Memory Bounded Open-Loop Planning in Large POMDPs Using Thompson Sampling

Thomy Phan  
LMU Munich  
thomy.phan@ifi.lmu.de

Lenz Belzner  
MaibornWolff  
lenz.belzner@maibornwolff.de

Marie Kiermeier  
LMU Munich  
marie.kiermeier@ifi.lmu.de

Markus Friedrich  
LMU Munich  
markus.friedrich@ifi.lmu.de

Kyrill Schmid  
LMU Munich  
kyrill.schmid@ifi.lmu.de

Claudia Linnhoff-Popien  
LMU Munich  
linhoff@ifi.lmu.de

Abstract

State-of-the-art approaches to partially observable planning like POMCP are based on stochastic tree search. While these approaches are computationally efficient, they may still construct search trees of considerable size, which could limit the performance due to restricted memory resources. In this paper, we propose Partially Observable Stacked Thompson Sampling (POSTS), a memory bounded approach to open-loop planning in large POMDPs, which optimizes a fixed size stack of Thompson Sampling bandits. We empirically evaluate POSTS in four large benchmark problems and compare its performance with different tree-based approaches. We show that POSTS achieves competitive performance compared to tree-based open-loop planning and offers a performance-memory tradeoff, making it suitable for partially observable planning with highly restricted computational and memory resources.

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. The agent has to reason about the history of past observations and actions, and maintain a belief state as a distribution of possible states. POMDPs have been widely used to model decision making problems in the context of planning and reinforcement learning (Ross et al. 2008).

Solving POMDPs exactly is computationally intractable for domains with enormous state spaces and long planning horizons. First, the space of possible belief states grows exponentially w.r.t. the number of states $N$, since that space is $N$-dimensional, which is known as the curse of dimensionality (Kaelbling, Littman, and Cassandra 1998). Second, the number of possible histories grows exponentially w.r.t. the horizon length, which is known as the curse of history (Pineau, Gordon, and Thrun 2006).

In the last few years, Monte-Carlo planning has been proposed to break both curses with statistical sampling. These methods construct sparse trees over belief states and actions, representing the state-of-the-art for efficient planning in large POMDPs (Silver and Veness 2010; Somani et al. 2013; Bai et al. 2014). While these approaches avoid exhaustive search, the constructed closed-loop trees can still become arbitrarily large for highly complex domains, which could limit the performance due to restricted memory resources (Powley, Cowling, and Whitehouse 2017). In contrast, open-loop approaches only focus on searching action sequences and are independent of the history and belief state space. Open-loop approaches are able to achieve competitive performance compared to closed-loop planning, when the problem is too large to provide sufficient computational and memory resources (Weinstein and Littman 2013; Perez Liebana et al. 2015; Lecarpentier et al. 2018). However, open-loop planning has been a less popular choice for decision making in POMDPs so far (Yu et al. 2005).

In this paper, we propose Partially Observable Stacked Thompson Sampling (POSTS), a memory bounded approach to open-loop planning in large POMDPs, which optimizes a fixed size stack of Thompson Sampling bandits.

To evaluate the effectiveness of POSTS, we formulate a tree-based approach, called Partially Observable Open-Loop Thompson Sampling (POOLTS) and show that POOLTS is able to find optimal open-loop plans with sufficient computational and memory resources.

We empirically test POSTS in four large benchmark problems and compare its performance with POOLTS and other tree-based approaches like POMCP. We show that POSTS achieves competitive performance compared to tree-based open-loop planning and offers a performance-memory tradeoff, making it suitable for partially observable planning with highly restricted computational and memory resources.

