDPs that would be ignored by the open-loop search otherwise, as shown in Fig. 4a (Bareinboim, Forney, and Pearl 2015). According to (Forney, Pearl, and Bareinboim 2017), the preferred actions do not need to be optimal, though. Therefore, if $0 < \eta < 1$, then CORAL can converge to an optimal counterfactual policy π^* given a sufficient simulation budget n_b. This is because all value estimates eventually converge to stationary values given a stationary rollout policy $\pi_{rollout}$, thus enabling convergence of all MABUCs in the tree (Kocsis and Szepesvári 2006; Bai et al. 2014).

Table 1 lists all tree traversal policies for open-loop MCTS that are considered in this paper: The intent policy π_{intent} optimizes the *observational value* (Fig. 1a) by selecting actions from $\mathcal{A}_{pref}(s)$. The standard traversal policy π_{tree} optimizes the *interventional value* (Fig. 1b) as a weighted average value over all possible intents, without explicit consideration of π_{intent}. The counterfactual policy $\pi_{counterfactual}$ optimizes the *counterfactual value* (Fig. 1c) by explicitly querying π_{intent}, as sketched in Fig. 3.

Unlike standard open-loop MCTS, CORAL considers states indirectly through π_{intent} for tree traversal (without storing them), as shown in Fig. 4b, enabling a *better informed search*. Thus, CORAL can potentially compete with closed-loop planning in a more memory-efficient way.

If a standard open-loop MCTS allocates X nodes, then CORAL produces at least $2X$ nodes due to maintaining one MAB for π_{intent} and another one for $\pi_{counterfactual}$ per node. In the worst case, CORAL produces X^2 nodes, where $|\mathcal{A}|$ counterfactual MABs are allocated per node. The worst case only occurs when $\mathcal{A}_{pref} = \mathcal{A}_{legal}$ for all states $s_t \in \mathcal{S}$ and the observational values on the diagonal in Fig. 3 are same. In our experiments, reported in Section 5.3, CORAL produces far less than X^2 nodes when preferred actions are available.

5 Experiments[1]

5.1 Evaluation Environments

For each environment, we use the discount factor $\gamma = 0.95$ and respective preferred action sets $\mathcal{A}_{pref}$, as suggested in (Silver and Veness 2010).

[1] https://github.com/thomyphan/counterfactual-planning.

Tag is a gridworld challenge where the agent has to find and tag a target that intentionally moves away. The agent knows its own position and can only observe the target when sharing the same position. The agent can either wait or move to the four adjacent positions with a reward of -1. In addition, the agent can perform a tag action and is rewarded with $+10$ if the target is successfully tagged and -10 otherwise.

$\mathcal{A}_{pref}$ avoids moving into walls and only prefers tagging when the target was observed in a corner.

RockSample[K,L] consists of a $K \times K$ grid with L rocks (Smith and Simmons 2004). Each rock can be *good* or *bad*, which is unknown to the agent. The agent has to sample good rocks and avoid bad rocks. It has a noisy sensor, which produces an observation $o_t \in \{good, bad\}$ for a particular rock. The probability of sensing the correct rock state decreases exponentially with the agent's distance to it. Sampling gives a reward of $+10$ if the rock is good and -10 otherwise. If a good rock is sampled, it becomes bad. Moving and sensing are not rewarded. Moving past the east edge of the grid gives a reward of $+10$, and the episode terminates.

$\mathcal{A}_{pref}$ prefers sampling rocks that emitted more *good* than *bad* observations. If all rocks emit more *bad* than *good* observations, then $\mathcal{A}_{pref}$ prefers moving east to the goal area.

