Efficient Exploitation of Factored Domains in Bayesian Reinforcement Learning for POMDPs

Sammie Katt
Northeastern University
katt.s@husky.neu.edu

Frans A. Oliehoek
University of Liverpool, and
Delft University of Technology
f.a.oliehoek@tudelft.nl

Christopher Amato
Northeastern University
camato@ccs.neu.edu

ABSTRACT

While the POMDP has proven to be a powerful framework to model and solve partially observable stochastic problems, it assumes accurate and complete knowledge of the environment. When such information is not available, as is the case in many real world applications, one must learn such a model. The BA-POMDP considers the model as part of the hidden state and explicitly considers the uncertainty over it, and as a result transforms the learning problem into a planning problem. This model, however, grows exponentially with the underlying POMDP size, and becomes intractable for non-trivial problems. In this article we propose a factored framework, the FBA-POMDP that represents the model as a Bayes-Net, drastically decreasing the number of parameters required to describe the dynamics of the environment. We demonstrate that the our approach allows solvers to tackle problems much larger than possible in the BA-POMDP.

1 INTRODUCTION

Robust decision-making agents in any non-trivial system must reason over uncertainty in various dimensions, including the current state, the outcome of possible actions, and the dynamics of the environment. The outcome and state uncertainty are elegantly captured by POMDPs [6], which enable planning in stochastic and partially observable environments.

The POMDP, however, assumes direct access to the system dynamics, which unfortunately are often not easily available. When such a model is not available, the problem turns into a Reinforcement Learning challenge, where one must consider both the potential benefit of learning (exploration) and exploiting current knowledge. To answer the exploration-exploitation trade-off problem in a principled way, model-based Bayesian RL maintains a posterior distribution over the unknown model parameters, encouraging explicit reasoning over the uncertainty over them. Recently these models have been extended to partially observable models [9], but fail to scale to bigger domains unless some structural assumptions are made a-priori. Other work has considered a factored representation of the state space, effectively reducing the number of unknown parameters [8]. Their formulation, however, does not accommodate for environments with partially hidden states or where the perception of the state is noisy.

In this work we propose a Bayesian approach towards learning a factored representation of the dynamics of a system, called the Factored Bayes-Adaptive POMDP, which allows for simultaneous planning and learning in a partially observable stochastic environment with unknown dynamics. This method utilizes Bayes-Nets to describe dynamics, which exploits conditionally independence relationships, effectively decreasing the number of unknown parameters to learn. Additionally we describe a solution method based on a Monte-Carlo Tree Search solution method and a mechanism for maintaining a belief in the FBA-POMDP, and demonstrate through experiments that this approach is able to learn and solve problems that previous work is unable to tackle.

2 BACKGROUND

Here we first provide a summary of the preliminary literature. Section 2.1 describes the POMDP and BA-POMDP framework, followed by a review on Bayesian Networks in Section 2.3.

2.1 The POMDP & BA-POMDP

The POMDP [6] is a general model for decision-making in stochastic and partially observable domains, with execution unfolding over (discrete) time steps. At each step in a POMDP, the agent selects an action that triggers a state transition in the system, which generates some reward and observation. The observation is perceived by the agent and the next time step commences. Formally, a POMDP is described by the tuple $(S, A, Z, D, R, \gamma, h)$, 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] 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. In this description of the POMDP, D captures the probability of transitioning from state s to the next state s' and generating observation z in the process (for each action a).

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 BA-POMDP [9] $(S', A, Z, D', R, \gamma, h)$ is yet another, larger, POMDP that characterizes the problem of optimal sequential decision-making in the original underlying POMDP with uncertainty over

[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.

the dynamics D in the form of Dirichlet distributions.[2] Conceptually, if one could observe both states and observations, then it is possible to maintain a vector χ with the counts of the occurrences for all $\langle s, a, s', z\rangle$ tuples, where 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 the using regular POMDP formalism. That is, the count vector is included as part of the hidden state of the POMDP. While the observation and action space remain unchanged, the state (space) of the BA-POMDP now includes the counts: $\bar{s} = \langle s, \chi\rangle$. The dynamics of the BA-POMDP, $\bar{D} = P(s', \chi, z|s, \chi, a)$, factorize to $P(s', z|s, \chi, a)P(\chi'|s, \chi, a, s', z)$, where $P(s', z|s, \chi, a)$ corresponds to the expectation of $D_{sa}^{s'z}$ according to χ:

$$P(s', z|s, \chi, a) = P_\chi(s', z|s, a) = \mathbf{E}[\chi_{sa}^{s'z}] = \frac{\chi_{sa}^{s'z}}{\sum_{s'z} \chi_{sa}^{s'z}} \tag{1}$$

Then, 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 is 1), and if we let $\mathbb{I}_{a,b}$ denote the Kronecker delta that indicates (is 1 when) $a = b$, we can write $P(\chi|s, \chi, a, s', z)$ as $\mathbb{I}_{\chi', \chi+\delta_{sa}^{s'z}}$.