Background

Partially Observable Markov Decision Processes

A POMDP is defined by a tuple $M = \langle \mathcal{S}, \mathcal{A}, P, R, \mathcal{O}, \Omega, b_0 \rangle$, where $\mathcal{S}$ is a (finite) set of states, $\mathcal{A}$ is the (finite) set of actions, $P(s_{t+1} | s_t, a_t)$ is the transition probability function, $R(s_t, a_t)$ is the scalar reward function, $\mathcal{O}$ is a (finite) set of observations, $\Omega(o_{t+1} | s_{t+1}, a_t)$ is the observation probability function, and $b_0$ is a probability distribution over initial states $s_0 \in \mathcal{S}$. It is always assumed, that $s_t, s_{t+1} \in \mathcal{S}$, $a_t \in \mathcal{A}$, and $o_t, o_{t+1} \in \mathcal{O}$ at time step $t$. 

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
A history $h_t = \{a_0, o_1, \ldots, a_{t-1}, o_t\}$ is a sequence of actions and observations. A belief state $b_{h_t}(s_t)$ is a sufficient statistic for history $h_t$ and defines a probability distribution over states $s_t$ given $h_t$. $B$ is the space of all possible belief states. $b_0$ represents the initial belief state $b_{h_0}$. The belief state can be updated by Bayes theorem:

$$b_{h_t}(s_t) = \eta \Omega(a_t | s_t, a) \sum_{s \in S} P(s_t | s, a) b_h(s_t)$$  \hspace{1cm} (1)

where $\eta = \frac{1}{\Omega(a_t, b_{h_{t+1}})}$ is a normalizing constant, $a = a_{t-1}$ is the last action, and $h$ is the history without $a$ and $o_t$.

The goal is to find a policy $\pi : B \rightarrow A$, which maximizes the return $G_t$ at state $s_t$ for a horizon $T$:

$$G_t = \sum_{k=0}^{T-1} \gamma^k \cdot \mathcal{R}(s_{t+k}, a_{t+k})$$  \hspace{1cm} (2)

where $\gamma \in [0, 1]$ is the discount factor. If $\gamma < 1$, then present rewards are weighted more than future rewards.

The value function $V^*(b_t) = \mathbb{E}_\pi [G_t | b_t]$ is the expected return conditioned on belief states given a policy $\pi$. An optimal policy $\pi^*$ has a value function $V^{\pi^*} = V^*$ with $V^*(b_t) \geq V^{\pi'}(b_t)$ for all $b_t \in B$ and $\pi' \neq \pi^*$.

**Multi-armed Bandits**

Multi-armed Bandits (MABs or bandits) are fundamental decision making problems, where an agent has to repeatedly select an arm among a given set of arms in order to maximize its future pay-off. MABs can be considered as problems with a single state $s$, a set of actions $a \in A$, and a stochastic reward function $\mathcal{R}(s, a) := X_a$, where $X_a$ is a random variable with an unknown distribution $f_{X_a}(x)$. To solve a MAB, one has to determine the action, which maximizes the expected reward $\mathbb{E}[X_a]$. The agent has to balance between sufficiently trying out actions to accurately estimate their expected reward and to exploit its current knowledge on all arms by selecting the arm with the currently highest expected reward. This is known as the exploration-exploitation dilemma, where exploration can lead to actions with possibly higher rewards but requires time for trying them out, while exploitation can lead to fast convergence but possibly gets stuck in a local optimum. In this paper, we will cover UCB1 and Thompson Sampling as MAB algorithms.

**UCB1** In UCB1, actions are selected by maximizing the upper confidence bound of action values $UCB1(a) = \overline{X}_a + c \sqrt{\frac{\log(N_{total})}{N_a}}$, where $\overline{X}_a$ is the current average reward when choosing $a$, $c$ is an exploration constant, $N_{total}$ is the total number of action selections, and $N_a$ is the number of times action $a$ was selected. The second term represents the exploration bonus, which becomes smaller with increasing $N_a$ (Auer, Cesa-Bianchi, and Fischer 2002).

UCB1 is a popular MAB algorithm and widely used in various challenging domains (Kocsis and Szepesvári 2006; Bubeck and Munos 2010; Silver et al. 2016; 2017).