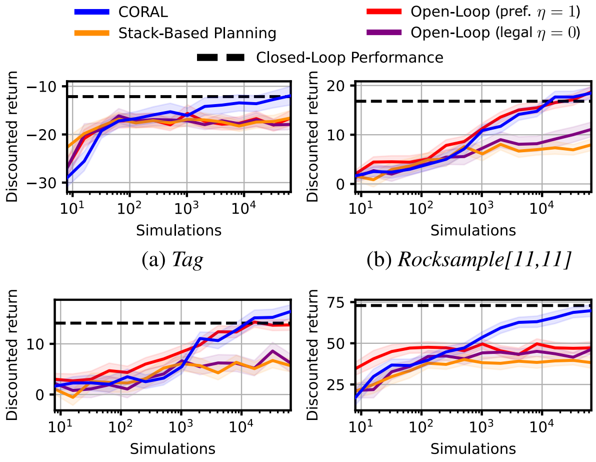
PocMan is a partially observable version of *PacMan* (Silver and Veness 2010). The agent navigates in a 17×19 maze and has to eat randomly distributed food pellets and power pills. There are four ghosts moving randomly in the maze. If the agent is within the visible range of a ghost, it is getting chased by the ghost and dies if it touches the ghost, terminating the episode with a reward of -100. Eating a power pill enables the agent to eat ghosts for 15 time steps. In that case, the ghosts will run away if the agent is under the effect of a power pill. At each time step, a reward of -1 is given. Eating food pellets gives a reward of $+10$, and eating a ghost gives $+25$. The agent can only perceive ghosts if they are in its direct line of sight in each cardinal direction or within a hearing range. Also, the agent can only sense walls and food pellets that are adjacent to it.

$\mathcal{A}_{pref}$ prefers moves that avoid any ghost without the effect of a power pill and moves toward observed ghosts otherwise.

5.2 Methods

Closed-Loop Algorithms We use POMCP, POMCPOW, and AR-DESPOT as state-of-the-art baselines (Sunberg and Kochenderfer 2018; Somani et al. 2013). Our POMCP and POMCPOW implementations are based on the code from (Silver and Veness 2010), using Thompson Sampling for π_{tree}, as specified in Algorithm 1 (Bai et al. 2014). For AR-DESPOT, we run the code from (Somani et al. 2013). For all approaches, $\pi_{rollout}$ randomly selects actions from $\mathcal{A}_{legal}(s')$, depending on the simulated state $s' \in \mathcal{S}$. In all cases, each simulation step has at most one expansion step, where new nodes are added to the search tree. Thus, the tree size should increase linearly w.r.t. n_b.

Open-Loop MCTS We use the open-loop MCTS implementation from (Phan et al. 2019a), where actions are selected with Thompson Sampling, using Algorithm 1. $\pi_{rollout}$ randomly selects actions from $\mathcal{A}_{legal}(s')$, depending on the currently simulated state $s' \in \mathcal{S}$. Like closed-loop MCTS, the search tree size increases linearly w.r.t. n_b, but with fewer nodes, since open-loop trees ignore observations.



(a) *Tag* (b) *Rocksample[11,11]*

(c) *Rocksample[15,15]* (d) *Pocman*

Figure 5: Performance of CORAL, open-loop MCTS variants, and closed-loop search over 100 runs for different n_b. Shaded areas show the 95% confidence interval.

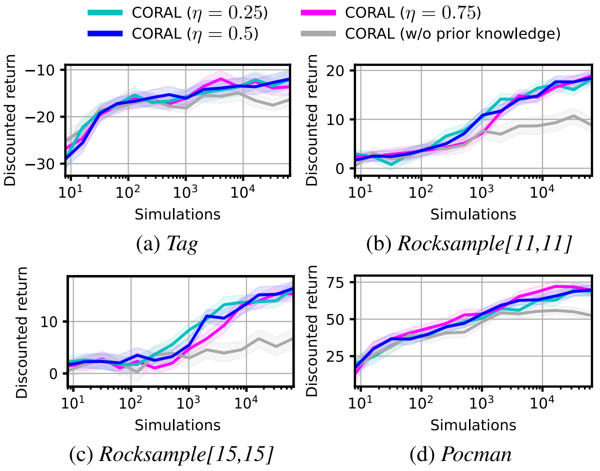
Stack-Based Planning We use the stack-based planning implementation from (Phan et al. 2019a) based on a sequence of bandits N_t with $0 \leq t \leq T-1$. Starting at s_t, all bandits N_t use Thompson Sampling to select actions. Given a horizon of T, stack-based planning always maintains T Thompson Sampling bandits, regardless of the budget n_b.