2.2 Solving BA-POMDPs

For an agent to make rational and informed decisions it can base its actions only on the action-observation history. Because an interaction with the environment can be of arbitrarily length, it is infeasible to compute the optimal action for any possible history. Alternatively, the agent may keep track of a probability distribution over the current state, also called the *belief* b. Given an initial belief b_0, and some sort of *belief update* $\tau : (b, a, z) \rightarrow b'$, the agent solution to the problem is a *policy* that maps any belief to an action $\pi : b \rightarrow a$. The next paragraph describes how one could go about solving a BA-POMDP, assuming some belief tracking mechanism is given. After which we discuss the most common belief update schemes for the BA-POMDP.

While the BA-POMDP is a POMDP, typical POMDP solvers are unable to deal with the (uncountably) large size of this model. In particular offline planners, which attempt to solve the problem for each possible belief or history sequence, struggle with the state space. Fortunately, the complexity of Monte-Carlo Tree Search[2, 3] (MCTS) is independent of the state space and thus has proven to be a reliable approach in the form of the BA-POMCP solution method [7]. BA-POMCP estimates the expected value of each possible action at each time step in an online fashion. It does so by building up a look-ahead tree from *simulated* interactions with environment. Each such simulation first traverses through the action-observation branches in the current tree, and ends with extending the tree with a set of nodes once it reaches a leaf. At that point it evaluates the leaf, which is typically done via a *roll-out* with the environment until a terminal state or some maximum depth has been reached.

A simulation starts with sampling a state from the current belief and traverses the tree by picking actions using UCB [1] and sampling state-observation pairs using the counts as described in Algorithm 1. At the end of each simulation the accumulated return is propagated back up into the tree and used to update the statistics at each node: the number of times the node has been visited and the average (discounted) return. The algorithm picks the action at the root of the tree that has accumulated the highest average return, after the action selection time has terminated or a fixed number of simulations have ran.

Algorithm 1 SIMULATE($\bar{s}, d, h$)

if ISTERMINAL(h) || d equals max_depth **then**
  return 0
end if
//Action selection uses statistics stored at node h:
$a \leftarrow$ UCBACTIONSELECTION(h)
$R \leftarrow R(\bar{s}, a)$
$\bar{s}', z \leftarrow$ STEP($\bar{s}, a$)
$h' \leftarrow (h, a, z)$
if $h' \in Tree$ **then**
  $r \leftarrow R + \gamma$ SIMULATE($\bar{s}', d + 1, h'$)
else
  CONSTRUCTNODE(h')
  $r \leftarrow R + \gamma$ ROLLOUT($\bar{s}', d + 1, h'$)
end if
//Update statistics:
$N(h, a) \leftarrow N(h, a) + 1$
$Q(h, a) \leftarrow \frac{N(h,a)-1}{N(h,a)} Q(h, a) + \frac{1}{N(h,a)} r$
return r

Note that, in BA-POMCP, the state sampled at the start of a simulation $\bar{s}$ consist of both a domain state s and a set of counts χ. The counts are key to the simulations, as they provide a model for the interactions with the environment (algorithm 2).

Algorithm 2 STEP($\bar{s} = \langle s, \chi\rangle, a$) (for BA-POMCP)

$D_{sa} \leftarrow \mathbf{E}[\chi_{sa}]$
$\langle s', z\rangle \sim D_{sa}$
$\chi_{sa}^{s'z} \leftarrow \chi_{sa}^{s'z} + 1$
$s \leftarrow s'$
return $\bar{s}, z$