**Thompson Sampling** Thompson Sampling is a Bayesian approach to balance between exploration and exploitation of actions (Thompson 1933). The unknown reward distribution of $X_a$ of each action $a \in A$ is modeled by a parametrized likelihood function $P_a(x | \theta)$ with a parameter vector $\theta$. Given a prior distribution $P_\theta(\theta)$ and a set of past observed rewards $D_\theta = \{x_{a_1}, x_{a_2}, \ldots, x_{a_N}\}$, the posterior distribution $P_\theta(\theta | D_\theta)$ can be inferred by using Bayes rule $P_\theta(\theta | D_\theta) \propto \prod_{i} P_a(x_{a_i} | \theta) P_\theta(\theta)$. The expected reward of each action $a \in A$ can be estimated by sampling $\theta \sim P_\theta(\theta | D_\theta)$ from the posterior. The action with the highest sample expected reward $\mathbb{E}_\theta [X_a]$ is selected.

Thompson Sampling has been shown to be an effective and robust algorithm for making decisions under uncertainty (Chapelle and Li 2011; Kaufmann, Korda, and Munos 2012; Bai, Wu, and Chen 2013; Bai et al. 2014).

**Planning in POMDPs**

Planning searches for an (near-)optimal policy given a model $M$ of the environment $N$, which usually consists of explicit probability distributions of the POMDP. Unlike offline planning, which searches the whole (belief) state space to find the optimal policy $\pi^*$, local planning only focuses on finding a policy $\pi_t$ for the current (belief) state by taking possible future (belief) states into account (Weinstein and Littman 2013). Thus, local planning can be applied online at every time step at the current state to recommend the next action for execution. Local planning is usually restricted to a time or computation budget $n_o$ due to strict real-time constraints (Bubeck and Munos 2010; Weinstein and Littman 2013; Perez Liebana et al. 2015).

In this paper, we focus on local Monte-Carlo planning, where $M$ is a generative model, which can be used as a black box simulator (Kocsis and Szepesvári 2006; Silver and Veness 2010; Weinstein and Littman 2013; Bai et al. 2014). Given $s_t$ and $a_t$, the simulator $M$ provides a sample $(s_{t+1}, a_{t+1}, r_t) \sim M(s_t, a_t)$. Monte-Carlo planning algorithms can approximate $\pi^*$ and $V^*$ by iteratively simulating and evaluating action sequences without reasoning about explicit probability distributions of the POMDP.

Local planning can be closed- or open-loop. Closed-loop planning conditions the action selection on histories of actions and observations. Open-loop planning only conditions the action selection on previous sequences of actions $r_T = [a_1, \ldots, a_T]$ (also called open-loop plans or simply plans) and summarized statistics about predecessor (belief) states (Bubeck and Munos 2010; Weinstein and Littman 2013; Perez Liebana et al. 2015). An example is shown in Fig. 1. A closed-loop tree for a domain with $\Omega(s_{t+1} | s_t, a_t) = 0.5$ is shown in Fig. 1a, while Fig. 1b shows the corresponding open-loop tree which summarizes the observation nodes of Fig. 1a within the blue dotted ellipses into history distribution nodes. Open-loop planning can be further simplified by only regarding statistics about the expected return of actions at specific time steps (Fig. 1c). In that case, only a stack of $T$ statistics is used to sample plans for simulation and evaluation (Weinstein and Littman 2013).
Related Work

Tree-based approaches to open-loop planning condition the action selection on previous action sequences as shown in Fig. 1b (Bubeck and Munos 2010; Perez Liebana et al. 2015; Lecarpentier et al. 2018). Such approaches have been thoroughly evaluated for fully observable problems, but have been less popular for partially observable problems so far (Yu et al. 2005). POSTS is based on stacked open-loop planning, where a stack of $T$ distributions over actions is maintained to generate open-loop plans with high expected return (Weinstein and Littman 2013; Belzner and Gabor 2017). Unlike previous approaches, POSTS is a memory-bounded open-loop approach to partially observable planning.

(Yu et al. 2005) proposed an open-loop approach to planning in POMDPs by using hierarchical planning. An open-loop plan is constructed at an abstract level, where uncertainty w.r.t. particular actions is ignored. A low-level planner controls the actual execution by explicitly dealing with uncertainty. POSTS is more general, since it performs planning directly on the original problem and does not require the POMDP to be transformed for hierarchical planning.

