Learning in POMDPs with Monte Carlo Tree Search

Sammie Katt [1] **Frans A. Oliehoek** [2] **Christopher Amato** [1]

Abstract

The POMDP is a powerful framework for reasoning under outcome and information uncertainty, but constructing an accurate POMDP model is difficult. Bayes-Adaptive Partially Observable Markov Decision Processes (BA-POMDPs) extend POMDPs to allow the model to be learned during execution. BA-POMDPs are a Bayesian RL approach that, in principle, allows for an optimal trade-off between exploitation and exploration. Unfortunately, BA-POMDPs are currently impractical to solve for any non-trivial domain. In this paper, we extend the Monte-Carlo Tree Search method POMCP to BA-POMDPs and show that the resulting method, which we call BA-POMCP, is able to tackle problems that previous solution methods have been unable to solve. Additionally, we introduce several techniques that exploit the BA-POMDP structure to improve the efficiency of BA-POMCP along with proof of their convergence.

1. Introduction

The Partially Observable Markov Decision Processes (POMDP) (Kaelbling et al., 1998) is a general model for sequential decision making in stochastic and partially observable environments, which are ubiquitous in real-world problems. A key shortcoming of POMDP methods is the assumption that the dynamics of the environment are known a priori. In real-world applications, however, it may be impossible to obtain a complete and accurate description of the system. Instead, we may have uncertain prior knowledge about the model. When lacking a model, a prior can be incorporated into the POMDP problem in a principled way, as demonstrated by the *Bayes-Adaptive POMDP* framework (Ross et al., 2011).

The BA-POMDP framework provides a Bayesian approach to decision making by maintaining a probability distribution over possible models as the agent acts in an online reinforcement learning setting (Duff, 2002; Wyatt, 2001). This method casts the Bayesian reinforcement learning problem into a POMDP planning problem where the hidden model of the environment is part of the state space. Unfortunately, this planning problem becomes very large, with a continuous state space over all possible models, and as such, current solution methods are not scalable or perform poorly (Ross et al., 2011).

Online and sample-based planning has shown promising performance on non-trivial POMDP problems (Ross et al., 2008). Online methods reduce the complexity by considering the relevant (i.e., reachable) states only, and sample-based approaches tackle the complexity issues through approximations in the form of simulated interactions with the environment. Here we modify one of those methods, *Partial Observable Monte-Carlo Planning* (POMCP) (Silver & Veness, 2010) and extend it to the *Bayes-Adaptive* case, leading to a novel approach: BA-POMCP.

In particular, we improve the sampling approach by exploiting the structure of the BA-POMDP resulting in *root sampling* and *expected models* methods. We also present an approach for more efficient state representation, which we call *linking states*. Lastly, we prove the correctness of our improvements, showing that they converge to the true BA-POMDP solution. As a result, we present methods that significantly improve the scalability of learning in BA-POMDPs, making them practical for larger problems.

2. Background

First, we discuss POMDPs and BA-POMDPs in Sections 2.1 and 2.2.

2.1. POMDPs

Formally, a POMDP is described by a tuple (S, A, Z, D, R, γ, h), where S is the set of states of the environment; A is the set of actions; Z is the set of observations; D is the 'dynamics function' that describes the dynamics of the system in the form of transition probabilities $D(s',z|s,a)$;[1]

[1]Northeastern University, Boston, Massachusetts,USA [2]University of Liverpool, UK. Correspondence to: Sammie Katt <katt.s@husky.neu.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

[1] This formulation generalizes the typical formulation with separate transition T and observation functions O: $D = \langle T, O \rangle$. In our experiments, we do employ this typical factorization.

R is the immediate reward function $R(s,a)$ that describes the reward of selecting a in s; $\gamma \in (0,1)$ is the discount factor; and h is the horizon of an episode in the system.

The goal of the agent in a POMDP is to maximize the expected cumulative (discounted) reward, also called the expected return. The agent has no direct access to the system's state, so it can only rely on the *action-observation history* $h_t = \langle a_0,z_1,\ldots,a_{t-1},z_t\rangle$ up to the current step t. It can use this history to maintain a probability distribution over the state, also called a belief, $b(s)$. A solution to a POMDP is then a mapping from a belief b to an action a, which is called a *policy* π. Solution methods aim to find an optimal policy, a mapping from a belief to an action with the highest possible expected return.

The agent maintains its belief during execution through *belief updates*. A belief update calculates the posterior probability of the state s' given the previous belief over s and action-observation pair $\langle a,z\rangle$: $b'(s) = P(s'|b(s),a,z)$. This operation is infeasible for large spaces because it enumerates over the entire state space. A common approximation method is to represent the belief with (unweighted) *particle filters* (Thrun, 1999). A particle filter is a collection of K particles (states). Each particle represents a probability of $\frac{1}{K}$; if a specific state x occurs n times in a particle filter, then $P(x) = \frac{n}{K}$. The precision of the filter is determined by the amount of particles K. To update such a belief after each time step with observation z, a standard approach is to utilize *rejection sampling*. Given some action a, the agent repeatedly samples a state s from particle filter, then simulates the execution of a on s through D, and receives a (simulated) new state s'_{sim} and observation z_{sim}. s' is added to the new belief only when $z_{sim} == z$, and rejected otherwise. This process repeats until the new belief contains K particles.

Partially Observable Monte-Carlo Planning (POMCP) (Silver & Veness, 2010), is a scalable method which extends Monte Carlo tree search (MCTS) to solve POMDPs. POMCP is one of the leading algorithms for solving general POMDPs. At each time step, the algorithm performs online planning by incrementally building a lookahead tree that contains (statistics that represent) $Q(h,a)$, where h is the action-observation history h to reach that node. It samples hidden states s at the root node (called 'root sampling') and uses that state to sample a trajectory that first traverses the lookahead tree and then performs a (random) rollout. The return of this trajectory is used to update the statistics for all visited nodes. These statistics include the number of times an action has been taken at a history ($N(h,a)$) and estimated value of being in that node ($Q(h,a)$), based on an average over the returns.

