Adaptive Thompson Sampling Stacks for Memory Bounded Open-Loop Planning

Thomy Phan, **Thomas Gabor**, **Robert Müller**, **Christoph Roch** and **Claudia Linnhoff-Popien**
LMU Munich
{thomy.phan, thomas.gabor, robert.mueller, christoph.roch, linnhoff}@ifi.lmu.de

Abstract

We propose *Stable Yet Memory Bounded Open-Loop (SYMBOL) planning*, a general memory bounded approach to partially observable open-loop planning. SYMBOL maintains an adaptive stack of Thompson Sampling bandits, whose size is bounded by the planning horizon and can be automatically adapted according to the underlying domain without any prior domain knowledge beyond a generative model. We empirically test SYMBOL in four large POMDP benchmark problems to demonstrate its effectiveness and robustness w.r.t. the choice of hyperparameters and evaluate its adaptive memory consumption. We also compare its performance with other open-loop planning algorithms and POMCP.

1 Introduction

Partially Observable Markov Decision Processes (POMDPs) are useful to model many real-world problems, where the actual state is unknown to a decision making agent due to limited and noisy sensors. The agent has to consider the history of its past observations and actions to maintain a belief state as a distribution of possible states [Kaelbling *et al.*, 1998].

Solving POMDPs exactly is computationally intractable for domains with extremely large state spaces and long planning horizons due to the *curse of dimensionality*, where the space of possible belief states grows exponentially w.r.t. the number of states [Kaelbling *et al.*, 1998], and the *curse of history*, where the number of possible histories grows exponentially w.r.t. the horizon length [Pineau *et al.*, 2006].

Monte-Carlo planning algorithms are popular approaches to decision making in large POMDPs due to breaking both curses with statistical sampling and black box simulation [Silver and Veness, 2010; Somani *et al.*, 2013]. State-of-the-art algorithms like POMCP construct sparse *closed-loop* search trees over belief states and actions. Although these approaches are computationally efficient, the search trees can become arbitrarily large in highly complex domains. In such domains, performance could be limited by restricted memory resources, which is common in sensor networks or IoT settings with intelligent devices that have to make decisions with very limited resources and perception capabilities.

Open-loop planning is an alternative approach to efficient decision making, which only optimizes sequences of actions independently of the belief state space. Although open-loop planning generally converges to suboptimal solutions, it has been shown to be competitive against closed-loop planning in practice, when the problem is too large to provide sufficient computational resources [Weinstein and Littman, 2013; Perez Liebana *et al.*, 2015; Lecarpentier *et al.*, 2018]. However, open-loop approaches have been rarely used in POMDPs so far, despite their potential to efficient planning in very large domains [Yu *et al.*, 2005; Phan *et al.*, 2019].

In this paper, we propose *Stable Yet Memory Bounded Open-Loop (SYMBOL) planning*, a general memory bounded approach to partially observable open-loop planning. SYMBOL maintains an adaptive stack of Thompson Sampling bandits, which is matched to successive time steps of the decision process. The stack size is bounded by the planning horizon and can be automatically adapted on demand according to the underlying domain without any prior domain knowledge beyond a generative model.

We empirically test SYMBOL in four large benchmark problems to demonstrate its effectiveness and robustness w.r.t. the choice of hyperparameters and evaluate its adaptive memory consumption. We also compare its performance with other open-loop planning algorithms and POMCP.

2 Background

2.1 POMDPs

A POMDP is defined by a tuple $M = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \Omega, b_0 \rangle$, 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 function, $r_t = \mathcal{R}(s_t, a_t) \in \mathbb{R}$ is the 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 probabilitiy distribution over initial states $s_0 \in \mathcal{S}$ [Kaelbling *et al.*, 1998]. 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.

A *history* $h_t = [a_0, o_1, ..., o_{t-1}, a_{t-1}, o_t]$ 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. $\mathcal{B}$ is the space of all possible belief states. The belief state can be updated by Bayes theorem $b_{h_t}(s_t) = \eta\Omega(o_t|s_t, a) \sum_{s\in\mathcal{S}} \mathcal{P}(s_t|s, a)b_h(s)$, where $\eta = \frac{1}{\Omega(o_{t+1}|b_h, a)}$ is a normalizing constant, $a = a_{t-1}$ is the last action, and $h = h_{t-1}$ is the history without a and o_t.

The goal is to find a policy $\pi : \mathcal{B} \rightarrow \mathcal{A}$, which maximizes the expectation of return G_t for a horizon T:

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

where $\gamma \in [0, 1]$ is the discount factor.