Belief tracking: For finite state spaces, one can compute the exact belief naively by iterating over all the possible new states using the model's dynamics. This approach is expensive and practical only in very small environments. As a feasible alternative, it is common to use *particle filters* [10] as an approximation method instead. The particles may be associated with some weight w, which represents the probability of that state: $\frac{w}{\sum_{i=1}^{K} w_i}$. There are numerous methods to update the particle filter when observing z after executing action a, such as *Rejection Sampling* and *Importance Sampling*. Rejection Sampling in general consist of sampling a value from some process, and then rejects or accepts it according to some condition. Here the process consists of simulating an interaction with the environment, where the condition is whether the generated observation matches that of the environment. More specifically, we first sample a particle from our particle filter, followed by a simulated step in the model, where a new state $\bar{s}'$ and observation z_{sim} is generated together with an update particle. If that observation equals the actual observation perceived from the environment, then we add $\bar{s}'$ to our

[2] [9] follows the standard T & O POMDP representation, we use the combined D formalism for brevity.

new particle filter. If not, then we *reject* (ignore) that particle. This operation continues until the number of particles in the new belief equals K (algorithm 3).

Algorithm 3 REJECTION SAMPLING(K, b, a, z)

$b' \leftarrow \{\}$
while $size(b')$ is not K **do**
  $\bar{s} \sim b$
  $\langle z_{sim}, \bar{s}' \rangle \sim D(\bar{s}, a)$
  if z_{sim} equals z **then**
    add $\bar{s}'$ to b'
  end if
end while
return b'

An alternative popular approach is *Importance Sampling*, which assigns *weights* to each particle, representing its probability within the distribution. There are multiple ways of updating the belief with this method, of which the one described by algorithm 4 is used in this paper: all particles go through a simulation step, during which we calculate the probability that this particle would have generated the perceived observation, and assign that probability as weight. This process concludes with a re-sampling step, which samples K particles from the new posterior to generate a 'flat' filter, where all K particles represent the same $\frac{1}{K}$ weight.

Algorithm 4 IMPORTANCE SAMPLING(K, b, a, z)

$b' \leftarrow \{\}$
for $i \in 1 \ldots K$ **do**
  $\bar{s} \sim b$
  $\langle z_{ignored}, \bar{s}' \rangle \sim D(\bar{s}, a)$
  $w \leftarrow O(z|a, \bar{s}')$
  add $\langle \bar{s}', w \rangle$ to b'
end for
return $resample(b')$

2.3 Bayes-Networks

Bayes-Networks (BNs) are graphical models that define joint distributions over n variables $X=\{x_0, \ldots, x_n\}$ according to some (non-cyclic) structure (or topology) G, which induces the conditional dependencies. The probability that a variable x_i takes on some value v is specified by the *Conditional Probability Tables* (CPTs)

$$P_{G,\theta}(x_i{=}v|PV_G(x_i){=}E_i) = \theta_G^{i,v|E_i} \tag{2}$$

and depends on the value of the parents $PV_G(x_i) = E_i$. The joint probably of some set of variables X is then described by

$$P_{G,\theta_G}(X|PV_G(X)) = \prod_{i \in 1 \ldots n} P_{G,\theta}(x_i|PV_G(x_i)) \tag{3}$$

Example: To illustrate the advantage of a compact representation of a process, consider figure 1 which shows an example of 2 BNs that describe different structures for an imaginary transition function between states with 3 features $\{X_1, X_2 \,\&\, X_3\}$. Features in highly structured problems have few transition conditions, which result in smaller BNs (fewer connections) and thus fewer parameters. The graph (*left*) in figure 1, for example, contains $|X_1| \times |X_2| \times |X_3|$ parameters to describe the conditional probabilities for node X_2 in $t+1$, while (*right*) graph 1 only requires $|X_1| \times |X_2|$ parameters.

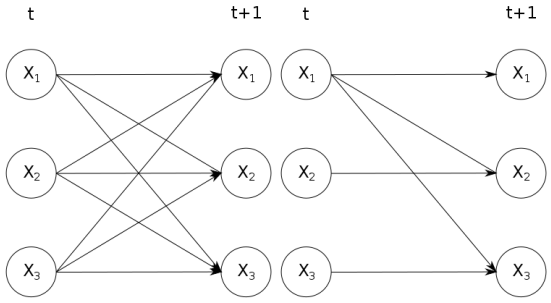

Figure 1: Two Bayesian Networks. The arrow $x_i \leftarrow x_j$ indicates that x_j depends on x_i. *Right* contains fewer parameters compared to *Left*

