Scaling Internal-State Policy-Gradient Methods for POMDPs

Douglas Aberdeen DOUGLAS.ABERDEEN@ANU.EDU.AU
Research School of Information Science and Engineering, Australian Nat. University, ACT 0200, Australia

Jonathan Baxter JBAXTER@PANSCIENT.COM
Panscient Pty Ltd, Adelaide, Australia

Abstract

Policy-gradient methods have received increased attention recently as a mechanism for learning to act in partially observable environments. They have shown promise for problems admitting memoryless policies but have been less successful when memory is required. In this paper we develop several improved algorithms for learning policies with memory in an infinite-horizon setting — directly when a known model of the environment is available, and via simulation otherwise. We compare these algorithms on some large POMDPs, including noisy robot navigation and multi-agent problems.

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, particularly observation noise. The only performance feedback given to the agents is a scalar reward signal. The rewards can be noisy and delayed from the actions that caused them. The agent's goal is to learn a policy that maximises the long-term average reward.

Unlike the situation with fully observable MDPs, the problem of finding optimal policies for POMDPs is PSPACE-complete (Papadimitriou & Tsitsiklis, 1987), even when the dynamics are 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 probability distributions over the state space or *belief 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.

All three algorithms use finite state controllers (FSCs) to store information about the history of observations. Both the transitions between the internal states of the controller given an observation, and the distribution over the agent's actions given a particular internal state, are governed by parameterised classes of functions. All three algorithms are *policy-gradient* algorithms in the sense that the agent adjusts the function 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 FSC. The third algorithm—Exp-GPOMDP—reduces the variance of IState-GPOMDP by computing true expectations over internal-state trajectories.

We give convergence theorems and present experimental results for several different problems including a multi-agent task with 21,632 states requiring cooperation for its solution. We have omitted proofs for brevity: they may be found in the technical report Aberdeen and Baxter (2002).

2. POMDPs with Internal-State

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. A natural model for this problem is that of POMDPs. For ease of exposition we consider finite POMDPs,[1] consisting of: states $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$

[1] Extensions to infinite POMDPs are relatively straightforward, but add significant mathematical complexity

of the world; actions $\mathcal{U} = \{1, \ldots, |\mathcal{U}|\}$ available to the policy in each state; observations $\mathcal{Y} = \{1, \ldots, |\mathcal{Y}|\}$ and a (possibly stochastic) reward $r(i)$ for each state $i \in \mathcal{S}$.

Each action $u \in \mathcal{U}$ determines a stochastic matrix $Q(u) = [q(j|i,u)]$ 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 with probability $\nu(y|i)$.

The policy has a finite set—$\mathcal{G} = \{1, \ldots, |\mathcal{G}|\}$—of internal states, or I-states, with which to store information about the observation history. The policy is stochastic with the probability of choosing action u given observation y, I-state g, and parameters $\theta \in \mathbb{R}^{n_\theta}$, written as $\mu(u|\theta, g, y)$. The I-state itself evolves stochastically as a function of the current observation and a second set of parameters $\phi \in \mathbb{R}^{n_\phi}$. The probability of making a transition from I-state $g \in \mathcal{G}$ to I-state $h \in \mathcal{G}$ is $\omega(h|\phi, g, y)$. With these definitions the agent may be viewed as a stochastic *finite state controller* or FSC. The evolution of world/I-state pairs (i, g) (Figure 1) is Markov, with an $|\mathcal{S}||\mathcal{G}| \times |\mathcal{S}||\mathcal{G}|$ transition probability matrix $P(\theta, \phi) = [p(j, h|\theta, \phi, i, g)]$ whose entries are given by

$$p(j, h|\theta, \phi, i, g) = \sum_{y,u} \nu(y|i)\omega(h|\phi, g, y)\mu(u|\theta, h, y)q(j|i, u). \tag{1}$$

Assumption 1. *Each $P(\theta, \phi)$ has a unique stationary distribution $\pi(\theta, \phi) := [\pi_{1,1}(\theta, \phi), \ldots, \pi_{|\mathcal{S}|,|\mathcal{G}|}(\theta, \phi)]'$ satisfying $\pi'(\theta, \phi)P(\theta, \phi) = \pi'(\theta, \phi)$.*

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

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

Assumption 4. *The ratios*

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

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

Parameterized distributions satisfying the final two assumption include the softmax distribution.

We seek $\theta \in \mathbb{R}^{n_\theta}$ and $\phi \in \mathbb{R}^{n_\phi}$ maximising the *long-term average reward*

$$\eta(\theta, \phi) := \lim_{T\to\infty} \frac{1}{T}\mathbb{E}_{\theta,\phi}\left[\sum_{t=1}^{T} r(i_t)\right], \tag{2}$$

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

$$\eta(\theta, \phi) = \sum_{i=1}^{|\mathcal{S}|}\sum_{g=1}^{|\mathcal{G}|} \pi_{i,g}(\theta, \phi)r(i) = \pi'(\theta, \phi)r, \tag{3}$$

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

without fundamentally altering the model.

