Journal of Machine Learning Research ? (2002) ?–? Submitted 4/02; Published ??/??

Internal-State Policy-Gradient Algorithms for Partially Observable Markov Decision Processes

Douglas Aberdeen DOUGLAS.ABERDEEN@ANU.EDU.AU
Research School of Information Science and Engineering
Building 115, Daley Rd
Australian National University
A.C.T, 0200, AUSTRALIA

Jonanthan Baxter JONATHANNEBAXTER@YAHOO.COM
Panscient Pty Ltd
Adelaide, Australia

Editor: ??

Abstract

Policy-gradient algorithms are attractive as a scalable approach to learning approximate policies for controlling partially observable Markov decision processes (POMDPs). POMDPs can be used to model a wide variety of learning problems, from robot navigation to speech recognition to stock trading. The downside of this generality is that exact algorithms are computationally intractable, motivating approximate methods. Existing policy-gradient methods have worked well for problems admitting good memory-less solutions, but have failed to scale to large problems requiring memory. This paper develops novel algorithms for learning policies with memory. We present a new policy-gradient algorithm that uses an explicit model of the POMDP to estimate gradients, and demonstrate its effectiveness on problems with tens of thousands of states. We also describe three new Monte-Carlo algorithms that learn by interacting with their environment. We compare these algorithms on non-trivial POMDPs, including noisy robot navigation and multi-agent settings.

Keywords: Partially Observable Markov Decision Processes, Finite State Controllers, Reinforcement Learning, Policy Gradient

1. Introduction

Partially observable Markov decision processes (POMDPs) provide a framework for agents that learn how to interact with their environment in the presence of multiple forms of uncertainty. The only performance feedback given to the agent is a scalar reward signal. Rewards can be noisy and delayed from the actions that caused them.

Unlike the situation with fully observable Markov decision processes (MDPs), the problem of finding optimal policies for POMDPs is PSPACE-complete (Papadimitriou and Tsitsiklis, 1987), even when the POMDP is known. Thus, approximate methods are required even for relatively small problems with known dynamics. There are two main classes of approximate algorithms: value-function methods that seek to approximate the value of *belief states* — probability distributions over world-states — see Hauskrecht (2000) for a good overview; and policy-based methods that search for a good policy within some restricted class of policies. In this paper we introduce three new approximate algorithms of the latter variety.

Solving POMDPs in the most general situation requires the agent to have memory, or internal state, which can take the form of a trace of past observation/action pairs, or a belief state describing the current probability of each world state. Another form of agent memory is a finite state controller (FSC), which augments the agent with a set of internal states that store information about the previous history of observations. FSCs can be considerably more efficient memory devices than either history traces or belief states, because they can be configured to only store *relevant* information about the world-state and observation history. For example, in order to avoid being run-down by a car while crossing the road, I need only remember that I am standing on the edge of the road, and that I have seen a car approaching, not the color of the car nor the brand of toothpaste with which I brushed my teeth this morning.

In the FSC model examined here, both the transitions between the internal states of the controller, and the distribution over the agent's actions given a particular controller internal state, are governed by a parameterised class of policies. The agent's goal is to adjust these parameters in order to maximize its *long-term average reward.* Thus, the agent learns to use its internal states to store information that is releveant for maximizing its long-term reward, while simultaneously learning the optimal actions to take given its current internal state.

All the algorithms described here are *policy-gradient* algorithms in the sense that the agent adjusts the policy parameters in the direction of the gradient of the long-term average reward. The algorithms differ in the way the gradient is computed. The first algorithm—GAMP—relies on a known model of the POMDP to compute the gradient. It is based on a series matrix expansion of an exact expression for the gradient. The second algorithm—IState-GPOMDP—computes the gradient stochastically by sampling from the POMDP and the internal-state trajectories of the agent's FSC. The third algorithm—Exp-GPOMDP—reduces the variance of IState-GPOMDP by computing true expectations over internal-state trajectories.

These algorithms are designed to scale computationally to large state/observations spaces. To achieve this the algorithms make use of the ability of FSCs to automatically filter out large amounts of hidden state which do not contribute to the optimal policy. Equivalently, this can be viewed as automatic state aggregation, clustering together all environment states which do not need to be distinguished (Singh et al., 1995). Similar approaches to date have only yielded results on trivial internal-state problems and are restricted to finite-horizon tasks. We investigate some of the reasons that these algorithms have failed to scale and present results for infinite-horizon problems including a memory-less multi-agent task with 21,632 states and a noisy robot navigation task with a combined world/internal state space of 4180 states.

In Section 2 we formally introduce partially observable Markov decision processes and discuss existing algorithms for learning internal-state controllers. We pay specific attention to FSCs — describing the extension of the POMDP model to agents with embedded FSCs. Section 3 introduces GAMP, our model-based policy-gradient algorithm. Section 4 develops the model free IState-GPOMDP algorithm. Section 5 develops IOHMM-GPOMDP and Exp-

GPOMDP which both maintain internal belief states. Finally, Section 7 compares the use of these algorithms on small to medium size tasks from the POMDP literature.

2. Partially Observable Markov Decision Processes

2.1 POMDP Definition

Our setting is that of an agent taking actions in a world according to a parameterised policy. The agent seeks to adjust the policy in order to maximise the long-term average reward. Formally, the most natural model for this problem is a partially observable Markov decision process or POMDP. For ease of exposition we consider finite POMDPs,[1] consisting of:

- n_s states $\mathcal{S} = \{1, \ldots, n_s\}$ of the world.
- n_u or actions (or controls) $\mathcal{U} = \{1, \ldots, n_u\}$ available to the policy.
- n_o observations $\mathcal{Y} = \{1, \ldots, n_o\}$.
- A (possibly stochastic) reward $r(i)$ for each state $i \in \mathcal{S}$.

Each action $u \in \mathcal{U}$ determines a stochastic matrix $Q(u) = [q(j|i,u)]_{i=1\ldots n_s, j=1\ldots n_s}$, where $q(j|i,u)$ denotes the probability of making a transition from state $i \in \mathcal{S}$ to state $j \in \mathcal{S}$ given action $u \in \mathcal{U}$. For each state i, an observation $y \in \mathcal{Y}$ is generated independently according to a probability distribution $\nu(i)$ over observations in $\mathcal{Y}$. We denote the probability of observation y by $\nu(y|i)$.

2.2 Reactive Policies

To ensure the average reward is a differentiable function of policy parameters, it is often necessary for the policy to be stochastic. In most cases of practical interest (such as a Gibbs soft-max distribution) the policy space is parameterized so that deterministic policies lie on the boundary of parameter space "at infinity". In this way the agent can approach deterministic policies arbitrarily closely.

Memory-less or *reactive* policies have no internal state and simply choose an action as a function of the current observation y. We denote the probability of choosing action u given observation y and parameters $\theta \in \mathbb{R}^{n_p}$ as $\mu(u|\theta, y)$. With these definitions, a POMDP controlled by a reactive policy as follows:

1. Let $i_0 \in \mathcal{S}$ denote the initial state of the world.
2. At time step t, generate an observation $y_t \in \mathcal{Y}$ with probability $\nu(y_t|i_t)$.
3. Generate an action u_t with probability $\mu(u_t|\theta, y_t)$.
4. Generate a new world state i_{t+1} with probability $q(i_{t+1}|i_t, u_t)$.
5. $t = t + 1$, goto 2.

1. For the most part, generalizations to countably or uncountably infinite POMDPs are reasonably straightforward (Baxter and Bartlett, 2001).

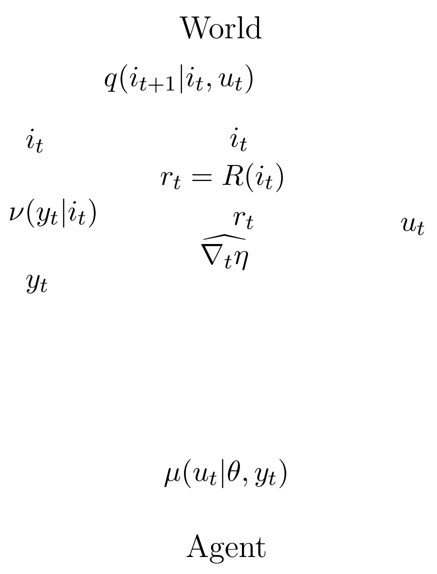

Figure 1: Evolution of a POMDP controller parameterised by θ

Figure 1 illustrates this process. Conceptually, the world issues rewards after each state transition. Rewards may depend on the action and the previous state, that is, $r_t = r(i_t, u_t)$; but without loss of generality we will assume rewards are a function of the updated state $r_t = r(j) = r(i_{t+1})$ so that r_0 is received after the first action is complete.[2]

Our goal is to find $\theta \in \mathbb{R}^{n_p}$ maximising the *long-term average reward*:

$$\eta(\theta) := \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_\theta \left[\sum_{t=1}^{T} r(i_t) \right], \tag{1}$$

where $\mathbb{E}_\theta$ denotes the expectation over all state trajectories $(i_0, i_1, \dots)$ when the policy is $\mu(u|\theta, y)$. Under certain ergodicity and smoothness assumptions to be discussed later, $\eta(\theta)$ is a differentiable function of the policy parameters θ and so lends itself to optimization via gradient-ascent.

2.3 Agents With Internal State

In the presence of observation noise, purely reactive policies are generally suboptimal; instead, the agent must remember features of its history. A simple example is an agent in a symmetric building. If it only receives observations about what is in its line of sight then many places in the building will appear identical. However, if it remembers the last time it saw a landmark, such as the front doors of the building, then it can infer where it is from its memory of how it moved since seeing the landmark.

2. The more general case can be achieved by augmenting the state with information about the last action, implicitly making the reward a function of the last action.

2.3.1 History Traces

The simplest way to add state to the agent is to have the agent remember past observations and actions $\hbar_T = \{(y_0, u_0), (y_1, u_1), \dots, (y_T, u_T)\}$. This is the approach taken by methods such as utile distinction trees McCallum (1996) and prediction suffix trees Ron et al. (1994).

However, explicitly storing $\hbar_T$ results in inefficient memory use because the agent potentially needs to store an infinite amount of history in infinite-horizon settings. Also, the complexity of representing policies in the worst case grows exponentially with the amount of history that needs to be remembered. While we know that worst-case POMDPs are exponential in history length, many problems of practical interest admit far simpler solutions.

2.3.2 Belief State Methods

Instead of explicitly storing history traces, the agent can instead maintain a *belief state* or probability distribution over world-states $i \in \mathcal{S}$. The belief state is a sufficient statistic in the sense that it contains all the information necessary for the agent to act optimally (Åström, 1965). One popular class of POMDP algorithms operate by performing value iteration in the space of belief states. Algorithms of this type were first introduced by Sondik (1971) and further investigated by Kaelbling et al. (1996), Cassandra (1998). See Murphy (2000) for a good review.

For large state spaces it becomes intractable to represent the exact value-function over belief states (Hansen and Feng, 2000). This motivates the use of approximation schemes which include techniques for computing policy values at a limited number of belief states — such as the simplex boundaries for the Most Likely State algorithm (Nourbakhsh et al., 1995) — and interpolating between points (Sutton et al., 2000, Parr and Russell, 1995, Hauskrecht, 1997, Brafman, 1997). Other approximation methods look at factoring the belief state into combinations of *state variables* (Sallans, 2000, Poupart and Boutilier, 2001, Poupart et al., 2001, Boutilier and Poole, 1996).

2.3.3 Finite State Control of POMDPs

An alternative to recording $\hbar_T$ or maintaining belief states is to incorporate a finite state controller into the agent. Let $\mathcal{G} = \{1, \dots, n_g\}$ be the internal states (I-states) that the agent can occupy. The global POMDP state is given by the tuple comprising of the current I-state $g_t \in \mathcal{G}$, and the world state $i_t \in \mathcal{S}$.

The stochastic policy is now a function μ mapping I-states $g \in \mathcal{G}$ and observations $y \in \mathcal{Y}$ into probability distributions over the controls $\mathcal{U}$. We denote the probability under μ of control u by $\mu(u|\theta, g, y)$.

The I-state evolves stochastically as a function of the current observation and a second set of parameters $\phi \in \mathbb{R}^{n_i}$. Specifically, we assume the existence of a function ω such that the probability of making a transition from I-state $g \in \mathcal{G}$ to I-state $h \in \mathcal{G}$ is $\omega(h|\phi, g, y)$. Thus the I-state is updated at each step based on the observation and the current internal state. The evolution of the POMDP is now represented by Figure 2. Many parameterisations of $\omega(h|\phi, g, y)$ and $\mu(u|\theta, h, y)$ are possible though we will restrict ourselves to table lookup and multi-layer perceptrons with distributions generated by the *soft-max* function defined in Section 3.5.1.

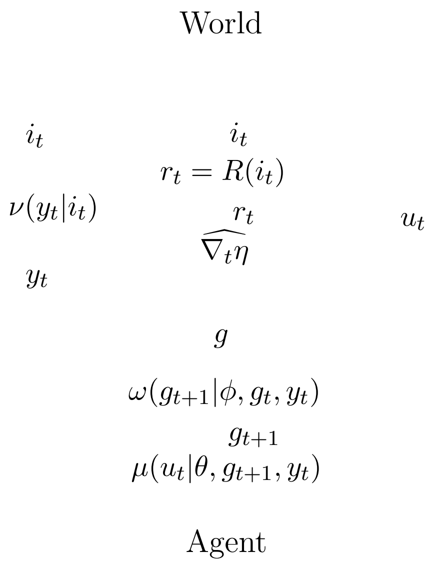

Figure 2: Evolution of a finite state POMDP controller.

The controller learns to use the I-states to remember only what is needed in order to act optimally. This process can be viewed as an automatic quantisation of the belief state space to provide the optimal policy *representable by n_g I-states.* As $n_g \to \infty$ we can represent the optimal policy arbitrarily accurately. Another way to view this process is the direct learning of a *policy graph* (Meuleau et al., 1999b), which is a directed and possibly cyclic graph where each node is labelled with a single action and transitions out of each node are labelled with observations. Policy-gradient methods for FSCs perform a search on the space of policy graphs (Meuleau et al., 1999b), where n_g is the number of policy graph nodes.

It is possible for a large POMDP to be controlled well by a relatively simple policy graph. Consider the Load/Unload problem Peshkin et al. (1999) shown in Figure 3(a). The observations alone do not allow the agent to determine if it should move left or right while it is in the middle `NULL` observation states. However, if the agent remembers whether it last visited the load or unload location, using 1 bit of memory, then it can act correctly. The smallest policy graph is shown in Figure 3(b). This policy graph suffices no matter how many intermediate locations there are between the load and unload locations.

FSCs are equivalent to a number of other approaches including external memory algorithms which use actions to set memory bits where n bits of memory provides 2^n I-states (Peshkin et al., 1999, Peshkin, 2000). Some architectures use recurrent neural networks which are similar to finite state controllers (Lin and Mitchell, 1992) and evolutionary search has been used to learn FSC agents for small POMDPs (Moriarty et al., 1999) and some relatively large POMDPs (Glickman and Sycara, 2001).

2.4 Policy-Gradient Methods

The complexity of finding good policies using value iteration in belief-state space motivates the use of local optimisation procedures such as policy-gradient methods. Unlike the ma-

(a) (b)

Figure 3: 3(a) The load/unload problem with 6 locations. The agent receives a reward of 1 each time it passes through the unload location `U` after having first passed through the load location `L`. In each state the agent has a choice of two actions: either `left` or `right`. 3(b) The policy graph learned for the Load/Unload problem. Each node represents an I-state. The "Left" state is interpreted as: *I have a load so move left,* and the "Right" state as: *I dropped my load so move right.* The dashed transitions are used during learning but not by the final policy.

jority of history and belief state techniques, policy-gradient methods do not attempt to associate values with policies, but instead adjust the policy directly. This is a potentially simpler task since it is often easier to learn *how to act* than it is to learn the *value of acting*. In addition, policy-gradient approaches guarantee convergence to at least a local maximum whereas value-based methods may diverge, especially when using function approximation (Baird and Moore, 1999, Marbach and Tsitsiklis, 1999). However, one reason to prefer value methods is their lower variance in comparison to policy-gradient methods. Intuitively, reduced variance comes from the extra constraints that the Bellman equation imposes on policy values Baird and Moore (1999).

Estimating performance gradients through Monte-Carlo simulation dates back to at least Aleksandrov et al. (1968) and Rubinstein (1969), using the *likelihood-ratio* method. Extensions of the likelihood-ratio method to *regenerative processes* (including Markov Decision Processes) were given by Glynn (1986, 1990), Glynn and L'Ecuyer (1995) and Reiman and Weiss (1986, 1989), and independently for *episodic* POMDPs by Williams (1992), who introduced the REINFORCE algorithm. These methods compute unbiased estimates of the performance gradient, bounding the implications of actions by observing visits to recurrent states $i^* \in \mathcal{S}$.

When the agent cannot observe when visits recurrent states occur, or when visits occur infrequently, we resort to biased estimates of the gradient. This is the approach we adopt from Section 4. As will be discussed, a discounted eligibility trace is used to bound the implications of actions, but the discount factor introduces a bias. Specifically, we generalise the GPOMDP algorithm of Baxter and Bartlett (2001) to agents which incorporate FSCs.

Similar biased estimators were developed by Kimura et al. (1997), Kimura and Kobayashi (1998) and Marbach (1998), which both use reward baselines to lower the variance of the estimates. Other methods reduce variance include *actor-critic* techniques (Sutton and Barto, 1998, Sutton et al., 2000, Konda and Tsitsiklis, 2000). The more general setting of additive control variates has been studied for GPOMDP by Greensmith et al. (2002) and these methods can be applied to reduce the variance of the Monte-Carlo algorithms developed in this paper. Adding internal state to policy-gradient methods exacerbates the variance problem, an issue addressed in Section 4.