CORAL Our CORAL implementation is based on the open-loop MCTS implementation from (Phan et al. 2019a), where we modify the bandit routines, namely the action selection and the update (Lines 20-25 and 29-34 in Algorithm 2) to implement MABUCs. The search tree size increases linearly w.r.t. n_b with fewer nodes than closed-loop MCTS but expectedly more nodes than standard open-loop MCTS.

5.3 Results

We ran experiments for different simulation budgets n_b. For each budget n_b, we tested 100 random instances to report the average performance and the 95% confidence interval. Unless stated otherwise, we always set $\eta = 50\%$ for CORAL.

Performance Evaluation We first compare CORAL against standard open-loop MCTS with preferred ($\eta = 1$) or legal actions ($\eta = 0$), regarding different simulation budgets n_b. The results are shown in Fig. 5. We also provide the average performance of all closed-loop algorithms using the maximum simulation budget of $n_b = 2^{16}$. CORAL is always among the best performing open-loop approaches when $n_b > 1000$ and eventually achieves a similar performance level as closed-loop planning. Standard open-loop MCTS performs better when using preferred actions as integrated human knowledge and can keep up with CORAL in *Rocksample[11,11]* and *Rocksample[15,15]*. Stack-based planning performs worst in all scenarios.



(a) *Tag*

(b) *Rocksample[11,11]*

(c) *Rocksample[15,15]*

(d) *Pocman*

Figure 6: Performance of CORAL over 100 runs for different simulation budgets n_b, training ratios η, and absence of human knowledge such that $\mathcal{A}_{pref} = \mathcal{A}_{legal}$. Shaded areas show the 95% confidence interval.

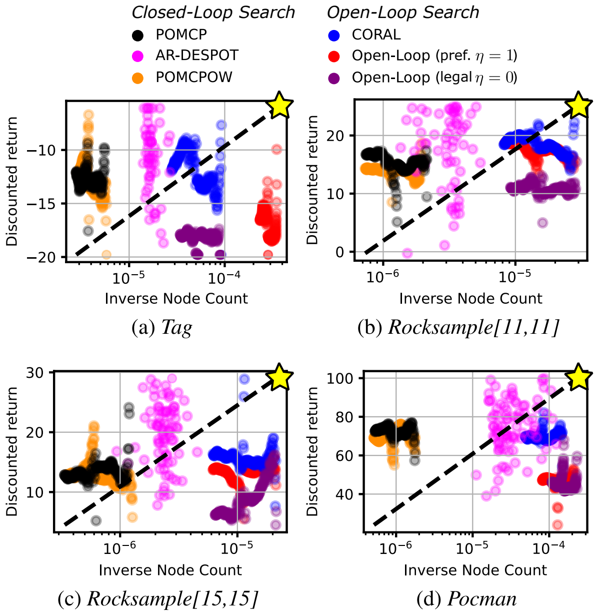
Parameter Evaluation Next, we evaluate the influence of different training ratios $\eta \in \{25\%, 50\%, 75\%\}$ as well as the absence of preferred actions based on human knowledge by setting $\mathcal{A}_{pref} = \mathcal{A}_{legal}$. The results are shown in Fig. 6. All CORAL variants with $0 < \eta < 1$ achieve roughly similar performance. The absence of human knowledge generally leads to inferior performance, indicating that $\mathcal{A}_{pref}$ needs to capture the structure of the underlying problem sufficiently well to make effective counterfactual decisions.

Performance-Memory Tradeoff Finally, we evaluate the performance relative to the *inverse number of nodes*, i.e., one divided by the number of nodes created during the search. We use *lazy node initialization* for all closed-loop algorithms and CORAL, where bandit nodes are only counted when explicitly visited or invoked during the search. The results are shown in Fig. 7. All closed-loop algorithms are mainly aligned to the left because they construct significantly larger trees than the open-loop algorithms. AR-DESPOT constructs the smallest closed-loop trees while having notably high variance in performance. The CORAL clusters are always centered around the same height as the closed-loop clusters, indicating that CORAL achieves competitive performance despite constructing significantly smaller trees. Due to the lazy node initialization, CORAL only constructs slightly larger trees than the standard open-loop variants.