Learning BNs: For our purposes we are interested in Bayesian approaches for learning BNs given some data D: $P(G, \phi_G|D)$. This is typically specified as the joint $P(G|D)P(\theta_G|G, D)$. We make the standard assumption that the prior $P(\theta_G|G)$ is factorized into a product of Dirichlets: $P(\theta_G^{i|E}|G) = Dirichlet(\phi_G^{i|E})$. $P(\theta_G|G, D)$ is then easily computed by incrementing $\phi_G^{i,v|E}$ for each instance of $\{X_i{=}\, v|PV_G(X_i){=}E\}$ in D. The posterior distribution over the topology $P(G|D)$ can rarely be computed in closed form, as there are $O(n!2^{\binom{n}{2}})$ [8] possible structures. In this work we consider the Metropolis-Hastings algorithm as an approximation method [4]. Given a proposal distribution $q(G'|G)$ from which we can sample moves from one graph G to another G', a prior distribution over all graphs $P(G)$ and a scoring metric for how well a graph explains the data $P(D|G)$. Then Metropolis-Hastings specifies that we can approximate $P(G|D)$ by sampling graphs G' with probability $min\{1, \frac{P(D|G')P(G')q(G|G')}{P(D|G)P(G)q(G'|G)}\}$. To simplify this computation, one typically assumes a uniform prior $P(G)$ and a symmetric proposal distribution $q(G'|G)$ that consists of either adding or removing an edge in G. Lastly, $P(D|G)$ can be computed in closed form with the likelihood-equivalence Bayesian Dirichlet score metric (BDe) [5].

3 BAYESIAN RL IN FACTORED POMDPS

The BA-POMDP allows the agent to do informed decision making despite the challenges it faces due to uncertainty. The number of parameters grow quadratic in the state space: $O(|S|^2|A||\Omega|)$, of which only 1 is updated after a new observation. We argue that many applications allow for generalization through conditional independence in the dynamics.

To illustrate this, consider a vacuum robot in a two-dimensional grid, tasked with cleaning dirty cells. Whenever the agent attempts to move east, its new location is the current location plus one step to the right (assuming the move is always successful). This is the case, regardless of the y-position and is also independent of which cells are dirty. In the regular BA-POMDP belief updates, however,

such transition would only affect the parameters concerned with that particular state, where the agent is on that specific y-position, and certain cells are dirty. If the agent ever returns to a *similar* state, but with a different set of dirty cells, it will not be able leverage the knowledge it could have, had it known that a new location (when going east) is only dependent on the current x-position.

The next section (3.1) introduces a novel model, the FBA-POMDP, to exploit structure in Bayesian model-based Reinforcement Learning. In section 3.2 we propose an extension of the BA-POMCP method adapted to the factorized model, and 3.3 describes a belief tracking methods specifically designed for FBA-POMDP.

3.1 The Factored BA-POMDP

First let us describe the state and observation space (S, Ω) in terms of *features* $x = \{x_1 \dots x_n\}$, $y = \{y_1 \dots y_m\}$, for n state and m observation features. We then describe D, of the underlying POMDP, in terms of a collection of BNs, one for each action a, and their parameters, $\langle G, \theta_G \rangle$ (figure 2 describes one such graph G^a).

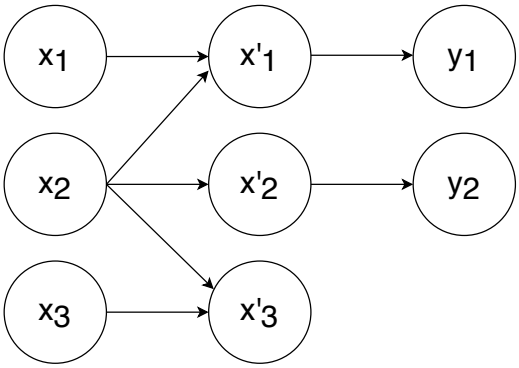

Figure 2: A possible instantiation of a dynamics model D in a factored POMDP for a domain with 3 state features and 2 observation features