Because this lookahead tree can be very large, the search is directed to the relevant parts by selecting the actions inside the tree that maximize the 'upper confidence bounds' (Auer et al., 2002): $U(h,a) = Q(h,a) + c\sqrt{\log(N(h)+1)/N(h,a)}$. Here, $N(h)$ is the number of times the history has been visited. At the end of each simulation, the discounted accumulated return is used to update the estimated value of all the nodes in the tree that have been visited during that simulation. POMCP terminates after some criteria has been met, typically defined by a maximum number of simulations or allocated time. The agent then picks the action with the highest estimated value ($\max_a Q(b,a)$). POMCP can be shown to converge to an ϵ-optimal value function. Moreover, the method has demonstrated good performance in large domains with a limited number of simulations. The extension of POMCP that is used in this work is discussed in Section 3.

2.2. BA-POMDPs

Most research concerning POMDPs has considered the task of *planning*: given a full specification of the model, determine an optimal policy (e.g., (Kaelbling et al., 1998; Shani et al., 2012)). However, in many real-world applications, the model is not (perfectly) known in advance, which means that the agent has to learn about its environment during execution. This is the task considered in *reinforcement learning* (RL) (Sutton & Barto, 1998).

A fundamental RL problem is the difficulty of deciding whether to select actions in order to learn a better model of the environment, or to exploit the current knowledge about the rewards and effects of actions. In recent years, Bayesian RL methods have become popular because they can provide a principled solution to this exploration/exploitation trade-off (Wyatt, 2001; Duff, 2002; Engel et al., 2005; Poupart & Vlassis, 2008; Vlassis et al., 2012).

In particular, we consider the framework of Bayes-Adaptive POMDPs (Ross et al., 2007; 2011). BA-POMDPs use Dirichlet distributions to model uncertainty over transitions and observations[2] (typically assuming the reward function is chosen by the designer and is known). In particular, if the agent could observe both states and observations, it could maintain a vector χ with the counts of the occurrences for all $\langle s,a,s',z\rangle$ tuples. We write $\chi_{sa}^{s'z}$ for the number of times that s,a is followed by s',z.

While the agent cannot observe the states and has uncertainty about the actual count vector, this uncertainty can be represented using regular POMDP formalisms. That is, the count vector is included as part of the hidden state of a specific POMDP, called a BA-POMDP. Formally, a BA-POMDP is a tuple $\langle \bar{S},A,\bar{D},\bar{R},Z,\gamma,h\rangle$ with some modified components in comparison to the POMDP. While the observation and action space remain unchanged, the state (space) of the BA-POMDP now includes Dirichlet parame-

[2] (Ross et al., 2007; 2011) follow the standard T & O POMDP representations, but we use our combined D formalism.

ters: $\bar{s} = \langle s, \chi \rangle$, which we will refer to as augmented states. The reward model remains the same, since it is assumed to be known, $\bar{R}(\langle s',\chi' \rangle,a) = R(s,a)$. The dynamics functions, $\bar{D}$, however, is described in terms of the counts in $\bar{s}$, and is defined as follows

$$D_\chi(s',z|s,a) \triangleq \mathbf{E}[D(s',z|s,a)|\chi] = \frac{\chi_{sa}^{s'z}}{\sum_{s'z} \chi_{sa}^{s'z}}. \quad (1)$$

These expectations can now be used to define the transitions for the BA-POMDP. If we let $\delta_{sa}^{s'z}$ denote a vector of the length of χ containing all zeros except for the position corresponding to $\langle s,a,s',z \rangle$ (where it has a one), and if we let $\mathbb{I}_{a,b}$ denote the Kronecker delta that indicates (is 1 when) $a = b$, then we can define $\bar{D}$ as $\bar{D}(s',\chi',z|s,\chi,a) = D_\chi(s',z|s,a)\mathbb{I}_{\chi',\chi+\delta_{sa}^{s'z}}$.

Remember that these counts are not observed by the agent, since that would require observations of the state. The agent can only maintain belief over these count vectors. Still, when interacting with the environment, *the ratio of the true—but unknown—count vectors will converge to coincide with the true transition and observation probabilities in expectation*. It is important to realize, however, that this convergence of count vector ratios does not directly imply learnability by the agent: even though the ratio of the count vectors of the true hidden state will converge, *the agent's belief over count vectors might not*.

BA-POMDPs are infinite state POMDP models and thus extremely difficult to solve. Ross et al. (2011) introduced a technique to convert such models to finite models, but these are still very large. Therefore, Ross et al. propose a simple lookahead planner to solve BA-POMDPs in an online manner. This approach approximates the expected values associated with each action at the belief by applying a lookahead search of depth d. This method will function as the comparison baseline in our experiments, as no other BA-POMDP solution methods have been proposed.

3. BA-POMDPs via Sample-based Planning

Powerful methods, such as POMCP (Silver & Veness, 2010), have significantly improved the scalability of POMDP solution methods. At the same time the most practical solution method for BA-POMDPs, the aforementioned lookahead algorithm, is quite limited in dealing with larger problems. POMDP methods have rarely been applied to BA-POMDPs (Amato & Oliehoek, 2015), and no systematic investigation of their performance has been conducted. In this paper, we aim to address this void, by extending POMCP to BA-POMDPs, in an algorithm that we refer to as *BA-POMCP*. Moreover, we propose a number of novel adaptations to BA-POMCP that exploit the structure of the BA-POMDP. In this section, we first lay out the basic adaptation of POMCP to BA-POMDPs and then describe the proposed modifications that improve its efficiency.