π can be evaluated with a value function $V^{\pi}(b_{h_t}) = \mathbb{E}_{\pi}[G_t|b_{h_t}]$, which is the expected return conditioned on belief states. An optimal policy π^* has a value function V^*, where $V^*(b_{h_t}) \geq V^{\pi'}(b_{h_t})$ for all $b_{h_t} \in \mathcal{B}$ and all $\pi' \neq \pi^*$.

2.2 Multi-armed Bandits

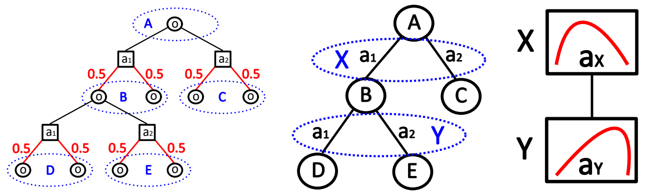
Multi-armed Bandits (MABs) are decision making problems with a single state s. An agent has to repeatedly select an action $a \in \mathcal{A}$ in order to maximize its expected reward $\mathbb{E}[\mathcal{R}(s, a) = X_a]$, where X_a is a random variable with an unknown distribution $f_{X_a}(x)$. The agent has to balance between exploring actions to estimate their expected reward and exploiting its knowledge on all actions by selecting the action with the currently highest expected reward. This is 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. UCB1 and Thompson Sampling are possible approaches to solve MABs.

UCB1. selects actions 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 average reward of action 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 is the *exploration bonus*, which becomes smaller with increasing n_a [Auer *et al.*, 2002; Kocsis and Szepesvári, 2006]. UCB1 is a popular MAB algorithm and widely used in various challenging domains [Kocsis and Szepesvári, 2006; Bubeck and Munos, 2010; Silver and Veness, 2010].

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 $a \in \mathcal{A}$ is modeled by a parametrized likelihood function $P_a(x|\theta)$ with parameter vector θ. Given a prior distribution $P_a(\theta)$ and a set of past observed rewards $D_a = \{x_1, x_2, ..., x_{n_a}\}$, the posterior distribution $P_a(\theta|D_a)$ can be inferred by using Bayes rule $P_a(\theta|D_a) \propto \prod_i P_a(x_i|\theta)P_a(\theta)$. The expected reward of each action $a \in \mathcal{A}$ can be estimated by sampling $\theta \sim P_a(\theta|D_a)$ to compute $\mathbb{E}_{\theta}[X_a]$. The action with the highest 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 *et al.*, 2012].

2.3 Online Planning in POMDPs

Planning searches for an (near-)optimal policy given a model $\hat{M}$ of the environment M, which usually consists of explicit probability distributions of the POMDP. Unlike *global planning*, which searches the whole (belief) state space to find an optimal policy π^*, *local planning* only focuses on finding a policy π_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 *nb* due to strict real-time constraints [Bubeck and Munos, 2010; Weinstein and Littman, 2013].



(a) closed-loop tree (b) open-loop tree (c) stacked

Figure 1: Illustration of closed- and open-loop planning schemes. (a) Closed-loop tree with state observations (circular nodes) and actions (rectangular nodes). Red links correspond to stochastic observations made with a probability of 0.5. (b) Open-loop tree with links as actions and history distribution nodes according to the blue dotted ellipses in Fig. 1a. (c) Open-loop approach with a stack of action distributions according to the blue dotted ellipses in Fig. 1b.

We focus on local *Monte-Carlo planning*, where $\hat{M}$ is a generative model, which can be used as black box simulator [Silver and Veness, 2010]. 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 algorithms can approximate π^* and V^* by iteratively simulating and evaluating actions 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 $p_T = [a_1, ..., a_T]$ (also called *open-loop plans* or simply *plans*) and summarized statistics about predecessor (belief) states [Bubeck and Munos, 2010; Perez Liebana *et al.*, 2015]. An example from [Phan *et al.*, 2019] is shown in Fig. 1. A closed-loop tree for $\Omega(o_{t+1}|s_t, a_t) = 0.5$ 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, a *stack* of T statistics is used to sample and evaluate plans [Weinstein and Littman, 2013].

Partially Observable Monte-Carlo Planning (POMCP) is a closed-loop approach based on *Monte-Carlo Tree Search (MCTS)* [Silver and Veness, 2010]. POMCP uses a search tree of histories with *o-nodes* representing observations and *a-nodes* representing actions (Fig. 1a). The tree is traversed by selecting *a-nodes* with a policy π_{tree} until a leaf *o-node* $o_t \in \mathcal{O}$ is reached, which is expanded, and its value $\hat{V}(h_t)$ is estimated with a rollout by using a policy $\pi_{rollout}$. $\pi_{rollout}$ can be used to integrate domain knowledge into the planning process to focus the search on promising states [Silver and

Veness, 2010]. The observed rewards are recursively accumulated (Eq. 1) to update the value estimate of each node in the simulated path. The original version of POMCP uses UCB1 for π_{tree} and converges to the optimal best-first tree given sufficient computation [Silver and Veness, 2010].