2.5 Mathematical Preliminaries

The evolution of world/I-state pairs (i, g) is Markov, with an $n_s n_g \times n_s n_g$ transition probability matrix $P(\theta, \phi)$ whose entries $p(j, h|\theta, \phi, i, g)|_{i,j=1\ldots n_s;g,h=1\ldots n_g}$ are given by

$$p(j, h|\theta, \phi, i, g) = \sum_{y,u} \nu(y|i)\omega(h|\phi, g, y)\mu(u|\theta, h, y)q(j|i, u). \tag{2}$$

We make the following assumptions:

Assumption 1 *Each $P(\theta, \phi)$ has a unique stationary distribution $\pi(\theta, \phi) := [\pi_{1,1}(\theta, \phi), \ldots, \pi_{n_s,n_g}(\theta, \phi)]$ satisfying the* balance equations

$$\pi(\theta, \phi)P(\theta, \phi) = \pi(\theta, \phi). \tag{3}$$

Assumption 2 *We assume the magnitudes of the rewards, $|r(i)|$, are uniformly bounded by $R < \infty$ $\forall i \in \mathcal{S}$.*

Assumption 3 *The derivatives,[5 $\frac{\partial\mu(u|\theta,h,y)}{\partial\theta_k}$ and $\frac{\partial\omega(h|\phi,g,y)}{\partial\phi_k}$ are uniformly bounded by $U < \infty$ and $Q < \infty$ $\forall g, h \in \mathcal{G}$, $u \in \mathcal{U}$, $y \in \mathcal{Y}$, $\theta \in \mathbb{R}^{n_p}$ and $\phi \in \mathbb{R}^{n_i}$.*

Assumption 4 *The ratios*

$$\frac{\left|\frac{\partial\mu(u|\theta,h,y)}{\partial\theta_k}\right|}{\mu(u|\theta, h, y)} \quad \textit{and} \quad \frac{\left|\frac{\partial\omega(h|\phi,g,y)}{\partial\phi_k}\right|}{\omega(h|\phi, g, y)}$$

are uniformly bounded by $D < \infty$ and $B < \infty$ respectively, $\forall g, h \in \mathcal{G}$, $u \in \mathcal{U}$, $y \in \mathcal{Y}$, $\theta \in \mathbb{R}^{n_p}$ and $\phi \in \mathbb{R}^{n_i}$.

Assumption 1 ensures that the Markov chain generated by $P(\theta, \phi)$ has a unique recurrent class, which mainly just makes the statement of the theorems more compact (the non-unique case is an easy generalisation). Assumption 4 is needed because those ratios will be used by the algorithms. Any stochastic controller or I-state function that uses a soft-max distribution based on underlying real parameters will satisfy these conditions, as will many others.

We now extend Equation (1) to internal state, where we seek $\theta \in \mathbb{R}^{n_p}$ and $\phi \in \mathbb{R}^{n_i}$ maximising the long-term average reward:

$$\eta(\theta, \phi) := \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_{\theta,\phi}\left[\sum_{t=1}^{T} r(i_t)\right]. \tag{4}$$

where $\mathbb{E}_{\theta,\phi}$ denotes the expectation over all trajectories $(i_0, g_0), (i_1, g_1), \ldots$, with transitions generated according to $P(\theta, \phi)$. Under assumption 1, $\eta(\theta, \phi)$ is independent of the starting state (i_0, g_0) and is equal to:

$$\eta(\theta, \phi) = \sum_{i=1}^{n_s} \sum_{g=1}^{n_g} \pi_{i,g}(\theta, \phi) r(i) = \pi(\theta, \phi) r, \tag{5}$$

where $r := [r(1,1), \ldots, r(n_s, n_g)]'$ and $r(i,g)$ is defined to be $r(i)$ $\forall g \in \mathcal{G}$.

3. Model Based Policy-Gradient

It is realistic to expect that we may have a model of the world in some applications. For example, manufacturing plants may be reasonably well modelled. Much work has been done on the state identification problem in control theory (Ogata, 1990). If we have a model we expect to be able to produce better gradient estimates with less effort. Thus it is useful to develop policy-gradient techniques which incorporate the use of a model to compute the gradient. In this section we provide one such algorithm which is limited to discrete state spaces, but which is feasible for many thousands of states, an order of magnitude improvement over model-based value-function algorithms which can handle tens to hundreds of states (Geffner and Bonet, 1998).

3.1 Gradient of POMDPs with Internal State

For the purposes of this discussion the model of the POMDP is represented by the global state transition matrix $P(\theta, \phi)$, whose entries are given by (2). This matrix has square dimension of $n_s n_g$ and incorporates our knowledge of the underlying world MDP, the observation hiding process ν, any internal state and the current parameters.

Dropping the dependence on θ, ϕ, we can rewrite (5) as $\eta = \pi r$ so $\nabla\eta = \nabla\pi r$. Differentiating both sides of (3) yields $(\nabla\pi)[I - P] = \pi\nabla P$ which should be understood as $n_p + n_i$ equations, one for each of the θ and ϕ parameters. Combining these derivatives yields

$$\nabla\eta = \pi(\nabla P)[I - P]^{-1}, \tag{6}$$

which involves inverting the singular matrix $[I - P]$. We avoid this problem by conditioning the matrix as follows. Let e denote the $n_s n_g$-dimensional column vector consisting of all 1's so $e\pi$ is the $n_s n_g \times n_s n_g$ matrix with the stationary distribution π in each row. Since

$$\begin{aligned}(\nabla\pi)e &= \sum_{i,g} \nabla\pi_{i,g} \\ &= \nabla \sum_{i,g} \pi_{i,g} \\ &= \nabla 1 \\ &= 0,\end{aligned}$$

we obtain $(\nabla\pi)e\pi = 0$. Thus, we can rewrite (6) as

$$\nabla\eta = \pi\nabla P \left[I - (P - e\pi)\right]^{-1} r. \tag{7}$$

A quick induction argument shows that $[P - e\pi]^n = P^n - e\pi$ which converges to 0 as $n \to \infty$ by Assumption 1. So by a classical matrix theorem, $[I - (P - e\pi)]^{-1}$ is invertible.

3.2 The GAMP Algorithm

Equation (7) can be viewed as a system of $n_s n_g$ linear equations. For brevity let $A = [I - P + e\pi]$, $x = A^{-1}r$ and $\nabla\eta = \pi\nabla P x$. Computing A^{-1} exactly is $O(n_s^3 n_g^3)$ hence intractable for more than a few 100's of states. Approximate methods for solving $Ax = r$ are well studied. For example, we can rewrite (7) as

$$\nabla\eta = \lim_{N\to\infty} \pi \left[\sum_{n=0}^{N} (\nabla P) P^n\right] r, \tag{8}$$

and truncate N to produce an estimate. We can bring the r vector inside the sum and, if N is finite, we can take the ∇P term out of the sum to give the approximation

$$x_N = \sum_{n=0}^{N} P^n r. \tag{9}$$

The sum can be efficiently evaluated by successively computing $v_{n+1} = P v_n$ where $v_0 = r$. Each v_n is accumulated into x. Because this is a series of matrix-vector multiplications we end up with an algorithm which has worst case complexity $O((n_s n_g)^2 N)$. This is a form of Richardson iteration, a simple iterative method for solving systems of linear equations (Kincaid and Cheney, 1991, §4.6). An $O(n^2)$ algorithm is still expensive but the matrix P is usually very sparse since only a small subset of $\mathcal{S}$ tends to have non-zero probability of being reached from some other state. For example, in the robot navigation domain of Cassandra (1998), the move forward action leads to one of at most 3 next states, regardless of the size of the world state. By using sparse matrix data structures and sparse multiplication algorithms we can obtain orders of magnitude speed up depending on the POMDP.

We have also experimented with the GMRES Krylov subspace method for iteratively computing x. This method can be thought of as a more advanced version of Richardson iteration that computes estimates of x_{n+1} based on multiple previous estimates $\{x_0, \dots, x_n\}$ (Greenbaum, 1997, §2)(Ipsen and Meyer, 1998). While GMRES is promising for very large systems, there are complex pre-conditioning issues for obtaining the best accuracy and performance which we have not yet investigated so our experiments use Equation (9) to compute x.

Evaluating π involves computing the leading left eigenvector of P, which is expensive if done exactly. We use the power method (Anton and Rorres, 1991, §9.5) for iteratively computing $\pi_{n+1} = \pi_n P$. The iteration stops when the $\|\pi_{n+1} - \pi_n\|_{L_\infty}$ falls below some threshold. A better but more complex method for computing π, which we have not yet implemented, is the Lanczos method (Greenbaum, 1997, §2.5) which is related to Krylov subspace methods.

A surprisingly expensive operation is evaluating Equation (2) for each element of each $\partial P/\partial\theta_k$, which is in the worst case $O((n_s n_g)^2 n_p n_o n_u)$. Sparsity again helps since $q(j|i,u)$ and $\nu(y|i)$ often evaluate to 0. Noise in transitions and observations decreases sparsity. As the sparsity of the POMDP decreases, the complexity of computing ∇P grows faster than that of computing π and x. Simple tricks such as pre-computing all values of $\nabla\omega(h|\phi, g, y)$ and $\nabla\mu(u|\theta, h, y)$, combining the operations of computing ∇P with multiplication by π and x, and using sparsity, allows systems of $n_g n_s > 20,000$ to be feasibly tackled on today's desktop computers.

The overall procedure is summarised by Algorithm 1, named GAMP for Gradient Approximation for Modelled POMDPs. Lines 2–7 compute P; 8–13 compute x_N using Richardson iteration; 14–18 estimate π using the power method and 19–29 compute $\pi(\nabla P)x_N$. Combining the computation of ∇P with the final step of multiplying $\pi(\nabla P)x_N$ avoids explicitly storing large ∇P matrices. Practical implementations require sparse representations and matrix multiplications. The loops for computing P and ∇P in Algorithm 1 are shown in a simplified form for clarity. They should be constructed to take maximum advantage of the factors in lines 5,23,26 that are often 0.

3.3 Asymptotic Convergence of GAMP

Because P is a stochastic matrix P^N converges exponentially quickly to π. The exact rate is governed by the mixing time τ of the POMDP which we define with the help of the following definitions from Barlett and Baxter (2000).

Definition 1 *The total variation distance between two discrete probability distributions P, Q on a set $\mathcal{S}$ is*

$$d(P,Q) = \sum_{j \in \mathcal{S}} |p(j) - q(j)|.$$

Definition 2 *We say that a stochastic process $\{X_t\}$ is exponentially mixing with time constant τ (τ-mixing for short) if*

$$d(P_i^n, \pi) \leq \exp(-\lfloor \frac{n}{\tau} \rfloor),$$

where P_i^n is the i'th row of P^n.

Intuitively τ can be though of as a measure of complexity of matrix P defined by the POMDP and the agent. The following theorem bounds the error in the GAMP gradient estimates as a function of τ.

Theorem 3 *Let*

$$\widehat{\nabla_N \eta} := \pi(\nabla P) \left[\sum_{n=0}^{N} P^n \right] r,$$

then

$$\|\nabla\eta - \widehat{\nabla_N \eta}\|_\infty \leq BRn_u\tau \frac{\exp(-\lfloor \frac{N}{\tau} \rfloor)}{1 - \exp(-1)}.$$

This theorem is proved in Appendix A.1. Empirical evidence suggests that this is not a tight bound. In particular some effort may show that the factor $n_u\tau$ can be removed. The difficulty of calculating τ for an arbitrary POMDP makes it difficult to use this theorem to establish the required N *a priori*. In practice we check for convergence of x by stopping when the angle $\angle(x_{n+1}, x_n) < \epsilon$.

Algorithm 1 GAMP$(Q, \nu, r, \theta, \phi, \epsilon_x, \epsilon_\pi)$

```
1: Given:
     • State transition probabilities $q(j|i,u) \quad \forall j, i \in \mathcal{S}, u \in \mathcal{U}$.
     • Observation probabilities $\nu(y|i) \quad \forall i \in \mathcal{S}, y \in \mathcal{Y}$.
     • Policy parameters $\theta \in \mathbb{R}^{n_p}$.
     • FSC parameters $\phi \in \mathbb{R}^{n_i}$.
     • Iteration termination thresholds $\epsilon_x, \epsilon_\pi$.
2: for each $((i,g),(j,h))$ do
3:   $p_{(i,g)(j,h)} = 0$
4:   for each $(y,u)$ do
5:     $p_{(i,g)(j,h)} = p_{(i,g)(j,h)} + \nu(y|i)\omega(h|\phi,g,y)\mu(u|\theta,h,y)q(j|i,u)$
6:   end for
7: end for
8: $v = r, x = r, \bar{x} = 0$
9: while $\angle(x,\bar{x}) > \epsilon_x$ do
10:   $\bar{x} = x$
11:   $v = Pv$
12:   $x = x + v$
13: end while
14: $\bar{\pi} = \frac{1}{n_g n_s}, \pi = \bar{\pi}P$
15: while $\max_i |\pi_i - \bar{\pi}_i| > \epsilon_\pi$ do
16:   $\bar{\pi} = \pi$
17:   $\pi = \bar{\pi}P$
18: end while
19: $\Delta = [0]$
20: for each $((i,g),(j,h))$ do
21:   for each $(y,u)$ do
22:     for each $\theta_k \in \theta$ do
23:       $\Delta_k = \Delta_k + \pi_{(i,g)}\nu(y|i)\omega(h|\phi,g,y)\frac{\partial\mu(u|\theta,h,y)}{\partial\theta_k}q(j|i,u)x_{j,h}$
24:     end for
25:     for each $\phi_l \in \phi$ do
26:       $\Delta_l = \Delta_l + \pi_{(i,g)}\nu(y|i)\frac{\partial\omega(h|\phi,g,y)}{\partial\phi_l}\mu(u|\theta,h,y)q(j|i,u)x_{j,h}$
27:     end for
28:   end for
29: end for
30: $\widehat{\nabla_N\eta} = \Delta$
```

Figure 4: Angular error between the GAMP estimate after N iterations and the exact gradient. This plot is based on the Pentagon POMDP with $n_s = 209$ and $n_g = 5$ making P a 1045×1045 element matrix.

3.4 GAMP in Practice

Figure 3.4 demonstrates empirically how quickly the GAMP algorithm converges on an internal-state problem with 1045 states. We computed the exact matrix inversion $[I - P + e\pi]^{-1}$ for the Pentagon problem defined in Section 7.3 with $n_s = 209$, $n_g = 5$. The initial parameters were set such that $\theta_k = \phi_l = 0$, $\forall k, l$. The Pentagon problem allows all observations from all states by adding noise, making the observation stochastic matrices dense. Noise is also added to the state transitions, allowing the agent to not move or move too far, as well as moving correctly. The added noise and internal state make this a good challenge for GAMP. Even the noisy Pentagon transition probabilities are very sparse with only 25,875 of 1,092,025 elements of P having non-zero probabilities.

This experiment was run on an unloaded Pentium II @ 433 MHz. When computing the exact gradient, computing π requires 315 s (wall clock time), computing the matrix inversion requires 10.5 s and computing ∇P requires 36 s. When computing the approximate gradient with $N = 500$, the Richarsdon Iteration inversion is computed in 1.41 s. A larger saving comes from approximating π. For his experiment we used a value of $\epsilon_\pi = 0.0001$. This required 1319 iterations taking 3.50 s instead of 315 s. The angular error in the gradient at $N = 500$ is $0.420°$ taking 11.3% of the time the true gradient requires. The speedup becomes greater as $n_s n_g$ grows. If π is computed exactly the error is reduced by $0.016°$, demonstrating that in this case the majority of the error is introduced by approximating x.

3.5 A Large Multi-Agent Problem

Model based methods for POMDPs have been restricted to at most few hundred states with 10's of observations and actions (Geffner and Bonet, 1998). This section demonstrates that

Figure 5: Plan of the factory floor for the multi-agent problem. The dashed arrows show a possible route traced by the optimal agents.

GAMP can learn the optimal policy for a noisy multi-agent POMDP with 21,632 states, 1024 observations and 16 actions.

The problem is shown in Figure 3.5: a factory floor with 13 grid locations to which 2 robots have access. The robots are identical except that one is given priority in situations where both robots want to move into the same space. They can turn left or right, move 1 position ahead, or wait were they are. One agent learns to move unfinished parts from the left shaded area to the middle area, where the part is processed instantly and then the second agent moves the processed parts from the middle to the right shaded area. The middle processing machine can handle only 1 part at a time, so if the first agent drops off a part at the middle before the second agent has picked up the last part dropped at the middle, the new part is discarded.

The large state space arises from the combined state of the two independent agents plus the global state. Each agent can be loaded or unloaded in 13 states with 4 orientations, giving each agent $2 \times 13 \times 4 = 104$ states. The global state indicates if a part is waiting at the middle processing machine and the state of the 2 agents, giving $2 \times 104^2 = 21,632$ states.

A reward of 1 is received for dropping off a processed part at the right. The agents only need to exit the loading or drop off locations to pick up or drop loads. To receive the maximum reward the agents must cooperate without explicit communication, the actions of the first agent allowing the second agent to receive rewards.

The observations for each agent consist of 4 bits describing whether their path is blocked in each of the 4 neighbouring positions, and a 5th bit describing if the agent is in the uppermost corridor (which is necessary to break the symmetry of the map). The combined observations are 10 bits, or $n_o = 1024$. The actions for each agent are {`move forward, turn left, turn right, wait`}, resulting in a total of $n_u = 16$ actions.

Uncertainty is added with a 10% chance of the agents' action failing, resulting in no movement, and a 10% chance of the agents' sensors completely failing, receiving a "no walls" observation. This problem was designed to be solved by a reactive policy. Section 7 demonstrates GAMP on problems which require memory to solve.

3.5.1 Experimental Protocol

These experiments were run on an unloaded AMD Athlon @ 1.3GHz. GAMP required less than 47 Mbytes of RAM to run this problem. All the experiments in this paper use a conjugate gradient ascent algorithm with an exponential line search (see Appendix B.2). The experiments of this section use the faster value based line search, but all other experiments use the more robust GSEARCH algorithm (Baxter et al., 2001) (see Appendix B.2). Unless stated, all experiments use policies and FSCs parameterised by a tables indexed by (y, b). Each index provides a vector of n_g or n_u real numbers (initialised to 0) for $\omega(h|\phi, g, y)$ and $\mu(u|\theta, h, y)$ respectively. The *soft-max* function is then used to generate distributions from the indexed vector. Computing the gradient of the soft-max function with respect to ϕ and θ gives us $\nabla\omega(h|\phi, g, y)$ and $\nabla\mu(u|\theta, h, y)$ which is needed to compute ∇P.

Definition 4 *Given a vector $\theta \in \mathbb{R}^N$, the probability of event $u \in 1, \ldots, N$ according to the soft-max distribution is*

$$P(u) := \frac{\exp(\theta_u)}{\sum_{u'=1}^{N} \exp(\theta_{u'})}. \tag{10}$$

$$\frac{\partial P(u)}{\partial \theta_k} = P(u)(\chi_u(k) - P(u)),$$

where $\chi_u(k)$ is the indicator function

$$\chi_u(k) := \begin{cases} 1 & \text{if } k = u, \\ 0 & \text{otherwise.} \end{cases}$$

IState-GPOMDP is a model free Monte-Carlo algorithm (described in Section 4) for estimating $\nabla\eta$ of FSC agents. To compare GAMP to a model free algorithm IState-GPOMDP with no memory was run 10 times with different random seeds. A gradient estimation time of 4×10^6 steps, value estimation time of 2×10^6 steps and a discount factor $\beta = 0.99$ was used. We set $\theta_k = 0 \ \forall k$ and a quadratic penalty of $q = 0.0001$ is used to prevent the parameters growing too quickly and entering a sub-optimal maxima (see Appendix B.3).