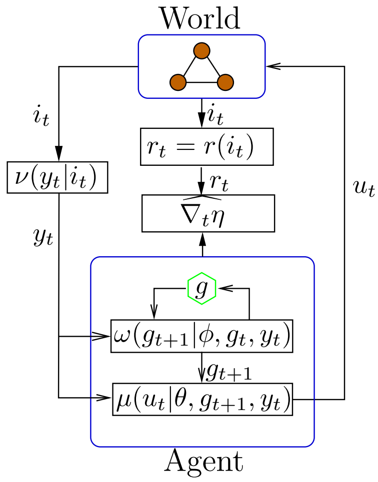

Figure 1. Evolution of a finite state POMDP controller.

2.1 Policy-Gradient Approaches

Policy-gradient methods do not seek to associate values with policies, instead they adjust the policy parameters ϕ and θ directly in the direction of increasing average reward, $\eta(\theta, \phi)$. Under mild assumptions, they ensure convergence to at least a local maximum of $\eta(\theta, \phi)$. This behaviour is in contrast to value-function based methods that can diverge, especially when using function approximation (Baird & Moore, 1999).

By adjusting the FSC parameters ϕ, the agent learns to use I-states to remember only what is needed to act optimally. This process is an automatic quantisation of the *belief state* space, providing the best policy *representable by $|\mathcal{G}|$ I-states*. As $|\mathcal{G}| \to \infty$ we can represent the optimal policy arbitrarily accurately. Learning FSC parameters is equivalent to searching the space of *policy graphs* (Meuleau et al., 1999). Policy search appeals intuitively because it is often easier to learn *how* to act than it is to learn the *value* of acting. In many cases it is possible for a large POMDP to be controlled well by a relatively simple policy graph, even though the corresponding value function is extremely complex.

3. Model-Based Policy-Gradient

In some situations an accurate model of the world may be available. For example, manufacturing plants are often well modelled. In this section we present an algorithm for computing estimates of the gradient of the average reward given a model of the world. This algorithm is feasible for many thousands of states, an improvement over existing model-based algorithms which typically handle at most hundreds of states(Geffner & Bonet, 1998).

Dropping the explicit dependence on θ, ϕ, we can rewrite (3) as $\eta = \pi' r$, hence $\nabla\eta = (\nabla\pi')r$, which should be understood as $n_\theta + n_\phi$ equations, one for each of the θ and ϕ parameters. Differentiating both sides of the balance equations $\pi' P = \pi'$ (Assumption 1), yields $(\nabla\pi')[I - P] = \pi'\nabla P$. We condition $[I - P]$ to make it invertible. Let e denote the $|\mathcal{S}||\mathcal{G}|$-dimensional column vector consisting of all "1s", and $e\pi'$ the $|\mathcal{S}||\mathcal{G}| \times |\mathcal{S}||\mathcal{G}|$ matrix with the stationary distribution π' in each row. Since $\sum_{i,g} \pi_{i,g} = 1$ we have $\sum_{i,g} \nabla\pi_{i,g} = 0$ and so $(\nabla\pi')e\pi' = 0$. Hence, we can write $(\nabla\pi')[I - P + e\pi'] = \pi'\nabla P$. Justification for the invertibility of $[I - P + e\pi']$ may be found in (Baxter & Bartlett, 2001). Substituting into the expression for $\nabla\eta$ yields

$$\nabla\eta = \pi'(\nabla P)\left[I - P + e\pi'\right]^{-1} r. \tag{4}$$

Now let $x := [I - P + e\pi']^{-1} r$, so that $\nabla\eta = \pi'(\nabla P)x$. In general, computing x is $O(|\mathcal{S}|^3|\mathcal{G}|^3)$; infeasible for $|\mathcal{S}||\mathcal{G}|$ greater than a few 100's of states. To finesse this difficulty we turn to approximate methods. Because $[I - P + e\pi']^{-1}$ is invertible we can approximate it with the series matrix expansion $x_N = \sum_{n=0}^{N} (P^n - e\pi')\, r$ (in the limit x_N approaches x), which is essentially Richardson iteration (Kincaid & Cheney, 1991, §4.6). Since $(\nabla P)e\pi' = 0$ we can simplify this to $x_N = \sum_{n=0}^{N} P^n r$. The sum can be efficiently evaluated by iteratively computing $v_{n+1} := P v_n$, with $v_0 := r$, and accumulating. Because this is a series of matrix-vector multiplications, the computation has worst case complexity $O(|\mathcal{S}|^2|\mathcal{G}|^2 N)$. P is typically sparse since only a subset of world-states $j \in \mathcal{S}$ have non-zero probabilities of being reached from some other state. For example, robots typically transit to a small set future states regardless of $|\mathcal{S}||\mathcal{G}|$. Thus practical POMDPs exhibit complexity closer to $O(c|\mathcal{S}||\mathcal{G}|N)$ where $c \ll |\mathcal{S}||\mathcal{G}|$.