[Lecarpentier *et al.*, 2018] formulates an open-loop variant of MCTS using UCB1 as π_{tree}, called *Open-Loop Upper Confidence bound for Trees (OLUCT)*, which could be easily extended to POMDPs by constructing a tree, which summarizes all *o-nodes* to history distribution nodes (Fig. 1b).

Open-loop planning generally converges to suboptimal solutions in stochastic domains, since it ignores (belief) state values $V(b_{h_t})$ and optimizes the summarized values $V(N_t)$ of each node N_t (Fig. 1b) instead [Lecarpentier *et al.*, 2018]. If the problem is too complex to provide sufficient computation budget *nb* or memory capacity, then open-loop approaches are competitive against closed-loop approaches, since they need to explore a much smaller search space to find an appropriate solution [Weinstein and Littman, 2013; Perez Liebana *et al.*, 2015; Lecarpentier *et al.*, 2018].

3 Related Work

Previous stack based approaches to open-loop planning maintain a fixed size stack of T statistics over actions to sample open-loop plans with high expected return [Weinstein and Littman, 2013; Belzner and Gabor, 2017; Phan *et al.*, 2019]. While these approaches work well in practice, their convergence properties remain unclear because of the non-stationarity of the sampling statistics and the underlying state distributions due to the simultaneous adaptation of each statistic. Furthermore, the required number of statistics is highly domain dependent and hard to prespecify. SYMBOL maintains an *adaptive stack* of Thompson Sampling bandits, which automatically adjusts its size according to the underlying domain without prior domain knowledge. The creation and adaptation of each bandit depends on the *convergence of all preceding* bandits to preserve a stationary state and reward distribution for proper convergence of all bandits.

[Yu *et al.*, 2005] proposed an open-loop approach to decision making 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. SYMBOL is more general, since it performs planning directly on the *original problem* by using a generative model for black box optimization and does not require the POMDP to be transformed for hierarchical planning.

[Powley *et al.*, 2017] proposed a memory bounded version of MCTS with a fixed size 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. SYMBOL does not require a pool to reuse states or nodes but maintains an *adaptive stack* of Thompson Sampling bandits. The bandits adapt according to the *temporal dependencies* between actions, while the size of the bandit stack is bounded by the planning horizon and *automatically adapts* itself according to the underlying domain.

Algorithm 1 Generalized Thompson Sampling

```
procedure ThompsonSampling($N_t$)
    for $a_t \in \mathcal{A}$ do
        Infer $\langle \mu_1, \lambda_1, \alpha_1, \beta_1 \rangle$ from prior and $\overline{X_{a_t}}, \sigma^2_{a_t}, n_{a_t}$
        $\mu_{a_t}, \tau_{a_t} \sim \mathcal{NG}(\mu_1, \lambda_1, \alpha_1, \beta_1)$
    return $argmax_{a_t \in \mathcal{A}}(\mu_{a_t})$

procedure UpdateBandit($N_t, G_t, a_t$)
    $\langle \overline{X_{old,a_t}}, \overline{X_{a_t}} \rangle \leftarrow \langle \overline{X_{a_t}}, (n_{a_t}\overline{X_{old,a_t}} + G_t)/(n_{a_t}+1) \rangle$
    $n_{a_t} \leftarrow n_{a_t} + 1$
    $\sigma^2_{a_t} \leftarrow [(n_{a_t}-1)\sigma^2_{a_t} + (G_t - \overline{X_{old,a_t}})(G_t - \overline{X_{a_t}})]/n_{a_t}$
    $\delta_{a_t} \leftarrow |\overline{X_{a_t}} - \overline{X_{old,a_t}}|$
    return $\delta_{a_t}$
```

4 Adaptive Thompson Sampling Stacks

4.1 Generalized Thompson Sampling

We use a variant of Thompson Sampling, which works for arbitrary reward distributions as proposed in [Bai *et al.*, 2013; Bai *et al.*, 2014] by assuming that X_{a_t} 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. $\langle \mu, \tau \rangle$ 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 $\tau \sim \mathit{Gamma}(\alpha, \beta)$ 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 $P(\theta) = \mathcal{NG}(\mu_0, \lambda_0, \alpha_0, \beta_0)$ 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 $\mu_1 = \frac{\lambda_0\mu_0 + n\overline{X}}{\lambda_0 + n}$, $\lambda_1 = \lambda_0 + n$, $\alpha_1 = \alpha_0 + \frac{n}{2}$, and $\beta_1 = \beta_0 + \frac{1}{2}(n\sigma^2 + \frac{\lambda_0 n(\overline{X} - \mu_0)^2}{\lambda_0 + n})$. $\overline{X}$ 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 $a_t \in \mathcal{A}$ to sample an estimate μ_{a_t} for the expected return. The action with the highest μ_{a_t} is selected. The complete formulation is given in Algorithm 1. A MAB N_t stores $\overline{X_{a_t}}$, $\sigma^2_{a_t}$, and n_{a_t} for each action $a_t \in \mathcal{A}$. In *UpdateBandit*, the absolute difference $\delta_{a_t} = |\overline{X_{a_t}} - \overline{X_{old,a_t}}|$ between the old and the new mean value of $\overline{X_{a_t}}$ is returned to evaluate the convergence of N_t.

