# Planning with Hidden Parameter Polynomial MDPs

**Clarissa Costen, Marc Rigter, Bruno Lacerda, Nick Hawes**

Oxford Robotics Institute, University of Oxford
clarissa, mrigter, bruno, nickh@robots.ox.ac.uk

## Abstract

For many applications of Markov Decision Processes (MDPs), the transition function cannot be specified exactly. Bayes-Adaptive MDPs (BAMDPs) extend Markov Decision Processes (MDPs) to consider transition probabilities governed by latent parameters. To act optimally in BAMDPs, one must maintain a belief distribution over the latent parameters. Typically, this distribution is described by a set of sample MDPs and associated weights which represent the likelihood of a sample MDP being the true underlying MDP. However, this method is sensitive to the size of the set of MDPs. As the latent parameter space's dimension increases, the number of sample MDPs required to sufficiently cover the space of possible worlds increases exponentially. Thus, we propose an alternative approach for maintaining the belief over the latent parameters. In particular, we consider BAMDPs where the transition probabilities can be expressed in closed form as a polynomial of the latent parameters. We outline a method to maintain a closed-form belief distribution for the latent parameters which results in an accurate belief representation. Furthermore, the closed-form representation does away with the need to tune the number of sample MDPs required to represent the belief. We evaluate two domains and empirically show that the polynomial, closed-form, belief representation results in better plans than a sampling-based belief representation.

## Introduction

Markov Decision Processes (MDPs) (Puterman 1994) are a common framework for sequential decision-making under uncertainty. In this paper, we consider an agent acting in an environment where the parameters governing the underlying MDP model are unknown. In such environments, the agent must reason over the uncertainty in the parameters, and balance exploration of the environment with gathering reward. An example of this is a robot planning over a domain with unknown weather conditions. As the robot interacts with the environment, the observations made by the robot can be used by it to plan for future steps. Bayes-Adaptive MDPs (BAMDPs) (Duff 2002) are used to model such problems in a way that allows for optimally trading off exploration and exploitation.

In BAMDPs, latent parameters are used to represent unknown dynamics that influence the transition probabilities. For problems where there is no prior knowledge of the transitions, the latent parameters are used to represent the transitions between states. Therefore, in a BAMDP, the state space of the original MDP is augmented by a posterior distribution which represents the belief over the true value of the latent parameters. As transitions are observed, the posterior distribution over the latent parameters is refined. Solving the BAMDP optimally results in the optimal behaviour in expectation under the initial belief. By reasoning about the information currently known about the MDP in the form of the posterior, BAMDPs provide an elegant and optimal solution to the exploration-exploitation tradeoff. One challenge in developing algorithms for BAMDPs is the need to keep track of the posterior distribution over the latent parameters. There are several methods to do so, depending on the class of BAMDP needed to be solved.

In a BAMDP where every transition probability is independent, each transition probability is a latent parameter. The posterior distribution over the latent parameters is represented by a Dirichlet distribution, which allows us to maintain an exact closed-form representation of the posterior distribution. However, for many problems, the transitions cannot be assumed to be independent of each other, and there is no prior information about the transitions. In BAMDPs where the latent parameters are used to describe global dependencies between transitions, the posterior distribution can be approximated by a set of *particles*, i.e. a set of sampled MDPs (Guez et al. 2014). This method relies on the MDP samples providing adequate coverage of the space of possible parameters. As the number of latent parameters increases, so does the dimensionality of the belief distribution over the parameters. Thus, the number of samples must increase exponentially to represent the belief accurately.

In this paper, we present a new class of BAMDPs, hidden parameter polynomial MDPs (HP$^2$-MDP). In a HP$^2$-MDP, the transition probabilities are defined by polynomial functions of the latent parameters. MDPs where the transitions are described by polynomial functions are traditionally used in probabilistic model checking for parameter synthesis, where the variables of the polynomials are assumed to be known and controllable (Junges et al. 2018). We re-frame these models such that the variables represent latent aspects of the environment that can be estimated during execution. This allows for an exact closed-form representation of the belief over

the latent variables, whilst also enabling the specification of dependencies between transitions. The closed-form belief representation can accurately deal with high-dimensional latent parameter spaces, resulting in better action selection. Furthermore, our method removes the need for hand-tuning the number of samples used to represent the belief.

We empirically show, in two domains, that incorporating the belief representation of HP$^2$-MDP in typical BAMDP solvers, such as Bayes-adaptive Monte-Carlo planning (BAMCP) (Guez, Silver, and Dayan 2012), yields improvement in planning performance. The first domain is a benchmark problem proposed in the original BAMCP paper. The second, more realistic, domain uses data from forecast ocean currents to simulate a problem where an ocean glider is planning a route to a goal location.

## Related Work

### MDPs with Hidden Factors

Markov decision processes (MDPs) have been used in planning under uncertainty (Mausam and Kolobov 2012) to help agents make optimal action choices. When MDPs are used to describe a system, we assume that the state is fully observable to the agent. However, this assumption cannot be satisfied in cases where the sensors used to determine the state are unreliable (Kaelbling, Littman, and Cassandra 1998; Hsiao, Kaelbling, and Lozano-Perez 2007), or the states are dependent on unknown factors, such as a human's likelihood to assist a robot (Nanavati et al. 2021; Costen et al. 2022). Partially Observable MDPs (POMDPs) (Kaelbling, Littman, and Cassandra 1998), and specialisations thereof are typically used to describe such systems.

In POMDPs, the state space is hidden from the agent. The agent receives stochastic observations, which are used to maintain a belief distribution over the hidden state space. The belief distributions can be updated to reflect hidden state transitions, which can be used in scenarios such as tracking the location of a human through observations made from an unreliable sensor in an autonomous driving setting (Wray, Witwicki, and Zilberstein 2017). The belief distribution is updated by reasoning over the hidden state transitions and the stochastic observations. This can be difficult, as the hidden state is dynamic, and thus hard to keep track of through stochastic observations.