We attempted to run the exact Incremental Pruning algorithm (Zhang and Liu, 1996) on the Factory problem however the program aborted during the first iteration of dynamic programming after consuming all 256 MB of memory.[3] Storing just one double precision vector of length n_s requires 169 Kb and exact algorithms quickly generate many thousands of vectors for large problems. Exact algorithms based on factored belief states could work well since the problem naturally decomposes into a few *state variables* (Boutilier and Poole, 1996, Hansen and Feng, 2000, Poupart and Boutilier, 2001), however we do not assume that the state variables are known *a priori*.

3.5.2 Results

The agents learn to move in opposing circles around the factory which reduces the chances of collision (shown by the dashed lines in Figure 3.5). They also learn to wait when their

3. We used Anthony Cassandra's pomdp-solve V4.0 code.

Table 1: Results for multi-agent factory setting POMDP. η values are multiplied by 1×10^2. IState-GPOMDP was run 10 times.

Algorithm	*mean* η	*max.* η	*var.*	*secs to* $\eta = 5.0$
GAMP	6.51	6.51	0	1035
IState-GPOMDP	5.85	6.36	1.79×10^{-5}	1286
Hand	6.51			

sensors fail, using the wait action to gather information. Table 1 shows a comparison between IState-GPOMDP with no memory and GAMP for this problem. The last line shows that GAMP has learnt a policy of equivalent value to the best policy we designed by hand.

The most common local maxima occurred when the agents learnt early in training that `forward` is a generally useful action, even when the sensors fail, and the agents failed to unlearn the move forward behaviour in favour of the wait behaviour when their sensors fail. Compared to IState-GPOMDP, GAMP consistently learns a superior policy in less time. The performance of IState-GPOMDP can be improved at the cost of more time.

We might expect the performance advantage of GAMP to be greater considering that GAMP has a lot more information about the problem. An intuition for the modest improvement arises from a nice feature of Monte-Carlo algorithms: their ability to focus search on the relevant parts of state space (Kaelbling et al., 1996). Except in the early stages of training IState-GPOMDP encounters only those states which are encountered by a reasonable agent, effectively learning in a pruned state space. GAMP always considers the entire state space, even those states which may be unreachable given a particular start state.

For this reason GAMP is less forgiving about violations of Assumption 1. For example, when designing the Factory problem above, care must be taken to ensure that impossible situations, such as two robots occupying the same location, have transitions to legal situations even though IState-GPOMDP would not encounter those impossible situations. If the state space grows very large then IState-GPOMDP should out-perform GAMP unless a form of state space pruning is used. For the smaller state spaces in the scenarios of Section 7 GAMP exhibits greater performance improvement over Monte-Carlo methods than the multi-agent factory problem. Section 7 also demonstrates that GAMP can successfully learn policies with memory.

The comparison of convergence times for IState-GPOMDP and GAMP does not account for the time it takes IState-GPOMDP to gather real experience from the world. The ability to train agents without slow and possibly hazardous interactions with the world is a key advantage of model-based algorithms.

3.6 Related Work

Dynamic programming model-based algorithms have been studied extensively (Murphy, 2000). Monte-Carlo policy-gradient algorithms have also been well studied, but little work has examined policy-gradient approaches when given a model. The Linear-Q and SPOVA algorithms (Littman et al., 1995, Parr and Russell, 1995) use gradient ascent to perform parameter updates for function approximators which learn the value of belief states but this work varies from ours because it learns values instead of policies and still simulates a trajectory through the world, updating the belief state using the model at each step. The most similar approach to ours directly computes the gradient of the *discounted sum of rewards* ∇J_β, to learn a finite state controller (Meuleau et al., 1999a).

4. Model Free Policy Gradient

Without knowledge of the POMDP transition and observation models we resort to Monte Carlo like methods for estimating $\nabla\eta$. These methods simulate trajectories through the POMDP, gathering noisy information about the true gradient at each step. Much work has been done in this area for MDPs including the value-based TD(λ) family of algorithms (Sutton and Barto, 1998).

As discussed in 2.3 agents require internal state to act optimally in partially observable environments. Peshkin et al. (1999) and Peshkin (2000) extend REINFORCE to policies with memory which we now extend to the infinite-horizon setting. Recall from Section 2.3.3 that the policy can be augmented with a finite state controller to create an agent with memory, allowing it to potentially remember events infinitely far into the past in order to resolve the hidden state of the POMDP.

The task is now to estimate the gradient of the long term average reward using simulation. The gradient is with respect to the parameters θ and ϕ, $\nabla\eta(\theta,\phi) = \left[\nabla^\theta(\theta,\phi), \nabla^\phi(\theta,\phi)\right]$; where θ parameterises the policy $\mu(u|\theta,h,y)$ and ϕ parameterises the stochastic finite state controller $\omega(h|\phi,g,y)$.

4.1 Estimating the Gradient by Simulation

Algorithm 2 computes estimates $[\Delta_T^\theta, \Delta_T^\phi]$ of an approximation to $\nabla\eta$. At each step an observation y_t is received and the distribution ω is sampled to choose g_{t+1}. The gradient of $\omega(g_{t+1}|\phi, g_t, y_t)$ is added into the trace z_t^ϕ which is discounted by $\beta \in [0,1)$ at each step to give more weight to recent I-state choices. The same process is followed with the new I-state to choose an action u_t from μ. At each step the immediate reward of action u_t is multiplied by the current traces and averaged to form the gradient estimate. The discount factor is necessary to solve the *temporal credit assignment* problem and reflects the assumption that rewards are exponentially more likely to be generated by recent actions. Algorithms such as REINFORCE avoid discount factors by assuming finite-horizon POMDPs, bounding the period in which actions can cause rewards. The following theorem establishes that IState-GPOMDP estimates *an approximation* to the gradient of η, given by (4) which converges to the approximate gradient as the number of estimation steps goes to ∞.

Theorem 5 *Define the discounted reward as*

$$J_\beta(i,g) := \mathbb{E}\left[\sum_{t=0}^{\infty} \beta^t r(i_t, g_t) | i_0 = i, g_0 = g\right], \tag{11}$$

where the expectation is over all world/I-state trajectories. Let $\Delta_T := \left[\Delta_T^\theta, \Delta_T^\phi\right]$ be the estimate produced by IState-GPOMDP after T iterations. Then under Assumptions 1–4, $\lim_{T\to\infty} \Delta_T = \pi(\nabla P)J_\beta$ with probability 1.

This is proved in Appendix A.2.2. The next theorem establishes that as $\beta \to 1$, the approximation $\pi(\nabla P)J_\beta$ is in fact $\nabla\eta$.

Theorem 6 *For P parameterised by FSC parameters ϕ and policy parameters θ*

$$\lim_{\beta\to 1} \pi(\nabla P)J_\beta = \nabla\eta.$$

The proof is the same as the memory-less setting by Baxter and Bartlett (2001). We now have $\Delta_T \xrightarrow{T\to\infty} \pi\nabla P J_\beta \xrightarrow{\beta\to 1} \nabla\eta$, however the variance of Δ_T scales as $1/\left[T(1-\beta)\right]$, reflecting the increasing difficulty of the credit assignment problem. Fortunately, $\widehat{\nabla_\beta\eta}$ is guaranteed to be a good approximation to $\nabla\eta$ provided $1/(1-\beta)$ exceeds the mixing time τ. The proofs are a generalisation from the memory-less GPOMDP algorithm (Baxter and Bartlett, 2001) which is retrieved from IState-GPOMDP by setting $n_g = 1$.

4.2 Finite I-state Error

Finite memory approaches introduce approximation errors when the agent's memory is insufficient to resolve the relevant hidden state. One side effect is that, like reactive policies for POMDPs (Jaakkola et al., 1995), stochastic policies and FSCs may outperform deterministic policy graphs when insufficient I-states are used. This is for the same reason that good reactive policies for POMDPs may be stochastic (Singh et al., 1994): there is insufficient memory to resolve enough uncertainty about the state in order to pick the best action.

Consider Figure 6 which shows a fragment of a policy graph in which a lost agent uses a compass to face north before moving on. As the number of I-states is reduced the policy degrades until for $n_g = 2$ the agent cannot decide between moving forward or turning, representing a stochastic policy.

An interesting result would be to bound the error in the long-term reward between an arbitrary policy and a policy constructed with n_b I-states. Theorem 3 in Sondik (1978) takes a step toward such a result by bounding the error between the value of a policy and the value of an approximate policy with a finite number of hyperplanes (equivalent to I-states) in the cost function. Unfortunately, this bound is in terms of the number of epochs of the value-iteration algorithm rather than the number of hyperplanes. However, this result may serve as a starting point.

5. Internal Belief States

So far we have assumed that the internal memory of the system consists of a finite state machine and that the agent samples an I-state at each time step according to ω. Multiple steps describe a trajectory through the state space of the finite state machine. The key idea of this section is that we do not need to sample I-states. The finite state controller is completely observable, allowing us to compute expectations over all possible I-state trajectories. At each time step we can update the probability of occupying each I-state and use this *belief* over I-states as our controller memory.

A related advantage is that belief states are more informative than sampled I-states. It requires $\log_2 n_g$ bits to encode a sampled I-state, but it requires pn_g bits to encode n_g

Algorithm 2 IState-GPOMDP

```
1: Given:
    • Parameterised class of FSCs $\{\omega(h|\phi, g, y) : \phi \in \mathbb{R}^{n_i}\}; g \in \mathcal{G}; h \in \mathcal{G}; y \in \mathcal{Y}$.
    • Parameterised class of randomised policies $\{\mu(u|\theta, h, y) : \theta \in \mathbb{R}^{n_p}\}; h \in \mathcal{G}; y \in \mathcal{Y}$.
    • POMDP which when controlled by $\mu(u|\theta, h, y)$ and $\omega(h|\phi, g, y)$ generates a parameterised class of Markov chains satisfying Assumption 1.
    • Discount $\beta \in [0, 1)$.
    • Arbitrary initial state $i_0$ and I-state $g_0$.
    • Observation sequence $y_0, y_1, \dots$ generated by the POMDP with $u_0, u_1, \dots$ generated randomly according to $\mu$, where $g_{t+1}$ is generated randomly according to $\omega$.
    • Bounded reward sequence $r(i_0), r(i_1), \dots$, where $i_0, i_1, \dots$ is the (hidden) sequence of states of the environment.
2: Set $z_0^\theta = 0, z_0^\phi = 0, \Delta_0^\theta = 0$ and $\Delta^\phi = 0$ ($z_0^\theta, \Delta_0^\theta \in \mathbb{R}^{n_p}, z_0^\phi, \Delta_0^\phi \in \mathbb{R}^{n_i}$).
3: while $t < T$ do
4:     Observe $y_t$ from the world.
5:     Choose $g_{t+1}$ from $\omega$.
6:     Choose $u_t$ from $\mu$.
7:     $z_{t+1}^\phi = \beta z_t^\phi + \frac{\nabla\omega(g_{t+1}|\phi, g_t, y_t)}{\omega(g_{t+1}|\phi, g_t, y_t)}$
8:     $z_{t+1}^\theta = \beta z_t^\theta + \frac{\nabla\mu(u_t|\theta, g_{t+1}, y_t)}{\mu(u_t|\theta, g_{t+1}, y_t)}$
9:     $\Delta_{t+1}^\theta = \Delta_t^\theta + \frac{1}{t+1}\left[r(i_{t+1}) z_{t+1}^\theta - \Delta_t^\theta\right]$
10:    $\Delta_{t+1}^\phi = \Delta_t^\phi + \frac{1}{t+1}\left[r(i_{t+1}) z_{t+1}^\phi - \Delta_t^\phi\right]$
11:    Issue action $u_t$.
12:    $t++$.
13: end while
```

(a) (b) (c)

Figure 6: Figure 6(a) shows a 4 I-state policy-graph fragment for a lost Agent which must move north. If the number of I-states is reduced to 3 (Figure 6(b)) then the policy is still deterministic but degraded because it may turn left 3 times instead of right once. In Figure 6(c) the agent cannot determine if it is facing north when it is in the upper I-state, thus must sometimes should forward and sometimes turn.

floating point numbers of precision p. The loss of information incurred when sampling I-states amounts to a form of internal partial observability because we throw away information about how likely the I-state trajectory is. This section covers two methods for learning when internal memory is a belief vector.

5.1 Hidden Markov Models for Policy Improvement

Because partially observability hides the true state of the POMDP we can use existing algorithms which reveal hidden state. Hidden Markov models (HMMs) are a candidate which have previously been used in conjunction with value based RL. HMMs are a generative model, that is, they model stochastic sequences without being driven by the available data. Bengio describes an extension to HMMs, called Input/Output HMMs, which allow them to model one sequence while being driven by a related sequence (Bengio and Frasconi, 1996, McCallum et al., 2000). The idea is that the driving sequence contains information useful to prediction.

IOHMMs have been used to solve small POMDPs where actions drive the state transitions and the IOHMM predicts the observations (Chrisman, 1992, McCallum, 1996). HMMs can also be layered at different time scales for hierarchical reinforcement learning (Theocharous et al., 2000). An alternative scheme is to drive state transitions with observations and use them to generate actions (Shelton, 2001c,a).

5.1.1 Predicting Rewards

A problem with IOHMM methods is that estimation of the hidden state ignores the most useful indicator of policy performance: the reward. Predicting rewards reveals the hidden state relevant to predicting policy performance, and which is consequently relevant to choosing actions that lead to high reward.

To include the reward we propose to drive transition probabilities by observations — and optionally actions — and then use the IOHMM to predict rewards. The idea is that the hidden state revealed by predicting rewards is the most relevant to maximising η. The *forward probability* calculation in the estimation phase of IOHMM parameter updates computes the probability of I-state occupation given all previous observations and rewards, denoted α_t. This quantity is equivalent to the I-state belief and is used in $\mu(u|\theta, \alpha, y)$ to compute an action distribution. Because $\alpha_t \in [0, 1]^{n_g}$ we take the function approximation approach for parameterising $\mu(u|\theta, \alpha, y)$. This paper uses linear controllers.

5.1.2 The IOHMM-GPOMDP Algorithm

Algorithm 3 presents one way of using IOHMMs to predict rewards. The IOHMM-GPOMDP algorithm interleaves IOHMM re-estimation phases and memory-less IState-GPOMDP phases. The IOHMM phase re-estimates the IOHMM transition and output probabilities using a single trajectory of observations (and possibly actions) of length T_h from the POMDP. The IState-GPOMDP phase updates the I-state belief at each time using the previous action's reward and the current observation. The updated I-state belief is used to compute an action distribution.

We have deliberately left the training details of the IOHMM in line 7 of Algorithm 3 unspecified because there are many reasonable approaches that can be taken. The novel contribution of the IOHMM-GPOMDP algorithm is to use IOHMMs for predicting rewards. Our IOHMM implementation is based upon n_g states where n_g is fixed *a priori*. Methods for automatically determining n_g are discussed by Chrisman (1992), McCallum (1996). Rewards can take a wide range of values between $(R, -R)$ thus continuous emission distributions such as mixtures of Gaussians per state (Rabiner, 1989) are sensible. Using a single Gaussian per state to model rewards is equivalent to the assumption that each state models only a single reward value. If there are only a small number of instantaneous reward values then discrete output IOHMMs can also be used. As suggested by Rabiner (1989) we found it important to use sensible initialisations of the emission distributions in order to obtain IOHMM convergence to the global maxima. Sensible initialisations are often easy to devise with the assumption that each IOHMM state models a particular POMDP reward state.

5.1.3 Convergence of the IOHMM-GPOMDP Algorithm

Algorithm 3 always runs IOHMM training to convergence. This allows us to state that once the memory-less GPOMDP algorithm estimates the gradient to be close to zero, both the ϕ and θ parameters have converged since neither set of parameters will change.

Unfortunately the re-estimation of the IOHMM parameters can lead to an arbitrarily large decrease in the performance of the policy. Consider a world with a straight road 5 sections long and assume the optimal policy is to move from on end to the other (essentially the Load/Unload problem of Section 7.1). Initially both the IOHMM and $\mu(u|\theta, \alpha, y)$ are initialised with random parameters. The following can occur:

Algorithm 3 IOHMM-GPOMDP

```
1: Given:
     • Parameterised class of randomised policies $\{\mu(u|\theta,\alpha,y) : \theta \in \mathbb{R}^{n_p}\}$ ; $\alpha \in [0,1]$; $y \in \mathcal{Y}$.
     • An IOHMM with $n_g$ states, inputs $y_t \in \mathcal{Y}$, outputs $r_t \in [-R,R]$ and parameterised by $\phi = [\phi_i, \phi_o]$, the IOHMM state transition and output parameters respectively.
     • POMDP which when controlled by $\mu(u|\theta,\alpha,y)$ generates a parameterised class of Markov chains satisfying Assumption 1.
     • Discount $\beta \in [0,1)$.
     • Arbitrary initial state $i_0$ and internal belief $\alpha_0$.
     • $[-R,R]$ bounded reward sequence $r(i_0), r(i_1), \dots$, where $i_0, i_1, \dots$ is the hidden sequence of states of the environment.
     • Observation sequence $y_0, y_1, \dots$ generated by the POMDP with $u_0, u_1, \dots$ generated randomly according to $\mu(u_t|\theta,\alpha_{t+1},y_t)$, where $\alpha_{t+1}$ is the forward probability vector of the IOHMM after conditioning on *output* $r_{t-1}$ and *input* observation $y_t$.
     • Step size $\gamma > 0$.
2: Set $z_0 = 0$, and $\Delta = 0$ $(z_0, \Delta_0 \in \mathbb{R}^{n_p})$.
3: Set $\phi_i$ to uniform initial transitions.
4: Set $\phi_o$ to appropriate initial output probabilities.
5: while $\|\Delta_{T_g}\| > \epsilon$ do
6:    Execute policy for $T_h$ steps, $\hbar = \{(y_0, r_0), \dots, (y_{T_h-1}, r_{T_h-1})\}$
7:    train_IOHMM_to_convergence($\phi$ ,$\hbar$)
8:    while $t < T_g$ do
9:       Observe $y_t$ from the world
10:      Observe $r_{t-1}$ from the world
11:      $\alpha_t(j) = \sum_{i=1}^{n_g} \alpha_{t-1}(i) P(r_{t-1}|\phi_o, i) P(j|\phi_i, i, y_t)$
12:      Choose $u_t$ from $\mu(u_t|\theta, \alpha_t, y_t)$
13:      $z_{t+1} = \beta z_t + \frac{\nabla\bar{\mu}(u_t|\theta,\phi,y_t)}{\bar{\mu}(u_t|\theta,\phi,y_t)}$
14:      $\Delta_{t+1} = \Delta_t + \frac{1}{t+1}\left[r(i_{t+1}) z_{t+1} - \Delta_t\right]$
15:      $t++$
16:   end while
17:   $\theta \leftarrow \theta + \gamma \Delta_{T_g}$
18: end while
```

1. With random $\mu(u|\theta, \alpha, y)$ the IOHMM learns that rewards are more likely when `move left` actions occur after the "`L`" observation, or `move right` actions occur after the "`U`" observation.