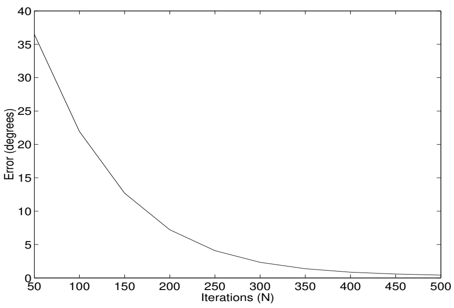

Figure 2. Angular error between $\widehat{\nabla_N\eta}$ after N iterations and the exact gradient for the Pentagon problem.

The remainder of the computation of $\nabla\eta$ involves computing $\pi'\nabla P$. Computing ∇P is in the worst case $O(|\mathcal{S}|^2|\mathcal{G}|^2(n_\theta + n_\phi)|\mathcal{Y}||\mathcal{U}|)$. Sparsity again helps since $q(j|i,u)$ and $\nu(y|i)$ are often 0, achieving practical complexities of $O(c|\mathcal{S}||\mathcal{G}|(n_\theta + n_\phi)|\mathcal{U}|)$ where $c \ll |\mathcal{S}||\mathcal{G}||\mathcal{Y}|$. The stationary distribution π is the leading left eigenvector of P, which is expensive to compute exactly. We use the power method: $\pi'_{n+1} = \pi'_n P$, stopping when $\|\pi'_{n+1} - \pi'_n\|_\infty < \epsilon$. The algorithm consisting of approximating x and π by iteration[2] and computing $\widehat{\nabla_N\eta} = \pi'(\nabla P)x_N$ has been named GAMP for Gradient Approximation of Modelled POMDPs.

3.1 Asymptotic Convergence of GAMP

Under assumption 1, P^N converges exponentially quickly to π. The exact rate is governed by the mixing time of the POMDP, defined as the smallest τ which satisfies $\max_i \sum_{j\in\mathcal{S}} |P^N_{ij} - \pi_j| \le \exp(-\lfloor \frac{N}{\tau} \rfloor)$. We can use τ to bound the estimate error after N iterations.

Theorem 1.

$$\|\widehat{\nabla_N\eta} - \nabla\eta\|_\infty \le BR|\mathcal{U}|\tau \frac{\exp(-\lfloor \frac{N}{\tau} \rfloor)}{1 - \exp(-1)}.$$

The difficulty of calculating τ for an arbitrary POMDP makes it hard to use this theorem to establish N *a priori*. In practice we check for convergence of x every k steps, stopping when $\|x_{N+k} - x_N\|_\infty < \epsilon$.[3] Figure 2 shows how quickly GAMP converges on a problem with 1045 states. We inverted $[I - P + e\pi']$ exactly for the Pentagon problem defined in Section 6.2 with $|\mathcal{S}| = 209$, $|\mathcal{G}| = 5$. Even the noisy Pentagon transition probabilities are very sparse with only 2.4% of P containing non-zero elements. This experiment was run on a Pentium II @ 433 MHz. When computing $\nabla\eta$: π takes 315 s (wall clock time), $[I - P + e\pi']^{-1}$

[2]Advanced iteration techniques such as Krylov subspace methods for x and the Lanczos method for π are worthy of further investigation.

[3]It is easy to verify that $\|x_{N+1} - x_N\|$ is decreasing.

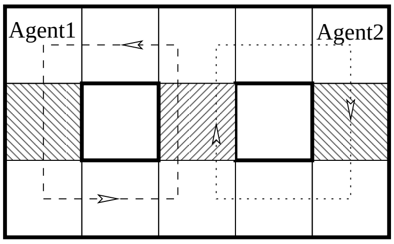

Figure 3. Plan of the factory floor with 13 locations. The dashed lines show an optimal policy.

takes 10.5 s and ∇P takes 36 s. When computing $\widehat{\nabla_N \eta}$: π takes 3.50 s ($\epsilon = 0.0001$, 1319 iterations), and Richardson matrix inversion takes 1.41 s ($N = 500$). 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 $|\mathcal{S}||\mathcal{G}|$ grows. Approximating π accounts for 0.016° of the error.