(Powel, Cowling, and Whitehouse 2017) proposed a memory bounded version of MCTS with a state pool to add, discard, or reuse states depending on their visitation frequency. However, this approach cannot be easily adapted to tree-based open-loop approaches, because it requires (belief) states to be identifiable. POSTS does not require a pool to reuse states or nodes, but only maintains a fixed size stack of Thompson Sampling bandits, which adapt according to the temporal dependencies between actions.

Open-Loop Search with Thompson Sampling

Generalized Thompson Sampling We use a variant of Thompson Sampling, which works for arbitrary reward distributions as proposed in (Bai, Wu, and Chen 2013; Bai et al. 2014) by assuming that $X_a$ follows a Normal distribution $\mathcal{N}(\mu, \frac{1}{\tau})$ with unknown mean $\mu$ and precision $\tau = \frac{1}{\lambda}$, where $\sigma^2$ is the variance. $(\mu, \tau)$ 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 $\tau$ is a Gamma distribution $\tau \sim Gamma(\alpha, \beta)$ and the conditional distribution over $\mu$ given $\tau$ is a Normal distribution $\mu \sim \mathcal{N}(\mu_0, \frac{1}{\lambda\tau}$).

Given a prior distribution $P(\theta) = \mathcal{NG}(\mu_0, \lambda_0, \alpha_0, \beta_0)$ and $n$ observations $D = \{x_1, ..., x_n\}$, the posterior distribution is defined by $P(\theta|D) = \mathcal{NG}(\mu_1, \lambda_1, \alpha_1, \beta_1)$, where $\mu_1 = \frac{\lambda_0\mu_0 + n\overline{x}}{\lambda_0+n}, \lambda_1 = \lambda_0 + n, \alpha_1 = \alpha_0 + n, \beta_1 = \beta_0 + \frac{n\sigma^2 + n\overline{x}^2 - \mu_0^2}{\lambda_0\overline{x}^2}$, $\overline{x}$ is the mean of all values in $D$ and $\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2$ is the variance.

The posterior is inferred for each action $a_i \in \mathcal{A}$ to sample an estimate $\mu_a$, for the expected return. The action with the highest estimate is selected. The complete formulation is given in Algorithm 1.

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 Take-mura 2014). If no knowledge is available, then uninformative priors should be chosen, where all possibilities can be
Gamma λ h

this reason, we approximate the belief state large state spaces due to the curse of dimensionality. For updates may be computationally infeasible in POMDPs with Monte Carlo Belief State Update a sufficiently large value (Bai et al. 2014).

Monte Carlo Belief State Update The belief state can be updated exactly according to Eq. 1. However, exact Bayes updates may be computationally infeasible in POMDPs with large state spaces due to the curse of dimensionality. For this reason, we approximate the belief state \( b_t \) for history \( h_t \) with a particle filter as described in (Silver and Veness 2010). The belief state \( b_t \) is represented by a set \( b_{hti} \) of \( K \) sample states or particles. After execution of \( a_t \) and observation of \( a_{t+1}, \) the particles are updated by Monte Carlo simulation. Sampled states \( s_{t+1} \sim b_{hti} \) are simulated with \( a_t \) such that \( \langle s_{t+1}, a_t, r_t \rangle \sim M(s_t, a_t) \). If \( o_{t+1} = o_t+1, \) then \( s_{t+1} \) is added to \( b_{hti, a_t, o_t+1} \).

POOLTS To evaluate the effectiveness of POSTS compared to other open-loop planners, we first define Partially Observable Open-Loop Thompson Sampling (POOLTS) and show that POOLTS is able to converge to an optimal open-loop plan, if sufficient computational and memory resources are provided. POOLTS is a tree-based approach based on OLUCT from (Lecarpentier et al. 2018). Each node \( N_t \) represents a Thompson Sampling bandit and stores \( \overline{x}_{a_t}, s_a, \) and \( n_{a_t} \) for each action \( a_t \in A \).