Hidden Goal MDPs (Fern et al. 2014) and POMDP-lite (Chen et al. 2016) attempt to simplify the problem by splitting the state space into a static hidden state space and a stochastic observable state space. The hidden state is unknown at the start of the run, and the agent updates their belief over the hidden states as observable state transitions occur. As the hidden state is fixed, the agent does not have to reason over hidden state transitions, reducing the complexity. Each hidden state corresponds to an MDP, and the hidden goal MDPs and POMDP-lites can be framed as a problem where the agent is planning over a set of MDPs, where the true MDP is unknown.

Similarly, robust MDPs (Tamar, Mannor, and Xu 2014) and contextual MDPs (Hallak, Di Castro, and Mannor 2015) represents an MDP governed by uncertain parameters, which are a member of a known set. While the robust MDP solvers attempt to plan for the worst-case scenario, contextual MDP solvers aim to minimise the regret; the gap between the cumulative reward collected by the agent, and the cumulative reward the agent would have collected if they knew the true parameters (Rigter, Lacerda, and Hawes 2021a). However, this assumes that the system can be a discrete set of MDPs.

Doshi-Velez and Konidaris (2016) present a Hidden Parameter MDP, where the transitions are functions governed by latent parameters. By approximating the transition functions to Gaussian processes, they outline a method of maintaining a belief over the latent parameters.

### Bayes-Adaptive MDPs

BAMDPs allow the agent to reason over a continuous set of MDPs the system is in (Duff 2002; Guez, Silver, and Dayan 2012). This is done by parameterizing the transition probabilities with a set of continuous parameters, which can more accurately reflect a real system. However, many systems discretize the parameter space (Nanavati et al. 2021) to reduce the complexity of the problem. In other approaches where transition probabilities are assumed to be independent, every transition probability is represented as a latent parameter (Rigter, Lacerda, and Hawes 2021b; Grover, Basu, and Dimitrakakis 2020).

In general BAMDPs, the belief over the latent parameters is represented through sampled-based methods (Guez et al. 2014). This allows us to solve BAMDPs where there is dependence between transitions. In cases where the transitions are assumed to be independent, an exact form of the belief can be maintained. In such BAMDPs, the probability distribution over the parameters is represented by Dirichlet distributions. As transitions occur, these Dirichlet distributions are updated. While we can maintain an exact form of the belief distribution for such BAMDPs, we are unable to describe more complex relationships between the transition probabilities.

### Polynomial MDPs

In polynomial Markov Chains (MC) and MDPs (Winkler et al. 2019), the transition function is expressed as a polynomial function of a set of parameters. This allows us to add dependencies between transitions and allows us to narrow the uncertainty on future transition probabilities. These polynomial MDPs have been widely used in the verification community to express spaces of possible behaviour (Hahn, Han, and Zhang 2011a). They have been used in probabilistic model checking to provide theoretical guarantees to the reachability of goals over the parameter space (Junges et al. 2018; Hahn, Han, and Zhang 2011b). In the verification problem, we check whether a known sub-space of the parameter space for the polynomial MDP can satisfy a given specification. Other works consider the parameter synthesis problem (Junges et al. 2019), where they aim to partition the parameter space into regions where the specification is satisfied or not. Both the verification and synthesis problems assume that the true values of the parameters are known, or within a given bound. In this work, we re-frame the polynomial MDP as a BAMDP, where the parameters are hidden, and we must reason over the posterior probability distribution over the parameters.

## Preliminaries

In this paper, we consider stochastic shortest path (SSP) MDPs (Mausam and Kolobov 2012), where the objective is to minimise the expected cumulative cost to reach a set of goal states. This is done to match the domains used for empirical evaluation. However, note that the approach presented here is independent of the objective and can be applied to reward maximisation problem straightforwardly. Thus, we will refer to the model simply as an MDP.

When an MDP's transition dynamics is governed by a set of latent parameters, $\Theta$, a BAMDP can be used to represent the world. Consider a MDP with an uncertain transition function $\mathcal{M} = \langle S, \overline{s_0}, A, \{T_\theta\}_{\theta\in\Theta}, C, G\rangle$, where $S$ is a set of states; $s_0 \in S$ is the initial state; A is a set of actions; $\{T_\theta\}_{\theta\in\Theta}$ is a set of possible transition functions, governed by a set of latent parameters in $\Theta$; $C : S \times A \to \mathbb{R}_{\geq 0}$ is a cost function; and $G \subset S$ is a set of goal states. The parameter space is $\Theta = \mathbb{R}^N$, i.e. the latent parameters are $N$-dimensional vectors, and $\theta$ defines a possible transition function $T_\theta : S \times A \times S \to [0, 1]$ which gives the probability of transitioning to $s'$ given that $a$ was executed at $s$ and the parameter values are $\theta$, i.e., $T_\theta(s, a, s') = P(s' \mid s, a, \theta)$. We assume a prior distribution $P_0(\theta)$ over the parameter space. The agent maintains a posterior distribution, or belief, over $\Theta$ at each timestep $t$, $P_t(\theta)$. This is determined by the observed history, $h_t = s_0 a_0 s_1 a_1 \cdots a_{t-1} s_t$ using Bayes' rule, $P_t(\theta) = P(\theta|h_t) \propto P(h_t|\theta)P_0(\theta)$. A BAMDP factors the history into the state space, and explicitly models how the transition probabilities change with the history of the agent.

**Definition 1 (BAMDP)** *A BAMDP is defined by the tuple, $\langle S^+, s_0^+, A, T^+, C^+, G^+\rangle$, where:*

- $S^+ = S \times \mathcal{H}$*, where $\mathcal{H}$ is the set of all histories;*
- $s_0^+ = (s_0, h_0)$*, where $h_0 = s_0$ is the initial history;*
- $A$ *is the set of actions;*
- $T^+((s, h_t), a, (s', h_t a s')) = \int_{\theta\in\Theta} T_\theta(s, a, s')P_t(\theta)d\theta$ *is the transition function, where $P_t(\theta) \propto P(h_t|\theta)P_0(\theta)$;*
- $C^+((s, h_t), a) = C(s, a)$ *is the cost function;*
- $G^+ = \{(s, h) \in S^+ \mid s \in G\}$ *is the set of goal states.*