3.2 A Multi-Agent Problem

Figure 3 represents a factory floor occupied by 2 robots which are identical except that one is given priority in situations where both robots want to move into the same space. The four actions for each agent are {`move forward, turn left, turn right, wait`}, resulting in a cross product of $|\mathcal{U}| = 16$. One agent learns to move unfinished parts from the left shaded location to the middle, where the part is processed instantly, ready for the second agent to move the processed part from the middle to the right shaded location. Agents only need to exit the shaded locations to load or unload. Each agent can be empty or carrying a part in any of 13 locations and 4 orientations ($2 \times 13 \times 4 = 104$ states). The global state is the state of the 2 agents plus whether a part is waiting at the middle, giving $104^2 \times 2 = 21,632$ states. A reward of 1 is received for dropping off a processed part at the right and 0 otherwise. To receive the maximum reward the robots must cooperate without explicit communication. The robots see 4 bits describing which directions are blocked by walls, and a 5th bit describing if the agent is in the top corridor (breaking the symmetry of the map), giving $|\mathcal{Y}| = 1024$. Uncertainty is added with a 10% chance of the robots' actions failing, resulting in no movement, and a 10% chance of the robots' sensors failing, receiving a "no walls" observation. This problem can be solved without internal state. Section 6 demonstrates GAMP on problems which require memory to solve.

These experiments were run on an AMD Athlon @ 1.3 GHz. GAMP required less than 47 Mb of RAM to run this problem. Unless stated, all experiments

Table 1. Results for multi-agent factory setting POMDP over 10 runs. η values are multiplied by 10^2.

Algorithm	*mean* η	*max* η	*secs to* $\eta = 5$
GAMP	6.51	6.51	1035
Hand	6.51	NA	NA

use policies and FSCs parameterised by a table indexed by (y, g). Each index provides a set $\mathcal{M}$ of $|\mathcal{G}|$ or $|\mathcal{U}|$ real numbers (initialised to 0), for $\omega(h|\phi, g, y)$ and $\mu(u|\theta, h, y)$ respectively. The *softmax* function is then used to generate distributions according to $Pr(m) = \frac{\exp(m)}{\sum_{m' \in \mathcal{M}} \exp(m')}$. Computing the gradient of the softmax 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. All experiments use Polak-Ribiére conjugate-gradient ascent (Fine, 1999, §5.5.2) with a line search (see Aberdeen and Baxter (2002)).

The agents learn to move in opposing circles (Figure 3) around the factory, reducing the chance of collision. They also learn to wait when their sensors fail, using the wait action to gather information. Table 1 shows GAMP consistently learns the hand designed policy.

It is important to note that GAMP does not use samples to estimate gradients. This is critical in applications where random world interactions are slow, expensive or dangerous. Fewer samples may be needed to estimate the model parameters, prior to running GAMP, than to run model free algorithms. Meuleau et al. (1999) computes the gradient of the *discounted* reward, avoiding evaluation of π, but running a matrix inverse iteration for each parameter. Thus we perform two iterative solves to their $n_\theta + n_\phi$.

4. Model-less Policy-Gradient

Without knowledge of the POMDP transition and observation probabilities, we resort to Monte-Carlo like methods for estimating $\nabla\eta$. These methods sample trajectories through the POMDP, gathering information about the true gradient at each step. The corresponding value-based algorithms, such as TD(λ), have been successful for large MDPs. We now develop the IState-GPOMDP algorithm which estimates $\nabla\eta$ by interacting with the world. The algorithm extends Williams REINFORCE with memory (Peshkin et al., 1999) to the infinite-horizon case.

4.1 The IState-GPOMDP Algorithm

Algorithm 1 computes estimates $[\Delta_T^\theta, \Delta_T^\phi]$ of an approximation to $\nabla\eta$. At each step an observation y_t is received and ω is evaluated 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 $\beta \in [0, 1)$ to give more weight to recent I-state choices. The same process with the new I-state chooses an action u_t. 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 more likely to be generated by recent actions. Williams REINFORCE avoids discount factors by assuming finite-horizon POMDPs. The following theorem establishes that IState-GPOMDP estimates *an approximation* to the gradient of η (2).

Theorem 2. *Define the discounted reward as*

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

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,* $\lim_{T\to\infty} \Delta_T = \pi'(\nabla P)J_\beta$ *w.p.1.*

The next theorem establishes that as $\beta \to 1$, the approximation $\pi'(\nabla P)J_\beta$ is $\nabla\eta$.

Theorem 3. *For P parameterised by ϕ and θ*

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

Thus $\Delta_T \xrightarrow{T\to\infty} \pi'\nabla P J_\beta \xrightarrow{\beta\to 1} \nabla\eta$, however the variance of Δ_T scales as $1/\left[T(1-\beta)\right]$, reflecting the increasing difficulty of the credit assignment problem as $\beta \to 1$. Fortunately, $\pi'(\nabla P)J_\beta$ 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 memoryless GPOMDP algorithm (Baxter & Bartlett, 2001) which is retrieved from IState-GPOMDP by setting $|\mathcal{G}| = 1$.