Algorithm 1 BA-POMCP($\bar{b}$,num_sims)

1: //$\bar{b}$ is an augmented belief (e.g., particle filter)
2: $h_0 \leftarrow ()$ ▷ The empty history (i.e., now)
3: **for** $i \leftarrow 1 \ldots num\_sims$ **do**
4: //First, we *root sample* an (augmented) state:
5: $\bar{s} \leftarrow$ SAMPLE($\bar{b}$) ▷ reference to a particle
6: $\bar{s}' \leftarrow$ COPY($\bar{s}$)
7: SIMULATE($\bar{s}', 0, h_0$)
8: **end for**
9: $a \leftarrow$ GREEDYACTIONSELECTION(h_0)
10: **return** a

Algorithm 2 SIMULATE($\bar{s}, d, h$)

1: **if** ISTERMINAL(h) || $d == max\_depth$ **then**
2: return 0
3: **end if**
4: //Action selection uses statistics stored at node h:
5: $a \leftarrow$ UCBACTIONSELECTION(h)
6: $R \leftarrow R(\bar{s},a)$
7: $z \leftarrow$ STEP($\bar{s}, a$) //*modifies* $\bar{s}$ to sampled next state
8: $h' \leftarrow (h,a,z)$
9: **if** $h' \in Tree$ **then**
10: $r \leftarrow R + \gamma$ SIMULATE($\bar{s}', d+1, h'$)
11: **else**
12: CONSTRUCTNODE(h') //Initializes statistics
13: $r \leftarrow R + \gamma$ ROLLOUT($\bar{s}', d+1, h'$)
14: **end if**
15: //Update statistics:
16: $N(h,a) \leftarrow N(h,a) + 1$
17: $Q(h,a) \leftarrow \frac{N(h,a)-1}{N(h,a)} Q(h,a) + \frac{1}{N(h,a)} r$
18: **return** r

BA-POMCP BA-POMCP, just like POMCP, constructs a lookahead tree through simulated experiences (Algorithm 1). In BA-POMDPs, however, the dynamics of the system are inaccessible during simulations, and the belief is a probability distribution over augmented states. BA-POMCP, as a result, must sample augmented states from the belief $\bar{b}$, and use copies of those states ($\bar{s} = \langle s,\chi \rangle$) for each simulation (Algorithm 2). We will refer to this as *root sampling of the state* (line 6). The copy is necessary, as otherwise the STEP function in Algorithm 2 would alter the belief $\bar{b}$. It is also expensive, for χ grows with the state, action and observation space, to $|S|^2 \times |A| + |S| * |A| * |\Omega|$ parameters. In practice, this operation becomes a bottleneck to the runtime of BA-POMCP in larger domains.

To apply POMCP on BA-POMDPs, where the dynamics are unknown, we modify the STEP function, proposing several variants. The most straightforward one, NAIVESTEP is employed in what we refer to as 'BA-POMCP'. This method, shown in Algorithm 3, is similar to BA-MCP (Guez et al., 2012): essentially, it samples a dynamic model D_{sa} which specifies probabilities $\Pr(s',z|s,a)$ and subsequently samples an actual next state and observation from that distribution. Note that the underlying states and observations are all represented simply as an index, and hence the assignment on line 5 is not problematic. However, the

Algorithm 3 BA-POMCP-STEP($\bar{s} = \langle s, \chi \rangle, a$)

1: $D_{sa} \sim \chi_{sa}$
2: $\langle s', z \rangle \sim D_{sa}$
3: //In place updating of $\bar{s} = \langle s, \chi \rangle$
4: $\chi_{sa}^{s'z} \leftarrow \chi_{sa}^{s'z} + 1$
5: $s \leftarrow s'$
6: **return** z

Algorithm 4 R-BA-POMCP-STEP ($\bar{s} = \langle s, \chi \rangle, a$)

1: //Sample root sampled function
2: $s', z \sim \dot{D}_{s,a}$
3: $s \leftarrow s'$
4: **return** z

cost of the model sampling operation in line 1 is.

Root Sampling of the Model BA-MCP (Guez et al., 2012) addresses the fully observable BRL problem by using POMCP on an augmented state $\bar{s} = \langle s, T \rangle$, consisting of the observable state, as well as the hidden true transition function T. Application of POMCP's root sampling of state in this case leads to 'root sampling of a transition function': Since the true transition model T does not change during the simulation, one is sampled at the root and used during the entire simulation. In the BA-POMCP case, root sampling of a state $\bar{s} = \langle s, \chi \rangle$ does not lead to a same interpretation: no model, but counts are root sampled and they *do* change over time.

We use this as inspiration to introduce a similar, but clearly different, (since this is not *root sampling of state*) technique called *root sampling of the model* (which we will refer to as just 'root sampling'). The idea is simple: every time we root sample a state $\bar{s} = \langle s, \chi \rangle \sim \bar{b}$ at the beginning of a simulation (line 5 in Algorithm 1), we directly sample a $\dot{D} \sim Dir(\chi)$, which we will refer to as the root-sampled model $\dot{D}$ and it is used for the rest of the simulation.

We denote this root sampling in BA-POMCP as 'R-BA-POMCP'. The approach is formalized by R-BA-POMCP-STEP (Algorithm 4). Note that no count updates take place (cf. line 4 in Algorithm 3). This highlights an important advantage of this technique: since the counts are not used in the remainder of the simulation, the copy of counts (as part of line 6 of Algorithm 1) can be avoided altogether. Since this copy operation is costly, especially in larger domains, where the number of states, action and observations and the number of counts is large, this can lead to significant savings. Finally, we point out that, similar to what Guez et al. (2012) propose, $\dot{D}$ can be constructed lazily: the part of the model $\dot{D}$ is only sampled when it becomes necessary.