If sufficient domain knowledge for defining the prior is unavailable, the prior should be chosen such that all possibilities can be sampled (almost) uniformly [Bai *et al.*, 2014]. This can be achieved by choosing the priors such that the variance of the resulting Normal distribution $\mathcal{N}(\mu_0, \frac{1}{\lambda_0\tau})$ becomes infinite ($\frac{1}{\lambda_0\tau_0} \rightarrow \infty$ and $\lambda_0\tau \rightarrow 0$). Since τ follows a Gamma distribution $\mathit{Gamma}(\alpha_0, \beta_0)$ with expectation $\mathbb{E}[\tau] = \frac{\alpha_0}{\beta_0}$, α_0 and β_0 should be chosen such that $\frac{\alpha_0}{\beta_0} \rightarrow 0$. 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 small enough and β_0 should be sufficiently large [Bai *et al.*, 2014].

4.2 SYMBOL

Stable Yet Memory Bounded Open-Loop (SYMBOL) planning is a partially observable open-loop approach, which optimizes an adaptive stack of *nMAB* Thompson Sampling bandits (with $nMAB \leq T$) to maximize the expected return.

Initially beginning with a single MAB N_1, a simulation starts at state s_1, which is sampled from an approximated belief state $\hat{b}_{h_t} \approx b_{h_t}$. In this work, we use a particle filter $\hat{b}_{h_t}$ to approximate b_{h_t} as proposed in [Silver and Veness, 2010]. The first *nMAB* actions a_t are sampled from all MABs in the current stack. The remaining $T - nMAB$ actions are sampled from a rollout policy $\pi_{rollout}$, which can be random or enhanced with domain knowledge [Silver and Veness, 2010]. The sampled plan $p_T = [a_1, ..., a_T]$ is evaluated with $\hat{M}$ to observe rewards $r_1, ..., r_T$, which are accumulated to returns $G_1, ..., G_T$ according to Eq. 1. The first *nMAB* returns are used to update the MAB stack. A MAB N_t is only updated or created, when all of its predecessors N_{t-k} with $0 < k < t$ converged. We assume that N_t converged, if $\overline{\delta}_{a_t} < \epsilon$, where $\overline{\delta}_{a_t}$ is the average of the last κ values of δ_{a_t} from previous updates to N_t (Algorithm 1). The parameter κ copes with the non-stationarity of the return values G_t to update each MAB, which is caused by the adaptation of the action selection.

The complete formulation of SYMBOL is given in Algorithm 2, where h_t is the action-observation history, T is the planning horizon, *nb* is the computation budget, κ is the *convergence tolerance* represented by the number of MAB updates to be considered, and ϵ is the *convergence threshold*.

Algorithm 2 SYMBOL Planning

```
procedure SYMBOL($h_t, T, nb, \kappa, \epsilon$)
    $nMAB \leftarrow 1$, Create first MAB $N_1$
    while $nb > 0$ do
        $nb \leftarrow nb - 1$
        $s_1 \sim \hat{b}_{h_t}$
        Simulate($s_1, T, \kappa, \epsilon$)
    return $argmax_{a_1 \in \mathcal{A}}(\overline{X_{a_1}})$

procedure Simulate($s_1, T, \kappa, \epsilon$)
    $t \leftarrow 1$
    while $t \leq T$ and $s_t$ is no terminal state do
        if $t \leq nMAB$ then
            $a_t \leftarrow ThompsonSampling(N_t)$
        else
            $a_t \leftarrow \pi_{rollout}(s_t)$
        $\langle s_{t+1}, r_t, o_{t+1} \rangle \sim \hat{M}(s_t, a_t)$        ▷ Simulate action
        $t \leftarrow t + 1$
    $H \leftarrow t$
    $G_{t+1} \leftarrow 0$
    for $t \in H, ..., 1$ do        ▷ Accumulate rewards (Eq. 1)
        $G_t \leftarrow r_t + \gamma G_{t+1}$
    for $t \in 1, ..., H$ do
        if $t \leq nMAB + 1$ and $t \leq T$ and $\overline{\delta}_{a_{t-1}} < \epsilon$ then
            if $t > nMAB$ then
                $nMAB \leftarrow nMAB + 1$, Create new $N_t$
            $\delta_{a_t} \leftarrow UpdateBandit(N_t, G_t, a_t)$
            Update $\overline{\delta}_{a_t}$ with average of last $\kappa$ values of $\delta_{a_t}$
        else
            break        ▷ Keep successor MABs stationary
```

