School of Informatics, University of Edinburgh

Institute for Adaptive and Neural Computation

Probabilistic inference for solving (PO)MDPs

by

Marc Toussaint, Stefan Harmeling and Amos Storkey

Informatics Research Report 0934

School of Informatics **December 2006**
http://www.informatics.ed.ac.uk/

Probabilistic inference for solving (PO)MDPs

Marc Toussaint, Stefan Harmeling and Amos Storkey

Informatics Research Report 0934

SCHOOL *of* INFORMATICS
Institute for Adaptive and Neural Computation

December 2006

Abstract : The development of probabilistic inference techniques has made considerable progress in recent years, in particular with respect to exploiting the structure (e.g., factored, hierarchical or relational) of discrete and continuous problem domains. We show that these techniques can be used also for solving Markov Decision Processes (MDPs) or partial observable MDPs (POMDPs) when formulated in terms of a structured dynamic Bayesian network (DBN). The approach is based on an equivalence between maximization of the expected future return in the time-unlimited MDP and likelihood maximization in a related mixture of finite-time MDPs. This allows us to use expectation maximization (EM) for computing optimal policies, using arbitrary inference techniques in the E-step. Unlike previous approaches we can show that this actually optimizes the discounted expected future return for arbitrary reward functions and without assuming an ad hoc finite total time.

We first develop the approach for standard MDPs and demonstrate it using exact inference on a discrete maze and Gaussian belief state propagation in non-linear stochastic optimal control problems. Then we present an extension for solving POMDPs. We consider an agent model that includes an internal memory variable used for gating reactive behaviors. Using exact inference on the respective DBN, the EM-algorithm solves complex maze problems by learning appropriate internal memory representations.

Keywords : Markov Decision Process, probabilistic inference, POMDP, planning

1 Introduction

The problems of planning in stochastic environments and inference in Markovian models are closely related, in particular in view of the challenges both of them face: scaling to large state spaces spanned by multiple state variables, or realizing planning (or inference) in continuous state spaces. Both fields developed techniques to address these problems. For instance, in the field of planning, they include work on Factored Markov Decision Processes (Boutilier et al. 1995; Koller & Parr 1999; Guestrin et al. 2003; Kveton & Hauskrecht 2005), abstractions (Hauskrecht et al. 1998), and relational models of the environment (Zettlemoyer et al. 2005). On the other hand, recent advances on inference techniques show how structure can be exploited both for exact inference as well as making efficient approximations. Examples are message passing algorithms (loopy BP, Expectation Propagation), variational approaches, approximate belief representations (particles, Assumed Density Filtering, Boyen-Koller) and arithmetic compilation (see, e.g., Minka 2001; Murphy 2002; Chavira et al. 2006).

In view of these similarities one may ask whether existing techniques for probabilistic inference can directly be translated to solving MDPs or POMDPs. From a complexity theoretic point of view, the equivalence between inference and planning is well-known (see, e.g., Littman et al. 2001). Bui, Venkatesh, & West (2002) have used inference on Abstract Hidden Markov Models for policy recognition, i.e., for reasoning about executed behaviors, but do not address the problem of computing optimal policies from such inference. Attias (2003) proposed a framework which suggests a straight-forward way to translate the problem of planning to a problem of inference: A Markovian state-action model is assumed, which is conditioned on a start state x_0 and a goal state x_T. Here, however, the total time T has to be fixed *ad hoc* and the MAP action sequence that is proposed as a solution is not optimal in the sense of maximizing an expected future reward. Verma & Rao (2006) used inference to compute plans (considering the maximal probable explanation (MPE) instead of the MAP action sequence) but again the total time has to be fixed and the plan is not optimal in the expected return sense.

We provide a framework that translates the problem of maximizing the discounted expected future return in the time-unlimited MDP (or general DBN) into a problem of likelihood maximization in a related mixture of finite-time MDPs. This allows us to use expectation maximization (EM) for computing optimal policies, utilizing arbitrary inference techniques in the E-step. Unlike previous approaches we can show that this actually optimizes the discounted expected future return for arbitrary reward functions and without assuming an ad hoc finite total time. The approach is generally applicable on any DBN-description of the problem whenever we have efficient inference techniques for this structure. DBNs allow us to consider structured representations of the environment (the world state) as well as the agent (or multiple agents, in fact).