Although the state-history pairs in $S^+$ are redundant because the current state can be extracted from the history, we use the $(s, h)$ notation for clarity as in (Guez, Silver, and Dayan 2012). The goal of the BAMDP is defined as minimising the expected cumulative cost to reach a state in $G^+$. Therefore, setting $V^{\pi^*}(s, h_t) = 0$ for all $(s, h_t) \in G^+$, the optimal policy selects the action that minimises the value function according to:

$$V^{\pi^*}(s, h_t) = \arg\min_a C(s, a) + \int_{\theta\in\Theta} \sum_{s'\in S} P(\theta|h_t)T_\theta(s, a, s') \cdot V^{\pi^*}(s', h_t a s')d\theta. \quad (1)$$

To solve the BAMDP via value iteration, the posterior distribution for every history must be computed, which quickly becomes computationally infeasible. Therefore, BAMDPs are typically solved using a sample-based solver, such as BAMCP (Guez, Silver, and Dayan 2012). The BAMCP algorithm is adapted from Monte-Carlo Tree Search (MCTS) (Kocsis and Szepesvári 2006), and is based on running trials from the root node to estimate the Q-values of the available actions. Each trial simulates a run through the model and records the cost collected. The average cost over the trials is used to estimate the Q-values. As the number of trials increases, the estimated Q-value converges towards the true value. The BAMCP algorithm is outlined in Algorithm 1. It is an online algorithm where a search procedure (line 5) is interleaved with action execution and updating of the belief over the parameters given the observed action outcomes (line 6 and 7). The search procedure is based on sampling a possible parameter instantiation from the belief distribution, and then simulating a run according to the correspondent transition function, repeating this process until a user-defined timeout (lines 11–14). A key part of BAMCP is maintaining and updating the belief given the online observations. We identify the two types of BAMDPs considered more often, where the posterior over the belief is maintained in distinct methods.

Algorithm 1: Bayes-Adaptive Monte Carlo Planning

1: **Function**{Run}$\{P_0(\theta), s_0, \langle S, A, \{T_\theta\}_{\theta\in\Theta}, C, G\rangle\}$
2: $t \leftarrow 0$
3: $h_0 \leftarrow s_0$
4: **while** $s_t \notin G$ **do**
5: $\quad a_t \leftarrow$ `Search(`$P_t(\theta), s_t$`)`
6: $\quad s_{t+1} \leftarrow$ execute action $a_t$ and observe outcome.
7: $\quad P_{t+1}(\theta) \leftarrow$ `UpdateBelief(`$P_t(\theta), a_t, s_{t+1}$`)`
8: $\quad t \leftarrow t + 1$
9: **end while**
10: **Function**{Search}$\{P_t(\theta), s_t\}$
11: **repeat**
12: $\quad \theta_{sample} \sim P_t(\theta)$.
13: $\quad$ `Simulate(`$s_t, T_{\theta_{sample}}$`)`
14: **until** `Timeout()`
15: **return** $\arg\max_a Q(s_t, a)$

**BAMDP-L** The simplest form of a BAMDP, presented in (Duff 2002; Ross et al. 2011), considers an MDP where the transition functions are independent. We refer to this type of BAMDP as BAMDP-L, as the latent parameters are the local transition probabilities, and there are no global parameters. The solvers for BAMDP-L represent the unknown transition functions using Dirichlet distributions, which are updated as transitions are observed. These Dirichlet distributions are sampled in Algorithm 1 line 12. The properties of the BAMDP-L allow us to maintain a closed-form distribution for the posterior distribution in the BAMCP algorithm.

**BAMDP-G** In BAMDP-Gs, the transition probabilities are functions of global latent parameters. In this case, it is often necessary to approximate the belief. A typical method for this is particle-based, where the posterior over the belief is described by a set of $M$ sample MDPs (Guez et al. 2014). Each MDP has a weight describing the likelihood of it being the true description of the domain. At the start of a run, we sample $M$ time from the prior belief $P_0(\theta)$ to get a set of

parameter instantiations, $U = \{\theta^1, \cdots, \theta^M\}$. These sampled parameter values are used to generate a set of MDPs, $U_{MDP} = \{\mathcal{M}^1, \cdots, \mathcal{M}^M\}$, with the transition function of $\mathcal{M}^k$ equal to $T_{\theta^k}$. Each MDP $\mathcal{M}^k$ is a particle with an associated weight function $w_k : \mathcal{H} \to \mathbb{R}_{>0}$ which represents how probable $\mathcal{M}^k$ is, given an observed history. We initialise $w_k(h_0) = \frac{1}{M} \forall k \in \{1, \cdots, M\}$. These weights are then updated using Bayes' rule as the agent observes state-action transitions, via

$$w_k(has') \propto T_{\theta^k}(s, a, s') \cdot w_k(h). \tag{2}$$

However, the accuracy of the representation using particles is dependent on $M$, as this affects how well the parameter space is covered: as the number of parameters increases, the value of $M$ required to accurately represent the belief grows exponentially. In this work, we tackle this issue by proposing a class for which we can maintain a closed-form representation of the belief for a set of global latent parameters.

## HP$^2$-MDP

We consider an MDP where the transition probabilities can be expressed by a polynomial of a set of global parameters, which are unknown to the agent.

**Definition 2 (HP$^2$-MDP)** *A hidden parameter polynomial MDP (HP$^2$-MDP) is defined by the tuple $\langle S, s_0, A, \Theta, P_0(\theta), \{T_\Theta, C, G\rangle$, where:*

- *$S$, $s_0$ and $A$ are the set of states, initial state and set of actions, respectively;*
- *$\Theta = [0,1]^N$ is the parameter space of a set of $N$ global parameters. We denote elements of $\Theta$ as $\theta = (\theta_1, \cdots, \theta_N)$;*
- *$P_0(\theta)$ is the prior belief over $\Theta$. $P_0(\theta) \in Pol(\Theta)$, i.e., it is a polynomial function over $\Theta$;*
- *$T_\Theta : S \times A \times S \to Pol(\Theta)$ is the (polynomial) set of possible transition functions;*
- *$C : S \times A \to \mathbb{R}$ is the cost function and $G \subset S$ is the set of goal states.*