Thompson Sampling bandits are able to converge, if their reward distributions are stationary [Agrawal and Goyal, 2013]. A reward distribution is stationary, if both the underlying state distribution and the successor policy π_{succ} are stationary. The former is ensured, if all preceding MABs converged, since their actions affect the underlying state distribution. The latter is ensured by keeping all successing MABs fixed (they are not updated unless the predecessors converged) and by using a stationary rollout policy $\pi_{rollout}$.

Starting from the first MAB N_1, we know that the state distribution $\hat{b}_{h_t}$ is stationary. If the successor policy π_{succ} is stationary as well, N_1 will converge to the best action given π_{succ} [Agrawal and Goyal, 2013]. By induction, we can show the same for all successor MABs, given that all predecessor MABs converged. If N_t significantly changed such that $\overline{\delta}_{a_t} \geq \epsilon$, all MABs N_{t+k} with $k > 0$ must remain fixed to ensure a stationary reward distribution to enable convergence of N_t first. By adjusting the parameters κ and ϵ, the size and speed of convergence of the MAB stack can be controlled.

5 Experiments

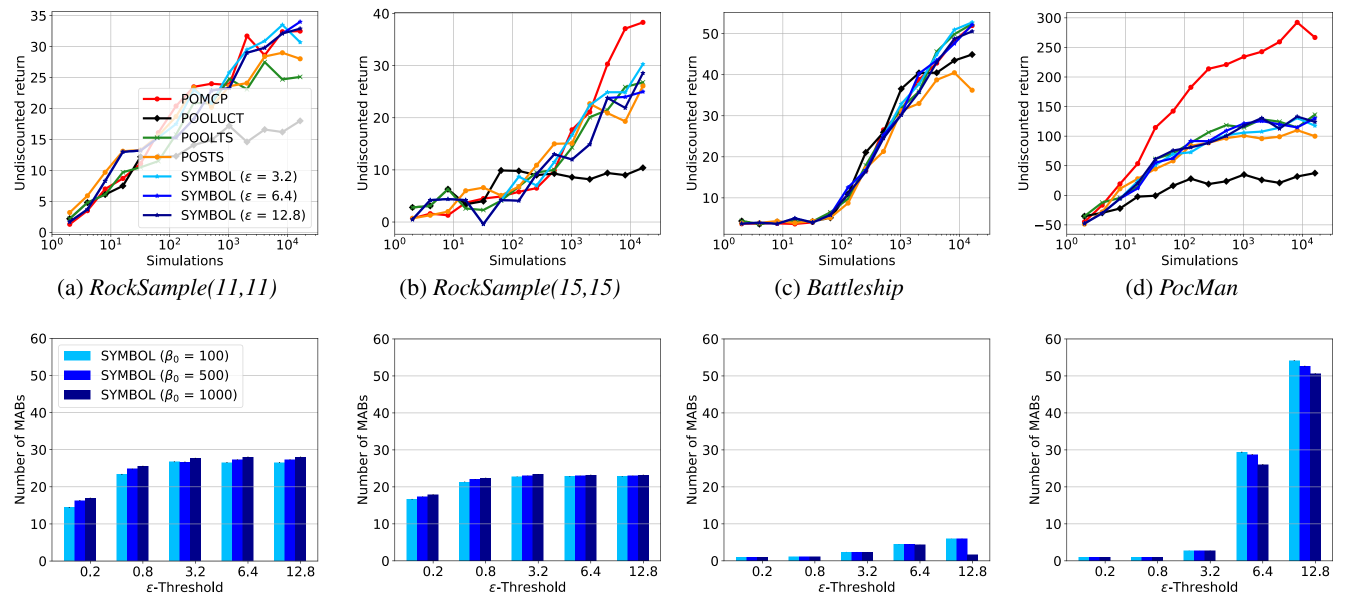
5.1 Evaluation Environments

We tested SYMBOL in different POMDP benchmark problems [Silver and Veness, 2010; Somani *et al.*, 2013]. We always set γ as proposed in [Silver and Veness, 2010].

The *RockSample(n,k)* problem simulates an agent moving in an $n \times n$ grid, which contains 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 $o_t \in \{good, bad\}$ 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 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 10×10 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 17×19 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 $\gamma = 0.95$.