2. μ becomes a good policy based on the current IOHMM.

3. The change in policy parameters θ, changes the extrema of the IOHMM. Consequently the IOHMM can move into a different local maxima based on the sample $\hbar$ gathered using the old ϕ. The IOHMM may learn that a reward usually occurs every 5 steps.

4. The policy represented by θ is no longer optimal under the IOHMM represented by the new ϕ. The reward may drop to a level worse than the uniform random policy.

5. The policy is no longer near optimal and rewards no longer occur every 5 steps, thus the IOHMM must re-learn the concept it learnt in the first step.

Thus, allowing the IOHMM and $\mu(u|\theta, \alpha, y)$ to bootstrap from each other can result in learning cycles which do not converge. If, during Step 4, GPOMDP quickly learns a new policy that is good under the new ϕ, then we may achieve convergence. However we do not usually want GPOMDP to completely converge in one iteration because it will generally end up in a pool local minimum. To avoid this we penalise large values of θ (see Appendix B.3).

5.1.4 Drawbacks of the IOHMM-GPOMDP Algorithm

We have just observed IOHMM-GPOMDP may fail to converge. A second drawback is that being able to successfully predict rewards does not necessarily lead to revealing the hidden state necessary for optimal policies. Consider the deterministic POMDP shown in Figure 7(a). Solid lines are deterministic transitions. Dotted lines are followed with probability 0.5. Actions only impact rewards when the current observation is `a`. To chose the optimal action we must recall one past observation. If the previous observation was `d`, then execute action `u2`, otherwise execute action `u1`. This is a simple policy requiring $n_g = 2$ and it is easily learnt by the IState-GPOMDP algorithm of Section 2. Now consider the task of learning an IOHMM to predict rewards. Figure 7(b) shows an optimal IOHMM with $n_g = 3$. This IOHMM waits for observation `c` or `b` while predicting 0 reward. Observation `c` always implies a reward of -1 after the next action and `b` always implies a reward of 1 after the next action. This IOHMM successfully reveals the hidden state associated with observation `r` because the IOHMM forward probability will indicate which of the two `r` world states we are in. However it does not allow us to implement an optimal policy because we need to know whether to issue action `u1` or `u2` before seeing observations `a` or `b`. Adding states to the IOHMM does not help because the IOHMM is not required to remember the critical observations `d` and `e` in order to optimise its performance.

These two drawbacks cause poor performance of IOHMM-GPOMDP on non-trivial problems, as we will observe in Section 7.

5.2 The Exp-GPOMDP Algorithm

One reason for the high variance of existing policy gradient methods is the noise introduced through estimating the gradient by simulating a trajectory through the environment state

(a) (b)

Figure 7: 7(a) A POMDP with $\mathcal{U} = \{\texttt{u1, u2}\}$ and $\mathcal{Y} = \{\texttt{a,b,c,d,e,r}\}$. The states with observation r, have non-zero rewards shown. This POMDP can always be solved by IState-GPOMDP with $n_g = 2$ but will not be solved by the *optimal* IOHMM shown in Figure 7(b), or one with more or less states.

space.[4] Thus far we have viewed internal states as augmenting the global state space, resulting in the method of sampling both world states and I-states. However, we can do better than simply sampling one long I-state trajectory. Recall the reasons for introducing simulation in the first place:

1. The POMDP model may not be available;
2. dynamic programming becomes infeasible for large n_s.

Since we always know the current I-state transition model and the number of I-states is typically small compared to the world state space, the main reasons for using simulation do not apply. In short, we can use the current I-state FSC model to compute the gradient equivalent to that we would compute if we could sample *all possible* I-state trajectories. Equivalently, we are taking the expectation of the IState-GPOMDP gradient across all possible I-state trajectories. Since sampling I-state trajectories exacerbates the high variance problem, we expect that computing the expectation over all possible trajectories will reduce the variance, reducing the number of simulation steps required.

The algorithm modifies the IState-GPOMDP view by considering the discrete I-state update process to be a hidden component of the policy μ. The parameters ϕ of ω are now embedded in μ. At the top level the new algorithm is identical to memory-less IState-GPOMDP. The change is in the way we compute $\mu(u|\theta, \phi, y)$ and in the way we represent the I-state: as a belief vector of I-state occupation probabilities. We denote the belief vector α, equivalent to the forward probability of an IOHMM.

4. The GAMP algorithm of Section 3.2 is an exception since it avoids simulating trajectories by direct gradient estimation from the POMDP model.

Definition 7 *Let $\alpha_t(g|\phi, \hbar_t)$ be the probability that $g_t = g$ given the current parameters and the observation/action history $\hbar_t$.*

The new I-state update is simply

$$\alpha_{t+1}(h|\phi, \hbar_{t+1}) = \sum_{g \in \mathcal{G}} \alpha_t(g|\phi, \hbar_t)\omega(h|\phi, g, y). \tag{12}$$

If the initial world state of the system is deterministic then we make the I-state belief deterministic, setting $\alpha_0(g) = 1$ for arbitrary g and $0 \ \forall g' \neq g$. If we have no prior information about the initial state we choose $\alpha_0(g) = 1/n_g, \ \forall g$. Now we write down the new form of the policy which takes the expectation of μ over internal states

$$\bar{\mu}(u|\theta, \phi, y) = \sum_{g \in \mathcal{G}} \alpha_{t+1}(g|\phi, \hbar_{t+1})\mu(u|\theta, g, y). \tag{13}$$

Alternatively, $\bar{\mu}$ could be any bounded differentiable function of θ, ϕ, and y which incorporates α. We can view these equations as representing a form of IOHMM, where $\bar{\mu}$ is the probability of emitting symbol (action) u given all previous observations. To finish the algorithm we need to compute $\nabla\bar{\mu}$ and use it in the IState-GPOMDP algorithm in place of $\nabla\mu$. This is straight forward after noting that $\nabla\alpha_{t+1}$ can be updated recursively from $\nabla\alpha_t$ and computing

$$\begin{aligned}
\nabla^{\theta}\bar{\mu} &= \sum_{g \in \mathcal{G}} \alpha_{t+1}(g|\phi, \hbar_{t+1})\nabla\mu(u|\theta, g, y), \\
\nabla^{\phi}\bar{\mu} &= \sum_{g \in \mathcal{G}} \left(\nabla\alpha_{t+1}(g|\phi, \hbar_{t+1})\right)\mu(u|\theta, g, y).
\end{aligned}$$

The algorithm, named Exp-GPOMDP, is shown in Algorithm 4.

The combination of (12) and (13) imply that each simulated step will now have a complexity of at least $O(n_g(n_u + n_g))$ compared to the IState-GPOMDP complexity of $O(n_u + n_g)$ per step. Experimental results of Section 7 show that while the wall clock time of Exp-GPOMDP may be greater than IState-GPOMDP, the former requires fewer simulation steps, which is desirable when world interactions are expensive. Finally, we note that Exp-GPOMDP has the same convergence to local maxima guarantees as IState-GPOMDP and the same bias/variance tradeoff for choosing β.

5.2.1 Related Work

Using a recursively updated forward probability variable α to compute an expectation over all possible paths through a state lattice is an important aspect of hidden Markov Model training (Poritz, 1988). By viewing state transitions as being driven by the observations, and viewing actions as symbols generated by the HMM, the algorithm above looks like a method of training Input/Output HMMs without using the backward probability component of HMM training. Dropping the backward component allows infinite-horizon problems to be tackled. If we restrict ourselves to finite-horizon tasks then it makes sense to perform the full HMM style update, updating the gradient for each control step based on *future* as well as past experience. This is exactly what the work of Shelton (2001b,c) does when incorporating internal state into REINFORCE algorithm. Our work can be viewed as modifying theirs to cope with non-episodic tasks.

Algorithm 4 Exp-GPOMDP

```
1: Given:
     • Same requirements as IState-GPOMDP (Algorithm 2).
2: Set $z_0 = 0$, and $\Delta = 0$ ($z_0, \Delta_0 \in \mathbb{R}^{n_p+n_i}$).
3: for each internal state $g$ do
4:    Set $\alpha_0(g) = 1/n_g$ and $\nabla\alpha_0(g) = 0$.
5: end for
6: while $t < T$ do
7:    Observe $y_t$ from the world.
8:    for each internal state $h$ do
9:       $\alpha_{t+1}(h|\phi, \hbar_{t+1}) = \sum_{g\in\mathcal{G}} \alpha_t(g|\phi, \hbar_t)\omega(h|\phi, y_t, g)$.
10:      $\nabla\alpha_{t+1}(h|\phi, \hbar_{t+1}) = \sum_{g\in\mathcal{G}} \nabla\alpha_t(g|\phi, \hbar_t)\omega(h|\phi, y_t, g) + \alpha_t(g|\phi, \hbar_t)\nabla\omega(h|\phi, y_t, g)$.
11:   end for
12:   Choose $u_t$ from $\bar{\mu}(u_t|\theta, \phi, y_t) = \sum_{g\in\mathcal{G}} \alpha_{t+1}(g|\phi, \hbar_t)\mu(u|\theta, h, y)$.
13:   $z_{t+1} = \beta z_t + \frac{\nabla\bar{\mu}(u_t|\theta,\phi,y_t)}{\bar{\mu}(u_t|\theta,\phi,y_t)}$
14:   $\Delta_{t+1} = \Delta_t + \frac{1}{t+1}[r(i_{t+1})z_{t+1} - \Delta_t]$
15:   Issue action $u_t$.
16:   $t{+}{+}$.
17: end while
```

6. Small FSC Gradients

We observed that policy-gradient FSC methods initialised with *small random parameter values* failed to learn to use the I-states for non-trivial problems due to small gradients. Increasing $\max_l \phi_l$ (that is, the range of the random number generator), helps somewhat but increasing this value too much results in the agent starting near a poor local maximum. This problem arises from having a collection of I-states which are initially undifferentiated. IState-GPOMDP fails to prefer one I-state over another for recording important events.

If a policy's action probabilities $\mu(u|\theta, h, y)$ are the same for each I-state g, the gradient of the average reward with respect to the policy's I-state parameters will be zero. Hence policies whose starting distributions are close to uniform will be close to a point of zero gradient with respect to the internal-state parameters, and will tend to exhibit poor behaviour in gradient-based optimisation.

In addition, if the I-state transitions are close to uniform then the conditional probability over I-state trajectories given most observation/action trajectories will be close to uniform. Thus, it will be difficult for the controller to determine which I-state transitions to adjust in order to maximise reward. The following theorem formalises this argument.

Theorem 8 *If we choose θ and ϕ such that $\omega(h|\phi, g, y) = \omega(h|\phi, g', y)\ \forall g, g', y$ and $\mu(u|\theta, h, y) = \mu(u|\theta, h', y)\ \forall h, h', y$ then $\nabla^{\phi}\eta = 0$.*

This theorem is proved in Appendix A.3. Even if the conditions of the theorem are met when we begin training, they may be violated by updates to the parameters θ during training, allowing a useful finite state controller to be learnt. Appendix A.3 also analyses this phenomenon, establishing an additional condition for perpetual failure learn a finite state controller, despite changes to θ during learning. The additional condition for lookup tables is simply that $\omega(h|\phi, g, y) = 1/n_g \quad \forall g, h, y$.

The same problem has been observed in the setting of learning value functions when the policy can set memory bits (Lanzi, 2000). Multiple trajectories through the memory states have the same reward termed *aliasing on the payoffs*. A solution was hinted at by Meuleau et al. (1999a) when it was noted that finite state controllers were difficult to learn using Monte Carlo approaches unless strong structural constraints were imposed.

6.1 Sparse FSCs

To overcome the problem of random internal state trajectories, we use a modified I-state representation: a large sparse I-state FSC where all I-states have out-degree $k \ll n_g$. The future I-states that can be reached from the current I-state must depend on the current observation. This trick imposes the necessary constraints on the class of learnable FSCs without requiring domain knowledge. Each observation/action trajectory will generate minimally overlapping distributions of I-state trajectories. Intuitively, this allows the correlation of high-reward observation/action trajectories to a set of I-state trajectories.

The complexity of each IState-GPOMDP step now has complexity $O(n_u k)$, allowing the use of many I-states provided k is small. The complexity of Exp-GPOMDP grows with $O(n_g(n_u + k))$, restricting the number of I-states it is practical to use. For both algorithms increasing n_g results in more parameters, requiring more simulation steps during early training.

Setting $k = 1$ is an interesting case which forces a single I-state trajectory for each observation/action trajectory, however I-state trajectories may not be unique, or may merge, causing the FSC to "forget" previous observations. If we use $k = 1$ we need large $n_g$to avoid merging I-state trajectories. This approach is valid for finite-horizon tasks and is equivalent to a finite memory controller.

6.2 Empirical Effects of Sparse FSCs

Theorem 11 shows that the gradient of the stochastic finite state controller is 0 when the transitions probabilities are independent of the internal state. This includes the apparently sensible choice of initial controller which sets all transitions to equal probability (equivalent to $\theta_k = \phi_l = 0 \ \forall k, l$) for lookup tables and many other sensible parameterisations. A natural question is what happens to the gradient as we approach the uniform case. Figure 8 provides an empirical answer. We see that the gradient estimated by IState-GPOMDP with 4 I-states on the Load/Unload problem of Section 7.1 smoothly increases as we diverge from uniform distributions for all algorithm variants. The dense ω shows results for the worst case scenario fully-connected FSC. The rightmost point represents $\theta_k = \phi_l = 0 \ \forall k, l$. The sparse ω line shows the results for $k = 2$. Det μ shows the effect of setting μ *a priori* to be a deterministic function of (y, g). This is an alternative way to break the zero gradient conditions but requires stronger assumptions about the best FSC policy. The IOHMM line shows the magnitude of the θ gradients, computed after one IOHMM parameter estimation phase. The increased gradient magnitude shows the benefit of computing gradients for smaller set of parameters (the IOHMM estimates the remaining parameters). The dive in

Figure 8: Degradation of the magnitude of the gradient for IState-GPOMDP as FSC distributions and action distributions approach uniformity. The lower lines have points labelled with the range of the random initial values for parameters. Results are averaged over 100 runs.

gradient for $\max_l \phi_l = \max_k \theta_k > 2$ shows the effect of pushing the initial parameters into a local maxima. The sparse ω line shows that we can choose parameters which generate near uniform distributions provided we use the sparse FSC trick.

The sparsity introduces an extra parameter k, the out-degree of each I-state. Figure 6.2 shows how the magnitude of the gradient varies as k is increased for the Load/Unload problem to be described in Section 7.1. This problem is a useful benchmark because only 1 bit of internal state is required to formulate the best policy; though in this experiment we set $n_g = 10$ to demonstrate the effects of changing k. The top line represents $\|\nabla\eta\|$ and the lower line is $\|\Delta_T\|$ estimated using Exp-GPOMDP with an estimation time of $T = 10^5$ steps and $\beta = 0.8$. The initial parameters are set to 0 so that all generated distributions are initially uniform. As k is increased the gradient magnitude drops until at $k = 10$ we have a dense FSC and $\|\nabla\eta\|$ is within machine tolerance of 0 and $\|\Delta_T\| = 3.13 \times 10^{-7}$ which indicates the level of noise in the estimate since the true gradient is 0. The strongest gradient is at $k = 1$ but at least for infinite-horizon tasks this choice is unsuitable because the agent needs to learn useful loops in the FSC which is impossible if it has no choice of I-state transitions. Thus $k = 2$ or $k = 3$ are reasonable choices for k.

Introducing sparsity restricts the class of policies hence generating purely random FSCs may create an FSC which cannot represent a good n_g I-state policy. It is useful to generate sparse FSCs using heuristics such as: "I-states should always be able to make self-transitions," representing the situation where no additional information needs to be stored at that step. Domain knowledge can also be encoded in the choice of initial FSC.

Figure 9: Effect of increasing the connectivity for the Load/Unload problem with $n_g = 10$. Results are averaged over 30 runs.

Section 7.2 demonstrates of the effect of sparse transitions on the Heaven-Hell problem, for which we could not achieve convergence without using sparse transitions.

7. Empirical Comparisons

This section examines the relative performance of the four algorithms discussed in this paper on three POMDPs from the literature. We begin with the simple Load/Unload problem, move on to the small but challenging Heaven/Hell problem and finish with a medium size noisy robot navigation problem. For all problems in this section the initial parameters are set to 0.

7.1 Load/Unload

7.1.1 Problem Description

The Load/Unload problem of Peshkin et al. (1999) is described in Figure 3(a). It is a simple POMDP which requires internal state to solve. There is no noise added to the problem and we require only $n_g = 2$ to learn the best finite state controller represented by the policy graph in Figure 3(b).

7.1.2 Experimental Protocol

With the exception of IOHMM-GPOMDP, all the algorithms in this paper will reliably solve the simple Load/Unload problem given sufficient resources. We set the number of gradient estimation steps to be 5000, deliberately under estimating the steps required in order to gauge the relative performance of the algorithms. No quadratic penalty was used for these runs. A discount factor of $\beta = 0.8$ is sufficient for this simple problem. All runs used $n_g = 4$ for a total of $4 \times 10 = 40$ states. We did not use the minimum possible $n_g = 2$ because this value is too small to allow a sparse FSC with $k > 1$.

Table 2: Results over 100 runs on the Load/Unload scenario with 5 positions. η values are multiplied by 10.

Algorithm	*mean* η	*max.* η	*var.* $\times 10^3$	*sec. to* 2.0
GAMP dense	0.500	0.500	0	— (0)
GAMP $k = 2$	2.39	2.50	1.16	0.22 (96)
IState-GPOMDP dense	0.540	2.48	0.383	2.75 (1)
IState-GPOMDP $k = 2$	1.15	2.50	7.86	2.05 (31)
Exp-GPOMDP dense	0.521	0.571	6.05×10^{-3}	— (0)
Exp-GPOMDP $k = 2$	2.18	2.50	3.40	9.75 (82)
IOHMM-GPOMDP	1.41	2.50	6.96	5.53 (46)
Inc. Prune.	2.50	2.50	0	3.27 (100)

For the IOHMM-GPOMDP runs we took samples of 1000 steps in order to re-estimate the IOHMM parameters. The Load/Unload IOHMM indexes state transitions distributions by the current observation and IOHMM state. In order to achieve convergence the IOHMM had to distinguish between rewards for loading and rewards for unloading. To do this we used a reward of 1.0 for loading and 2.0 for unloading. This allowed us to use discrete emission distributions with 3 symbols: $r_t = 0$, $r_t = 1$ and $r_t = 2$. This does not change the optimal policy and the quoted results have been adjusted back to the normal reward schedule.