To be well-formed, the transition functions of a HP$^2$-MDP must be valid probability distributions over the set of states, i.e., for all $s \in S$ and $a \in A$ that can be executed in $s$:

$$\sum_{s' \in S} T_\Theta(s, a, s')(\theta) = 1 \; \forall \theta \in \Theta. \tag{3}$$

HP$^2$-MDP allows us to represent a wide range of BAMDPs (with global latent parameters) while maintaining a closed-form posterior distribution. While the BAMDP has a parameter space of $\Theta = \mathbb{R}^N$, in the HP$^2$-MDP, the parameter space is restricted to $\Theta = [0,1]^N$. This change in parameter space is required to maintain a closed-form distribution, to confine the possible parameter values to a fixed range. In the case of a parameter with a range outside of $[0,1]$, the parameter can be scaled to fit the above constraint. Equation 3 imposes constraints on the space of valid parameter values. We cover the case where $a)$ the parameters are independent of each other, and $b)$ when the parameters follow $\sum_{i=1}^N \theta_i = 1$. For each case, we will provide an example illustrating when these constraints would be used. Other linear constraints generated by Equation 3 can be computed by combining the methods used in the cases outlined above. We give an example of this by showing that BAMCP-L is a type of HPP-MDP.

### Independent Global Parameters

We consider a HP$^2$-MDP $\mathcal{M}$, where the $N$ parameters are independent, and have a prior belief, $P_0(\theta)$.

**Example 1 (Independent Parameters)** *Consider an underwater glider planning a route from a start location to a finish location in the sea. The longitudinal and latitudinal velocity of the water at each location is known, but the effect of these on the glider's movement is unknown. We assume a linear relationship between the water's speed and the effect on the glider, and the effect of the longitudinal and latitudinal water velocity are respectively expressed by $\theta_h, \theta_v$. The effect of the longitudinal and latitudinal water velocity is independent, where the longitudinal water velocity will only affect the glider's movement in the east-west direction, and the latitudinal water velocity will only affect the glider in the north-south direction. For example, a point in the sea may have a longitudinal water velocity of $-0.3ms^{-1}$, and a latitudinal water velocity of $0.6ms^{-1}$. If the agent chose the action to travel west, the transition probabilities would become the following: $P_{stationary} = 0.3\theta_h \cdot (1 - 0.6\theta_v)$, $P_{west} = (1 - 0.3\theta_h) \cdot (1 - 0.6\theta_v)$, $P_{north} = 0.3\theta_h \cdot 0.6\theta_v$ and $P_{north-west} = (1 - 0.3\theta_h) \cdot 0.6\theta_v$.*

**Closed-Form Belief Update** Example 1 illustrates a case where the latent parameters are independent. In such cases, the posterior belief after transitioning from a state-action pair $s, a$ to state $s'$ follows Bayes' rule, and follows:

$$P_{t+1}(\theta \mid s_t, a_t, s_{t+1}) = \frac{P_t(\theta) T_\Theta(s_t, a_t, s_{t+1})(\theta)}{\int_{\theta' \in \Theta} P_t(\theta') T_\Theta(s_t, a_t, s_{t+1})(\theta') d\theta'}. \tag{4}$$

Assuming the prior belief, $P_t(\theta)$, to be polynomial, we have that $P_t(\theta) T_\Theta(s_t, a_t, s_{t+1})(\theta)$ is a polynomial. Thus, we can consider its expansion into $J$ terms, $\sum_{j=1}^J \beta_j \prod_{i=1}^N \theta_i^{a_i^j}$, where $\beta_j$ is the coefficient of the $j$-th term, and $a_i^j$ is the order of $\theta_i$ in the $j$-th term. In the independent global parameter case, the valid parameter space is a hyper-cube, where each parameter is in $[0,1]$. Therefore, the denominator can be expressed as:

$$\int_0^1 \cdots \int_0^1 \sum_{j=1}^J \beta_j \prod_{i=1}^N \theta_i'^{a_i^j} d\theta' = \sum_{j=1}^J \beta_j \prod_{i=1}^N \frac{1}{a_i^j + 1}. \tag{5}$$

From this, the belief can be written as:

$$P_{t+1}(\theta) = \frac{\sum_{j=1}^J \beta_j \prod_{i=1}^N \theta_i^{a_i^j}}{\sum_{j=1}^J \beta_j \prod_{i=1}^N \frac{1}{a_i^j + 1}} \tag{6}$$

Since $P_0(\theta)$ is polynomial by definition, a simple induction argument yields that $P_{t+1}(\theta)$ is a polynomial of the form stated in Equation 6.

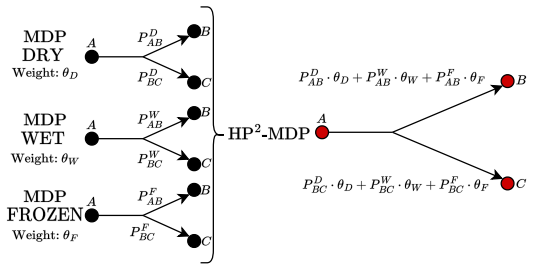



Figure 1: HP$^2$-MDP example with dependent parameters.

## Global parameters Bound by a Simplex

In other scenarios, there will be dependencies between the latent parameters. The example below shows a scenario where this may be true.