A simulation starts at a state \( s_t \), which is sampled from the current belief state \( b_{ht} \). The belief state is approximated by a particle filter \( b_{ht} \), as described above. An open-loop tree (Fig. 1b) is iteratively constructed by traversing the current tree in a selection step by using Thompson Sampling to select actions. When a leaf node is reached, it is expanded by a child node \( N_{new} \) and a rollout is performed by using a policy \( \pi_{n_{new}} \) until a terminal state is reached or a maximum search depth \( T \) is exceeded. The observed rewards are accumulated to returns \( G_t \) (Eq. 2) and propagated back to update the corresponding bandit of every node in the simulated path. When the computation budget \( n_b \) has run out, the action \( a_t \) with the highest expected return \( \overline{x}_{a_t} \) is selected for execution. The complete formulation of POOLTS is given in Algorithm 2.

Algorithm 1: Generalized Thompson Sampling

\[
\text{procedure}\ ThompsonSampling(N_t) \\
\text{for } a_t \in A \text{ do} \\
\quad \text{Infer } (\mu_t, \lambda_t, \mu_0, \beta_0) \text{ from prior and } \overline{x}_{a_t}, \sigma^2_{a_t}, n_{a_t} \\
\text{return } \text{argmax}_{a_t\in A}(\mu_t) \\
\text{procedure}\ UpdateBandit(N_t, G_t) \\
\quad n_{a_t} + 1 \\
\quad (X_{old,a_t}, X_t) + (X_a, (n_{a_t} X_{old,a_t} + G_t)/(n_{a_t} + 1)) \\
\quad s_{a_t} - (n_{a_t} - 1)s_{a_t} + G_t - X_{a_t})/n_{a_t} \\
\]

Algorithm 2: POOLTS Planning

\[
\text{procedure}\ POOLTS(h_t, T, n_b) \\
\text{Create } N_0 \text{ for } h_t \\
\text{while } n_b > 0 \text{ do} \\
\quad n_b - 1 \\
\quad \text{return } b_{ht} \\
\quad TreeSearch(N_0, s_t, T, 0) \\
\text{return } \text{argmax}_{a_t\in A}(\overline{x}_{a_t}) \\
\text{procedure}\ TreeSearch(N_t, s_t, T, d) \\
\quad \text{if } d \geq T \text{ or } s_t \text{ is a terminal state then return } 0 \\
\quad \text{if } N_t \text{ is a leaf node then} \\
\quad \quad \text{Expand } N_t \\
\quad \quad \text{Perform rollout with } \pi_{n_{new}} \text{ to sample } G_t \\
\quad \quad \text{return } G_t \\
\quad \quad a_t - ThompsonSampling(N_t) \\
\quad \quad s_{t+1}, t_{a_t} \sim M(s_t, a_t) \\
\quad \quad R_t - TreeSearch(N_{t+1}, s_{t+1}, T, d + 1) \\
\quad \quad G_t - r_t + \gamma R_t \\
\quad \quad UpdateBandit(N_t, G_t) \\
\quad \quad \text{return } G_t \\
\]

(Kocsis and Szepesvári 2006; Bubeck and Munos 2010; Lecarpentier et al. 2018) have shown that tree search algorithms using UCB1 converge to the optimal closed-loop or open-loop plan respectively, if the computation budget \( n_b \) is sufficiently large. This is because the expected state or node values in the leaf nodes become stationary, given a stationary rollout policy \( \pi_{n_{new}} \). This enables the values in the preceding nodes to converge as well, leading to state- or node-wise optimal actions. By replacing UCB1 with Thompson Sampling, the tree search should still converge to the optimal closed-loop or open-loop plan, since Thompson Sampling also converges to the optimal action, if the return distribution of \( G_t \) becomes stationary (Agrawal and Goyal 2013). (Chapelle and Li 2011; Bai et al. 2014) empirically demonstrated that Thompson Sampling converges faster than UCB1, when rewards are sparse and when the number of arms is large.