The next two sections will first introduce this new approach in the context of an unstructured MDP. We establish the basic equivalence between maximizing expected future return and likelihood maximization. Section 3.4 also explains the relation to Policy Iteration in the unstructured MDP case. In section 4 we include some demonstrations of the basic approach using exact inference on a discrete maze and Gaussian belief state propagation in non-linear stochastic optimal control problems.

Then, in sections 5 and 6, we consider a non-trivial DBN representation of a POMDP problem. We propose a certain model of an agent (similar to finite state controllers, FSCs) that uses an internal memory variable as a sufficient representation of history which gates a reactive policy. In section 7 we use this to learn sufficient memory representations (e.g., counting aisles) and primitive reactive behaviors (e.g., aisle or wall following) in some partially observable maze problems. This can be seen in analogy to the classical Machine Learning paradigm of learning latent variable models of data for bootstrapping interesting internal representations (e.g., ICA) — but here generalized to the learning of latent variable models of behavior. Sections 8 and 9 will conclude with a more detailed discussion of our approach and related work.

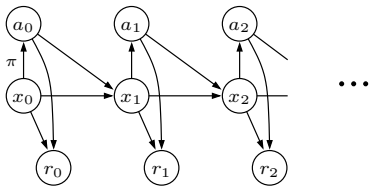

Figure 1: Dynamic Bayesian network for a MDP. The x states denote the state variables, a the actions and r the rewards.

2 Markov Decision Processes and likelihood maximization

A stationary Markov Decision Process (MDP, (Kaelbling, Littman, & Moore 1996)) is defined by the

initial state distribution	$P(x_0 = x) =: P_x$,
state transition	$P(x_{t+1} = x' \mid a_t = a, x_t = x) =: P_{x'ax}$,
rewards	$P(r_t = r \mid a_t = a, x_t = x) =: P_{rax}$,
policy	$P(a_t = a \mid x_t = x; \boldsymbol{\pi}) =: \pi_{ax}$.

We consider P_x, $P_{x'ax}$, and P_{rax} as known and call the expectation $R_{ax} = \mathrm{E}\{r \mid a, x\} = \sum_r r\, P_{rax}$ the reward function. In model-based Reinforcement Learning approaches such probabilities are estimated from experience (see, e.g., (Atkeson & Santamaría 1997)), but this is not addressed here. The random variables x_t and a_t can be discrete or continuous whereas the reward r_t is a real number. Figure 1 displays the dynamic Bayesian network for a time-unlimited Markov Decision Process (MDP).

The free parameter of this DBN is the policy $\boldsymbol{\pi}$ with numbers $\pi_{ax} \in [0, 1]$ normalized w.r.t. a. The problem we address is *solving the MDP*:

Definition 2.1. Solving an MDP means to find a parameter $\boldsymbol{\pi}$ of the graphical model in Figure 1 that maximizes the expected future return $V^{\boldsymbol{\pi}} = \mathrm{E}\{\sum_{t=0}^{\infty} \gamma^t\, r_t; \boldsymbol{\pi}\}$, where $\gamma \in [0, 1]$ is a discount factor.

The classical approach to solving MDPs is anchored in Bellman's equation, which simply reflects the recursive property of the future discounted return

$$\sum_{t=0}^{\infty} \gamma^t\, r_t = r_0 + \gamma \left[\sum_{t=1}^{\infty} \gamma^t\, r_t\right] \tag{1}$$

and consequently of its expectation conditioned on the current state,

$$V^{\boldsymbol{\pi}}(x) = \mathrm{E}\left\{\sum_{t=0}^{\infty} \gamma^t\, r_t \;\middle|\; x_0 = x; \boldsymbol{\pi}\right\} = \sum_{x',a} P_{x'ax}\, \pi_{ax}\, [R_{ax} + \gamma\, V^{\boldsymbol{\pi}}(x')] \,. \tag{2}$$

Standard algorithms for computing value functions can be viewed as iterative schemes that converge towards the Bellman equation or as directly solving this linear equation w.r.t. V by matrix inversion.