**Example 2 (Dependent Parameters)** *Consider a robot planning a route over a domain with unknown weather conditions. We have three MDPs, shown on the left in Figure 1, respectively describing the motion of the robot in dry, wet and frozen weather conditions. The underlying true MDP is a combination of the three MDPs, depending on the wetness and the temperature of the domain. Each MDP contributes a different proportion to the true MDP, which is represented by weights $\theta_D, \theta_W$ and $\theta_F$. In the true MDP, the transition probability from state $A$ to state $B$ would be $P^T_{AB} = P^D_{AB} \cdot \theta_D + P^W_{AB} \cdot \theta_W + P^F_{AB} \cdot \theta_F$. However, when these weights are unknown to the agent, this problem becomes a HP$^2$-MDP. The resulting HP$^2$-MDP is shown on the right side of Figure 1. The weights have a constraint of $\theta_D + \theta_W + \theta_F = 1$, and the agent must maintain a posterior distribution over these latent parameters.*

**Closed-Form Belief Update** When the parameters in $\mathcal{M}$ satisfy $\sum_{i=1}^{N} \theta_i = 1$, the denominator in Equation 4 must integrate over the $N-1$ simplex. The integral of the product of powers of the parameters over the $N-1$ simplex (Gupta and Richards 2001) can be expressed as:

$$\int_{\theta \in \Theta} \prod_{i=1}^{N} \theta_i^{a_i - 1} d\theta = \oint \prod_{i=1}^{N} \theta_i^{a_i - 1} d\theta = \frac{\prod_{i=1}^{N} \Gamma(a_i)}{\Gamma\left(\sum_{i=1}^{N} a_i\right)}, \quad (7)$$

where $\Gamma(x)$ is the gamma function. Therefore, the normalization constant, similar to Equation 5, is

$$\oint \sum_{j=1}^{J} \beta_j \prod_{i=1}^{N} {\theta'_i}^{a_i^j} d\theta' = \sum_{j=1}^{J} \beta_j \cdot \frac{\prod_{i=1}^{N} \Gamma(a_i^j + 1)}{\Gamma\left(\sum_{i=1}^{N} (a_i^j + 1)\right)}. \quad (8)$$

And thus, the belief over the latent parameter is

$$P_{t+1}(\theta) = \frac{\sum_{j=1}^{J} \beta_j \prod_{i=1}^{N} \theta_i^{a_i^j}}{\sum_{j=1}^{J} \beta_j \cdot \frac{\prod_{i=1}^{N} \Gamma(a_i^j + 1)}{\Gamma\left(\sum_{i=1}^{N} (a_i^j + 1)\right)}}. \quad (9)$$

## BAMDP-L: A Special Case of HP$^2$-MDP

Other linear relationships between the global parameters can be expressed by applying a combination of Equation 5 and 7. For example, consider the case where the set of parameters are $\theta = \{\theta_{sas'} \in [0,1] \forall s, a, s' \in S \times A \times S\}$, where $T_\Theta(s, a, s') = \theta_{sas'}$. This is the case when every transition probability is independent, and thus each state-action-state transition probability is defined as a latent parameter. The parameters satisfy the following relationship $\sum_{s' \in S} \theta_{s,a,s'} = 1$, as the transition probabilities from the same state-action pair must sum to 1. When the prior belief distribution is $P_0(\theta) = 1$, this is equivalent to BAMDP-L. Therefore, BAMDP-L can be seen as a special case of HP$^2$-MDP.

First, we consider the case with a general prior distribution. The posterior belief is

$$\begin{aligned} &P_{t+1}(\theta) \\ &= \frac{1}{\eta} P_t(\theta) T_\Theta(s_t, a_t, s_{t+1}) = \frac{1}{\eta} P_0(\theta) \prod_{t'=1}^{t} T_\Theta(s_{t'}, a_{t'}, s_{t'+1}) \\ &= \frac{1}{\eta} P_0(\theta) \prod_{t'=1}^{t} \theta_{s_{t'} a_{t'} s_{t'+1}} = \frac{1}{\eta} P_0(\theta) \prod_{\substack{(s,a,s') \in \\ S \times A \times S}} {\theta_{sas'}}^{\alpha^t_{sas'}} \end{aligned} \quad (10)$$

where $\eta$ is the normalisation constant, and $\alpha^t_{sas'}$ is the number of times the transition from state-action $sa$ to new state $s'$ has occurred in the $t$ steps. We generalise the prior distribution, expressing the polynomial as

$$P_0(\theta) = \sum_{j=1}^{J} \beta_j \prod_{(s,a,s') \in S \times A \times S} \theta_{sas'}^{a^j_{sas'}}. \quad (11)$$

The normalization constant for belief distribution can be simplified to

$$\begin{aligned} \eta &= \oint P_0(\theta) \prod_{(s,a,s') \in S \times A \times S} {\theta_{sas'}}^{\alpha^t_{sas'}} d\theta \\ &= \oint \sum_{j=1}^{J} \beta_j \prod_{(s,a,s') \in S \times A \times S} \theta_{sas'}^{a^j_{sas'} + \alpha^t_{sas'}} d\theta \\ &= \sum_{j=1}^{J} \beta_j \prod_{(s,a) \in S \times A} \oint \prod_{s' \in S} \theta_{sas'}^{a^j_{sas'} + \alpha^t_{sas'}} d\theta_{sa\cdot}, \end{aligned} \quad (12)$$

where the transition probabilities that share the starting state and action form a simplex. We can use Equation 7 to express the normalizing constant of the belief distribution as

$$\eta = \sum_{j=1}^{J} \beta_j \prod_{(s,a) \in S \times A} \frac{\prod_{s' \in S} \Gamma(a^j_{sas'} + \alpha^t_{sas'} + 1)}{\Gamma\left(\sum_{s' \in S} (a^j_{sas'} + \alpha^t_{sas'} + 1)\right)}. \quad (13)$$

When the prior belief is $P_0(\theta) = 1$ (where every value of $\theta$ is equally likely), then $J = 1, \beta_1 = 1$ and $a^j_{sas'} =$

$0 \forall (s, a, s') \in S \times A \times S$. The belief will simplify to

$$P_{t+1}(\theta) = \frac{\prod_{(s,a,s') \in S \times A \times S} \theta_{sas'}{}^{\alpha^t_{sas'}}}{\prod_{(s,a) \in S \times A} \frac{\prod_{s' \in S} \Gamma(\alpha^t_{sas'}+1)}{\Gamma\left(\sum_{s' \in S} (\alpha^t_{sas'}+1)\right)}} \tag{14}$$

This is equivalent to the posterior for a Dirichlet distribution used to solve a BAMDP-L.

**Sampling the Belief Distribution** The closed form expression shown in Equation 6 and 9 allows us to maintain an accurate representation of the belief distribution for HP$^2$-MDPs where the transition probabilities are expressed as a polynomial function of the global parameters. To solve HP$^2$-MDP, we can use the BAMCP algorithm, where we sample from the closed-form belief distribution at the root node instead of sampling a weights-based representation. While the weights-based representation gives an approximation to the belief, our method allows us to maintain an exact representation.