POSTS Partially Observable Stacked Thompson Sampling (POSTS) is an open-loop approach, which optimizes a stack of \( T \) Thompson Sampling bandits to search for high-quality open-loop plans (Fig. 1c). Each bandit \( N_t \) stores \( \overline{x}_{a_t}, s_{a_t}, \) and \( n_{a_t} \) for each action \( a_t \in A \).

Similarly to POOLTS, a simulation starts at a state \( s_t \), which is sampled from a particle filter \( b_{ht} \), representing the current belief state \( b_{ht} \). Unlike POOLTS, a fixed size stack of bandits \( N_1, ..., N_T \) is used to sample plans \( p_T = [a_1, ..., a_T] \). \( p_T \) is evaluated with the generative model \( M \) to observe immediate rewards \( r_1, ..., r_T \), which are accumulated to returns \( G_1, ..., G_T \) (Eq. 2). Each bandit \( N_t \) is then updated with the corresponding return \( G_t \). When the computation budget \( n_b \) has run out, the action \( a_t \) with the highest expected return \( \overline{x}_{a_t} \) is selected for execution. The complete
formulation of POSTS is given in Algorithm 3.

**Algorithm 3 POSTS Planning**

```
procedure POSTS(h, T, n)
    while n > 0 do
        n ← n − 1
        st ← \tilde{b}_b,
        Simulate(N, st, T, 0)
    return argmax_{a \in A} \hat{X}_{st}(a)
```

```
procedure Simulate(st, T, d)
    if d > 0 or st is a terminal state then return 0
    a_t ← ThompsonSampling(N_t)
    \langle s_{t+1}, r_t, a_{t+1} \rangle \sim M(s_t, a_t)
    R_t ← Simulate(s_{t+1}, T, d + 1)
    G_t ← r_t + \gamma R_t
    UpdateBandit(N_t, G_t)
    return G_t
```

The idea of POSTS is to only regard the temporal dependencies between the actions of an open-loop plan. The bandit stack is used to learn these dependencies with the expected (discounted) return. When a bandit \( N_t \) samples an action \( a_t \) with a resulting reward of \( r_t \), then all preceding bandits \( N_{t-k} \) with \( k < t \) are updated with \( G_{t-k} = r_{t-k} + \gamma r_{t-k+1} + \ldots + \gamma^k r_t \), using a discounted value of \( r_t \). This is because the actions sampled by all preceding bandits are possibly relevant for obtaining the reward \( r_t \). By only regarding these temporal dependencies, POSTS is memory bounded, not requiring a search tree to model dependencies between histories or history distributions (Fig. 1a and 1b).

## Experiments

### Evaluation Environments

We tested POSTS in the RockSample, Battleship, and Pac-Man domains, which are well-known POMDP benchmark problems for decision making in POMDPs (Silver and Veness 2010; Somani et al. 2013; Bai et al. 2014). For each domain, we set the discount factor \( \gamma \) as proposed in (Silver and Veness 2010). The results were compared with POMCP, POOLTS, and a partially observable version of OLUCT, which we call POOLUCT. The problem-size features of all domains are shown in Table 1.

The RockSample(n,k) problem simulates an agent moving in an \( n \times n \) grid containing \( k \) rocks (Smith and Simmons 2004). Each rock can be good or bad but the true state of each rock is unknown. The agent has to sample good rocks, while avoiding to sample bad rocks. It has a noisy sensor, which produces an observation \( a_t \in \{\text{good}, \text{bad}\} \) for a particular rock. The probability of sensing the correct state of the rock decreases exponentially with the agent’s distance to that rock. Sampling gives a reward of +10, if the rock is good and −10 otherwise. If a good rock was sampled, it becomes bad. Moving and sensing do not give any rewards. Moving past the east edge of the grid gives a reward of +10 and the episode terminates. We set \( \gamma = 0.95 \).