The Incremental Pruning algorithm results are provided to allow comparison with exact methods. We used Anthony Cassandra's `pomdp-solve` V4.0 source code which performs value-iteration, learning value functions represented by convex piecewise linear functions (Cassandra, 1998). This code outputs a policy graph which can be loaded by our code to obtain results for comparison.

All algorithms except IOHMM-GPOMDP and Incremental Pruning used lookup tables to represent $\omega(h|\phi, g, y)$ and $\mu(u|\theta, h, y)$. IOHMM-GPOMDP used a linear controller μ due to the real valued inputs. As with all parameterisations used in the paper, the soft-max function (10) is used to generate output distributions. The total number of parameters in the table case is 56 for dense FSCs and 32 for sparse FSCs with $k = 2$. IOHMM-GPOMDP used 64 parameters.

7.1.3 Results

Table 2 summarises the results for this problem on all the algorithms described in this paper, using both sparse FSCs and dense FSCs. The rightmost column represents the mean time in seconds taken to converge to a value of η which represents a reasonable policy. Only those runs which achieve the cut-off value are included in the figures for time to convergence. The number of runs which reach the cut-off is written in brackets next to the convergence time. This allows a comparison of the running times of different algorithms which may not always achieve a good policy.

There are several interesting results in Table 2. The first is the complete failure of GAMP to learn when a dense initial FSC is used. This is an expected consequence of Theorem 8 which tells us that the gradient of the ϕ parameters is always 0 in this case. As expected, the Exp-GPOMDP algorithm also fails in this case but both algorithms do well when we use sparse FSCs. The IState-GPOMDP algorithm with a dense FSC occasionally succeeds, which is simply due to noise in the gradient estimates moving the parameters in such a way that Theorem 8 no longer applies. This does not happen for GAMP and Exp-GPOMDP due to their lower variance.

The GAMP algorithm performs consistently well for sparse FSCs and is the fastest algorithm. As expected, the sparse version of Exp-GPOMDP performs significantly better than the sparse IState-GPOMDP algorithm.

We would not get the IOHMM-GPOMDP algorithm to converge reliably, and although it outperforms the IState-GPOMDP algorithm in this experiment, simply increasing the gradient estimation time will allow IState-GPOMDP to converge more reliably than IOHMM-GPOMDP. This is consistent with the problems identified in Section 5.1.4. Figure 10 shows how quickly the three simulation based algorithms converge. The Exp-GPOMDP algorithm clearly outperforms the other two. According to a single-tailed t-test the IOHMM-GPOMDP algorithm is statistically better than IState-GPOMDP with a confidence of 95%. The Exp-GPOMDP result is significantly better than IState-GPOMDP with a 99% confidence interval.

Unsurprisingly the exact Incremental Pruning algorithm works extremely well on this small problem, always obtaining the optimal policy. Using VAPS in an episodic setting of the Load/Unload problem achieved convergence within around 20,000 steps Peshkin et al. (1999). Their version of the problem provided information about which position the agent was in on the road, rather than the `null` observations we give for non-end points. The episodic constraint also makes the problem easier.

7.2 Heaven/Hell

7.2.1 Problem Description

The Heaven-Hell problem of Geffner and Bonet (1998), Thrun (2000) is shown in Figure 11. The agent starts in either position shown with probability 0.5. The state is completely observable except that the agent does not initially know if it starts in the left world or the right world. The agent must first visit the signpost which provides this information, and remember the sign direction until the top-middle state. The agent should then move in the direction the signpost points to receive a reward of 1, or -1 for the wrong direction. In theory 3 I-states are sufficient for optimal control.

Although this problem has only 20 world states, it contains two features which make it a difficult task. Firstly, it requires long-term memory. The signpost direction must be remembered for 5 time steps. Secondly, the temporal credit assignment problem is hard because the actions critical to receiving a consistent reward happen up to 11 steps before receiving the reward. There is nothing in the reward structure which informs the agent that visiting the signpost is good, that is, for a random policy the same average reward of 0 is received regardless of whether the agent saw the signpost.

Figure 10: Convergence plots comparing 100 runs of IState-GPOMDP and Exp-GPOMDP with 4 I-states and a connectivity of $k = 2$. This figure also shows the results for IOHMM-GPOMDP trained using an IOHMM with 4 states and 1000 observations per re-estimation. The variance bars show $0.2\sqrt{(var)}$ to reduce clutter.

Figure 11: The Heaven/Hell problem: the optimal policy is to visit the signpost to find out whether to move left or right at the top-middle state in order to reach heaven.

7.2.2 Experimental Protocol

Agents are parameterised by lookup tables with a total of 1540 parameters. Quadratic penalties were used to avoid local maxima (see Appendix B.3). GAMP used a penalty of $q = 1 \times 10^{-7}$. Good convergence was found to be sensitive to the penalty, with some runs failing when the penalty was set to 1×10^{-6} or 1×10^{-8}. IState-GPOMDP is less sensitive to the penalty and used $q = 1 \times 10^{-4}$. IState-GPOMDP used a discount factor of $\beta = 0.99$ and gradient estimation times of 10^7 steps. The global state space comprises of 20 world states and 20 I-states, for a total of $n_s n_g = 400$. We use more I-states than necessary to increase the ease with which the sparse FSC will learn cycles in the policy-graph of the appropriate length. An analogy can be drawn with multi-layer perceptrons where we often choose the number of hidden units to be larger than strictly required for the problem.

We ran Incremental Pruning for this problem, however it failed to converge after running for 100,000 seconds, producing policy graphs with $n_g > 2000$ states when the optimal infinite-horizon policy graph can be represented with $n_g = 24$.[5] This poor performance results from the failure of the algorithms to identify that the task is actually episodic and that the belief state can be reset after receiving a reward. Using a fixed number of I-states has forced our algorithms to identify useful cycles in the policy graph, hence identifying the episodic qualities of the POMDP.

7.2.3 Results

To demonstrate the necessity of sparse I-state transitions the experiments were run with dense initial FSCs and with FSCs with $k = 3$ possible next I-states for each combination of current I-state and observation. The stochasticity in the GAMP algorithm comes from the random choice of initial FSC. IState-GPOMDP contains the additional randomness of different trajectories taken through the global state space.

Table 3 shows the results of these experiments. Both algorithms find the optimum policy, but only GAMP finds it consistently. The two failed IState-GPOMDP runs had $\eta \approx 0$. All GAMP $k = 3$ runs succeeded. Also notice that the experiments with dense FSCs all failed with no runs converging. For a dense FSC, the initial gradient magnitude estimated by GAMP is within machine tolerance of 0. The agents typically learnt to use about half of the available I-states, learning to never to make transitions to the unused-states.

The wall clock times to convergence quoted for these experiments are not directly comparable. The IState-GPOMDP experiments were run using 94 processors of a 550 MHz dual CPU PIII Beowulf cluster. The GAMP experiments were run on a 1.3 GHz Athlon, roughly equivalent to 3 CPUs of the cluster. It is extraordinary that GAMP converged in less than 0.3% of the time required by IState-GPOMDP, with better results. This demonstrates the advantage of having a model of the world.

The same problem was tackled by Geffner and Bonet (1998) and a continuous state space version was tackled by Thrun (2000). Both of these papers assumed knowledge of the POMDP model and that the problem is episodic. Within our knowledge this is the first

5. Exact algorithms compute policy graphs equivalent to a deterministic $\omega(h|\phi, g, y)$ function and allowing only one action per node. Because we make the policy $\mu(u|\theta, h, y)$ a function of current observation as well as the I-state (equivalent to a policy-graph node), we can compute optimal policies with far fewer I-states. In the case of Heaven/Hell, $n_g = 3$ is sufficient.

Table 3: Results over 10 runs for the Heaven/Hell scenario. η values are multiplied by 10^2.

Algorithm	*mean* η	*max.* η	*var.* $\times 10^5$	*sec. to* 5.0
GAMP $k = 3$	9.01	9.09	0.514	34 (10)
GAMP dense	5.24×10^{-3}	5.24×10^{-3}	0	— (0)
IState-GPOMDP $k = 3$	6.49	9.09	10.8	11436 (8)
IState-GPOMDP dense	0.0178	0.0339	3.23×10^{-4}	— (0)
Optimum	9.09			

time the Heaven/Hell problem has been solved using a model free algorithm. We could not get the IOHMM-GPOMDP algorithm to solve this problem.

7.3 Pentagon

7.3.1 Problem Description

An interesting set of problems are defined by Cassandra (1998). In these problems a robot must navigate through corridors to reach a goal. The world is mapped into discrete locations. The robot occupies a location pointing North, South, East or West. The actions it can take are $\mathcal{U} = \{$`move forward`, `turn left`, `turn right`, `declare goal`$\}$. Actions do nothing with 11% probability, or move/turn one too many steps with 1% probability. This models the unreliability of the movement systems. There are 28 observations which indicate whether the locations immediately to the front and sides of the robot are reachable. The observations have a 12% chance of being misleading, modelling sensor uncertainty. We modified the problem slightly from the original definition to make it an infinite-horizon task and to make rewards control-independent. The modification means that the `declare goal` action in the correct state leads to a special state where a reward of 1 is received for any action before the agent is taken back to the initial state. No penalty is received for the `declare goal` action in the wrong state. In this experiment the agent always starts in the same place but the noise means the agent can quickly become confused about its location even if it has a perfect memory of past observations and actions.

Figure 12 shows the Pentagon navigation problem where the robot must move from the bottom left state to the extra state in the upper left of the middle. This problem exhibits a lot of symmetry, meaning the agent finds many states hard to distinguish by observations alone.

7.3.2 Experimental Protocol

This problem has 209 world states plus internal states. For 20 I-states the global state space size is $20 \times 209 = 4180$ states. The IState-GPOMDP algorithm required a gradient estimation time of 2×10^6 steps to achieve consistent gradient estimates and Exp-GPOMDP required 10^6 steps, demonstrating an effective reduction in variance. Each step in the gradient-ascent line search used an estimation time of one quarter of the standard estimation time. This speeds up convergence, taking advantage of the GSEARCH algorithm's ability to tolerate poor

Figure 12: The Pentagon robot navigation problem maze.

gradient estimates during the line search (see Appendix B.2). For the simulated algorithms a quadratic penalty of $q = 10^{-4}$ was used along with a discount factor of $\beta = 0.99$.

The Belief experiment uses the POMDP model and the observations to maintain a belief state over the 209 world states. The belief state is passed to IState-GPOMDP in place of y. The belief state sufficiently summarises all past events (Åström, 1965) thus the optimal policy is memory-less and we set the number of I-states to 1.[6] Because belief states are vectors in $[0, 1]^{n_s}$ we cannot use tables to parameterise policies. Instead we use a linear policy with 209 inputs and 4 outputs passed through a soft-max function to generate action distributions. The largest experiments required 3920 parameters to be estimated.

As in the Load/Unload experiment, the IOHMM experiment uses discrete transition distributions indexed by IOHMM state and observation. Unlike the Load/Unload experiment, reward emissions are generated by a single Gaussian for each state, bounded to $[-1, 1]$.

7.3.3 Results

The lower part of the table provides baseline figures. I-states approximate the policies achievable using a full belief state thus the Belief result of $\eta = 3.65$ (Table 4) is an empirical upper bound for the η we should be able to achieve using a large number of I-states. IState-GPOMDP with $n_g = 1$ is a memory-less policy which should lower bound the I-state results. Values between these empirical bounds show that I-states can be used to learn better than memory-less policies with a finite amount of memory. The MDP line gives the results for learning when the agent is told the true state of the world, giving us an empirical value for the best policy achievable in a fully observable environment. The Noiseless line is the theoretical maximum that can be achieved if no observation or transition noise is added.

The Incremental Pruning algorithm aborted after 5 value-iteration steps, consuming all 256 MB of memory on the system in 10,800 seconds. Cassandra (1998) contains results for the Pentagon problem which were obtained using the most likely state heuristic. This

6. Using the model to generate belief states which take the place of y in IState-GPOMDP is another form of model based policy-gradient algorithm with the potential to solve large problems, especially in combination with belief state factorisation. This idea was applied to gradient methods for learning value functions by Rodríguez et al. (1999).

Table 4: Results over 10 runs for various algorithms on the Pentagon scenario. η values are multiplied by 10^2.

Algorithm	n_g	k	*mean* η	*max.* η	*var.* $\times 10^5$	*sec. to* 2.0
GAMP	5	2	2.55	2.70	0.102	611 (10)
	10	2	2.50	2.63	0.128	4367 (10)
	20	2	2.50	2.80	0.250	31311 (10)
	20	3	2.89	3.00	0.0613	47206 (10)
IState-GPOMDP	5	2	2.06	2.42	2.81	649 (9)
	10	2	2.18	2.37	0.180	869 (9)
	20	2	2.12	2.28	0.137	1390 (9)
	20	3	2.15	2.33	0.138	1624 (9)
Exp-GPOMDP	5	2	1.96	2.33	4.01	1708 (7)
	10	2	2.19	2.40	0.151	6020 (9)
	20	2	2.11	2.27	0.105	29640 (8)
	20	3	2.26	2.36	0.0448	48296 (10)
IOHMM-GPOMDP	1		1.39	1.26	1.3	— (0)
IOHMM-GPOMDP	5		1.47	1.63	3.65	— (0)
IState-GPOMDP	1	1	1.35	1.37	0.00324	— (0)
Belief			2.67	3.65	7.78	2313 (7)
MDP			4.93	5.01	0.148	24 (10)
Noiseless			5.56	5.56		

method uses the belief state only to identify the most likely current state, performing value updates based on the assumption that the system is in that state, greatly simplifying the problem of representing the value function. To perform a comparison we ran our trained policies on the original POMDP and used the sum of discounted rewards criteria as used in the thesis. The maximum Belief GPOMDP result of 3.64 gives a discounted reward of 0.764. This result sits between Cassandra's results of 0.791 for a known start state and 0.729 for a uniform initial belief state. We do not reset the I-state after receiving rewards so the state in the initial position is not fixed, however the probability of occupying each I-state is not uniform, thus Cassandra's results bracketing ours is a reasonable result.

GAMP has the best finite-memory results because it has access to the POMDP model, however, it does not out-perform the Belief experiments which have access to the model and the *full belief state*. For large or infinite state spaces tracking the full belief state is intractable so finite memory methods such as GAMP are useful. The GAMP result of $\eta = 3.00$ for $n_g = 20$ and $k = 3$ is the best finite state controller, coming close to the best expected result of $\eta = 3.65$ while using only 20 I-states. In fact the mean value for GAMP is greater than the mean value for the Belief experiment, indicating that GAMP gradient estimates have lower variance and that the Belief experiments should have used a longer gradient estimation time. Increasing n_g and the FSC connectivity k generally improves the results due to the increasing richness of the parameterisation. However setting $k > 3$ caused failures due to small initial gradients.

The fact that the best IState-GPOMDP results come from $n_g < 20$ is because we did not scale the number of gradient estimation steps with the number of I-states. Exp-GPOMDP's reduced variance allowed it to demonstrate performance increases as n_g is increased, even when using fewer gradient estimation steps. Fewer estimation steps means fewer expensive interactions with the world. This claim is backed up by Figure 13 which demonstrates that for $n_g = 20$, Exp-GPOMDP produces a superior result in fewer steps than IState-GPOMDP.

8. Future Work

The state space of a POMDP grows exponentially with the number of *state variables* in the system. Factored belief state methods for POMDPs (Hansen and Feng, 2000) operate on state variables directly, greatly compacting the representation of the POMDP and hence allowing much larger problems to be tackled. To scale to hundreds of thousands of states algorithms such as GAMP should be extended to factored representations. Factored representations can also be used with IState-GPOMDP when observations are replaced by factored belief states computed using the world model (similar to the Belief experiment of Section 7.3).

We have so far assumed a fixed number of I-states, n_g. We would prefer algorithms that begin with a small number of I-states and add more as found necessary. A simple approach is to split states which have non-deterministic transitions or actions. Similar ideas have been applied to HMMs (Chrisman, 1992) and utile distinction trees (McCallum, 1996).

Extending the trick of sparse FSCs to function approximators like neural networks is relatively straight forward and we plan to apply it to problems with continuous state spaces such as speech processing.

Figure 13: Convergence of IState-GPOMDP and Exp-GPOMDP on the Pentagon problem with $n_g = 20$ and $k = 3$.

For applications that learn from real experience it is important to keep the number of interactions with the world to a minimum (Peshkin and Shelton, 2001). For example, it is not feasible to allow a robot to run around a building making millions of random high level decisions. Combining techniques like importance sampling (Glynn, 1996, Ortiz and Kaelbling, 2000, Shelton, 2001b) with Exp-GPOMDP is an interesting prospect for minimising world interactions.

9. Conclusion

We have described and empirically compared four policy-gradient algorithms for learning parameterised policies for POMDPs. All the algorithms incorporate internal state to help cope with the loss of the Markov property that occurs with partial observability. The GAMP algorithm makes use of a model of the environment to compute approximations to the gradient of the long term average reward. The IState-GPOMDP algorithm estimates the gradient without a model by interacting with the world. The IOHMM-GPOMDP algorithm reveals hidden state by attempting to predict rewards and the Exp-GPOMDP algorithm behaves similarly to IState-GPOMDP but it computes the expectation over all I-state trajectories instead of simulating paths. We also analysed as problem with finite state controller agents which occurs when the I-states are initially undifferentiated. We have demonstrated the ability of these algorithms to learn POMDPs with thousands of states, increasing by an order of magnitude the size of POMDPs previously learnt using FSCs.

10. Acknowledgements

The Authors would like to thank Professors Markus Hegland, Sebastian Thrun, Brian Anderson and Yutaka Yamamoto for their comments. Thanks also to Phillip Williams for mathematical assistance. Douglas Aberdeen is supported by an Australian Postgraduate Award.