In HP$^2$-MDP, the number of parameters defines the number of dimensions of the space we sample from. At lower dimensions, we compute the cumulative density function (CDF). We can sample the CDF to produce particles that follow the probability distribution. At higher dimensions, we use a Markov Chain Monte-Carlo (MCMC) (Chib and Greenberg 1995) to generate representative samples of $P(\theta)$.

**Belief Distribution Dimension Reduction** As the steps taken by the agent increase, the number of terms in the polynomial representing the belief increases as well. For problems with large horizons, the belief distribution may gain enough terms such that the MCMC algorithm becomes time-consuming. To minimise the time taken to sample the belief while maintaining an accurate representation, we apply dimension reduction to such problems.

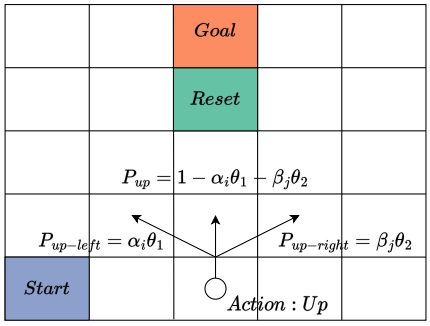
After a fixed number of steps, we use ordinary least-squares to fit our belief distribution to a polynomial of a lower degree. The fitted polynomial is used to represent the belief. After fitting the belief distribution, it can be updated using Bayes' rule as transitions occur. This allows us to use HP$^2$-MDP to represent domains with large state spaces and sample from the belief accurately, whilst bounding the size of its polynomial representation.

## Experiments

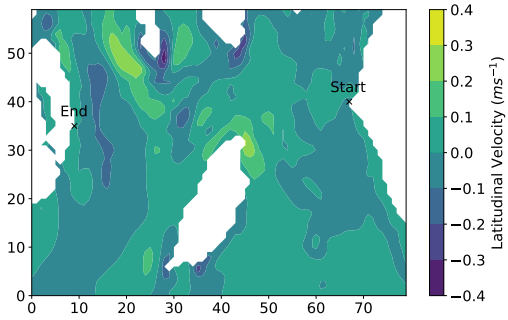
We use BAMCP (Algorithm 1) and apply HP$^2$-MDP, BAMDP-G and a particle-filter-based belief representation to two domains, showing empirically that HP$^2$-MDP results in better performance.

### Algorithms

We apply the following algorithms for updating the belief: **HP$^2$-MDP**, as presented in Definition 2. We used dimension reduction on the second domain. **BAMCP-G**, as presented in Guez et al. (2014), which uses the approach described in BAMDP-G to maintain the belief. **BAMCP-PF**, where we add noise into the samples when we update the posterior distribution. Similar to BAMCP-G, we sample the initial belief




(a) Grid5 problem with an example transition. $\alpha_i$ and $\beta_j$ are known to the agent.




(b) Latitudinal water velocity of ocean. The white regions represents land.

Figure 2: Domains used in experiments.

to generate the initial set of particles, $U = \{\mathcal{M}^1, \cdots, \mathcal{M}^M\}$. We assign a weight of $w_k(h_0) = \frac{1}{M}$ to particle $\mathcal{M}^k$. When an action-state pair has been observed, the weights are updated using Equation 2. Then, we use low-variance sampling to select $M$ samples from the updated distribution. We add a small perturbation ($\mathcal{E} \sim \mathcal{N}(0, 0.01)$) to the sample values to allow the posterior distribution to explore the latent parameter space. The new sampled $M$ particles are given the same weight of $\frac{1}{M}$. Between each step, the algorithms sample the belief and run simulations for time $\mathcal{T}$.

### Domains

**Grid5** Guez, Silver, and Dayan (2012) considered a $5 \times 5$ grid, where the only reward is located at the goal, directly next to a reset square. If the agent enters the reset square, they are sent to the start. The agent can take actions to navigate to neighbouring squares, and each action has a small probability of failure. The authors use Dirichlet distributions to model the unknown state transition probabilities. We modify the problem to introduce dependencies between the transitions.

We consider a $5 \times 5$ grid, as shown in Figure 2a. The transition probabilities of actions taken from the square at $(i, j)$ are determined by global latent parameters $\theta = (\theta_1, \theta_2)$, and local known parameters $\{\alpha_i, \beta_j\}$. Every step has a cost of 1, and the run ends at the goal square. $\theta_1$ and $\theta_2$ both are in the range $[0, 1]$ and are independent from each other. The prior distribution over the latent parameters is uniform.

| Algorithm | HP$^2$-MDP | BAMCP-G | | | | | BAMCP-PF | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $M$ | – | 100 | 200 | 300 | 400 | 500 | 100 | 200 | 300 | 400 | 500 |
| Failure Rate | **0.0** | 1.0 | 1.0 | 0.85 | 0.85 | 0.9 | 0.74 | 0.8 | 0.64 | 0.74 | 0.77 |
| Mean Cost | **17.6± 0.435** | – | – | 58.7± 3.64 | 50.3± 3.53 | 37.5± 3.32 | 49.6± 2.64 | 55.2± 2.85 | 48.8± 2.52 | 44.1± 2.73 | 41.8± 2.71 |

Table 1: Results from 100 simulation conducted in the SeaGrid domain. The bold values indicate the best performance.

**SeaGrid** We consider the problem outlined in Example 1, where a glider is planning a route from a start to a finish location. The domain is a $17 \times 13$ grid. The forecast for the longitudinal and latitudinal velocity of the water is known ahead of time, using data from Copernicus (2022). An example of the vertical velocity is shown in Figure 2b.

The influence the horizontal and vertical water velocity will have on the glider is unknown. Therefore, we consider two unknown latent parameters, $\theta = (\theta_h, \theta_v)$, describing the influence of the sea's velocity on the motion of the glider. The agent has an action choice of the cardinal directions and the choice to do nothing. The choice of doing nothing may result in the glider being pushed along by the water.