Algorithm 1 IState-GPOMDP

while $t < T$ **do**
  Observe y_t from the world.
  Draw g_{t+1} from ω and u_t from μ.
  $z_{t+1}^\phi = \beta z_t^\phi + \frac{\nabla\omega(g_{t+1}|\phi, g_t, y_t)}{\omega(g_{t+1}|\phi, g_t, y_t)}$
  $z_{t+1}^\theta = \beta z_t^\theta + \frac{\nabla\mu(u_t|\theta, g_{t+1}, y_t)}{\mu(u_t|\theta, g_{t+1}, y_t)}$
  $\Delta_{t+1}^\theta = \Delta_t^\theta + \frac{1}{t+1}\left[r(i_{t+1})z_{t+1}^\theta - \Delta_t^\theta\right]$
  $\Delta_{t+1}^\phi = \Delta_t^\phi + \frac{1}{t+1}\left[r(i_{t+1})z_{t+1}^\phi - \Delta_t^\phi\right]$
  Issue action u_t.
  $t{+}{+}$.
end while

4.2 Zero Gradient Regions

We observed that FSC methods with *small random initial parameters* fail to learn to use the FSC for non-trivial problems. In this case $\omega(h|\phi, g, y)$ and $\mu(u|\theta, h, y)$ are near uniform and therefore the distribution over I-state trajectories, given different observation histories, is also near uniform. Thus, it is difficult for the controller to distinguish high reward I-state trajectories and which I-state transitions to adjust in order to increase reward. This results in near zero gradient with respect to the internal-state parameters. The following theorem formalises this argument.

Theorem 4. *If we choose θ and ϕ such that* $\omega(h|\phi, g, y) = \omega(h|\phi, g', y)$ $\forall g, g', h, y$ *and* $\mu(u|\theta, h, y) = \mu(u|\theta, h', y)$ $\forall h, h', y, u$ *then* $\nabla^\phi\eta = 0$.

The conditions of the theorem may be violated by one iteration of gradient ascent. Aberdeen and Baxter (2002) establishes the slightly more restrictive conditions which result in *perpetual* failure to learn a finite state controller. We have investigated avoiding small gradient regions by using a large *sparse* I-state FSC where all I-states have out-degree $k \ll |\mathcal{G}|$. This trick ensures that distinct observation histories typically generate minimally overlapping distributions of I-state trajectories. The complexity of each IState-GPOMDP step grows linearly with k, not $|\mathcal{G}|$, allowing the use of many I-states to compensate for the loss in FSC richness. Section 6.1 contains an empirical demonstration of the effect of sparse transitions.

5. Internal Belief States

IState-GPOMDP discards information because it uses Monte-Carlo sampling to estimate the effect of internal-state transitions, even though the internal-state transition probabilities are known to the algorithm (they are given by ω). This results in an unnecessary increase in the variance of the algorithm's gradient estimates. To avoid this, at each time step we can update the probability of occupying each I-state and use this *belief over I-states* as the agent's memory. Our first attempt in this direction used Input/Output HMMs (Bengio & Frasconi, 1996), computing hidden state occupancy probabilities driven by the observations. The problem with existing IOHMM/POMDP algorithms such as Chrisman (1992), is that they ignore the most useful indicator of performance: the reward. Our initial approach used the IOHMM to *predict rewards*. However, this suffers from the same difficulties as value-function methods: the agent can learn to predict rewards well while failing to learn a good policy (Aberdeen & Baxter, 2002).

Exp-GPOMDP is an alternative which overcomes the limitations of predicting rewards. The algorithm is a partly Rao-Blackwellised version of IState-GPOMDP; computing the expectation only over I-state trajectories. Rao-Blackwellisation of Monte-Carlo estimators reduces their variance (Casella & Robert, 1996).

We replace μ with $\bar{\mu}$ which implicitly contains the I-state in the form of a belief over I-state occupation probabilities. The I-state update is still parameterised by a stochastic FSC described by ϕ, but the update to the internal *belief* state is not stochastic. Let $\alpha_t(g|\phi, y_t, \alpha_{t-1})$ be the probability that $g_t = g$ given the current parameters, observation and the previous I-state belief. The recursive I-state update is

$$\alpha_{t+1}(h|\phi, y, \alpha_t) = \sum_{g\in\mathcal{G}} \alpha_t(g|\phi, y, \alpha_{t-1})\omega(h|\phi, g, y). \tag{5}$$

If the initial world state of the system is deterministic then we make α_0 deterministic, otherwise we can set $\alpha_0(g) = 1/|\mathcal{G}|$, $\forall g$. The new form of the policy which takes the expectation of μ over internal states is

$$\bar{\mu}(u|\theta, \phi, y) := \sum_{g\in\mathcal{G}} \alpha_{t+1}(g|\phi, y, \alpha_t)\mu(u|\theta, g, y). \tag{6}$$

We compute $\nabla\bar{\mu}$ by application of the chain-rule to (6), where $\nabla\alpha_{t+1}$ is computed recursively from $\nabla\alpha_t$, shown in Algorithm 2. Each gradient estimation step has a complexity of at least $O(|\mathcal{G}|(|\mathcal{G}|+|\mathcal{U}|))$ compared to the IState-GPOMDP complexity of $O(|\mathcal{G}| + |\mathcal{U}|)$ per step. This becomes $O(|\mathcal{G}|(k + |\mathcal{U}|))$ using the sparse transition trick of Section 4.2, also overcoming the small gradient problem. Exp-GPOMDP has the same convergence properties as IState-GPOMDP.