Our general approach is to translate the problem of solving an MDP into a problem of likelihood maximization. There are different approaches for such a translation. One issue to be considered is that the quantity we want to maximize (the expected future return) is a *sum* of expectations of every time slice, whereas the likelihood in Markovian models is usually the *product* of observation likelihoods in each time slice. Equivalence could be achieved, e.g., by introducing exponentiated rewards as observation likelihoods. However, we follow a different approach based on a mixture of finite-time MDPs. This mixture model is in some respects different to the

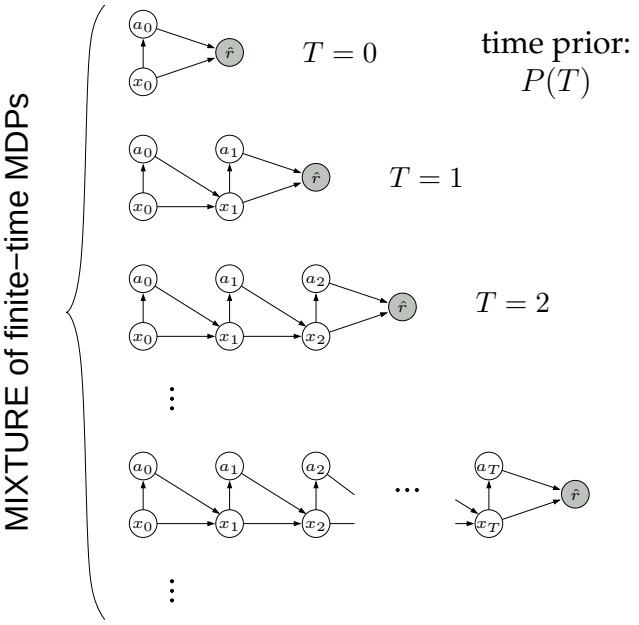

Figure 2: Mixture of finite-time MDPs.

original MDP but – and this is the important point – the likelihood of "observing reward" in this mixture model is proportional to the expected future return in the original MDP. The reasons for this choice of approach are related to inference (performing a backward pass) without pre-fixing a finite time horizon T, the handling of discounting rewards, and also to the resulting relations to standard Policy Iteration, as it will later become more clear.

We define each finite-time MDP as having the same probabilities P_x, $P_{x'ax}$ and π_{ax} as the original MDP but (i) it ends at a finite time T and (ii) it emits a single binary reward $\hat{r}$ only at the finial time step as given by

$$P(\hat{r}=1 \,|\, a_T=a, x_T=x) =: \widehat{R}_{ax} \,. \tag{3}$$

Figure 2 displays graphical models of some finite-time MDPs. Let $X = x_{0:T}$ and $A = a_{0:T}$ denote state and action trajectories of length T, then each finite-time MDP defines the joint

$$P(\hat{r}, X, A \,|\, T; \boldsymbol{\pi}) = P(\hat{r} \,|\, a_T, x_T)\; P(a_T \,|\, x_T; \boldsymbol{\pi}) \Big[\prod_{t=0}^{T-1} P(x_{t+1} \,|\, a_t, x_t)\; P(a_t \,|\, x_t; \boldsymbol{\pi}) \Big]\; P(x_0) \,. \tag{4}$$

Now let T be a random variable. The mixture of finite-time MDPs is given by the joint

$$P(\hat{r}, X, A, T; \boldsymbol{\pi}) = P(\hat{r}, X, A \,|\, T; \boldsymbol{\pi})\; P(T) \,. \tag{5}$$

Here, $P(T)$ is a prior over the total time. Note that this gives a distribution over random variables X and A of the space of variable length trajectories. We find

Theorem 2.1. *When introducing binary rewards $\hat{r}$ such that $\widehat{R}_{ax} \propto R_{ax}$ and choosing the discounting time prior $P(T) = \gamma^T(1-\gamma)$, maximizing the likelihood*

$$L^{\boldsymbol{\pi}} = P(\hat{r}=1; \boldsymbol{\pi}) \tag{6}$$

of observing reward in the mixture of finite-time MDPs is equivalent to solving the original MDP.