The transition probabilities during R-BA-POMCP differ from those in BA-POMCP, and it is not obvious that a policy based on R-BA-POMCP maintains the same guarantees. We prove in Section 4 that the solution of R-BA-POMCP in the limit converges to that of BA-POMCP.

Algorithm 5 E-BA-POMCP-STEP($\bar{s} = \langle s, \chi \rangle, a$)

1: //Sample from *Expected* model
2: $s', z \sim D_{\chi}(\cdot, \cdot | s, a)$
3: $\chi_{sa}^{s'z} \leftarrow \chi_{sa}^{s'z} + 1$
4: $s \leftarrow s'$
5: **return** z

Expected models during simulations The second, complementary, adaptation modifies the way models are sampled from the root-sampled counts in STEP. This version samples the transitions from the *expected* dynamics D_{χ} given in (1), rather than from a sampled dynamics function $D \sim Dir(\chi)$. The latter operation is relatively costly, while constructing D_{χ} is very cheap. In fact, this operation is so cheap, that it is more efficient to (re-)calculate it on the fly rather than to actually store D_{χ}. This approach is shown in Algorithm 5.

Linking States Lastly, we propose a specialized data structure to encode the augmented BA-POMDP states. The structure aims to optimize for the complexity of the count-copy operation in line 6 of Algorithm 1 while allowing modifications to $\bar{s}$.

The linking state s_l is a tuple of a system state, a pointer (or *link*) to an unmodifiable set of counts χ and a set of *updated* counts $\langle s, l, \delta \rangle$. l is a pointer to some set of counts χ, which remain unchanged during count updates (such as in the STEP function), and instead are stored in the set of updated counts, δ, as shown in Algorithm 6. The consequence is that the linking state copy-operation can safely perform a shallow copy of the counts χ, and must only consider δ, which is assumed to be much smaller.

Linking states can be used during the (rejection-sample-based) belief update at the beginning of each real time step. While the root-sampled augmented states (including δ in linking states) are typically deleted at the end of each simulation during L-BA-POMCP, each belief update potentially increases the size of δ of each particle. Theoretically, the number of updated counts represented in δ increases and the size of δ may (eventually) grow similar to the size of χ. Therefore, at some point, it is necessary to construct a new χ' that combines χ and δ (after which δ can be safely emptied). We define a new parameter for the maximum size of δ, λ, and condition to merge only if the size of δ exceeds λ. We noticed that, in practice, the number of merges is much smaller than the amount of copies in BA-POMCP. We also observed in our experiments that it is often the case that a specific (small) set of transitions are notably more popular than others and that δ grows quite slowly.

4. Theoretical Analysis

Here, we analyze the proposed root sampling of the dynamics function and expected transition techniques, and

Algorithm 6 L-BA-POMCP-STEP($s_l = \langle s, l, \delta \rangle, a$)

1: $D \sim \langle l, \delta \rangle$
2: $s', z \sim D_{s,a}$
3: $s \leftarrow s'$
4: $\delta_{sa}^{s'z} \leftarrow \delta_{sa}^{s'z} + 1$
5: **return** z

demonstrate they converge to the solution of the BA-POMDP. These main steps of this proof are similar to those in (Silver & Veness, 2010). We point out however, that the technicalities of proving the components are far more involved. Due to lack of space we will defer a detailed presentation of all these to an extended version of this paper.

The convergence guarantees of the original POMCP method are based on showing that, for an arbitrary rollout policy π, the *POMDP rollout distribution* (the distribution over full histories when performing root sampling of state) is equal to the *derived MDP rollout distribution (*the distribution over full histories when sampling in the belief MDP). Given that these are identical it is easy to see that the statistics maintained in the search tree will converge to the same number in expectation. As such, we will show a similar result here for expected transitions ('expected' for short) and root sampling of the dynamics function ('root sampling' below).

We define H_0 as the *full* history (also including states) at the root of simulation, H_d as the full history of a node at depth d in the simulation tree, and $\chi(H_d)$ as the counts induced by H_d. We then define the rollout distributions:

Definition 1. The ***expected*** *full-history* ***expected-transition*** *BA-POMDP rollout distribution* is the distribution over full histories of a BA-POMDP, when performing Monte-Carlo simulations according to a policy π, while sampling expected transitions. It is given by:

$$P^{\pi}(H_{d+1}) = D_{\chi(H_d)}(s_{d+1},z_{d+1}|a_s,s_d)\pi(a_d|h_d)P^{\pi}(H_d) \tag{2}$$

with $P^{\pi}(H_0) = b_0(\langle s_0,\chi_0 \rangle)$ the belief 'now' (at the root of the online planning).

Note that there are two expectations in the above definition: 'expected transitions' mean that transitions for a history H_d are *sampled* from $D_{\chi(H_d)}$. The other 'expected' is the expectation of those samples (and it is easy to see that this will converge to the expected transition probabilities $D_{\chi(H_d)}(s_{d+1},z_{d+1}|a_s,s_d)$). For root sampling of the dynamics model, this is less straightforward, and we give the definition in terms of the empirical distribution:

Definition 2. The *full-history* ***root-sampling (RS)*** *BA-POMDP rollout distribution* is the distribution over full histories of a BA-POMDP, when performing Monte-Carlo simulations according to a policy π *in combination with root sampling of the dynamics model* D. This distribution, for a particular stage d, is given by $\tilde{P}_K^{\pi}(H_d) \triangleq \frac{1}{K_d}\sum_{i=1}^{K_d}\mathbb{I}_{\left\{H_d = H_d^{(i)}\right\}}$, where K is the number of simulations that comprise the empirical distribution, K_d is the number of simulations that reach depth d (not all simulations might be equally long), and $H_d^{(i)}$ is the history specified by the i-th particle at stage d.