(a) *RockSample(11,11)* (b) *RockSample(15,15)* (c) *Battleship* (d) *PocMan*

(e) *RockSample(11,11)* (f) *RockSample(15,15)* (g) *Battleship* (h) *PocMan*

Figure 2: Sensitivity analysis of SYMBOL w.r.t. the ϵ-threshold with a horizon of $T = 100$ and $\kappa = 8$. (a-d) Average performance w.r.t. to different computation budgets *nb* compared to POMCP, POOLUCT, POOLTS, and POSTS (e-h) MAB stack size given $nb = 4096$.

5.2 Methods

We implemented different partially observable planning algorithms to compare with SYMBOL [1]. All algorithms with a rollout phase use a policy $\pi_{rollout}$ which randomly selects actions from a set of legal actions $a_t \in \mathcal{A}_{legal}(s')$, depending on the currently simulated state $s' \in \mathcal{S}$. Since open-loop planning can encounter different states at the same node or time step (Fig. 1), the set of legal actions $\mathcal{A}_{legal}(s_t)$ may vary for each state $s_t \in \mathcal{S}$. Thus, we mask out currently illegal actions, regardless of whether they have high action values.

POMCP. We use the POMCP implementation from [Silver and Veness, 2010]. π_{tree} selects actions from $\mathcal{A}_{legal}(s_t)$ with UCB1. In each simulation step, there is at most one expansion step, where new nodes are added to the search tree. Thus, the tree size should increase linearly w.r.t. *nb* in large POMDPs.

POOLUCT and POOLTS. are implemented as open-loop versions of POMCP (Fig. 1b), where actions are selected from $\mathcal{A}_{legal}(s_t)$ using UCB1 (POOLUCT) or Thompson Sampling (POOLTS) as node selection policy π_{tree}. Similarly to POMCP, the search tree size should increase linearly w.r.t. *nb*, but with less nodes, since open-loop trees store summarized information about history distributions (Fig. 1b).

SYMBOL. uses an adaptive stack of *nMAB* Thompson Sampling bandits N_t according to Algorithm 2. Starting at s_1, all MABs N_t apply Thompson Sampling to $\mathcal{A}_{legal}(s')$, depending on the currently simulated state $s' \in \mathcal{S}$. If $t >$ *nMAB*, then $\pi_{rollout}$ is used. Given a horizon of T, SYMBOL always maintains *nMAB* $\leq T$ MABs. Although *nMAB* depends on κ, ϵ, and *nb*, it never exceeds T (Algorithm 2).

[1]Code is available at https://github.com/thomyphan/planning

Partially Observable Stacked Thompson Sampling (POSTS). uses a *fixed size stack* of *nMAB* $= T$ Thompson Sampling bandits N_t [Phan *et al.*, 2019]. Similarly to SYMBOL, all MABs N_t apply Thompson Sampling to $\mathcal{A}_{legal}(s')$, depending on the currently simulated state $s' \in \mathcal{S}$. Unlike SYMBOL, all MABs N_t are updated *simultaneously* according to G_t regardless of the convergence of the preceding MABs as suggested in [Weinstein and Littman, 2013].

5.3 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 POOLUCT we set the UCB1 exploration constant c to the reward range of each domain as proposed in [Silver and Veness, 2010].

Since we assume no additional domain knowledge, we focus on uninformative priors with $\mu_0 = 0$, $\alpha_0 = 1$, and $\lambda_0 = 0.01$ [Bai *et al.*, 2014]. With this setting, β_0 controls the degree of initial exploration during the planning phase, thus we only vary β_0 for POOLTS, POSTS, and SYMBOL. To preserve readability of the figures, we only provide the best configuration of POOLTS and POSTS for comparison.

Hyperparameter Sensitivity

We evaluated the sensitivity of SYMBOL w.r.t. the convergence threshold ϵ. Fig. 2a-d show the performance of SYMBOL with $\epsilon \in \{3.2, 6.4, 12.8\}$ compared to all other approaches described in Section 5.2. All SYMBOL variants are able to keep up with their tree-based counterpart POOLTS, while outperforming POOLUCT. SYMBOL is able to keep up with POMCP in *RockSample(11,11)* and *Battleship*. POMCP outperforms all open-loop approaches in *PocMan*. SYMBOL scales better in performance with increasing *nb* than POSTS, which seems to converge prematurely after $nb > 1000$. Except in *PocMan*, SYMBOL scales slightly better with increasing *nb* when $\epsilon = 3.2$, probably due to more stable convergence of the MABs. Fig. 2e-h show the average stack sizes *nMAB* of SYMBOL for $nb = 4096$ [2], $\beta_0 \in \{100, 500, 1000\}$, and different ϵ. In *RockSample*, $20 < nMAB < 30$, when $\epsilon \geq 0.8$, but it does not grow any further. In *Battleship*, $nMAB < 10$, but the stack size slightly increases w.r.t ϵ . In *PocMan*, *nMAB* quickly increases w.r.t ϵ. β_0 does not have any significant impact on *nMAB*.