Given the way we defined the mixture, the proof is straight-forward and illustrated in Figure 3.

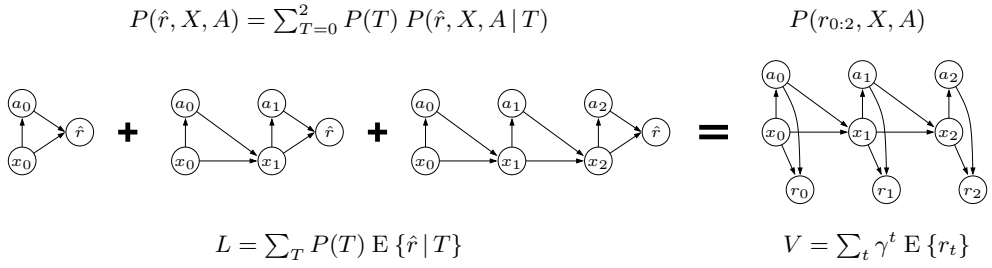

Figure 3: Equivalence between the mixture and the original MDP.

Proof. Each finite-time MDP defines a likelihood

$$L_T^{\boldsymbol{\pi}} = P(\hat{r}{=}1 \mid T; \boldsymbol{\pi}) = \mathrm{E}\{\hat{r} \mid T; \boldsymbol{\pi}\} \ ,$$

and for the full mixture of MDPs we have

$$L^{\boldsymbol{\pi}} = \sum_T P(T)\ L_T^{\boldsymbol{\pi}} = \sum_T P(T)\, \mathrm{E}\{\hat{r} \mid T; \boldsymbol{\pi}\} \ .$$

Since we chose $\widehat{R}_{ax}$ proportional to the reward function R_{ax} the expectations $\mathrm{E}\{\hat{r} \mid T; \boldsymbol{\pi}\}$ here are proportional to the reward expectations $\mathrm{E}\{r_t; \boldsymbol{\pi}\}$ at time slice $t = T$ of the original MDP. (We are taking the expectation w.r.t. a full probabilistic forward-sweep through the MDP, from time 0 to time T, given the policy $\boldsymbol{\pi}$; all MDPs share the same transition probabilities.) Hence

$$L^{\boldsymbol{\pi}} \propto \sum_t P(t)\, \mathrm{E}\{r_t; \boldsymbol{\pi}\} = (1-\gamma)\, V^{\boldsymbol{\pi}}$$

□

In appendix A the following points are discussed in some more details:

1. the interpretation of the mixture with death probabilities of the agent,
2. the difference between the models w.r.t. the correlation between rewards,
3. choosing the binary rewards such that $\widehat{R}_{ax} \propto R_{ax}$ for arbitrary reward functions,
4. and handling $\gamma = 1$.

3 EM-algorithm for unstructured MDPs

Formulating the objective function in terms of a likelihood allows us to apply expectation maximization (EM) to find optimal parameters $\boldsymbol{\pi}$. All action and state variables are considered hidden variables. The E-step will, for a given $\boldsymbol{\pi}$, compute posteriors over state-action sequences as well as T conditioned on "the data" $\hat{r} = 1$. The M-step then adapts the model parameters $\boldsymbol{\pi}$ to maximize the expected "data" log-likelihood (expectations then taken w.r.t. the posteriors calculated in the E-step). Conceptually, inference in this Markovian model is straight-forward. However, the special structure of the finite-time MDPs will allow for certain simplifications and spare us from performing separate inference sweeps in each finite-time MDPs. In particular, we will not have to pre-decide on a finite time horizon T_{max} before starting inference but rather can decide on a cutoff time on the fly, similar to deciding on a cutoff time during Policy Evaluation. For those reasons we discuss the inference process in some more detail below.