6 Discussion

In this paper, we presented CORAL, an open-loop MCTS variant using a causal bandit approach with unobserved confounders. CORAL first learns observational values using commonly integrated human knowledge to obtain a counterfactual policy via MABUCs to make a final decision.

Our results demonstrate that CORAL meaningfully combines human knowledge and bandit-based learning for open-loop MCTS. CORAL can outperform both open-loop extremes, namely the variant exclusively based on human knowledge ($\eta = 1$) as well as the variant without any human knowledge ($\eta = 0$). The performance gap between CORAL and the former open-loop extreme confirms that the provided human knowledge does not have to represent an (near-) optimal policy but merely needs to capture the structural properties of the problem sufficiently well (Forney, Pearl, and Bareinboim 2017). CORAL effectively overcomes the limitation of classic open-loop planning and achieves competitive performance to closed-loop alternatives.



(a) *Tag*

(b) *Rocksample[11,11]*

(c) *Rocksample[15,15]*

(d) *Pocman*

Figure 7: Performance of CORAL, closed-loop search, and open-loop MCTS variants relative to the inverse number of nodes created during the search. Each point represents a random domain instance with a simulation budget of $n_b \in \{2^{15}, 2^{16}\}$. The closer the points are to the **star in the top-right corner**, the better the performance-memory tradeoff.

CORAL offers a better performance-memory tradeoff than closed-loop planning and is generally robust w.r.t. the choice of the training ratio as long as $0 < \eta < 1$.

While CORAL requires human knowledge in the form of preferred actions or some pre-computed intent policy, we can simply rely on common heuristics, as used in practice (Perez Liebana et al. 2015). Since these heuristics exploit structural properties explicitly, we can use them in CORAL without directly reasoning about these properties ourselves (Fig. 4b). Thus, we can effectively reuse and improve prior human knowledge for memory-efficient planning.

CORAL requires a sufficient simulation budget, i.e., $n_b > 1000$, to learn a meaningful intent policy because the counterfactual learning phase depends on its quality.

For future work, we want to test the idea of CORAL with other heuristic search algorithms to better leverage human knowledge for different problem classes.

Acknowledgements

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1817189, 1837779, 1935712, 2121028, 2112533, and 2321786 as well as a gift from Amazon Robotics. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

References

Bai, A.; Wu, F.; and Chen, X. 2013. Bayesian Mixture Modelling and Inference based Thompson Sampling in Monte-Carlo Tree Search. In *Advances in Neural Information Processing Systems*, 1646–1654.

Bai, A.; Wu, F.; Zhang, Z.; and Chen, X. 2014. Thompson Sampling based Monte-Carlo Planning in POMDPs. In *Proceedings of the Twenty-Fourth International Conferenc on International Conference on Automated Planning and Scheduling*, 29–37. AAAI Press.

Bareinboim, E.; Forney, A.; and Pearl, J. 2015. Bandits with Unobserved Confounders: A Causal Approach. *Advances in Neural Information Processing Systems*, 28.

Barenboim, M.; and Indelman, V. 2023. Online POMDP Planning with Anytime Deterministic Guarantees. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Belzner, L.; and Gabor, T. 2017. Stacked Thompson Bandits. In *Proceedings of the 3rd International Workshop on Software Engineering for Smart Cyber-Physical Systems*, 18–21. IEEE Press.

Bubeck, S.; and Munos, R. 2010. Open Loop Optimistic Planning. In *COLT*, 477–489.

Cannizzaro, R.; and Kunze, L. 2023. CAR-DESPOT: Causally-Informed Online POMDP Planning for Robots in Confounded Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Forney, A.; Pearl, J.; and Bareinboim, E. 2017. Counterfactual Data-Fusion for Online Reinforcement Learners. In *International Conference on Machine Learning*, 1156–1164. PMLR.