Appendix A. Proofs

A.1 Proof of Theorem 3

Proof

$$\begin{aligned}
\|\widehat{\nabla}\eta - \nabla\eta\|_\infty &= \|\pi \sum_{n=N}^{\infty} \nabla P P^n r\| \\
&\leq \|\pi\| \|\sum_{n=N}^{\infty} \nabla P P^n r\|, \quad \text{Cauchy-Schwartz inequality,} \\
&\leq \|\pi\| \sum_{n=N}^{\infty} \|V_n\|, \quad \text{Triangle inequality,} \\
&\leq \sum_{n=N}^{\infty} \|V_n\|, \quad \|\pi\|_\infty \leq 1 \qquad (14)
\end{aligned}$$

where $\|V_n\|$ is defined by the following expression in which the vector ξ_n takes the place of $PP^n r$ and $\xi_{n,j}$ is the j'th element of ξ_n

$$\begin{aligned}
\|V_n\| &= \|\nabla P P^n r\| \\
&= \|\nabla P \xi_n\| \\
&= \|\sum_{j,y,u} \nu(y|i)\nabla\mu(u|\theta, y)q(j|i, u)\xi_{n,j}\|.
\end{aligned}$$

Now we show the computation for each element of ξ_n, defining it in terms of the sum of η and the variation of $\xi_{n,j}$ away from η

$$\begin{aligned}
\xi_n, i &= \sum_j P^n_{ij} r_j \\
&= \sum_j (\pi_j + (P_{ij} - \pi_j))r_j \\
&= \sum_j (\pi_j + \Delta_n(i, j))r_j \\
&= \eta + \sum_j \Delta_n(i, j) r_j.
\end{aligned}$$

Substituting back into $\|V_n\|$ we obtain

$$\begin{aligned}
\|V_n\| &= \|\sum_{j,y,u} \nu(y|i)\nabla\mu(u|\theta,y)q(j|i,u)(\eta + \sum_m \Delta_n(j,m)r_m)\| \\
&= \|\sum_{j,y,u} \nu(y|i)\nabla\mu(u|\theta,y)q(j|i,u)\eta + \\
&\qquad \sum_{j,y,u} \nu(y|i)\nabla\mu(u|\theta,y)q(j|i,u)\sum_m \Delta_n(j,m)r_m\| \\
&= \|\eta\sum_y \nu(y|i)\nabla\sum_u \mu(u|\theta,y)\underbrace{\sum_j q(j|i,u)}_{=1} + \\
&\qquad \sum_{j,y,u} \nu(y|i)\nabla\mu(u|\theta,y)q(j|i,u)\sum_m \Delta_n(j,m)r_m\| \\
&= \|\eta\sum_y \nu(y|i)\underbrace{\nabla\sum_u \mu(u|\theta,y)}_{=0} + \\
&\qquad \sum_{j,y,u} \nu(y|i)\nabla\mu(u|\theta,y)q(j|i,u)\sum_m \Delta_n(j,m)r_m\| \\
&= \|\sum_{j,y,u} \nu(y|i)\nabla\mu(u|\theta,y)q(j|i,u)\sum_m \Delta_n(j,m)r_m\|,
\end{aligned}$$

which has only one vector quantity $\mu(u|\theta,y)$ so we can take the scalar factors outside the norm, taking the absolute value of the $|\sum_m \Delta_n(j,m)|$ which is the only possibly negative quantity

$$\begin{aligned}
&= \|\sum_{j,y,u} \nu(y|i)\nabla\mu(u|\theta,y)q(j|i,u)\sum_m \Delta_n(j,m)r_m\| \\
&\leq \sum_{y,u} \nu(y|i)\|\nabla\mu(u|\theta,y)\|\sum_j q(j|i,u)|\sum_m \Delta_n(j,m)r_m| \\
&\leq R\sum_{y,u} \nu(y|i)U\sum_j q(j|i,u)|\sum_m \Delta_n(j,m)|. \qquad (15)
\end{aligned}$$

The last line makes use of the bounds on $\nabla\mu(u|\theta,y)$ and r_m stated in Assumptions 2 and 3. Now observe that

$$\begin{aligned}
|\sum_m \Delta_n(j,m)| &\leq \sum_m |\Delta_n(j,m)|, \quad \text{Triangle inequality} \\
&= \sum_m |P^n_{jm} - \pi_m| \\
&= d(P^n_j, \pi),
\end{aligned}$$

that is, the total variation distance from Definition 1. Returning to (15) and substituting the inequality above

$$\leq UR \sum_{y,u} \nu(y|i) \sum_j q(j|i,u) d(P_j^n, \pi),$$

which is further simplified by letting $\bar{d}(n) = \max_j d(P_j^n, \pi)$, giving

$$\begin{aligned}
&\leq UR \sum_{y,u} \nu(y|i) \sum_j q(j|i,u) \bar{d}(n) \\
&= UR\bar{d}(n) \sum_{y,u} \nu(y|i) \underbrace{\sum_j q(j|i,u)}_{=1} \\
&= UR\bar{d}(n) \underbrace{\sum_{y,u} \nu(y|i)}_{=n_u} \\
&= URn_u \bar{d}(n).
\end{aligned}$$

Now we substitute this value for $\|V_n\|$ into (14)

$$\begin{aligned}
\|\widehat{\nabla\eta} - \nabla\eta\| &\leq \sum_{n=N}^{\infty} URn_u \bar{d}(n) \\
&= URn_u \sum_{n=N}^{\infty} \bar{d}(n).
\end{aligned}$$

Now we make use of Definition 2, bounding $\bar{d}(n)$ in terms of τ of the matrix P

$$\begin{aligned}
&\leq URn_u \sum_{n=N}^{\infty} \exp(-\lfloor \frac{n}{\tau} \rfloor) \\
&= URn_u \left[\left(\sum_{l=\lfloor \frac{N}{\tau} \rfloor + 1}^{\infty} \sum_{k=0}^{\tau-1} \exp(-l) \right) + \sum_{k=N \mod \tau}^{\tau-1} \exp(-\lfloor \frac{N}{\tau} \rfloor) \right] && (16) \\
&\leq URn_u \sum_{l=\lfloor \frac{N}{\tau} \rfloor}^{\infty} \sum_{k=0}^{\tau-1} \exp(-l). && (17)
\end{aligned}$$

The last two lines re-write the bound so that the floor operator does not appear in the summation. We arrive at (16) by re-writing the first line as a sum of sums for which the $l = \lfloor n/\tau \rfloor$ value is constant. The second term of (16) is eliminated by noting that $k = N \mod \tau \leq \tau$ and combining it with the first sum by subtracting one from the initial summation index l. We can now observe we are performing the first sum τ times and apply a a standard series convergence result

$$
\begin{aligned}
&= URn_u\tau \sum_{l=\lfloor \frac{N}{\tau} \rfloor}^{\infty} \exp(-l) \\
&= URn_u\tau \sum_{l=\lfloor \frac{N}{\tau} \rfloor}^{\infty} \left(\frac{1}{\exp(1)}\right)^l \\
&= URn_u\tau \frac{\exp(-\lfloor \frac{N}{\tau} \rfloor)}{1-\exp(-1)}.
\end{aligned}
$$

■

A.2 IState-GPOMDP Proofs

A.2.1 Proof of Theorem 5

The following proof is copied from Baxter and Bartlett (2001) and is not the author's work.
Proof Recall from Equation (7) that

$$\nabla\eta = \pi\nabla P\left[I-(P-e\pi)\right]^{-1} r$$

A quick induction argument shows that $[P-e\pi]^n = P^n - e\pi$ which converges to 0 as $n \to \infty$ by Assumption 1. So by a classical matrix theorem, $[I-(P-e\pi)]^{-1}$ exists and is equal to $\sum_{n=0}^{\infty}[P^n - e\pi]$. Observe that $(\nabla P)e = 0$ so (7) can be rewritten as

$$\nabla\eta = \pi\left[\sum_{n=0}^{\infty}(\nabla P)P^n\right] r. \tag{18}$$

Now let $\beta \in [0,1]$ be a discount factor and consider the following modification to Equation (18)

$$\widehat{\nabla_\beta\eta} := \pi\left[\sum_{n=0}^{\infty}(\nabla P)(\beta P)^n\right] r. \tag{19}$$

Since $\lim_{\beta\to 1}\nabla_\beta\eta = \nabla\eta$, and since $(\beta P)^n = \beta^n P^n \to \beta^n e\pi \to 0$, we can take ∇P back out of the sum and write

$$\widehat{\nabla_\beta\eta} := \pi(\nabla P)\left[\sum_{n=0}^{\infty}\beta^n P^n\right] r. \tag{20}$$

But $[\sum_{n=0}^{\infty}\beta^n P^n]\, r = J_\beta$ where $J_\beta = [J_\beta(1,1),\ldots,J_\beta(n_s,n_g)]$ is the vector of expected discounted reward from each world/I-state pair (i,g):

$$J_\beta(i,g) := \mathbb{E}\left[\sum_{t=0}^{\infty}\beta^t r(i_t,g_t)|i_0 = i, g_0 = g\right], \tag{21}$$

where the expectation is over all sample paths $(i, g), (i_1, g_1), (i_2, g_2), \ldots$. This give us the required expression

$$\lim_{\beta \to 1} \pi(\nabla P)r = \nabla \eta.$$

■

A.2.2 Proof of Theorem 5

This proof is an easy generalisation of (Baxter and Bartlett, 2001, Thm. 4).

Proof Let $X_0, X_1, \ldots, X_t, \ldots$ denote the random process corresponding to the environment Markov chain generated by $P(\theta, \phi)$. Also, let $G_0, G_1, \ldots, G_t, \ldots$ denote the random process corresponding to the I-state Markov chain generated by $P(\theta, \phi)$. By Assumption 1, $\{X_t, G_t\}$ is asymptotically stationary, and we can write

$$\begin{aligned}
\pi \nabla P J_\beta &= \sum_{i,j} \pi(i,g) \nabla p_{(i,g)(j,h)}(\theta, \phi) J_\beta(j,h) \\
&= \sum_{i,j,y,u,g,h} \pi(i,g) q(j|i,u) \nu_y(i) \\
&\qquad \Big[\omega(h|\phi,g,y)\nabla^\theta \mu(u|\theta,h,y), \nabla^\phi \omega(h|\phi,g,y)\mu(u|\theta,h,y)\Big] J_\beta(j,h), \text{ from (2)} \\
&= \sum_{i,j,y,u,g,h} \pi(i,g) p_{(i,g)(j,h)}(u) \nu_y(i) \\
&\qquad \Bigg[\omega(h|\phi,g,y)\frac{\nabla^\theta \mu(u|\theta,h,y)}{\mu(u|\theta,h,y)} \mu(u|\theta,h,y) J_\beta(j,h), \\
&\qquad\quad \frac{\nabla^\phi \omega(h|\phi,g,y)}{\omega(h|\phi,g,y)} \omega(h|\phi,g,y)\mu(u|\theta,h,y) J_\beta(j,h)\Bigg].
\end{aligned}$$

There are two independent sets of gradient values for the θ and ϕ parameters. The remainder of the proof follows the gradient w.r.t. ϕ however the proof w.r.t. θ is the same. Dropping the ∇^θ components, we can rewrite the last expression the right-hand-side as

$$\sum_{i,j,y,u,g,h} \mathbb{E} \chi_i(X_t)\chi_j(X_{t+1})\chi_g(G_t)\chi_h(G_{t+1})\chi_u(U_t)\chi_y(Y_t)\frac{\nabla \omega(h|\phi,g,y)}{\omega(h|\phi,g,y)} J(t+1),$$

where $\chi_i(X)$ denotes the indicator function for state i of random process X

$$\chi_i(X_t) := \begin{cases} 1 & \text{if } X_t = i, \\ 0 & \text{otherwise,} \end{cases}$$

and the expectation is with respect to the joint distribution of $\{X_t, X_{t+1}, Y_t, G_t, G_{t+1}, U_t\}$ when X_t is stationary. Here $J(t+1)$ is the process governing the discounted infinite horizon reward

$$J(t+1) = \sum_{s=t+1}^{\infty} \beta^{s-t-1} r(X_s, G_s)$$

such that

$$J_\beta(j,h) = \mathbb{E}[J(t+1)|X_{t+1}=j, G_{t+1}=h]$$

follows from the boundedness of the rewards and Lebesgue's dominated convergence theorem.

If X_t is stationary then X_t is also ergodic and the joint process $\{X_t, X_{t+1}, Y_t, G_t, G_{t+1}, U_t\}$ is stationary and ergodic (because Y_t is i.i.d. given X_t, G_{t+1} is i.i.d. given G_t and Y_t, U_t is i.i.d. given G_{t+1} and Y_t, and X_{t+1} is i.i.d. given X_t and U_t). Since $\{Z_t\}$ is obtained by taking a fixed function of $\{X_t, X_{t+1}, Y_t, G_t, G_{t+1}, U_t\}$, it is also stationary and ergodic (see (Breiman, 1966, Proposition 6.31)). As $\left\|\frac{\nabla\omega(h|\phi,g,y)}{\omega(h|\phi,g,y)}\right\|$ is bounded by Assumption 4, from the ergodic theorem we obtain (with obvious notation):

$$\begin{aligned}
&\pi\nabla^\phi P J_\beta = \\
&\sum_{i,j,y,u,g,h} \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \chi_{ijuygh}(X_t, X_{t+1}, U_t, Y_t, G_t, G_{t+1}) \frac{\nabla\omega(h|\phi,g,y)}{\omega(h|\phi,g,y)} J(t+1) \\
&= \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \sum_{ijyugh} \chi_{ijuygh}(X_t, X_{t+1}, U_t, Y_t, G_t, G_{t+1}) \frac{\nabla\omega(h|\phi,g,y)}{\omega(h|\phi,g,y)} J(t+1) \\
&= \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla\omega(G_{t+1}|\phi,G_t,Y_t)}{\omega(G_{t+1}|\phi,G_t,Y_t)} J(t+1) \\
&= \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla\omega(G_{t+1}|\phi,G_t,Y_t)}{\omega(G_{t+1}|\phi,G_t,Y_t)} \left[\sum_{s=t+1}^{T} \beta^{s-t-1} r(X_s,G_s) + \sum_{s=T+1}^{\infty} \beta^{s-t-1} r(X_s,G_s)\right].
\end{aligned} \tag{22}$$

Concentrating on the second term in the right-hand-side of (22), observe that:

$$\begin{aligned}
&\left\|\frac{1}{T}\sum_{t=0}^{T-1} \frac{\nabla\omega(G_{t+1}|\phi,G_t,Y_t)}{\omega(G_{t+1}|\phi,G_t,Y_t)} \sum_{s=T+1}^{\infty} \beta^{s-t-1} r(X_s,G_s)\right\| \\
&\le \frac{1}{T}\sum_{t=0}^{T-1} \left\|\frac{\nabla\omega(G_{t+1}|\phi,G_t,Y_t)}{\omega(G_{t+1}|\phi,G_t,Y_t)}\right\| \sum_{s=T+1}^{\infty} \beta^{s-t-1} \left|r(X_s,G_s)\right| \\
&\le \frac{BR}{T}\sum_{t=0}^{T-1}\sum_{s=T+1}^{\infty} \beta^{s-t-1} \\
&= \frac{BR}{T}\sum_{t=0}^{T-1} \frac{\beta^{T-t}}{1-\beta} \\
&= \frac{BR\beta\left(1-\beta^T\right)}{T\left(1-\beta\right)^2} \\
&\to 0 \,\text{as}\, T\to\infty,
\end{aligned}$$

where R and B are the bounds on the magnitudes of the rewards and $\|\nabla\omega/\omega\|$ from Assumptions 2 and 4. Hence,

$$\pi\nabla^{\phi}PJ_{\beta} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\omega(G_{t+1}|\phi,G_t,Y_t)}{\omega(G_{t+1}|\phi,G_t,Y_t)}\sum_{s=t+1}^{T}\beta^{s-t-1}r(X_s,G_s),$$

almost surely. Unrolling the equation for Δ_T^{ϕ} in the IState-GPOMDP algorithm shows it is equal to

$$\frac{1}{T}\sum_{t=0}^{T-1}\frac{\nabla\omega(g_{t+1}|\phi,g_t,y_t)}{\omega(g_{t+1}|\phi,g_t,y_t)}\sum_{s=t+1}^{T}\beta^{s-t-1}r_{s-1},$$

hence $\Delta_T^{\phi}\to\pi\nabla^{\phi}PJ_{\beta}$ w.p.1 as required. The same procedure is followed to show that this is also true with respect to the parameters θ. ■

A.3 Conditions for Zero Gradient of the I-state Controller

To increase η by utilising internal state, for some (θ,ϕ) we require $\|\nabla^{\phi}\eta\|>0$. From (8), we have

$$\nabla\eta = \pi\left[\sum_{n=0}^{\infty}(\nabla P)P^n\right]r. \tag{23}$$

We must select (θ,ϕ) to provide an initial FSC and policy prior to training. A sensible choice for the FSC is one that makes the least assumptions about the task: a uniform FSC where any observation y is equally likely to lead to any next I-state h from any current I-state g. Here we shall prove that this and similarly sensible choices of initial FSC result in $\|\nabla^{\phi}\eta\|=0$.

Recall from (2) that the transition probability matrix $P(\theta,\phi)$ has dimension $n_s n_g\times n_s n_g$ and the entries $p_{(i,g)(j,h)}(\theta,\phi)|_{i,j=1\ldots n_s;g,h=1\ldots n_g}$ are given by

$$p_{(i,g)(j,h)}(\theta,\phi)=\sum_{y\in\mathcal{Y}}\sum_{u\in\mathcal{U}}\nu(y|i)\omega(h|\phi,g,y)\mu(u|\theta,h,y)q(j|i,u). \tag{24}$$

Also, $r=[r_1,r_2,\dot{s},r_{n_s}]'$ is a column vector of the rewards received for being in state i. In the case of I-states $r_{(i,g)}=r_i$.