Exp-GPOMDP is similar to the finite-horizon algorithm presented by Shelton (2001), an HMM gradient ascent algorithm where emissions are actions. In that paper the HMM backward probability is used as well as the forward probability α. In the infinite-horizon setting there is no natural point to begin the backward probability calculation so our algorithm uses only α.

Algorithm 2 Exp-GPOMDP

Set $z_0 = 0$, and $\Delta = 0$ ($z_0, \Delta_0 \in \mathbb{R}^{n_\theta + n_\phi}$).
for each $g \in \mathcal{G}$ **do**
  Set $\alpha_0(g) = 1/|\mathcal{G}|$ and $\nabla\alpha_0(g) = 0$.
end for
while $t < T$ **do**
  Observe y_t from the world.
  for each $h \in \mathcal{G}$ **do**
    $\alpha_{t+1}(h|\phi, \alpha_t) = \sum_{g\in\mathcal{G}} \alpha_t(g|\phi, \alpha_{t-1})\omega(h|\phi, g, y_t)$.
    $\nabla\alpha_{t+1}(h|\phi, \alpha_t) = \sum_{g\in\mathcal{G}} \nabla\alpha_t(g|\phi, \alpha_{t-1})\omega(h|\phi, g, y_t) + \alpha_t(g|\phi, \alpha_{t-1})\nabla\omega(h|\phi, g, y_t)$.
  end for
  Choose u_t from $\bar{\mu}(u_t|\theta, \phi, y_t) = \sum_{g\in\mathcal{G}} \alpha_{t+1}(g|\phi, \alpha_t)\mu(u|\theta, g, y_t)$.
  $z_{t+1} = \beta z_t + \frac{\nabla\bar{\mu}(u_t|\theta,\phi,y_t)}{\bar{\mu}(u_t|\theta,\phi,y_t)}$
  $\Delta_{t+1} = \Delta_t + \frac{1}{t+1}\left[r(i_{t+1})z_{t+1} - \Delta_t\right]$
  Issue action u_t.
  $t{+}{+}$.
end while

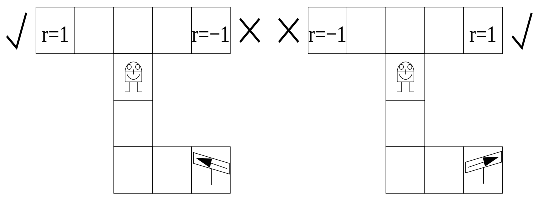

Figure 4. The Heaven/Hell problem: the optimal policy is to visit the signpost to find out whether to move left or right at the top-middle state in order to reach heaven.

6. Empirical comparisons

6.1 Heaven/Hell

The Heaven-Hell problem of Geffner and Bonet (1998) is shown in Figure 4. 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 is 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. In theory, 3 I-states are sufficient for optimal control. Two features make this problem hard: multi-stage memory is needed, and temporal credit assignment is difficult because the rewards are delayed up to 11 steps from relevant actions. We used lookup tables with softmax distributions yielding a total of 1540 parameters. The gradient estimation time in IState-GPOMDP was set to $T = 10^7$, with $\beta = 0.99$. We chose 20 internal states for the controller, yielding a total of $|\mathcal{S}||\mathcal{G}| = 400$.

To demonstrate the necessity of sparse I-state transitions (Section 4.2) the experiments were run with fully connected FSCs and randomly connected FSCs with an outdegree of $k = 3$. The randomness in the GAMP results comes only from the choice of initial FSC. Table 2 shows the results of these experiments. Both algorithms find the optimum policy, but only GAMP finds it consistently. The experiments with dense FSCs failed with GAMP producing gradient estimates within machine tolerance of 0.

Table 2. Results for the Heaven/Hell scenario over 10 runs. η values are multiplied by 10^2. Bracketed numbers indicate how many runs converged to $\eta = 5.0$.