In Battleship five ships of size 1, 2, 3, 4, and 5 respectively are randomly placed into a \( 10 \times 10 \) grid, where the agent has to sink all ships without knowing their actual positions (Silver and Veness 2010). Each cell hitting a ship gives a reward of +1. There is a reward of −1 per time step and a terminal reward of +100 for hitting all ships. We set \( \gamma = 1 \).

**PocMan** is a partially observable version of PacMan (Silver and Veness 2010). The agent navigates in a \( 17 \times 19 \) 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, which are adjacent to it. We set \( \gamma = 0.95 \).

### Methods

**POMCP** We use the POMCP implementation from (Silver and Veness 2010). \( \pi_{\text{tree}} \) selects actions from a set of legal actions \( a_t \in \mathcal{A}_{\text{legal}}(s_t) \) with UCB1. \( \pi_{\text{ rollout}} \) randomly selects actions from \( \mathcal{A}_{\text{legal}}(s') \), depending on the currently simulated state \( s' \in S \),

In each simulation step, there is at most one expansion step, where new nodes are added to the search tree. Thus, tree size should increase linearly w.r.t. \( n_b \), in large POMDPs.

**POOLTS and POOLUCT** POOLTS is implemented according to Algorithm 2, where actions are selected from a set of legal actions \( a_t \in \mathcal{A}_{\text{legal}}(s_t) \) with Thompson Sampling (Algorithm 1) in the first stage. \( \pi_{\text{ rollout}} \) randomly selects actions from \( \mathcal{A}_{\text{legal}}(s') \), depending on the currently simulated state \( s' \in S \). POOLUCT is similar to POOLTS but uses UCB1 as action selection strategy in the first stage.

Since, open-loop planning can encounter different states at the same node (Fig. 1), the set of legal actions may vary for each state \( s_t \in S \). We always mask out the statistics of currently illegal actions, regardless of whether they have high average action values, to avoid selecting them.

Similarly to POMCP, the search tree size should increase linearly w.r.t. \( n_b \), but with less nodes, since open-loop trees store summarized information about history distributions.

**POSTS** POSTS is implemented as a stack of Thompson Sampling bandits \( N_i \) with \( 1 \leq i \leq T \) according to Algorithm 3. Starting at \( s_t \), all bandits \( N_i \) apply Thompson Sampling to a set of legal actions \( \mathcal{A}_{\text{legal}}(s') \), depending on the currently simulated state \( s' \in S \). Similarly to POOLTS and POOLUCT, we mask out the statistics of currently illegal actions and only regard the value statistics of legal actions for selection during planning.

Given a horizon of \( T \), POSTS always maintains \( T \) Thompson Sampling bandits, independently of the computation budget \( n_b \).
### Table 1: The problem-size features of the benchmark domains *RockSample*, *Battleship*, and *PocMan*.

<table>
<thead>
<tr>
<th></th>
<th><em>RockSample</em>(11,11)</th>
<th><em>RockSample</em>(15,15)</th>
<th><em>Battleship</em></th>
<th><em>PocMan</em></th>
</tr>
</thead>
<tbody>
<tr>
<td># States</td>
<td>247,808</td>
<td>7,372,800</td>
<td>∼10^{12}</td>
<td>∼10^{36}</td>
</tr>
<tr>
<td># Actions</td>
<td>16</td>
<td>20</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td># Observations</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1024</td>
</tr>
</tbody>
</table>

**Results**

We ran each approach on *RockSample*, *Battleship*, and *PocMan* with different settings for 100 times or at most 12 hours of total computation. We evaluated the performance of each approach with the undiscounted return ($\gamma = 1$), because we focus on the actual effectiveness instead of the quality of optimization (Bai et al. 2014). For POMCP and POOLTS we set the UCB1 exploration constant $c$ to the reward range of each domain as proposed in (Silver and Veness 2010).

**Prior Sensitivity** Since we assume no additional domain knowledge, we focus on uninformative priors with $\mu_0 = 0$, $\alpha_0 = 1$, and $\lambda_0 = 0.01$ as proposed in (Bai et al. 2014). With this setting, $\beta_0$ controls the degree of initial exploration during the planning phase, thus its impact on the performance of POOLTS and POSTS is evaluated. The results are shown in Fig. 2 for $\beta_0 = 1000, 4000, 32000$ for POOLTS and POSTS.