Notation 1 *For the remainder of this section we will abuse notation to allow $\omega(h|\phi,g,y)=\omega(h|\phi,y)$ to mean $\omega(h|\phi,g,y)=\omega(h|\phi,g',y)\quad\forall g,g',y$ for the* specified choice of *θ or ϕ. In words we say that in this situation $\omega(h|\phi,y)$ is independent of the choice of I-state. When taking the gradient of these functions we will reintroduce the I-state dependence since the gradient may vary with the I-state even when the value does not.*

Lemma 9 *If we choose θ and ϕ such that $\omega(h|\phi,g,y)=\omega(h|\phi,y)$ $\forall g$ and $\mu(u|\theta,h,y)=\mu(u|\theta,y)$ $\forall h$ that is, independent of the current I-state, then $(\nabla^{\phi}P)P=0$.*

Proof

We start by re-writing (2), taking into account the simplified distributions and combining the two summations for brevity

$$p_{(i,g)(j,h)}(\theta,\phi) = \sum_{y,u} \nu(y|i)\omega(h|\phi,y)\mu(u|\theta,y)q(j|i,u). \tag{25}$$

Similarly we can write down the simplified gradient of the x'th parameter of ϕ, denoted ϕ_l

$$\begin{aligned}\frac{\partial p_{(i,g)(j,h)}(\theta,\phi)}{\partial \phi_x} &= \frac{\partial}{\partial \phi_l}\left[\sum_{y,u} \nu(y|i)\omega(h|\phi,y)\mu(u|\theta,y)q(j|i,u)\right] \\ &= \sum_{y,u} \nu(y|i)\frac{\partial \omega(h|\phi,g,y)}{\partial \phi_x}\mu(u|\theta,y)q(j|i,u).\end{aligned}$$

Now, the $(i,g)(j,h)$'th element of $(\frac{\partial P}{\partial \phi_l})P$ is the dot product of row (i,g) of $\frac{\partial P}{\partial \phi_l}$ with column (j,h) of P. Here (k,c) defines which dot product element we are computing

$$\begin{aligned}\left(\frac{\partial P}{\partial \phi_l}P\right)_{(i,g)(j,h)} &= \sum_{k\in\mathcal{S}}\sum_{c\in\mathcal{G}} \frac{\partial p_{(i,g)(k,c)}(\theta,\phi)}{\partial \phi_x} p_{(k,c)(j,h)} \\ &= \sum_{k\in\mathcal{S}}\sum_{c\in\mathcal{G}} \left(\sum_{y,u}\nu(y|i)\frac{\partial \omega(c|\phi,g,y)}{\partial \phi_x}\mu(u|\theta,y)q(k|i,u)\right) \\ &\qquad \left(\sum_{y,u}\nu(y|i)\omega(h|\phi,y)\mu(u|\theta,y)q(j|k,u)\right).\end{aligned}$$

From (25) we see that $p_{(k,c)(j,h)}(\theta,\phi)$ does not depend on c, so we define $p_{(k,c)(j,h)}(\theta,\phi) := p_{(k)(j,h)}(\theta,\phi)$ and continue by moving the sum over c inside $\frac{\partial p_{(i,g)(k,c)}(\theta,\phi)}{\partial \phi_x}$

$$\begin{aligned}&= \sum_{k\in\mathcal{S}}\left(\sum_{y,u}\nu(y|i)\mu(u|\theta,y)q(k|i,u)\sum_{c\in\mathcal{G}}\frac{\partial \omega(c|\phi,g,y)}{\partial \phi_x}\right)p_{(k)(j,h)}(\theta,\phi) \\ &= \sum_{k\in\mathcal{S}}\left(\sum_{y,u}\nu(y|i)\mu(u|\theta,y)q(k|i,u)\frac{\partial}{\partial \phi_l}\sum_{c\in\mathcal{G}}\omega(c|\phi,g,y)\right)p_{(k)(j,h)}(\theta,\phi) \\ &= \sum_{k\in\mathcal{S}}\left(\sum_{y,u}\nu(y|i)\mu(u|\theta,y)q(k|i,u)\frac{\partial}{\partial \phi_l}1\right)p_{(k)(j,h)}(\theta,\phi) \\ &= \sum_{k\in\mathcal{S}}\left(\sum_{y,u}\nu(y|i)\mu(u|\theta,y)q(k|i,u)0\right)p_{(k)(j,h)}(\theta,\phi) \\ &= [0].\end{aligned}$$

■

Lemma 10 *Under the same conditions as Lemma 9 and if r is a column vector of rewards with $r_{(i,a)} = r_i$, then $(\nabla^\phi P)r = 0$.*

Proof The (i,g)'th element of $(\frac{\partial P}{\partial \phi_l})r$ is the dot product of row (i,g) of $\frac{\partial P}{\partial \phi_l}$ with r. By definition $r_{(k,c)} = r_l$ is independent of c, giving us

$$\left(\frac{\partial P}{\partial \phi_l} r\right)_{(i,g)} = \sum_{k \in \mathcal{S}} \sum_{c \in \mathcal{G}} \left(\sum_{y,u} \nu(y|i) \frac{\partial \omega(c|\phi,g,y)}{\partial \phi_x} \mu(u|\theta,y) q(k|i,u) \right) r_l.$$

Now we move the sum over c inside $\frac{\partial p_{(i,g)(k,c)}(\theta,\phi)}{\partial \phi_x}$. The rest of the proof is the same as Lemma 9. ■

Theorem 11 *If we choose θ and ϕ such that $\omega(h|\phi,g,y) = \omega(h|\phi,y)\ \forall a$ and $\mu(u|\theta,h,y) = \mu(u|\theta,y)\ \forall b$ then $\nabla^\phi \eta = 0$.*

Proof We expand (23) and apply Lemmas 9 and 10

$$\begin{aligned}
\frac{\partial \eta}{\partial \phi_l} &= \pi \left[\frac{\partial P}{\partial \phi_l} + \sum_{n=1}^{\infty} \frac{\partial P}{\partial \phi_l} P^n \right] r \\
&= \pi \frac{\partial P}{\partial \phi_l} r + \pi \left[\sum_{n=1}^{\infty} \frac{\partial P}{\partial \phi_l} P^n \right] r \\
&= \pi[0] + \pi \left[\sum_{n=1}^{\infty} \frac{\partial P}{\partial \phi_l} P P^{n-1} \right] r \\
&= \pi \left[\sum_{n=1}^{\infty} [0] P^{n-1} \right] r \\
&= \pi[0] r \\
&= 0.
\end{aligned}$$

■

A.3.1 Conditions for Perpetual Zero Gradient

So far we have not shown any results about $\nabla^\theta \eta$. It is possible that $\|\nabla^\theta \eta\| > 0$ even under conditions for Theorem 11. There are at least three situations in which this can happen:

1. In some POMDPs it possible to increase η by changing the stationary distribution of actions, that is, emitting some actions more frequently than others regardless of the observation. If this is possible then $\|\nabla^\theta \eta\| > 0$ is possible.

2. If the μ process is a function of y then μ can learn the optimal reactive policy, ignoring h.

3. If we choose ϕ s.t. $\omega(h|\phi, g, y) = \omega(h|\phi, y) \neq 1/n_g$ then h tells us something about y even though it tells us nothing about the previous I-state, g. In this case $\omega(h|\phi, y)$ is a *partial observation hiding* process in the same way that $\nu(y|i)$ is a partial state hiding process. Because h is still a useful indicator of state $\nabla^\theta \eta$ may have different values for parameters related to different I-states, indicating that internal state can help to maximise η. A single iteration of gradient ascent may therefore cause the condition $\mu(u|\theta, h, y) = \mu(u|\theta, y)$ of Theorem 11 to be violated, such that the next computation of the gradient results in $\|\nabla^\phi \eta\| > 0$.

The last case is interesting because it allows us to initialise $\mu(u|\theta, h, y) = \mu(u|\theta, y)$ and $\omega(h|\phi, g, y) = \omega(h|\phi, y)$ and possibly still learn to utilise I-states despite that fact Theorem 11 tells us that these initialisations are sufficient for $\|\nabla^\phi \eta\| = 0$. This possibility has been verified experimentally.

When this happens depends on the choice of parameterisation for $\mu(u|\theta, h, y)$. We will now show that for a $\mu(u|\theta, h, y)$ parameterised by a real-valued table and a soft-max output distribution, and under slightly tighter initialisation conditions than Theorem 11, that it is not possible for these conditions to be violated and hence $\|\nabla^\phi \eta\| = 0$ from one gradient ascent iteration to another, resulting in perpetual failure to learn to utilise I-states.

Lemma 12 *Choose $\omega(h|\phi, g, y) = 1/n_g$ and some $\mu(u|\theta, h, y) = \mu(u|\theta, y)$. Then $\nabla^\theta \eta$ is independent of the I-state, that is, the equation for each element of $\nabla^\theta \eta$ is not a function of the I-state.*

Proof If we rewrite (2) using the simplified definitions we find there is no dependence on the I-state

$$\begin{aligned}\frac{\partial p_{(i,g)(j,h)}(\theta, \phi)}{\partial \phi_x} &= \frac{\partial}{\partial \phi_l}\left[\sum_{y,u} \nu(y|i)\frac{1}{n_g}\mu(u|\theta, y) q(j|i, u)\right] \\ &= \sum_{y,u} \nu(y|i)\frac{\partial \omega(h|\phi, g, y)}{\partial \phi_x}\mu(u|\theta, y) q(j|i, u),\end{aligned}$$

so we can define $p_{(k,c)(j,h)}(\theta, \phi) := p_{(k)(j)}(\theta, \phi)$. Now even if $\nabla^\theta \mu(u|\theta, h, y)$ is generally a function of g, $\nabla^\theta \eta$ is independent of g under the conditions given. This is because the dependence of the gradient, for any element p of θ, on the I-state is always marginalised out during the dot-product

$$\begin{aligned}\left(\frac{\partial P}{\partial \theta_p} P\right)_{(i,g)(j,h)} &= \sum_{k\in\mathcal{S}}\sum_{c\in\mathcal{G}}\left(\frac{1}{n_g}\sum_{u,y}\nu(y|i) q(k|i, u)\frac{\partial \mu(u|\theta, c, y)}{\partial \theta_p}\right) p_{(k)(j)} \\ &= \sum_{k\in\mathcal{S}}\left(\frac{1}{n_g}\sum_{u,y}\nu(y|i) q(k|i, u)\sum_{c\in\mathcal{G}}\frac{\partial \mu(u|\theta, c, y)}{\partial \theta_p}\right) p_{(k)(j)}\end{aligned}$$

At this point we observe that $(\frac{\partial P}{\partial \theta_p})P$ is completely independent of the internal state. Because the rewards are independent of the internal state it follows that $(\frac{\partial P}{\partial \theta_p})r$ is also independent of the internal state. If we can show that π is independent of internal state then follows from (23) that $\nabla^\theta \eta$ is independent of the internal state.

Since we have defined $\omega(h|\phi, g, y) = 1/n_g$, and with the further assumption that the initial distribution on I-states is $\pi(g) = 1/n_g$, then the probability of being in any I-state at any time is $1/n_g$. This allows us to state $\pi_{(i,g)} = \frac{\pi_i}{n_g}$. Therefore, all components of (23) are independent of the current internal state and thus $\nabla^\theta \eta$ is independent of the internal state.

■

After estimating the gradient we make a step $\Delta\theta := \gamma \nabla^\theta \eta$, where γ is a positive step size. Whether or not the conditions of Lemma 12 lead to the perpetual 0 gradient situation depends on whether $\mu(u|\theta + \Delta\theta, g, y)$ alters the conditions for Lemma 12. These conditions can be broken even when $\nabla^\theta \eta$ is independent of the I-state in the way Lemma 12 defines.

So we need to prove $\mu(u|\theta{+}\Delta\theta, g, y) = \mu(u|\theta{+}\Delta\theta, y)$ for individual choices of $\mu(u|\theta, h, y)$. We start by showing this is true for a lookup table indexed by (g, y) and which provides an $\mathbb{R}^{n_u}$ vector turned into an action distribution using a soft-max function (10) (equivalently a Boltzmann function with a temperature co-efficient of 1.0). Since the use of the soft-max distribution is a common choice for turning real number vectors into distributions, the following proof forms a basis for showing many common choices of $\mu(u|\theta, h, y)$ can lead to perpetual $\nabla^\phi \eta = 0$ situations.

Lemma 13 *Let $x, y \in \mathbb{R}^N$ and $u \in \mathcal{U}$ where $\mathcal{U} = \{1, \ldots, N\}$. Define $d(u) : \mathcal{U} \mapsto \mathbb{R}$. Assume*

$$\frac{\exp(x_u)}{\sum_{u'=1}^{N} \exp(x'_u)} = \frac{\exp(y_u)}{\sum_{u'=1}^{N} \exp(y'_u)} \qquad \forall u, \tag{26}$$

then

$$\frac{\exp(x_u + d(u))}{\sum_{u'=1}^{N} \exp(x'_u + d(u'))} = \frac{\exp(y_u + d(u))}{\sum_{u'=1}^{N} \exp(y'_u + d(u'))} \qquad \forall u. \tag{27}$$

Proof

For (26) to be true, the x_u's and y_u's must differ by at most some constant c. This can be shown with a short proof by contradiction starting with $x_u = y_u + c_u$ and assuming $\exists c_{u'} \neq c_u$ for which the equality holds. This fact and some algebra give us the result:

$$\begin{aligned}
\frac{\exp(x_u + d(u))}{\sum_{u'=1}^{N} \exp(x'_u + d(u'))} &= \frac{\exp(c)\exp(x_u + d(u))}{\sum_{u'=1}^{N} \exp(c)\exp(x'_u + d(u'))} \\
&= \frac{\exp(x_u + c + d(u))}{\sum_{u'=1}^{N} \exp(x'_u + c + d(u'))} \\
&= \frac{\exp(y_u + d(u))}{\sum_{u'=1}^{N} \exp(y'_u + d(u'))}
\end{aligned}$$

■

This lemma tells us that if we have two equal soft-max distributions generated by the possibly different vectors x and y, then adding a quantity independent of x or y (but possibly dependent on the element index) to both vectors results in distributions which remain equal (though they represent an altered distribution to the equality of (26)). This is useful because we wish to show that when we have two vectors generated by different I-states, which result in the same output distribution, then adding a quantity independent of the I-state does not change the equality of the distributions and hence the independence of the distributions with respect to the I-state conditioning.

Lemma 14 *For $\mu(u|\theta, h, y)$ parameterised by a lookup-table with soft-max output distribution $\mu(u|\theta, h, y)$, and under the conditions of Lemma 12 then $\mu(u|\theta + \Delta\theta, g, y) = \mu(u|\theta + \Delta\theta, y)$ implies $\|\nabla^{\phi}\eta\| = 0$ always.*

Proof Set $x_u = \theta_{ugy}$ and $y_u = \theta_{u\bar{g}y}$. If we set

$$d(u) = \gamma \frac{\partial \eta}{\partial \theta_{ugy}},$$

then the *left* hand side of (27) is equal to $\mu(u|\theta + \Delta\theta, g, y)$. If we set

$$d(u) = \gamma \frac{\partial \eta}{\partial \theta_{u\bar{g}y}},$$

then the *right* hand side gives us $\mu(u|\theta + \Delta\theta, \bar{g}, y)$. However, Lemma 13 requires $d(u)$ be the same on the left and right sides, which is precisely what Lemma 12 tells us is the case so

$$d(u) = \gamma \frac{\partial \eta}{\partial \theta_{ugy}} = \gamma \frac{\partial \eta}{\partial \theta_{u\bar{g}y}}.$$

Lemma 13 can be applied for all choices of g and $\bar{g}$, resulting in

$$\begin{aligned} \mu(u|\theta + \Delta\theta, g, y) &= \mu(u|\theta + \Delta\theta, \bar{g}, y) \quad \forall g, \bar{g} \\ &= \mu(u|\theta + \Delta\theta, y). \end{aligned}$$

■

In summary Lemma 14 tells us that if we meet the necessary conditions for Lemma 12, then $\|\nabla^{\phi}\eta\| = 0$ from one step of the GPOMDP algorithm to another, and by induction we have $\|\nabla^{\phi}\eta\| = 0$ always.

Because we have based the proof on the value of the true gradient, the result holds for any algorithm that estimates the gradient. This includes GPOMDP and I-state Williams REINFORCE (Peshkin et al., 1999).

The analysis above presents a method for avoiding 0 gradient regions of parameter space. We simply select initial controllers which violate the conditions of Theorem 11. By making the set of future states reachable from I-state g a subset of $\mathcal{G}$ and by ensuring that the subset we can reach is different for all g and y, we neatly avoid the problem and increase computational efficiency by taking advantage of the sparseness. This allows us to use very large I-state space without slowing down each step of the computation. However, large I-state spaces will require more steps.

Lemma 12 assumes a table-lookup controller with a soft-max output distribution. The following generalises the same argument to arbitrary functions with soft-max output distributions.

Theorem 15 *Let $f(\theta, g, y, u) : \mathbb{R}^{n_p} \times \mathcal{G} \times \mathcal{Y} \times \mathcal{U} \mapsto \mathbb{R}$. Then using a soft-max output distribution we can write*

$$\mu(u|\theta, h, y) = \frac{\exp(f(\theta, g, y, u))}{\sum_{u' \in \mathcal{U}} \exp(f(\theta, g, y, u'))}.$$

If $\mu(u|\theta, h, y)$ fulfils the conditions of Lemma 12, and if we can write $\mu(u|\theta, h, y)$ after a step in the gradient direction as

$$\mu(u|\theta, h, y) = \frac{\exp(f(\theta, g, y, u) + f(\Delta\theta, y, u))}{\sum_{u' \in \mathcal{U}} \exp(f(\phi, g, y, u') + f(\Delta\theta, y, u'))},$$

then $\|\nabla^{\phi}\eta\| = 0$ always.

Proof The theorem is a generalisation of Lemma 14. By allowing an arbitrary function $f(\theta, g, y, u)$ in place of θ_{ugy} and if a parameter step of $\Delta\theta$ is separable so that

$$f(\theta + \Delta\theta, g, y, u) = f(\theta, g, y, u) + f(\Delta\theta, y, u),$$

then the same argument as Lemma 14 holds. Note that we require the second term to be independent of g so that it takes the place of $d(u)$ in Lemma 14. ■

Using Theorem 15 we can easily show that 0 gradient problems exist for linear controllers

$$f(\theta, g, y, u) = \sum_{l=1}^{n_f} \psi_l(g, y)\theta_{gu}.$$

that is, a controller with n_f features $\{\psi_1(g, y), \ldots, \psi_{n_f}(g, y)\}$ and parameters θ_{lu} where $l = 1, \ldots, n_f, u = 1, \ldots, n_u$. Linear controllers are separable as required by the corollary.

One simple choice of $n_o + n_f + 1$ features we used in early experiments is

$$f(\theta, g, y, u) = \begin{cases} \chi_b(g) & g \leq n_g \\ \chi_y(g - n_g) & n_g < g \leq n_g + n_o \\ 1 & g = n_g + n_o + 1 \end{cases}$$

These features turn on a weight for each of the current I-state and observation as well as a bias weight, so we can alternatively write f as

$$f(\theta, g, y, u) = \theta_{bu} + \theta_{yu} + \theta_u;$$

and the additional parameter step component as

$$f(\Delta\theta, g, y, u) = \gamma \left(\frac{\partial\eta}{\theta_{bu}} + \frac{\partial\eta}{\theta_{yu}} + \frac{\partial\eta}{\theta_u} \right).$$

Theorem 12 tells us this is equal to $f(\Delta\theta, g', y, u)$ for all g, g'. So all conditions of Corollary 15 are satisfied with appropriate choices of initial θ and ϕ implying $\|\nabla^{\phi}\eta\| = 0$ always.