In case the structure G is known a-priori, it is easy to see that we could define a Bayes-Adaptive model with counts χ_G as belief over θ_G. The (important) difference with the BA-POMDP here is that χ_G has fewer parameters and thus the belief would converge quicker to the true model. However, this assumption is unrealistic, and we must consider the structure as part of the hidden state space too. As a result, the state space consists of the domain (factored) state space, the space of possible graph structures, and the possible count assignments: $\bar{s} = \langle x, G, \chi_G \rangle$. Now let us formulate the dynamics $\bar{D}$ of this model using the standard Bayes-rule:

$$\bar{D}(x, G, \chi_G, a, x', G', \chi'_{G'}, y) = \tag{4}$$

$$P(x', G', \chi'_{G'}, y | x, G, \chi_G, a) = P(x', y | x, G, \chi_G, a) \times \tag{5}$$

$$P(G' | x, G, \chi_G, a, x', y) \times \tag{6}$$

$$P(\chi'_{G'} | x, G, \chi_G, a, x', G', y) \tag{7}$$

$P(x', y| \dots)$ (5) corresponds to the expectation of the joint distribution $P_{\chi_G^a}(x', y|x)$, denoted as $D_{\chi^G}(x, a, x', y)$, and factorizes as a BN according to structure G:

$$D_{\chi^G}(x, a, x', y) = \prod_{n \in x', y} \hat{\theta}_{G^a}^{n|PV(n)}$$

where $\hat{\theta}$ are the expected CPT parameters $\mathbf{E}_{\chi_{G^a}}[\theta]$.

Then, if we define $\delta_G^{xax'y}$ as a set of zero counts except on the locations corresponding to the nodes $x'_1 \dots x'_n$ and $y_1 \dots y_m$ and their associated parent values, where they are 1. Then we define the count update equation $P(\chi'_{G'}| \dots)$ (7) as $\mathbb{I}_{\chi'_G, \chi_G + \delta_G^{xax'y}}$. Lastly, it is important to note that we assume that the topology of G is static, thus $P(G'| \dots)$ (6) is $\mathbb{I}_{G',G}$. Putting this together, the complete formal definition of the FBA-POMDP is:

- $\bar{A} = A$, $\bar{R} = R$, $\bar{\gamma} = \gamma$, $\bar{h} = h$: Identical to POMDP definition
- $\bar{\Omega}$: Y. Set of possible observations defined by their features.
- $\bar{S}$: $S \times G^A \times \chi_{G^A}$. The cross product of the domain's factored state space and the set of possible models in the form of BNs per action, where χ_{G^A} represent the counts to describe the probability over the conditional probabilities in G^A in the form of Dirichlet Distributions.
- $\bar{D}$: $P(x', G', \chi'_{G'}, y | x, G, \chi_G, a) =$

$$D_{\chi_G}(x, a, x', y) \mathbb{I}_{\chi'_G, \chi_G + \delta_G^{xax'y}} \mathbb{I}_{G',G}$$

3.2 Monte-Carlo Tree Search for FBA-POMDPs

Solving the FBA-POMDP faces similar challenges as BA-POMDP solvers do with respect to uncountable large (hyper-) state spaces and uncertainty over current state and the dynamics. So it is only natural to look at BA-POMDP solution methods for inspiration, and to extend Monte-Carlo Tree Search to the FBA-POMDP in a similar spirit as was done to design BA-POMCP. Recall that these methods could not directly be applied to the Bayes-Adaptive setting, as they assume access to the underlying dynamics. This was circumvented in BA-POMCP by sampling and utilizing the counts χ at each simulation as a model. Instead, in the factored setting, we sample a $\langle G, \chi_G, s \rangle$ tuple at the start of each simulation, and use those throughout to simulate steps. This is best illustrated in algorithm 5, which replaces the BA-POMCP step of algorithm 2 in the main loop (algorithm 1).

Algorithm 5 STEP ($\bar{s} = \langle G, \chi_G, x \rangle, a$) (for FBA-POMDP)

$D^{xa} \leftarrow \prod_n \mathbf{E}[\chi_{G^a}^{n|PV(n)}]$
$\langle x', z \rangle \sim D_{xa}$
//Increment count of each node - parent combination
for each node $n \in \chi_{G^a}$ and its value $v \in x$ **do**
  $\chi_{G^a}^{n,v|PV(n)} \leftarrow \chi_{G^a}^{n,v|PV(n)} + 1$
end for
$x \leftarrow x'$
return $\bar{s}$, z

Note that the others techniques that can be incorporated into BA-POMCP, such as *root sampling*, *linking states* and sampling directly from the expected model of χ, are equally applicable to FBA-POMCP.