Alg	*ave*	*max*	*var*	*secs to* $\eta = 5$
GAMP $k = 3$	9.01	9.09	0.0514	34 (10)
GAMP dense	0.005	0.005	$< .001$	(0)
IState $k = 3$	6.49	9.09	10.8	11436 (8)
IState dense	0.018	0.034	$< .001$	(0)
Optimum	9.09 (11 steps between rewards)			

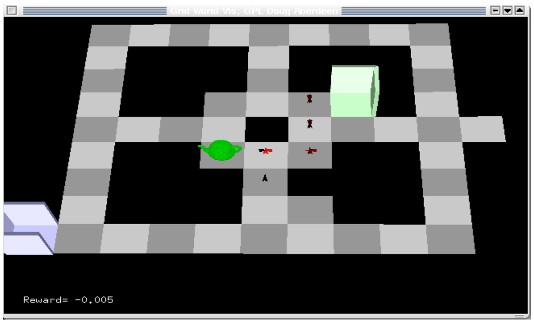

Figure 5. The Pentagon robot navigation problem maze.

The wall clock times to convergence quoted for these experiments are not directly comparable. IState-GPOMDP was run using 94 processors of a 550 MHz dual CPU PIII Beowulf cluster. GAMP was run on a 1.3 GHz Athlon, roughly equivalent to 3 CPUs of the cluster. This demonstrates the advantage of having a model of the world. Within our knowledge this is the first time the Heaven/Hell problem has been solved using a model-less algorithm.[4]

6.2 Pentagon

In this problem a robot must navigate through corridors to reach a goal (Figure 5). The world is mapped into discrete locations. The robot can point North, South, East or West. The actions are {`move forward`, `turn left`, `turn right`, `declare goal`} and they fail with 12% probability. There are 28 observations indicating if the locations in front and to the sides of the robot are reachable. Observations have a 12% chance of being wrong. We modified the original POMDP definition in (Cassandra, 1998) to make rewards control independent and to make the agent jump to the start state when it achieves the goal. No penalty is received for incorrect `declare goal` actions. The agent starts in the same place but noise means the agent quickly becomes confused about its location even with perfect memory of observations/actions. We have $|\mathcal{S}| = 209$ and $|\mathcal{G}| \leq 20$, giving at most 4180 states. The IState-GPOMDP algorithm required a minimum of 2×10^6 gradient estimation steps, while the Exp-GPOMDP algorithm required 10^6 steps. For comparison purposes we have reported results for Belief, a variation on IState-GPOMDP in which the observations y are replaced with the entire belief-state vector over world-states. Actions in Belief are generated by a softmax distribution that computes a linear combination of the world-state occupation probabilities for each possible action. Since the belief-state vector contains all information about observation histories, Belief does not maintain any internal state.

[4] We verified that Exp-GPOMDP can learn to visit the signpost. However, since 1 run takes $>$ 2 days on our cluster, we were prohibited from a formal comparison.

Table 3. Results on the Pentagon scenario over 10 runs. η values are multiplied by 10^2.

Alg	$\lvert\mathcal{G}\rvert$	*k*	*ave*	*max*	*var*	*secs to* 2
GAMP	5	2	2.55	2.70	0.0102	611 (10)
	10	2	2.50	2.63	0.0128	4367 (10)
	20	2	2.50	2.80	0.0250	31311 (10)
	20	3	2.89	3.00	0.00613	47206 (10)
IState	5	2	2.06	2.42	0.281	649 (9)
	10	2	2.18	2.37	0.0180	869 (9)
	20	2	2.12	2.28	0.0137	1390 (9)
	20	3	2.15	2.33	0.0138	1624 (9)
Exp	5	2	1.96	2.33	0.401	1708 (7)
	10	2	2.19	2.40	0.0151	6020 (9)
	20	2	2.11	2.27	0.0105	29640 (8)
	20	3	2.26	2.36	0.00448	48296 (10)
IState	1	1	1.35	1.37	0.00032	(0)
Belief			2.67	3.65	0.778	2313 (7)
MDP			4.93	5.01	0.00148	24 (10)

I-states approximate the policies achievable using a full belief state thus the Belief result of $\eta = 3.65$ (Table 3) is an approximate upper bound for the η we should be able to achieve using a large number of I-states. IState-GPOMDP with $|\mathcal{G}| = 1$ is a memoryless policy which should lower bound the I-state results. Values of η between these empirical bounds show that a finite number of I-states can be used to learn better than memoryless agents.

We ran our trained agents on the original POMDP to compare results with Cassandra (1998) which uses value-iteration and the most likely state heuristic. Our best Belief agent ($\eta = 3.65$) achieves a discounted reward of 0.764, requiring an average of 27 steps to achieve the goal. This result is bracketed by Cassandra's results of 0.791 (known start state) and 0.729 (uniform initial belief state).