Now, our main theoretical result is that these distributions are the same in the limit of the number of simulations:

Theorem 3. *The full-history RS-BA-POMDP rollout distribution (Def. 2) converges in probability to the quantity of Def. 1:*

$$\forall_{H_d} \quad \tilde{P}_{K_d}^{\pi}(H_d) \xrightarrow{p} P^{\pi}(H_d). \tag{3}$$

Sketch of Proof. (Due to length restrictions we omit some details) At the start of every simulation (we assume they start in some history H_0), RS-BA-POMCP samples a dynamics model $\dot{D}$. We consider probability $\tilde{P}^{\pi}(H_d)$ that RS-BA-POMCP generates a full history H_d at depth d. Clearly $\tilde{P}_{K_d}^{\pi}(H_d) \xrightarrow{p} \tilde{P}^{\pi}(H_d)$, and this quantity can be written out

$$\begin{aligned}\tilde{P}^{\pi}(H_d) &= \int \tilde{P}^{\pi}\left(H_d|\dot{D}\right) Dir(\dot{D}|\dot{\chi})d\dot{D} = \dots \\ &= \prod_{t=1}^{d}\pi(a_{t-1}|h_{t-0}) \prod_{\langle s,a \rangle} \frac{B(\chi_{sa}(H_0))}{B(\chi_{sa}(H_d)}\end{aligned} \tag{4}$$

with $B(\alpha) = \frac{\Gamma(\alpha_1+\ldots+\alpha_k)}{\Gamma(\alpha_1)\cdot\ldots\cdot\Gamma(\alpha_k)}$ the normalization term of a Dirichlet distribution with parametric vector α.

For ease of notation we continue the proof for stage $d+1$. I.e., we will show $\forall_{H_{d+1}} \quad \tilde{P}^{\pi}(H_{d+1}) \xrightarrow{p} P^{\pi}(H_{d+1})$. Note that a history $H_{d+1} = (H_d,a_d,s_{d+1},z_{d+1})$, *only* differs from H_d in that it has one extra transition for the $(s_d,a_d,s_{d+1},z_{d+1})$ quadruple, implying that $\chi(H_{d+1})$ only differs from $\chi(H_d)$ in the counts $\chi_{s_da_d}$ for s_da_d. Therefore (4) can be written in recursive form as

$$\tilde{P}^{\pi}(H_{d+1}) = \tilde{P}^{\pi}(H_d)\pi(a_d|h_d)\frac{B(\chi_{s_da_d}(H_d))}{B(\chi_{s_da_d}(H_{d+1}))} \tag{5}$$

Now, let us write $T = \sum_{(s',z)}\chi_{s_da_d}^{s'z}(H_d)$ for the total of the counts for s_d,a_d and $N = \chi_{s_da_d}^{s_{d+1}z_{d+1}}(H_d)$ for the number of counts for that such a transition was to $(s_{d+1}z_{d+1})$. Because H_{d+1} only has 1 extra transition: $\sum_{(s',z)}\chi_{s_da_d}^{s'z}(H_{d+1}) = T+1$ and since that transition was to $(s_{d+1}z_{d+1})$ the counts $\chi_{s_da_d}^{s_{d+1}z_{d+1}}(H_{d+1}) = N+1$. Now let us expand the rightmost term from (5):

$$\begin{aligned}&= \frac{\Gamma(T)/\prod_{s'z}\Gamma(\chi_{s_da_d}^{s'z}(H_d))}{\Gamma(T+1)/\prod_{s'z}\Gamma(\chi_{s_da_d}^{s'z}(H_{d+1}))} \\ &= \frac{\Gamma(T)}{\Gamma(T+1)}\frac{\Gamma(N+1)}{\Gamma(N)} = \frac{\Gamma(T)}{T\Gamma(T)}\frac{N\Gamma(N)}{\Gamma(N)} = \frac{N}{T} \\ &= \frac{\chi_{s_da_d}^{s_{d+1}z_{d+1}}(H_d)}{\sum_{(s',z)}\chi_{s_da_d}^{s'z}(H_d)} \triangleq D_{\chi(H_d)}(s_{d+1},z_{d+1}|a_s,s_d)\end{aligned}$$

where we used the fact that $\Gamma(x+1) = x\Gamma(x)$ (DeGroot, 2004). Therefore $\tilde{P}^{\pi}(H_{d+1})$

$$= \tilde{P}^{\pi}(H_d)\pi(a_d|h_d)D_{\chi(H_d)}(s_{d+1},z_{d+1}|a_s,s_d). \quad (6)$$

which is identical to (2) except for the difference in between $\tilde{P}^{\pi}(H_d)$ and $P^{\pi}(H_d)$. This can be resolved by forward induction with base step: $\tilde{P}^{\pi}(H_0) = P^{\pi}(H_0)$, and the induction step, i.e., show $\tilde{P}^{\pi}(H_{d+1}) = P^{\pi}(H_{d+1})$ given $\tilde{P}^{\pi}(H_d) = P^{\pi}(H_d)$, directly following from (2) and (6). Therefore we can conclude that $\forall_d \quad \tilde{P}^{\pi}(H_d) = P^{\pi}(H_d)$. Combining this with (4) proves the result. □

Corollary 4. *Given suitably chosen exploration constant (e.g., $c > \frac{Rmax}{1-\gamma}$), BA-POMCP with root-sampling of dynamics function converges in probability to the expected transition solution.*