We can also show a simple example that demonstrates that not all linear controllers are subject to the 0 gradient regions of parameter space we have analysed.[7] Let there be only 1 feature $\psi(\theta, g, y, u) = g$. If we set $\omega(h|\phi, g, y) = 1/n_g$ and $\theta = 0$ then all conditions for Theorem 11 are met, but $f(\Delta\theta, g, y, u) = g\Delta\theta_u$ which is not independent of g as required for Corollary 15.

As a final linear example, we retrieve Theorem 14 from Corollary 15 by defining $n_f = n_g n_o$ input features:

$$\psi_{\bar{g}\bar{y}}(g, y) = \chi_{\bar{g}}(g)\chi_{\bar{y}}(y);$$

which is simply a binary input for every possible combination of observation and internal state.

Appendix B. GPOMDP and Gradient Ascent Techniques

This section provides details of the original GPOMDP algorithm and also provides details of the gradient ascent techniques we use including the conjugate gradient algorithm and the GSEARCH algorithm.

B.1 The GPOMDP Algorithm

The GPOMDP algorithm is a policy-gradient algorithm for learning memoryless policies for controlling POMDPs (Baxter and Bartlett, 2001). It extends REINFORCE to infinite-horizon problems by eliminating the need to identify at least one recurrent state. The IState-GPOMDP algorithm is a generalisation of GPOMDP to include a finite state controller. To retrieve the original GPOMDP algorithm set $n_g = 1$, that is, only one internal state which makes the policy memoryless.

B.2 The GSEARCH Algorithm

Once a gradient estimate has been obtained the conjugate gradient algorithm (Fine, 1999, §5.5.2) computes the search direction h. The conjugate gradient algorithm (Algorithm 5) ensures that successive search directions are orthogonal, avoiding problems such as successive parameter updates "bouncing" between opposite walls of a ridge instead of in an uphill direction along the ridge top.

The parameters are updated using $\theta_{n+1} = \theta_n + h\gamma$ where γ is a positive step size. To choose γ we perform an exponential line search, trying a sequence of exponentially increasing values $\{\gamma_1, \gamma_2, \dots\}$, attempting to bracket the maximum value of η in the search direction. After bracketing the maxima quadratic interpolation is used to compute the final value for γ. The estimate of η at each γ_m is noisy due to the use of simulation. A more robust solution is to compute another gradient estimate $\nabla\eta(\theta_n + h\gamma_m)$. Let ϵ be the machine tolerance for 0. While

$$\operatorname{sgn}(\nabla\eta(\theta_n + h\gamma_m) \cdot h) > \epsilon$$

7. The use of the soft-max function means the gradient approaches 0 in the limit as $f(\theta, g, y, u) \to \infty$. Other 0 gradient situations may exist and are a well known problem of gradient optimisation.

Algorithm 5 ConjGrad$(\nabla, \theta, s_0, \epsilon)$

```
1: Given:
     • $\nabla\colon \mathbb{R}^K \to \mathbb{R}^K$: a (possibly noisy and biased) estimate of the gradient of the objective function to be maximised.
     • Starting parameters $\theta \in \mathbb{R}^K$ (set to maximum on return).
     • Initial step size $s_0 > 0$.
     • Gradient resolution $\epsilon$.
2: $g = h = \nabla(\theta)$
3: while $\|g\|^2 \geq \epsilon$ do
4:   GSearch$(\nabla, \theta, h, s_0, \epsilon)$
5:   $\Delta = \nabla(\theta)$
6:   $\gamma = (\Delta - g) \cdot \Delta / \|g\|^2$
7:   $h = \Delta + \gamma h$
8:   if $h \cdot \Delta < 0$ then
9:     $h = \Delta$
10:  end if
11:  $g = \Delta$
12: end while
```

we have not yet bracketed the maxima. Because we only care about the sign of the calculation a poor trial step gradient can be tolerated, allowing the search to be performed with fewer simulation steps than a value based search requires. See Algorithm 6 for details.

B.3 Quadratic Penalties

Gradient estimates early in learning often point to local maxima. Because line search algorithms attempt to find the γ that maximises η in one iteration, it is easy to end up caught in a local maximum. For output distribution functions such as (10) local maxima are associated with large parameter values. We can avoid these local maxima by adding a constraint term to the search direction which penalises large parameter values.

For example, in the multi-agent problem of Section 3.5 the initial gradient direction tells the agents to move forward when the sensors fail. Without a penalty term the line search pushes the parameters for this situation to values which result in near 0 gradients for those parameters during subsequent $\nabla\eta$ estimations. The end result is that the agents never learn to wait. However, if we restrict the initial parameter growth, the agents have a chance to correct their policies. An alternative intuition arises by observing that large parameter values result in near deterministic policies, reducing exploration during simulation. If an agent does not experience the preferred policy it cannot learn it.

We use a quadratic penalty term. Let $q \in [0, \infty)^{n_p}$ be a vector of penalties, with q_x giving the penalty for parameter θ_x. Let Q be a square matrix with q along the diagonal

Algorithm 6 GSearch($\nabla, \theta_0, \theta^*, s_0, \epsilon$)

```
1: Given:
     • $\nabla\colon \mathbb{R}^K \to \mathbb{R}^K$: a (possibly noisy and biased) estimate of the gradient of the objective function.
     • Starting parameters $\theta_0 \in \mathbb{R}^K$ (set to maximum on return).
     • Search direction $\theta^* \in \mathbb{R}^K$ with $\nabla(\theta_0) \cdot \theta^* > 0$.
     • Initial step size $s_0 > 0$.
     • Inner product resolution $\epsilon >= 0$.
2: $s = s_0$
3: $\theta = \theta_0 + s\theta^*$
4: $\Delta = \nabla(\theta)$
5: if $\Delta \cdot \theta^* < 0$ then
6:   Step back to bracket the maximum:
7:   repeat
8:     $s_+ = s$
9:     $p_+ = \Delta \cdot \theta^*$
10:    $s = s/2$
11:    $\theta = \theta_0 + s\theta^*$
12:    $\Delta = \nabla(\theta)$
13:  until $\Delta \cdot \theta^* > -\epsilon$
14:  $s_- = s$
15:  $p_- = \Delta \cdot \theta^*$
16: else
17:  Step forward to bracket the maximum:
18:  repeat
19:    $s_- = s$
20:    $p_- = \Delta \cdot \theta^*$
21:    $s = 2s$
22:    $\theta = \theta_0 + s\theta^*$
23:    $\Delta = \nabla(\theta)$
24:  until $\Delta \cdot \theta^* < \epsilon$
25:  $s_+ = s$
26:  $p_+ = \Delta \cdot \theta^*$
27: end if
28: if $p_- > 0$ and $p_+ < 0$ then
29:   $s = s_- - p_- \frac{s_+ - s_-}{p_+ - p_-}$
30: else
31:   $s = \frac{s_- + s_+}{2}$
32: end if
33: $\theta_0 = \theta_0 + s\theta^*$
```

and 0 elsewhere. We define the penalised η as

$$\bar{\eta} := \eta - \frac{\theta Q \theta'}{2}$$
$$\frac{\partial \bar{\eta}}{\theta_x} = \frac{\partial \eta}{\theta_x} - q_x \theta_x.$$

In practice we use the same penalty for all parameters, though in some situations, such as having independent sets of parameters θ and ϕ, non-uniform q could aid convergence.

To eventually settle into a maximum we reduce the penalty over time. We halve the penalty if $\bar{\eta}$ fails to increase by 2% over 3 or more iterations of GSEARCH.

References

V M Aleksandrov, V I Sysoyev, and V V Shemeneva. Stochastic optimaization. *Engineering Cybernetics*, 5:11–16, 1968.

Howard Anton and Chris Rorres. *Elementary Linear Algebra: applications version.* Wiley, New York, NY., 6 edition, 1991.

K J Åström. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10, 1965.

Leemon C. Baird and Andrew W. Moore. Gradient descent for general reinforcement learning. In *Advances in Neural Information Processing Systems 11.* MIT Press, 1999. `http://www.cs.cmu.edu/~leemon/papers/index.html`.

Peter L. Barlett and Jonathan Baxter. Estimation and approximation bounds for gradient based reinforcment learning. In *Thirteenth Annual Conference on Computational Learning Theory*, 2000. `http://discus.anu.edu.au/~bartlett/`.

Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.

Jonathan Baxter, Peter L. Bartlett, and Lex Weaver. Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:351–381, 2001.

Yoshua Bengio and Paolo Frasconi. Input-output HMM's for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, 1996. URL `citeseer.nj.nec.com/bengio95inputoutput.html`.

C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations, 1996.

Ronen I. Brafman. A heuristic variable grid solution method for POMDPs. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI '97)*, 1997.

Leo Breiman. *Probability.* Addison-Wesley, 1966.

Anthony Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes.* PhD thesis, Brown University, May 1998.

Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *National Conference on Artificial Intelligence*, pages 183–188, 1992. `citeseer.nj.nec.com/chrisman92reinforcement.html`.

Terrence L Fine. *Feedforward Neural Network Methodology.* Springer, New York, 1999.

Héctor Geffner and Blai Bonet. Solving large pomdps by real time dynamic programming. Working Notes Fall AAAI Symposium on POMDPS, 1998. `http://www.cs.ucla.edu/~bonet/`.

Matthew R. Glickman and Katia Sycara. Evolutionary search, stochastic policies with memory, and reinforcement learning with hidden state. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 194–201. Morgan Kaufmann, June 2001.

Peter W Glynn. Stochastic approximation for monte-carlo optimization. In *Proceedings of the 1986 Winter Simulation Conference*, pages 356–365, 1986.

Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33:75–84, 1990.

Peter W. Glynn. Importance sampling for monte carlo estimation of quantiles. Technical report, Dept. of Operations Research, Stanford University, 1996. URL `citeseer.nj.nec.com/glynn96importance.html`.

Peter W Glynn and Paul L'Ecuyer. Likelihood ratio gradient estimation for regenerative stochastic recursions. *Advances in Applied Probability, 27, 4 (1995)*, 27:1019–1053, 1995.

Anne Greenbaum. *Iterative Methods for Solving Linear Systems.* Frontiers in applied mathemtics. SIAM, Philadelphia, PA, 1997. ISBN 0-89871-396-X.

Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. In *Advances in Neural Information Processing Systems 13 (2001)*, Vancouver, BC, December 2002. MIT Press. In press.

Eric A. Hansen and Zhengzhu Feng. Dynamic programming for POMDPs using a factored state representation. In *Fith International Conference on Artificial Intelligence Planning and Scheduling*, pages 130–139, Breckenridge, Colorado, April 2000. URL `citeseer.nj.nec.com/hansen00dynamic.html`.

Milos Hauskrecht. Incremental methods for computing bounds in partially observable markov decision processes. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 734–739, Providence, Rhode Island, 1997. MIT Press. ISBN 0-262-51095-2.

Milos Hauskrecht. Value-function approximations for partially observable markov decsion processes. *Journal of Artificial Intelligence Research*, 13:33–94, August 2000.

Ilse C. F. Ipsen and Carl D. Meyer. The idea behind Krylov methods. *American Mathematical Monthly*, 105(10):889–899, 1998. URL `citeseer.nj.nec.com/135899.html`.

Tommi Jaakkola, Satinder P. Singh, and Michael I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 345–352. The MIT Press, 1995. URL `citeseer.nj.nec.com/jaakkola95reinforcement.html`.

Leslie Pack Kaelbling, Michael L.Littman, and Andrew W.Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, (4):237–285, May 1996.

Hajime Kimura and Shigenobu Kobayashi. Reinforcement learning for continuous action using stochastic gradient ascent. In *Intelligent Autonomous Systems (IAS-5)*, pages 288–295, 1998.

Hajime Kimura, Kazuteru Miyazaki, and Shigenobu Kobayashi. Reinforcement learning in POMDPs with function approximation. In *Proc. 14th International Conference on Machine Learning*, pages 152–160. Morgan Kaufmann, 1997.

David Kincaid and Ward Cheney. *Numerical Analysis*. Brooks/Cole Publishers, Pacific Grove, California, 1991. ISBN 0-534-13014-3.

V. Konda and J. Tsitsiklis. Actor-critic algorithms, 2000. URL `citeseer.nj.nec.com/434910.html`.

Pier Luca Lanzi. Adaptive agents with reinforcement learning and internal memory. In *Sixth International Conference on the Simulation of Adaptive Behavior (SAB2000)*, 2000. URL `citeseer.nj.nec.com/346913.html`.

Long-Ji Lin and Tom M. Mitchell. Memory approaches to reinforcement learning in non-Markovian domains. Technical Report CS-92-138, Carnegie Mellon, Pittsburgh, PA, 1992.

Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, San Francisco, CA, 1995. Morgan Kaufmann. URL `citeseer.nj.nec.com/littman95learning.html`.

Perter Marbach and John N. Tsitsiklis. Gradient-based optimisation of markov reward processes: Practical variants. In *38th IEEE Conference on Decsions and Control*, December 1999.

Peter Marbach. *Simulation-Based Methods for Markov Decision Processes*. PhD thesis, Laboratory for Information and Decision Systems, MIT, 1998.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of ICML-2000*, 2000. `http://www.cs.cmu.edu/~mccallum/`.

Andrew Kachites McCallum. *Reinformcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, 1996.

Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 127–136. Computer Science Dept., Brown University, Morgan Kaufmann, July 1999a.

Nicolas Meuleau, Leonid Peshkin, Kee-Eung Kim, and Leslie Pack Kaelbling. Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. Computer Science Dept., Brown University, Morgan Kaufmann, July 1999b.

David E. Moriarty, Alan C. Schultz, and John J. Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, (11):199–229, August 1999.

Kevin P. Murphy. A survey of pomdp solution techniques. Technical report, Dept. of Computer Science, U.C.Berkeley, September 2000. `http://www.cs.berkeley.edu/~murphyk/publ.html`.

I. Nourbakhsh, R. Powers, and S. Birchfield. DERVISH an office-navigating robot. *AI Magazine*, 16(2):53–60, 1995.

Katsuhiko Ogata. *Modern Control Engineering*. Prentice-Hall, New Jersey, U.S., 2nd edition, 1990. ISBN 0-12-589128-0.

Luis E. Ortiz and Leslie Pack Kaelbling. Adaptive importance sampling for estimation in structured]domains. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Articial Intelligence (UAI2000)*, pages 446–454. Morgan Kaufmann Publishers, 2000.

Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

Ronald Parr and Stuart Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1088–1094. Morgan Kaufmann, 1995. URL `citeseer.nj.nec.com/parr95approximating.html`.

Leonid Peshkin, Nicolas Meuleau, and Leslie Kaelbling. Learning policies with external memory. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference in Machine Learning*, pages 307–314. Morgan Kaufmann, 1999. `http://www.ai.mit.edu/~pesha/Public/papers.html`.

Leonid Peshkin and Christian R. Shelton. Learning from scarce experience. `http://www.ai.mit.edu/~pesha/Public/papers.html`, 2001.

Leonid M. Peshkin. Thesis proposal: Architectures for policy search. `http://www.ai.mit.edu/~pesha/Public/papers.html`, July 2000.

Alan B. Poritz. Hidden markov models: A guided tour. In *ICASSP '88*, pages 7–13. Morgan Kaufmann, 1988.

Pascal Poupart and Craig Boutilier. Vector-space analysys of belief-state approximation for POMDPs. In *Uncertainty in Artificial Intelligence 2001*, August 2001.

Pascal Poupart, Luis E. Ortiz, and Craig Boutilier. Value-directed sampling methods for monitoring pomdps. In *Uncertainty in Artificial Intelligence 2001*, August 2001. URL `citeseer.nj.nec.com/445996.html`.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech processing. In *Proceedings on the IEEE*, volume 77. IEEE, February 1989.

M I Reiman and A Weiss. Sensitivity analysis via likelihood ratios. In *Proceedings of the 1986 Winter Simulation Conference*, 1986.

M I Reiman and A Weiss. Sensitivity analysis for simulations via likelihood ratios. *Operations Research*, 37, 1989.

Andrés Rodríguez, Ronald Parr, and Daphne Koller. Reinforcement learning using approximate belief states, 1999.

Dana Ron, Yoram Singer, and Naftali Tishby. The power of amnesia. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 176–183. Morgan Kaufmann Publishers, Inc., 1994. URL `citeseer.nj.nec.com/ron94power.html`.

Reuven Y. Rubinstein. *Some Problems in Monte Carlo Optimization.* PhD thesis, 1969.

Brian Sallans. Learning factored representations for partially observable Markov decision processes. In S. A. Solla, T. K. Leen, and K-R. Muller, editors, *Neural Information Processing Systems*, volume 12. MIT Press, 2000. URL `citeseer.nj.nec.com/sallans00learning.html`.

Christian R. Shelton. Importance sampling estimates for policies with memory. Uncertainty in Artificial Intelligence Workskop, August 2001a. `http://www.ai.mit.edu/people/cshelton/papers/`.

Christian R. Shelton. Policy imporovemnt for pomdps using normalized importance sampling. Technical Report AI Memo 2001-002, MIT, Cambridge, MA, March 2001b. `http://www.ai.mit.edu/people/cshelton/papers/`.

Christian R. Shelton. Policy improvement for pomdps using normalized importance sampling. In *Uncertainty in Artificial Intelligence*, August 2001c.

S. Singh, T. Jaakkola, and M. Jordan. Learning without state-estimation in partially observable Markovian decision processes. In *Proceedings of ICML-11*, 1994.

Satinder P. Singh, Tommi Jaakkola, and Michael I. Jordan. Reinforcement learning with soft state aggregation. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 361–368. The MIT Press, 1995. URL `citeseer.nj.nec.com/article/singh95reinforcement.html`.

Edward J. Sondik. *The Optimal Control of Paritally Observable Markov Decision Processes.* PhD thesis, Stanford University, Standford, CA., 1971.

Edward J Sondik. The optimal control of partially observable Markov decision processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304, 1978.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge MA, 1998. ISBN 0-262-19398-1.

Richard S. Sutton, David McAllester, and Yishay Mansour Satinder Singh. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12:1057–1063, 2000.

Georgios Theocharous, Khashayar Rohanimanesh, and Sridhar Mahadevan. Learning and planning with hierarchical stochastic models for robot navigation. In *ICML 2000 Workshop on Machine Learning of Spatial Knowledge*, Stanford, 2000.

Sebastian Thrun. Monte Carlo POMDPs. In S. A. Solla, T. K. Leen, and K-R. Müller, editors, *Advances in Neural Information Processing Systems 12.* MIT Press, 2000. `http://citeseer.nj.nec.com/thrun99monte.html`.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

Nevin L. Zhang and Wenju Liu. Planning in stochastic domains: Problem characteristics and approximation. Technical Report HKUST-CS96-31, Dept. of Computer Science, Hong Kong University of Science and Technology, 1996. URL `citeseer.nj.nec.com/article/zhang96planning.html`.