We also experimented with the convergence tolerance $\kappa \in \{2, 4, 8, 16, 32\}$ but did not observe significantly different results than shown in Fig. 2. *nMAB* tends to decrease with increasing κ, which is due to the amount of time required to consider a MAB as converged. When $\kappa < 8$, then SYMBOL was less stable in all domains (except in *Battleship*), leading to high variance in performance when *nb* is large.

Performance-Memory Tradeoff

We evaluated the performance-memory tradeoff of all approaches by introducing a memory capacity *nMEM*, where the computation is interrupted, when the number of nodes exceeds *nMEM*. For POMCP, we count the number of *o-nodes* and *a-nodes* (Fig. 1a). For POOLTS and POOLUCT, we count the number of history distribution nodes (Fig. 1b). For SYMBOL and POSTS, we count *nMAB*. POSTS always uses a planning horizon of $min(T, nMEM)$ to satisfy the memory bound. The results are shown in Fig. 3 for $nb = 4096$, $T = 100$, $\beta_0 \in \{100, 500, 1000\}$ for POOLTS, POSTS, and SYMBOL, $\epsilon = 6.4$, and $\kappa = 8$.

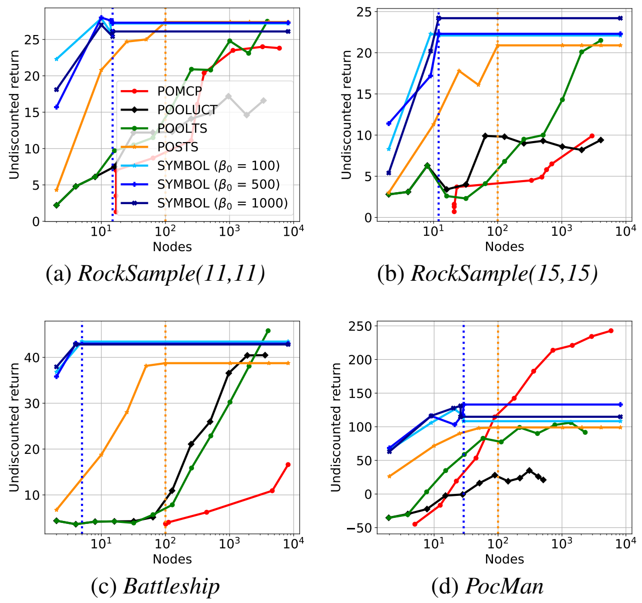
In *RockSample* and *Battleship*, POMCP is outperformed by SYMBOL and POOLTS. SYMBOL always performs best in these domains, when $nMEM \leq 1000$. POMCP performs best in *PocMan* by outperforming SYMBOL, when $nMEM > 100$ and POOLTS keeps up with SYMBOL, when $nMEM > 1000$. SYMBOL always outperforms POSTS, while using a lower maximum number of MABs. POSTS is only able to keep up with the best SYMBOL setting in *RockSample(11,11)* after creating 100 MABs, while SYMBOL only uses about 20 MABs for planning. POOLUCT performs worst except in *Battleship*, improving less and slowest with increasing *nMEM*.

6 Discussion

We presented SYMBOL, a general memory bounded approach to partially observable open-loop planning with an adaptive stack of Thompson Sampling bandits.

Our experiments show that SYMBOL is a good alternative to tree-based planning in POMDPs. SYMBOL is competitive against tree-based open-loop planning like POOLUCT and POOLTS and is able to keep up with POMCP in domains with large action spaces and low stochasticity like *RockSample* or *Battleship*. SYMBOL is robust w.r.t. the choice of

[2]Using budgets between 1024 and 16384 led to similar plots, thus we stick to $nb = 4096$ as suggested in [Phan *et al.*, 2019].



(a) *RockSample(11,11)* (b) *RockSample(15,15)* (c) *Battleship* (d) *PocMan*

Figure 3: Average performance of POMCP, POOLUCT, POOLTS, POSTS, and SYMBOL with memory bounds, $nb = 4096$, $T = 100$, $\kappa = 8$, and $\epsilon = 6.4$. The vertical dotted lines indicate the maximum number of MABs used by SYMBOL (blue) and POSTS (orange).

the hyperparameters β_0 and ϵ in terms of performance, with ϵ strongly affecting the memory consumption in domains with high stochasticity as shown for *PocMan* in Fig. 2h. ϵ should be sufficiently small to ensure stable convergence of the MABs, although more computation budget will be required to build up adequate MAB stacks, if ϵ is too small. Fig. 2e-h indicate that appropriate MAB stack sizes are highly domain dependent and cannot be generally specified beforehand without expensive parameter tuning. Thus, adaptive and robust approaches like SYMBOL seem to be promising for general and efficient decision making in POMDPs.