Proof. Since Theorem 3 guarantees the distributions over histories are the same in the limit, they will converge to the same values maintained in the tree. □

Finally, we see that these are solutions for the BA-POMDP:

Corollary 5. *BA-POMCP with expected transitions sampling, as well as with root sampling of dynamics function converge to an ϵ-optimal value function of a BA-POMDP: $V(\langle s,\chi\rangle,h) \xrightarrow{p} V^*_{\epsilon}(\langle s,\chi\rangle,h)$, where $\epsilon = \frac{precision}{1-\gamma}$.*

Proof. A BA-POMDP is a POMDP, so the analysis from Silver & Veness (2010) applies to the BA-POMDP, which means that the stated guarantees hold for BA-POMCP. The BA-POMDP is stated in terms of expected transitions, so the theoretical guarantees extend to the expected transition BA-POMCP, which in turn (4) implies that the theoretical guarantees extend to RS-BA-POMCP. □

Finally, we note that linking states does not affect they way that sampling is performed at all:

Proposition 6. *Linking states does not affect convergence of BA-POMCP.*

5. Empirical Evaluation

Experimental setup In this section, we evaluate our algorithms on a small toy problem—the classical Tiger problem (Cassandra et al., 1994)—and test scalability on a larger domain—a partially observable extension to the Sysadmin problem (Guestrin et al., 2003), called Partially Observable Sysadmin (POSysadmin). These domains will show performance on a very small problem and a more complex one.

In POSysadmin, the agent acts as a system administrator with the task of maintaining a network of n computers. Computers are either 'working' or 'failing', which can be deterministically resolved by 'rebooting' the computer. The agent does not know the state of any computer, but can 'ping' any individual computer. At each step, any of the computers can 'fail' with some probability f. This leads to a state space of size 2^n, an action space of $2n+1$, where the agent can 'ping' or 'reboot' any of the computers, or 'do nothing', and an observation space of 3 ($\{NULL, failing, working\}$). The 'ping' action has a cost of 1 associated with it, while rebooting a computer costs 20 and switches the computer to 'working'. Lastly, each 'failing' computer has a cost of 10 at each time step.

Table 1: Default experiment parameters

Parameter	Value
γ	0.95
$horizon$ (h)	20
# particles in belief	1000
f (computer fail %)	0.1
exploration const	$h * (\max(R) - \min(R))$
# episodes	100
λ = # updated counts	30

We conducted an empirical evaluation with aimed for 3 goals: The first goal attempts to support the claims made in Section 4 and show that the adaptations to BA-POMCP do not decrease the quality of the resulting policies. Second, we investigate the runtime of those modifications to demonstrate their contribution to the efficiency of BA-POMCP. The last part contains experiments that directly compare the performance per action selection time with the baseline approach of Ross et al. (2011). For brevity, Table 1 describes the default parameters for the following experiments. It will be explicitly mentioned whenever different values are used.

BA-POMCP variants Section 4 proves that the solutions of the proposed modifications (root-sampling (R-), expected models (E-) and linking states (L-)) in the limit converge to the solution of BA-POMCP. Here, we investigate the transient behaviour of these methods in practice. These experiments describe a scenario where the agent starts of with a noisy prior belief.

For the Tiger problem, the agent's initial belief over the transition model is correct (i.e., counts that correspond to the true probabilities with high confidence), but it provides an uncertain belief that underestimates the reliability of the observations. Specifically, it assigns 5 counts to hearing the correct observation and 3 counts to incorrect: the agent beliefs it will hear correctly with a probability of 62.5%. The experiment is run for with 100, 1000 & 1000 simulations and all combinations of BA-POMCP adaptations.

Figure 1 plots the average return over 10000 runs for a learning period of 100 episodes for Tiger. The key observation here is two-fold. First, all methods improve over

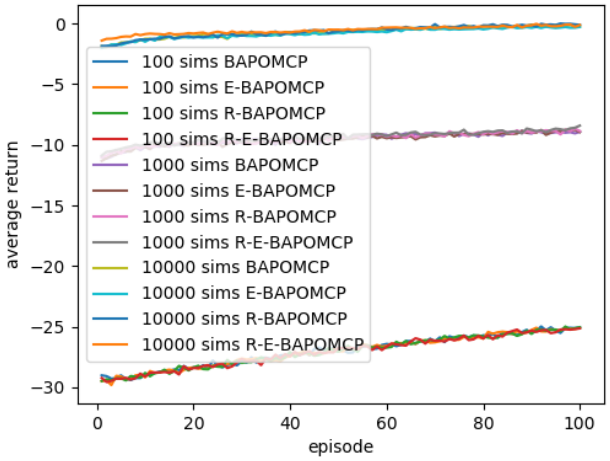

Figure 1: The average discounted return of BA-POMCP over 100 episodes on the Tiger problem for 100, 1000 & 10000 simulations

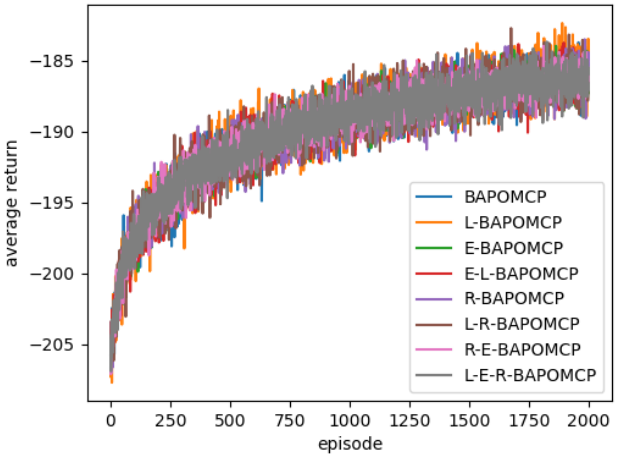

Figure 2: The average discounted of 100 simulations of BA-POMCP per time episodes on the Sysadmin problem