Honda, J.; and Takemura, A. 2014. Optimality of Thompson Sampling for Gaussian Bandits depends on Priors. In *Artificial Intelligence and Statistics*, 375–383.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial intelligence*, 101(1): 99–134.

Kocsis, L.; and Szepesvári, C. 2006. Bandit based Monte-Carlo Planning. In *ECML*, volume 6, 282–293. Springer.

Lecarpentier, E.; Infantes, G.; Lesire, C.; and Rachelson, E. 2018. Open Loop Execution of Tree-Search Algorithms. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2362–2368. IJCAI Organization.

Pearl, J. 2010. The Foundations of Causal Inference. *Sociological Methodology*, 40(1): 75–149.

Perez Liebana, D.; Dieskau, J.; Hunermund, M.; Mostaghim, S.; and Lucas, S. 2015. Open Loop Search for General Video Game Playing. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 337–344. ACM.

Phan, T.; Belzner, L.; Kiermeier, M.; Friedrich, M.; Schmid, K.; and Linnhoff-Popien, C. 2019a. Memory Bounded Open-Loop Planning in Large POMDPs Using Thompson Sampling. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 33(01): 7941–7948.

Phan, T.; Gabor, T.; Müller, R.; Roch, C.; and Linnhoff-Popien, C. 2019b. Adaptive Thompson Sampling Stacks for Memory Bounded Open-Loop Planning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 5607–5613. IJCAI Organization.

Pineau, J.; Gordon, G.; and Thrun, S. 2006. Anytime Point-based Approximations for Large POMDPs. *Journal of Artificial Intelligence Research*, 27: 335–380.

Powley, E.; Cowling, P.; and Whitehouse, D. 2017. Memory Bounded Monte Carlo Tree Search. *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Ross, S.; Pineau, J.; Paquet, S.; and Chaib-Draa, B. 2008. Online Planning Algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32: 663–704.

Silver, D.; and Veness, J. 2010. Monte-Carlo Planning in Large POMDPs. In *Advances in neural information processing systems*, 2164–2172.

Smith, T.; and Simmons, R. 2004. Heuristic Search Value Iteration for POMDPs. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, 520–527. AUAI Press.

Somani, A.; Ye, N.; Hsu, D.; and Lee, W. S. 2013. DESPOT: Online POMDP Planning with Regularization. In *Advances in neural information processing systems*, 1772–1780.

Sunberg, Z.; and Kochenderfer, M. 2018. Online Algorithms for POMDPs with Continuous State, Action, and Observation Spaces. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, 259–263.

Świechowski, M.; Godlewski, K.; Sawicki, B.; and Mańdziuk, J. 2023. Monte Carlo Tree Search: A Review of Recent Modifications and Applications. *Artificial Intelligence Review*, 56(3): 2497–2562.

Thompson, W. R. 1933. On the Likelihood that One Unknown Probability exceeds Another in View of the Evidence of Two Samples. *Biometrika*, 25(3/4): 285–294.

Wang, Z.; Xiao, X.; Xu, Z.; Zhu, Y.; and Stone, P. 2022. Causal Dynamics Learning for Task-Independent State Abstraction. In *International Conference on Machine Learning*, 23151–23180. PMLR.

Weinstein, A.; and Littman, M. L. 2012. Bandit-Based Planning and Learning in Continuous-Action Markov Decision Processes. In *Twenty-Second International Conference on Automated Planning and Scheduling*.

Weinstein, A.; and Littman, M. L. 2013. Open-loop Planning in Large-Scale Stochastic Domains. In *Proceedings* *of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 1436–1442. AAAI Press.

Wu, C.; Yang, G.; Zhang, Z.; Yu, Y.; Li, D.; Liu, W.; and Hao, J. 2021. Adaptive Online Packing-Guided Search for POMDPs. *Advances in Neural Information Processing Systems*, 34: 28419–28430.