When restricting the memory capacity, SYMBOL clearly outperforms all tree-based approaches, while requiring significantly less MABs. Although being bounded by $T = 100$ at most, SYMBOL always created much less MABs in all domains, resulting in extremely memory efficient planning (Fig. 3). POOLTS requires thousands of nodes to keep up with SYMBOL, while POMCP is only able to outperform SYMBOL in *PocMan* after creating more than 100 nodes, which still consumes much more memory than SYMBOL (Fig. 3d).

SYMBOL is able to outperform the fixed size stack approach POSTS, showing the effectiveness of the adaptive stack concept, where proper convergence is ensured by the convergence threshold ϵ and the convergence tolerance κ.

While state-of-the-art approaches to efficient online planning [Silver and Veness, 2010; Somani *et al.*, 2013; Bai *et al.*, 2014] heavily rely on sufficient memory resources in highly complex domains, SYMBOL is a memory bounded alternative, which maintains an adaptive stack of MABs. SYMBOL is able to automatically adapt its stack according to the underlying domain without any prior domain knowledge.

References

[Agrawal and Goyal, 2013] Shipra Agrawal and Navin Goyal. Further Optimal Regret Bounds for Thompson Sampling. In *Artificial Intelligence and Statistics*, pages 99–107, 2013.

[Auer *et al.*, 2002] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-Time Analysis of the Multiarmed Bandit Problem. *Machine learning*, 47(2-3):235–256, 2002.

[Bai *et al.*, 2013] Aijun Bai, Feng Wu, and Xiaoping Chen. Bayesian Mixture Modelling and Inference based Thompson Sampling in Monte-Carlo Tree Search. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2013.

[Bai *et al.*, 2014] Aijun Bai, Feng Wu, Zongzhang Zhang, and Xiaoping Chen. Thompson Sampling based Monte-Carlo Planning in POMDPs. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling*, pages 29–37. AAAI Press, 2014.

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

[Bubeck and Munos, 2010] Sébastien Bubeck and Rémi Munos. Open Loop Optimistic Planning. In *COLT*, pages 477–489, 2010.

[Chapelle and Li, 2011] Olivier Chapelle and Lihong Li. An Empirical Evaluation of Thompson Sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial intelligence*, 101(1):99–134, 1998.

[Kaufmann *et al.*, 2012] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson Sampling: An Asymptotically Optimal Finite-Time Analysis. In *International Conference on Algorithmic Learning Theory*, pages 199–213. Springer, 2012.

[Kocsis and Szepesvári, 2006] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo Planning. In *ECML*, volume 6, pages 282–293. Springer, 2006.

[Lecarpentier *et al.*, 2018] Erwan Lecarpentier, Guillaume Infantes, Charles Lesire, and Emmanuel Rachelson. Open Loop Execution of Tree-Search Algorithms. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2362–2368. IJCAI Organization, 7 2018.

[Perez Liebana *et al.*, 2015] Diego Perez Liebana, Jens Dieskau, Martin Hunermund, Sanaz Mostaghim, and Simon Lucas. Open Loop Search for General Video Game Playing. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 337–344. ACM, 2015.

[Phan *et al.*, 2019] Thomy Phan, Lenz Belzner, Marie Kiermeier, Markus Friedrich, Kyrill Schmid, and Claudia Linnhoff-Popien. Memory Bounded Open-Loop Planning in Large POMDPs Using Thompson Sampling. *33th AAAI Conference on Artificial Intelligence*, 2019.

[Pineau *et al.*, 2006] Joelle Pineau, Geoffrey Gordon, and Sebastian Thrun. Anytime Point-based Approximations for Large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006.

[Powley *et al.*, 2017] Edward Powley, Peter Cowling, and Daniel Whitehouse. Memory Bounded Monte Carlo Tree Search. *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2017.

[Silver and Veness, 2010] David Silver and Joel Veness. Monte-Carlo Planning in Large POMDPs. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[Smith and Simmons, 2004] Trey Smith and Reid Simmons. Heuristic Search Value Iteration for POMDPs. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 520–527. AUAI Press, 2004.

[Somani *et al.*, 2013] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP Planning with Regularization. In *Advances in neural information processing systems*, pages 1772–1780, 2013.

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

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

[Yu *et al.*, 2005] C Yu, Jason Chuang, Brian Gerkey, G Gordon, and Andrew Ng. Open-Loop Plans in Multi-Robot POMDPs. Technical report, Stanford CS Dept, 2005.