3.1 E-step

Forward-backward in all MDPs synchronously. In the E-step, we consider a fixed given policy $\boldsymbol{\pi}$ and all the quantities we compute depend on $\boldsymbol{\pi}$ even if not explicitly annotated. Let $P_{x'x} = P(x_{t+1}=x' \,|\, x_t=x;\boldsymbol{\pi}) = \sum_a P_{x'ax}\ \pi_{ai}$. For a single MDP of finite time T the standard forward and backward equations are

$$\alpha_0(x) = P_x\ , \qquad \alpha_t(x') = P(x_t=x;\boldsymbol{\pi}) = \sum_x P_{x'x}\ \alpha_{t-1}(x)\ , \tag{7}$$

$$\tilde{\beta}_T(x) = \sum_a \widehat{R}_{ax}\ \pi_{ax}\ , \qquad \tilde{\beta}_t(x) = P(\hat{r}=1 \,|\, x_t=x;\boldsymbol{\pi}) = \sum_{x'} P_{x'x}\ \tilde{\beta}_{t+1}(x')\ . \tag{8}$$

We find that all the α-quantities do not depend on T in any way, i.e., they are valid for all MDPs of any finite time T. This is not true for the $\tilde{\beta}$-quantities when defined as above. However, we can use a simple trick, namely define the β's to be indexed backward in time (with the 'time-to-go' τ), and get

$$\beta_\tau(x) := P(\hat{r}=1 \,|\, x_{T-\tau}=x;\boldsymbol{\pi})\ , \qquad \beta_0(x) = \sum_a \widehat{R}_{ax}\ \pi_{ax}\ , \qquad \beta_\tau(x) = \sum_{x'} P_{x'x}\ \beta_{\tau-1}(x')\ . \tag{9}$$

Defined in that way, all β-quantities do indeed not depend on T. For a specific MDP of finite time T, setting $\tau = T - t$ would allow us to retrieve the traditional forward-indexed $\tilde{\beta}$-quantities.

This means that we can perform the α- and β-propagation in parallel, incrementing t and τ synchronously, and start both without pre-determining T or some T_{max}. Although we introduce a mixture of MDPs we only have to perform a single forward and backward sweep. This procedure is, from the point of view of ordinary Hidden Markov Models, quite unusual—it is possible because in our setup we only condition on the very last state ($\hat{r}\ =\ 1$). Apart from realizing the additivity of $V^{\boldsymbol{\pi}}$ and the discounting by means of time prior, this is the main reason for why we considered the mixture of finite-time MDPs rather than the original time-unbounded MDP that emits rewards at every time slice.

Time posterior and choosing the cutoff. Say we have propagated α's forward for t iterations and β's backward for τ iterations. Then we can compute the T-conditioned likelihood for $T = t + \tau$

$$L^{\boldsymbol{\pi}}_{t+\tau} = P(\hat{r}=1 \,|\, T=t+\tau;\boldsymbol{\pi}) = \sum_x \alpha_t(x)\ \beta_\tau(x) \tag{10}$$

For instance, if there is no overlap between the distributions $\alpha_t(x)$ and $\beta_\tau(x)$ we have $L^{\boldsymbol{\pi}}_{t+\tau} = 0$ which means that there exists no trajectory of length $T = t+\tau$ from the initial start distribution P_x to rewards $\widehat{R}_{ax}$. In practice we will propagate α's and β's synchronously and compute $L^{\boldsymbol{\pi}}_{2k}$ after each step (both propagated for k steps). Monitoring $L^{\boldsymbol{\pi}}_{2k}$ on the fly allows us to decide on a cutoff time: as long as $L^{\boldsymbol{\pi}}_{2k} = 0$ we need to continue propagation. When $L^{\boldsymbol{\pi}}_{2k}$ is very small compared to the previously collected likelihood mass, $\gamma^{2k}\ L^{\boldsymbol{\pi}}_{2k} \ll \sum_{T=0}^{2k} \gamma^T\ L^{\boldsymbol{\pi}}_T$, we may choose to terminate the inference process and decide that the horizon $T_{\text{max}} = 2k$ is sufficient to estimate posteriors. Note that using Bayes rule we get the time posterior (which corresponds to model selection in the mixture model)

$$P(T \,|\, \hat{r}=1;\boldsymbol{\pi}) = \frac{P(\hat{r}=1 \,|\, T;\boldsymbol{\pi})}{P(\hat{r}=1;\boldsymbol{\pi})}\ P(T) = \frac{L^{\boldsymbol{\pi}}_T\ P(T)}{L^{\boldsymbol{\pi}}}\ . \tag{11}$$