Because the Belief algorithm has access to the world-state belief (using the model), it is unsurprising that it obtains the best maximum η. GAMP also uses a model but restricts memory to $|\mathcal{G}|$ states, resulting in the second highest maximum. GAMP may sometimes

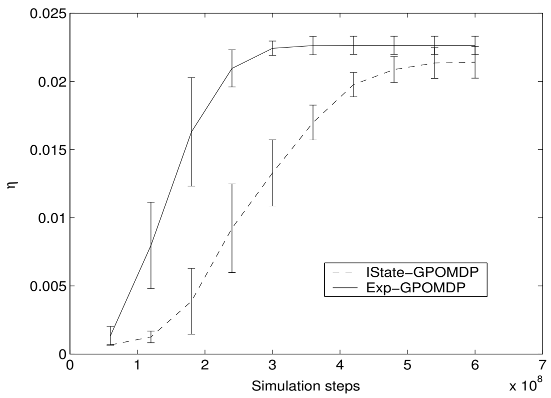

Figure 6. Convergence of IState-GPOMDP *vs.* Exp-GPOMDP for 10 Pentagon runs with $|\mathcal{G}| = 20$ and $k = 3$.

be preferable to Belief because of its zero-variance (the variance in the tables is due to the random choice of sparse FSC), supported by the fact that for $|\mathcal{G}| = 20$ the GAMP *mean* is better than Belief.

Increasing $|\mathcal{G}|$ and the FSC connectivity k generally improves the results due to the increasing richness of the parameterisation, however, IState-GPOMDP performed best for $|\mathcal{G}| < 20$ because we did not scale the number of gradient estimation steps with $|\mathcal{G}|$. Exp-GPOMDPs reduced variance allows it to improve consistently as $|\mathcal{G}|$ is increased while still using fewer estimation steps than IState-GPOMDP. Figure 6 shows that for $|\mathcal{G}| = 20$, Exp-GPOMDP produces a superior result in fewer steps than IState-GPOMDP. Setting $k > 3$ caused failures due to small initial gradients.

7. Conclusion

Memoryless algorithms are inadequate for all but trivial POMDPs. However, even for approximate algorithms, methods that use the belief state become intractable for large $|\mathcal{S}|$. We have proposed and compared 3 finite state controller algorithms which lie between memoryless algorithms and belief state algorithms. GAMP takes full advantage of a model to compute zero variance, low bias estimates of the performance gradient. Use of iterative approximations, sparse structures, and sparse FSCs allows GAMP to solve infinite-horizon POMDPs an order of magnitude larger than competing methods on a basic desktop computer. IState-GPOMDP and Exp-GPOMDP do not need a model and scale in two senses: the running time is constant per step and the number of simulation steps required does not necessarily increase with $|\mathcal{S}|$. Future work includes developing factored-state policy-gradient algorithms and incorporating further variance reduction methods.

References

Aberdeen, D., & Baxter, J. (2002). *Internal-state policy-gradient algorithms for infinite-horizon pomdps* (Technical Report). RSISE, Australian National University. `http://discus.anu.edu.au/~daa/papers.html`.

Baird, L. C., & Moore, A. W. (1999). Gradient descent for general reinforcement learning. *Advances in Neural Information Processing Systems 11*. MIT Press.

Baxter, J., & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, *15*, 319–350.

Bengio, Y., & Frasconi, P. (1996). Input-output HMM's for sequence processing. *IEEE Transactions on Neural Networks*, *7*, 1231–1249.

Casella, G., & Robert, C. P. (1996). Rao-Blackwellisation of sampling schemes. *Biometrika*, *83*, 81–94.

Cassandra, A. (1998). *Exact and approximate algorithms for partially observable Markov decision processes*. Doctoral dissertation, Brown University.

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. *National Conference on Artificial Intelligence* (pp. 183–188).

Fine, T. L. (1999). *Feedforward neural network methodology*. New York: Springer.

Geffner, H., & Bonet, B. (1998). Solving large POMDPs by real time dynamic programming. Working Notes Fall AAAI Symposium on POMDPs. `http://www.cs.ucla.edu/~bonet/`.

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

Kincaid, D., & Cheney, W. (1991). *Numerical analysis*. Pacific Grove, California: Brooks/Cole Publishers.

Meuleau, N., Kim, K.-E., Kaelbling, L. P., & Cassandra, A. R. (1999). Solving POMDPs by searching the space of finite policies. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence* (pp. 127–136). KTH, Sweden: Morgan Kaufmann.

Papadimitriou, C. H., & Tsitsiklis, J. N. (1987). The complexity of markov decision processes. *Mathematics of Operations Research*, *12*, 441–450.

Peshkin, L., Meuleau, N., & Kaelbling, L. (1999). Learning policies with external memory. *Proceedings of the Sixteenth International Conference in Machine Learning* (pp. 307–314). San Francisco: Morgan Kaufmann.

Shelton, C. R. (2001). Policy improvement for POMDPs using normalized importance sampling. *Uncertainty in Artificial Intelligence*. Seattle.