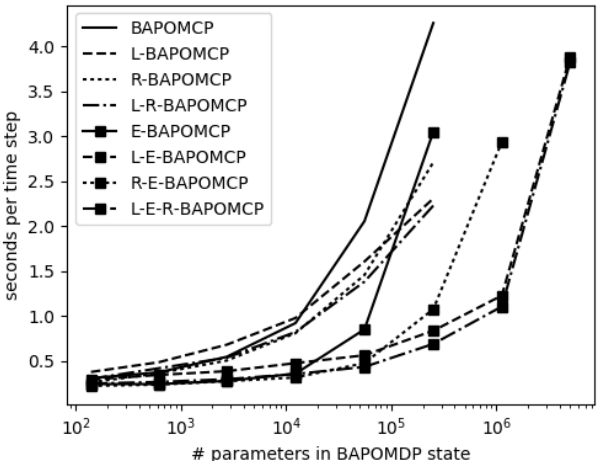

Figure 3: The average amount of seconds required for BA-POMCP given the the log of the amount of parameters (size) in the POSysadmin problem

time through refining their knowledge about D. Second, there are three distinct clusters of lines, each grouped by the number of simulations. This shows that all 3 variants (R/L/E-BA-POMCP) produce the same results and performance increases as the number of simulations increases.

We repeat this investigation with the (3-computer) POSysadmin problems, where we allow 100 simulations per time step. In this configuration, the network was fully connected with a failure probability $f = 0.1$. The (deterministic) observation function is assumed known a priori, but the prior over the transition function is noisy as follows: for each count c, we take the true probability of that transition (called p) and (randomly) either subtract or add .15. Note that we do not allow transitions with non-zero probability to fall below 0 by setting those counts to 0.001. Each Dirichlet distribution is then normalized the counts to sum to 20. With 3 computers, this results in $|S| \times |A| = 8 \times 7 = 56$ noisy Dirichlet distributions of $|S| = 8$ parameters.

Figure 2 shows how each method is able to increase its performance over time for POSysadmin. Again, the proposed modifications do not seem to negatively impact the solution quality.

BA-POMCP scalability While the previous experiments indicate that the three adaptations produce equally good policies, they do not support any of the efficiency claims made in Section 3. Here, we compare the scalability of BA-POMCP on the POSysadmin problem. The proposed BA-POMCP variants are repeatedly run for 100 episodes on instances of POSysadmin of increasing network size (3 to 10 computers), and we measure the average action selection time required for 1000 simulations. Note that the experiments are capped to allow up to 5 seconds per action selection, demonstrating the problem size that a specific method can perform 1000 simulations in under 5 seconds.

Figure 3 shows that BA-POMCP takes less than 0.5 seconds to perform 1000 simulations on an augmented state with approximately 150 parameters (3 computers), but is quickly unable to solve larger problems, as it requires more than 4 seconds to plan for a BA-POMDP with 200000 counts. BA-POMCP versions with a single adaptation are able to solve the same problems twice as fast, while combinations are able to solve much larger problems with up to 5 million parameters (10 computers). This implies not only that each individual adaptation is able to speed up BA-POMCP, but also that they complement one another.

Performance The previous experiments first show that the adaptations do not decrease the policy quality of BA-POMCP and second that the modified BA-POMCP methods improve scalability. Here we put those thoughts together and directly consider the performance relative to the action selection time. In these experiments we take the av-

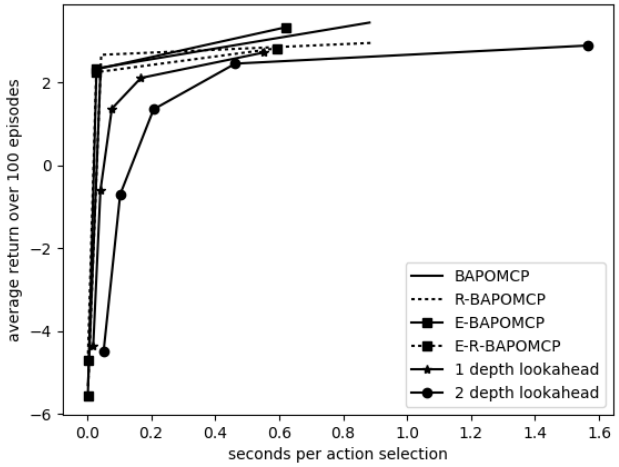

Figure 4: The average return over 100 episodes per action selection time of on the Tiger problem

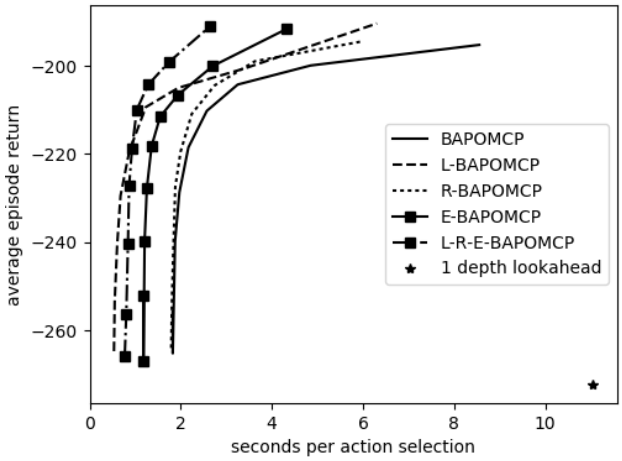

Figure 5: The average return over 100 episodes per action selection time of BA-POMCP on the POSysadmin problem

erage return over multiple repeats of 100 episodes and plot them according to the time required to reach such performance. Here BA-POMCP is also directly compared to the baseline lookahead planner by Ross et al. (2011).