![Figure 2](image_url)

Figure 2: Average performance of POSTS, POOLTS, POOLUCT, and POMCP with different prior values for $\beta_0$, different computation budgets $n_b$, and a horizon of $T = 100$.

In *RockSample*, POSTS slightly outperforms POOLTS and keeps up in performance with POMCP. POOLTS slightly outperforms POSTS and POMCP in *Battleship* with POSTS only being able to keep up when $n_b < 10^4$ or when $\beta_0 = 32000$. POMCP clearly outperforms all open-loop approaches in *PocMan*. POOLTS slightly outperforms POSTS in *PocMan* with POSTS only being to keep up, if $\beta_0 = 32000$. POOLUCT performed worst in all domains except *Battleship*, where it performs best with a computation budget of $n_b \leq 1024$. POSTS performs slightly better, if $\beta_0$ is large, but POOLTS seem to be insensitive to the choice of $\beta_0$ except in *PocMan*, where it performs better, if $\beta_0$ is large.

**Horizon Sensitivity** We evaluated the sensitivity of all approaches w.r.t. different horizons $T$. The results are shown in Fig. 3 for $n_b = 4096$ and $\beta_0 = 1000, 4000, 32000$ for POOLTS and POSTS.

![Figure 3](image_url)

Figure 3: Average performance of POSTS, POOLTS, POOLUCT, and POMCP with different planning horizons $T$ and a computation budget of $n_b = 4096$.

In *RockSample*(11,11), there is a performance peak at $T = 10$ for POMCP and POOLUCT, while for POSTS and POOLTS it is about $T = 50$. In all other domains, there seems to be a performance peak at $T = 100$ for most approaches. If $T > 100$, there is no significant improvement or even degrading performance for most approaches except for POMCP, which slightly improves in all domains but *RockSample*(11,11), if $T = 400$.

**Performance-Memory Tradeoff** We evaluated the performance-memory tradeoff of all approaches by introducing a memory capacity $n_{\text{mem}}$, where the computation is interrupted, when the number of nodes exceeds $n_{\text{mem}}$. For

\[1\] Using computation budgets between 1024 and 16384 led to similar plots, thus we stick to $n_b = 4096$ with all approaches requiring less than one second per action (Silver and Veness 2010).
In RockSample and Battleship, POMCP is outperformed by POSTS and POOLTS (and also POOLUCT in Battleship). POSTS always performs best in these domains, when \( n_{\text{mem}} < 1000 \). POMCP performs best in PocMan by outperforming POSTS, when \( n_{\text{mem}} > 100 \) and POOLTS keeps up with the best POSTS setting, when \( n_{\text{mem}} > 1000 \). POOLUCT performs worst except in Battleship, improving less and slowest with increasing \( n_{\text{mem}} \). It outperforms POMCP in RockSample(15,15), when \( n_{\text{mem}} \leq 1000 \) though. In PocMan, POOLUCT creates less than 550 nodes, when \( n_b = 4096 \), indicating that the search tree construction has converged and does not improve any further.

**Conclusion and Future Work**

In this paper, we proposed Partially Observable Stacked Thompson Sampling (POSTS), a memory bounded approach to open-loop planning in large POMDPs, which optimizes a fixed size stack of Thompson Sampling bandits.

To evaluate the effectiveness of POSTS, we formulated a tree-based approach, called POOLTS and showed that POOLTS is able to find optimal open-loop plans with sufficient computational and memory resources.

We empirically tested POSTS in four large benchmark problems and showed that POSTS achieves competitive performance compared to tree-based open-loop planners like POOLTS and POOLUCT, if sufficient resources are provided. Unlike tree-based approaches, POSTS offers a performance-memory tradeoff by performing best, if computational and memory resources are highly restricted, making it suitable for efficient partially observable planning.

For the future, we plan to extend POSTS to multi-agent systems (Phan et al. 2018).
References