3.3 Structure rejuvenation

The particle filter approach towards maintaining the belief as described in section 2.2 can be naturally extended to the FBA-POMDP case: each particle consists of a domain state, graph topology and corresponding counts $\langle s, G, \chi_G \rangle$, and the simulation step in line 4 and 4 in respectively algorithm 3 and 4 is sampled according to $\tilde{D}$ of the FBA-POMDP definition.

However, these methods only update the counts and are not particular well designed to represent or update the probability over the graph topology. We propose to address this issue through (particle) reinvigoration by means of the Metropolis-Hastings algorithm as described in section 2.3. We sample graphs $\langle G, \chi_G \rangle$ from our current belief and compare their BD score to a proposed modification $\langle G', \chi_{G'} \rangle$, given the prior $\langle G, \chi^0_G \rangle$ and $\langle G', \chi^0_{G'} \rangle$. Once K particles have been accepted we have rejuvenated our belief with new structures consistent with our history.

We propose to compute $\chi_{G'}$ by sampling a complete history $H = \langle x, a, x', y \rangle$ trajectory that is consistent with h, similarly to how the counts in the particles of the current belief have been accumulated through Rejection Sampling. This operation is expensive and linear in the size of the history, and should be used only when necessary. We use the overall likelihood L of the current belief to decide when to apply Metropolis-Hastings. L happens to be calculated directly in the form of the accumulated weight during Importance Sampling in line 5.

4 EXPERIMENTS

The previous section introduced a factored Bayes-Adaptive approach towards learning in POMDPs in combination with a specifically design solution and belief tracking method. Here we provide empirical support for this approach on an extension of the well-known Tiger problem [6].

4.1 Setup

Parameters: The parameters of FBA-POMCP and the belief update methods are consistent across the different approaches we compare. In particular, the Monte-Carlo Tree Search solution methods performs 4096 simulations and use a random policy during the roll-outs. The particle filters contain 1024 particles, and we have applied basic Importance Sampling whenever 'MH' is not explicitly mentioned as belief update algorithm. We resample particles using Metropolis-Hastings whenever the log likelihood falls below -10. In our experiments we compare the average return over 400 episodes, plotting the performance against time. The expected average return of a single episode with these parameters, given a perfect model, is approximately 3.1.

Factored Tiger Problem: The Tiger domain, arguably the smallest possible POMDP, describes a scenario where the agent is faced with the task of opening one out of two doors. Behind one door lurks a tiger, a danger and reward of -100 that must be avoided, while the other door opens up to a bag of gold for a reward of 10. The agent can choose to open either doors (which terminates the episode) or to *listen* for a noisy *observation*, for a cost of 1. This observation informs the agent of the location of the tiger with 85% accuracy. We propose to extend this domain with additional, uninformative, binary state features and call it the Factored Tiger problem. A state in the Factored Tiger problem with f extra attributes contains $1 + f$ features: $s = \{tiger{-}location, x_2, \ldots, x_{f+1}\}$. These state features have no effect on the observations, and do not change over time. The observation space consist of a single binary feature $y = \{tiger{-}location\}$ that indicates behind which door the tiger is heard. As a result the state space S increases by a factor 2 with each additional feature, while the complexity of the underlying dynamics remains unaffected. The agent, however, does not necessarily know this, as it is unfamiliar with the exact observation probabilities. Note that for f binary state features (of which only the location is informative), the agent must maintain a belief over $2 * 2^f$ counts: a count for each probability $p(o|s', a{=}listen)$, where s' grows 2^f. Here we consider the case of $f = 7$ uninformative binary state features.

We assume the reward and state transition model are known, but the parameters of the observation model are not. The agent's initial belief over those parameters are captured with the counts $c = \{c_t{=}5, c_f{=}3\}$, where c_t is the count for hearing the tiger's location correctly and count c_f is assigned to hearing the tiger behind the other door. This describes an expected accuracy of %62.5 (as done in [7, 9]). We consider both a prior where the topology of the observation function is known (P_k) and unknown (P_u), where it assigns a uniform probability distribution over the assignment of parents for the observation node. Note that graphs G^{listen}, where the observation does not depend on the tiger's location, cannot represent c; the counts for particles with such topologies in the agent's initial belief is uniform ($c_{unif} = \{4, 4\}$). As a result, approximately half of that prior assigns uninformative counts to the observation function.