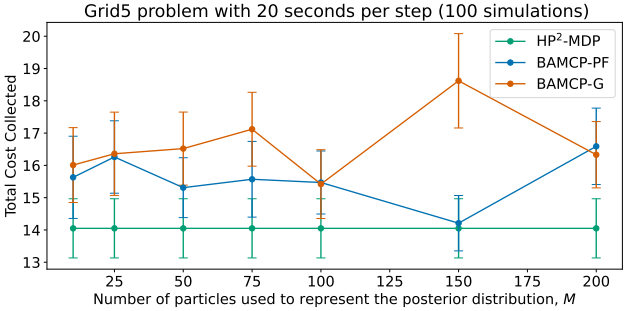



Figure 3: Costs collected in the Grid5 domain

## Results

**Grid5** We tested the HP$^2$-MDP without dimension reduction, BAMCP-G and BAMCP-PF algorithms on the Grid5 problem, and we set $\mathcal{T}$ to 20 seconds. We also varied the size of the set of samples used to represent the posterior distribution, $M$, for BAMCP-G and -PF. Each algorithm was tested on the domain 100 times. We ran the simulations on Linux OS with an Intel Core i9-11900H processor and 16GB of RAM.

The results are in Figure 3. The HP$^2$-MDP algorithm does not rely on sets of samples, thus the results do not vary with $M$. The BAMCP-G and -PF algorithms are shown to perform worse at low values of $M$, as the particles are sparsely distributed over the latent parameter space. As the value of $M$ increases, the posterior distribution is better represented, and thus able to return actions closer to the optimal strategy. However, at high values of $M$, the time taken to sample the posterior distribution increases such that the number of trials that can be completed until timeout is reduced. This leads to the algorithm returning an action without converging to the optimal policy, and thus higher costs.

We found that the BAMCP algorithms with particle-based belief updates lead to higher costs than the HP$^2$-MDP closed-form updates, where we varied $M$ between 10 and 200. BAMCP-PF performed marginally better than the BAMCP-G, as the particles were able to explore the parameter space. The BAMCP-PF generally resulted in higher costs than the HP$^2$-MDP algorithm, except in the case where $M = 150$, where the particle filter and the HP$^2$-MDP algorithm performed comparably. Both the BAMCP-G and -PF experience high variability in performance as $M$ is altered. To get the best performance from these algorithms, the optimal value of $M$ must be explored. In comparison, as HP$^2$-MDP is using an exact form of the posterior distribution, there is no need to fine-tune $M$ to use the algorithm.

**SeaGrid** In the seagrid problem, we compared HP$^2$-MDP to BAMCP-PF and BAMCP-G with $M = 100, 200, 300, 400$ and 500. We applied dimension reduction to HP$^2$-MDP, where the highest degree of the polynomial was fixed to 12. We increased the number of particles used, as the transition probabilities were more sensitive to variation in the parameter values. This meant a slight deviation from the true parameter values would result in significantly different transition probabilities. Therefore, sample-based methods would struggle to fully capture the true MDP without large values of $M$. We set $\mathcal{T}$ to 90 seconds and ran 100 simulations for each algorithm. We ran the trials on CentOS with an Intel Xeon Platinum 8268 CPU at 2.90 GHz Processor, with 16GB of RAM. We applied a limit of 75 steps before halting the simulation due to memory constraints for BAMCP-PF with $M = 500$. This is reflected in the failure rate, shown in Table 1.

We found that HP$^2$-MDP significantly outperforms the particle-based approaches. We attribute this to the size of the state space and the high dependence of the transition probabilities on the latent parameters. This meant that for low values of $M$, the samples were insufficient to cover the latent parameter space, resulting in poor performance. This was particularly pronounced in BAMCP-G, where the samples were fixed at the start of a trial, and thus the performance was heavily dependent on the existence of a sample similar to the underlying MDP.

## Conclusion

We have presented a new class for BAMDPs, HP$^2$-MDP, where the transitions are expressed as polynomial functions of the latent parameters. HP$^2$-MDP allows the posterior distribution over the latent parameters to be maintained in a closed-form and exact representation, and we outlined the method for updating the posterior distribution for HP$^2$-MDP. We have shown that our method outperforms current BAMDP solvers with particle-based representations of the posterior distribution in two domains.

## Acknowledgments

This work was supported by the Defence Science and Technology Laboratory, the EPSRC Programme Grant 'From Sensing to Collaboration' (EP/V000748/1), the Clarendon Fund at the University of Oxford, and a gift from Amazon Web Services. This document is an overview of UK MOD's Defence Science and Technology Laboratory (DSTL) sponsored research and is released for informational purposes only. The contents of this document should not be interpreted as representing the views of the UK MOD, nor should it be assumed that they reflect any current or future UK MOD policy.

## References

Chen, M.; Frazzoli, E.; Hsu, D.; and Lee, W. S. 2016. POMDP-lite for robust robot planning under uncertainty. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 5427–5433.

Chib, S.; and Greenberg, E. 1995. Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4): 327–335. Publisher: [American Statistical Association, Taylor & Francis, Ltd.].

Copernicus, M. S. 2022. Data | Copernicus Marine.

Costen, C.; Rigter, M.; Lacerda, B.; and Hawes, N. 2022. Shared Autonomy Systems with Stochastic Operator Models. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 4614–4620. Vienna, Austria: International Joint Conferences on Artificial Intelligence Organization. ISBN 978-1-956792-00-3.

Doshi-Velez, F.; and Konidaris, G. 2016. Hidden parameter markov decision processes: a semiparametric regression approach for discovering latent task parametrizations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, 1432–1440. New York, New York, USA: AAAI Press. ISBN 978-1-57735-770-4.

Duff, M. O. 2002. *Optimal Learning: Computational Procedures For Bayes-Adaptive Markov Decision Process*. Ph.D. thesis, University of Massachusetts.

Fern, A.; Natarajan, S.; Judah, K.; and Tadepalli, P. 2014. A Decision-Theoretic Model of Assistance. *Journal of Artificial Intelligence Research*, 50: 71–104.