First, we apply lookahead with depth 1&2 on the Tiger problem under the same circumstance as the first experiment for increasing number of particles (25, 50, 100, 200 & 500), which determines the runtime. The resulting average episode return is plotted against the action selection time in Figure 4.

The results show that most methods reach near optimal performance after 0.5 seconds action selection time. R-BA-POMCP and E-R-BA-POMCP perform worse than their counterparts BA-POMCP and E-BAPOMCP, which suggests that root sampling of the dynamics actually *slows down* BA-POMCP slightly. This phenomenon is due to the fact that the Tiger problem is so small, that the overhead of copying the augmented state and re-sampling of dynamics (during STEP function) that root sampling avoids is negligible and does overcome the additional complexity of root sampling. Also note that, even though the Tiger problem is so trivial that a lookahead of depth 1 suffices to solve the POMDP problem optimally, BA-POMCP still consistently outperforms this baseline.

The last experiment shows BA-POMCP and lookahead on the POSysadmin domain with 6 computers (which contains 55744 counts) with a failure rate of 0.05. The agent was provided with an accurate belief χ.[3] The results are shown in Figure 5.

We were unable to get lookahead search to solve this problem: the single instance which returned results in a reasonable amount of time (the single dot in the lower right corner) was with a lookahead depth of 1 (which is insufficient for this domain) with just 50 particles. BA-POMCP, however, was able to perform up to 4096 simulations within 5 seconds and reach an average return of approximately -198, utilizing a belief of 1000 particles. The best performing method, L-R-E-BA-POMCP requires less than 2 seconds for similar results, and is able to reach approximately -190 in less than 3 seconds. Finally, we see that each of the individual modifications outperform the original BA-POMCP, where Expected models seems to be the biggest contributor.

6. Conclusion

This paper provides a scalable framework for learning in Bayes-Adaptive POMDPs. BA-POMDPs give a principled way of balancing exploration and exploiting in RL for POMDPs, but previous solution methods have not scaled to non-trivial domains. We extended the Monte Carlo Tree Search method POMCP to BA-POMDPs and described three modifications—Root Sampling, Linking States and Expected Dynamics models— to take advantage of BA-POMDP structure. We proved convergence of the techniques and demonstrated that our methods can generate high-quality solutions on significantly larger problems than previous methods in the literature.

Acknowledgements

Research supported by NSF grant #1664923 and NWO Innovational Research Incentives Scheme Veni #639.021.336.

[3] We do not use the same prior as in the first BA-POMCP variants experiments since this gives uninformative results due to the fact that solution methods convergence to the optimal policy with respect to the (noisy) belief, which is different from the one with respect to the true model.

References

Amato, Christopher and Oliehoek, Frans A. Scalable planning and learning for multiagent POMDPs. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pp. 1995–2002, January 2015.

Auer, Peter, Cesa-Bianchi, Nicolo, and Fischer, Paul. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Cassandra, Anthony R, Kaelbling, Leslie Pack, and Littman, Michael L. Acting optimally in partially observable stochastic domains. In *AAAI*, volume 94, pp. 1023–1028, 1994.

DeGroot, Morris H. *Optimal Statistical Decisions*. Wiley-Interscience, 2004.

Duff, Michael O'Gordon. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts Amherst, 2002.

Engel, Yaakov, Mannor, Shie, and Meir, Ron. Reinforcement learning with kernels and Gaussian processes. In *Proceedings of the ICML'05 Workshop on Rich Representations for Reinforcement Learning*, pp. 16–20. Citeseer, 2005.

Guestrin, Carlos, Koller, Daphne, Parr, Ronald, and Venkataraman, Shobha. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, pp. 399–468, 2003.

Guez, Arthur, Silver, David, and Dayan, Peter. Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, pp. 1025–1033, 2012.

Kaelbling, Leslie Pack, Littman, Michael L, and Cassandra, Anthony R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1): 99–134, 1998.

Poupart, Pascal and Vlassis, Nikos A. Model-based Bayesian reinforcement learning in partially observable domains. In *ISAIM*, 2008.

Ross, Stephane, Chaib-draa, Brahim, and Pineau, Joelle. Bayes-Adaptive POMDPs. In *Advances in Neural Information Processing Systems*, pp. 1225–1232, 2007.

Ross, Stéphane, Pineau, Joelle, Paquet, Sébastien, and Chaib-Draa, Brahim. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

Ross, Stéphane, Pineau, Joelle, Chaib-draa, Brahim, and Kreitmann, Pierre. A Bayesian approach for learning and planning in partially observable Markov decision processes. *The Journal of Machine Learning Research*, 12: 1729–1770, 2011.

Shani, Guy, Pineau, Joelle, and Kaplow, Robert. A survey of point-based POMDP solvers. *Autonomous Agents and Multi-Agent Systems*, pp. 1–51, 2012.

Silver, David and Veness, Joel. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems*, pp. 2164–2172, 2010.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Thrun, Sebastian. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems*, volume 12, pp. 1064–1070, 1999.

Vlassis, Nikos, Ghavamzadeh, Mohammad, Mannor, Shie, and Poupart, Pascal. Bayesian reinforcement learning. In *Reinforcement Learning*, pp. 359–386. Springer, 2012.

Wyatt, Jeremy L. Exploration control in reinforcement learning using optimistic model selection. In *ICML*, pp. 593–600, 2001.