We run a total of 4 solution methods on the Factored Tiger problem with 7 extra uninformative binary features (figure 3). The *bapomdp* line corresponds to the baseline 'BA-POMCP' on the (flat) BA-POMDP model, whereas 'known structure' and 'unknown structure' describe the performance on the FBA-POMDP model with respectively prior knowledge of the structure P_k and a uniform distribution P_u. Lastly, we compare this to our implementation labelled 'MH', which starts with the uniform topology prior P_u.

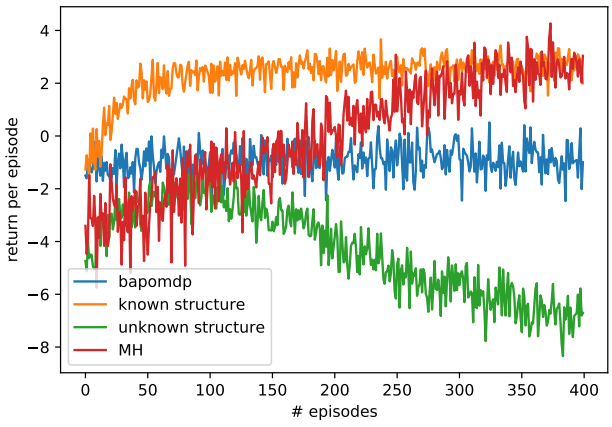

Figure 3: Average return on the Tiger problem

4.2 Analysis

Effect of knowing the structure: We first observe that the 'known structure' and '*bapomdp*' variant perform better at t=0. This verifies the earlier statement that a prior where half of the particles do not contain c_{prior} ($\langle 5, 3 \rangle$) leads to poorer performance.

Correct structure allows for efficient learning: Secondly, the data unsurprisingly shows that the highest performing approach is the 'known structure', which is a direct result of the compactness of the BNs in the belief. This shows that if you happen to know the structure of the domain a-priori, then that allows the number of unknown parameters to be as low as possible, leading to more efficient learning.

Resampling BNs is crucial: Section 3.3 mentioned that traditional methods of updating beliefs is not particularly effective when the prior describes uncertainty over the topology. In particular, for this problem it is important that the belief converges to a distribution that places a high probability on graphs where the edge connecting the location of the tiger to the observation feature. The 'unknown structure' variant (in green) shows that the performance (of the agent that only uses Importance Sampling in a factored setting) declines after 75 episodes. This is because, on rare occasions, the belief converges to a uniform (uninformative) observation model, which discouraged the agent to listen, and resulted in a policy that opened doors randomly (with an expected return of −45).

Factorization is important: Lastly we show that it is important to utilize factorization when solving and learning in larger POMDPs. The BA-POMDP (blue) is unable to learn in a problem as (arguably) simple as the Tiger problem, when faced with a larger state space. Our method, which combines a factored representation and structure resampling, is able to solve the problem in approximately 325 episodes, despite the number of unknown parameters.

5 CONCLUSION

This paper addresses the void of frameworks in the field of Bayesian Reinforcement Learning intersecting factored models and partially observable domains. Our approach describes the dynamics of the POMDP in terms of graphical models, and maintains a joint belief over the state, and both the CPTs and the structure simultaneously. Alongside the framework we introduce a solution method, which consists of an extension to the Monte-Carlo Tree Search method for FBA-POMDPs: FBA-POMCP, in addition to Metropolis-Hastings belief tracking algorithm. These methods were tested on the Factored Tiger, an extension of the traditional Tiger problem that can scale arbitrarily in size. The results show the significance of representing and recognizing independent features, as FBA-POMDP agents are able to learn in scenarios where the BA-POMDP is not feasible.

REFERENCES

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

[2] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[3] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. 2008. Monte-Carlo Tree Search: A New Framework for Game AI.. In *AIIDE*.

[4] Nir Friedman and Daphne Koller. 2003. Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine learning* 50, 1-2 (2003), 95–125.

[5] David Heckerman, Dan Geiger, and David M Chickering. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine learning* 20, 3 (1995), 197–243.

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

[7] Sammie Katt, Frans A Oliehoek, and Christopher Amato. 2017. Learning in POMDPs with Monte Carlo Tree Search. In *International Conference on Machine Learning*. 1819–1827.

[8] Stéphane Ross and Joelle Pineau. 2008. Model-based Bayesian reinforcement learning in large structured domains. *arXiv preprint arXiv:1206.3281* (2008).

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

[10] Sebastian Thrun. 1999. Monte Carlo POMDPs. In *NIPS*, Vol. 12. 1064–1070.