Grover, D.; Basu, D.; and Dimitrakakis, C. 2020. Bayesian Reinforcement Learning via Deep, Sparse Sampling. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 3036–3045. PMLR. ISSN: 2640-3498.

Guez, A.; Heess, N.; Silver, D.; and Dayan, P. 2014. Bayes-Adaptive Simulation-based Search with Value Function Approximation. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Guez, A.; Silver, D.; and Dayan, P. 2012. Efficient Bayes-Adaptive Reinforcement Learning using Sample-Based Search. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Gupta, R. D.; and Richards, D. S. P. 2001. The History of the Dirichlet and Liouville Distributions. *International Statistical Review / Revue Internationale de Statistique*, 69(3): 433–446. Publisher: [Wiley, International Statistical Institute (ISI)].

Hahn, E. M.; Han, T.; and Zhang, L. 2011a. Synthesis for PCTL in Parametric Markov Decision Processes. *NASA Formal Methods - Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011. Proceedings*, 146–161.

Hahn, E. M.; Han, T.; and Zhang, L. 2011b. Synthesis for PCTL in Parametric Markov Decision Processes. In Bobaru, M.; Havelund, K.; Holzmann, G. J.; and Joshi, R., eds., *NASA Formal Methods*, volume 6617, 146–161. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-20397-8 978-3-642-20398-5. Series Title: Lecture Notes in Computer Science.

Hallak, A.; Di Castro, D.; and Mannor, S. 2015. Contextual Markov Decision Processes. Technical Report arXiv:1502.02259, arXiv. ArXiv:1502.02259 [cs, stat] type: article.

Hsiao, K.; Kaelbling, L. P.; and Lozano-Perez, T. 2007. Grasping pomdps. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 4685–4692. IEEE.

Junges, S.; Abraham, E.; Hensel, C.; Jansen, N.; Katoen, J.-P.; Quatmann, T.; and Volk, M. 2019. Parameter Synthesis for Markov Models. Technical Report arXiv:1903.07993, arXiv. ArXiv:1903.07993 [cs] type: article.

Junges, S.; Jansen, N.; Katoen, J.-P.; Topcu, U.; Zhang, R.; and Hayhoe, M. 2018. Model Checking for Safe Navigation Among Humans. In McIver, A.; and Horvath, A., eds., *Quantitative Evaluation of Systems*, Lecture Notes in Computer Science, 207–222. Cham: Springer International Publishing. ISBN 978-3-319-99154-2.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2): 99–134.

Kocsis, L.; and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In Fürnkranz, J.; Scheffer, T.; and Spiliopoulou, M., eds., *Machine Learning: ECML 2006*, Lecture Notes in Computer Science, 282–293. Berlin, Heidelberg: Springer. ISBN 978-3-540-46056-5.

Mausam; and Kolobov, A. 2012. Planning with Markov Decision Processes: An AI Perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1): 1–210.

Nanavati, A.; Mavrogiannis, C.; Weatherwax, K.; Takayama, L.; Cakmak, M.; and Srinivasa, S. 2021. Modeling Human Helpfulness with Individual and Contextual Factors for Robot Planning. In *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation. ISBN 978-0-9923747-7-8.

Puterman, M. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Ltd.

Rigter, M.; Lacerda, B.; and Hawes, N. 2021a. Minimax Regret Optimisation for Robust Planning in Uncertain Markov Decision Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13): 11930–11938. Number: 13.

Rigter, M.; Lacerda, B.; and Hawes, N. 2021b. Risk-Averse Bayes-Adaptive Reinforcement Learning. *ArXiv*.

Ross, S.; Pineau, J.; Chaib-draa, B.; and Kreitmann, P. 2011. A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes. *The Journal of Machine Learning Research*, 12(null): 1729–1770.

Tamar, A.; Mannor, S.; and Xu, H. 2014. Scaling Up Robust MDPs using Function Approximation. In *Proceedings of the 31st International Conference on Machine Learning*, 181–189. PMLR. ISSN: 1938-7228.

Winkler, T.; Junges, S.; Pérez, G. A.; and Katoen, J.-P. 2019. On the Complexity of Reachability in Parametric Markov Decision Processes. *arXiv:1904.01503 [cs]*. ArXiv: 1904.01503.

Wray, K. H.; Witwicki, S. J.; and Zilberstein, S. 2017. Online Decision-Making for Scalable Autonomous Systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 4768–4774. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-0-3.

Algorithm 2: Markov Chain Monte-Carlo

**Function**{MCMC}{$P(\Theta)$}
$\Theta_1 \sim$ `Random(N)`
$\Theta_{i-1} \leftarrow \Theta_1$
$Samples \leftarrow [\Theta_1]$
**repeat**
  $\Theta_i \sim$ `Random(N)`
  $ratio = P(\Theta_i)/P(\Theta_{i-1})$
  **if** $ratio > 1$ **then**
    $Samples.Append(\Theta_i)$
    $\Theta_{i-1} \leftarrow \Theta_i$
  **else if** $ratio >$ `Random()` **then**
    $Samples.Append(\Theta_i)$
    $\Theta_{i-1} \leftarrow \Theta_i$
  **else**
    $Samples.Append(\Theta_{i-1})$
    $\Theta_i \leftarrow \Theta_{i-1}$
  **end if**
**until** `Timeout()`
**return** $Samples$
**EndFunction**

# Appendix

## How to do the maths in equation 5

## MCMC algorithm

We use a MCMC algorithm to generate samples from a closed-form posterior distribution with high parameter space dimensions. In a MCMC, a random sample $\Theta_1$ is generated, and the value of the posterior distribution $P(\Theta_1)$ at this point is calculated. This is added to the set of samples. Then, a new random sample $\Theta_2$ is generated, and has a posterior distribution value of $P(\Theta_2)$. If $P(\Theta_2) > P(\Theta_1)$, so the second sample has a higher likelihood of being sampled, and $\Theta_2$ is added to the set of samples. If $P(\Theta_2) < P(\Theta_1)$, $\Theta_2$ is accepted with a probability of $\frac{P(\Theta_2)}{P(\Theta_1)}$, otherwise, the previous sample, $\Theta_1$ is added to the set of samples. The full algorithm is shown in Algorithm 2.