Action, state, and state occupancy posteriors. Let

$$q_{t\tau}(a,x) = P(\hat{r}=1 \,|\, a_t=a, x_t=x, T=t+\tau;\boldsymbol{\pi}) = \begin{cases} \sum_{x'} P_{x'ax}\ \beta_{\tau-1}(x') & \tau > 1 \\ \widehat{R}_{ax} & \tau = 0 \end{cases}\ . \tag{12}$$

As expected, this quantity is independent of t because we conditioned on x_t and the history before time t (including its length t) becomes irrelevant in the Markov chain. We may use the simpler notation $q_\tau(a,x) = q_{t\tau}(a,x)$. Multiplying with the time prior and eliminating the total time we get the *action-conditioned likelihood*

$$P(\hat{r}=1 \,|\, a_t=a, x_t=i; \boldsymbol{\pi}) = \frac{1}{C} \sum_{\tau=0}^{\infty} P(T=t+\tau)\, q_\tau(a,i)\,, \tag{13}$$

where $C = \sum_{\tau'} P(T=t+\tau')$, and which is for the discounted time prior independent of t because $P(t+\tau) = \gamma^t P(\tau)$ which is absorbed in the normalization. Further, with Bayes rule we get the *action posterior*

$$P(a_t=a \,|\, x_t=x, \hat{r}=1; \boldsymbol{\pi}) = \frac{\pi_{ax}}{C'} \sum_{\tau=0}^{\infty} P(T=t+\tau)\, q_\tau(a,x)\,, \tag{14}$$

where $C' = P(\hat{r}=1 \,|\, x_t=x; \boldsymbol{\pi}) \sum_{\tau'} P(T=t+\tau')$ can be computed from normalization, and which is also independent of t.

We can also compute

$$\gamma_{t\tau}(x) = P(x_t=x \,|\, \hat{r}=1, T=t+\tau; \boldsymbol{\pi}) = \frac{1}{L^{\boldsymbol{\pi}}_{t+\tau}}\, \beta_\tau(x)\, \alpha_t(x)\,, \tag{15}$$

where the likelihood $L^{\boldsymbol{\pi}}_{t+\tau}$ plays the role of a normalization (as for standard HMMs). In the experimental results we will display the posterior probability of visiting a certain state x within a rewarded trajectory, the *state occupancy*, which is given as

$$P(x \in X \,|\, \hat{r}=1; \boldsymbol{\pi}) = \sum_T \frac{P(T)\, L^{\boldsymbol{\pi}}_T}{L^{\boldsymbol{\pi}}} \left[1 - \prod_{t=0}^{T} \left[1 - \gamma_{t,T-t}(x)\right]\right]. \tag{16}$$

3.2 M-step

The exact M-step in an EM-algorithm maximizes the expected complete data log-likelihood

$$Q(\pi, \pi^*) = \sum_T \sum_{X,A} P(\hat{r}=1, X, A, T; \boldsymbol{\pi}) \log P(\hat{r}=1, X, A, T; \boldsymbol{\pi}^*) \tag{17}$$

w.r.t. the new parameters $\boldsymbol{\pi}^*$, where expectations over the latent variables (T, X, A) are taken w.r.t. the posterior given the old parameters $\boldsymbol{\pi}$. Although the mixture of finite-time MDPs complicates the derivation of the M-step, the result is in strong analogy to the standard HMM,

Theorem 3.1. *In the MDP case, the exact M-step is to assign the new parameters to the posterior action probability*

$$\pi^*_{ax} = \frac{\pi_{ax}}{C'} \sum_{\tau=0}^{\infty} P(T=\tau)\, q_\tau(a,x)\,, \tag{18}$$

Proof. From (5) in more compact notation the full joint reads

$$P(\hat{r}=1, X, A, T; \boldsymbol{\pi}) = P(T) \left[\widehat{R}_{ax} \pi_{ax}\right]_{t=T} \left[\prod_{t=0}^{T-1} P_{x'ax}\, \pi_{ax}\right] \left[P_x\right]_{t=0}. \tag{19}$$

where the brackets are to clarify the time indices, e.g., $[P_{x'ax}]_t \equiv P_{x_{t+1}a_t x_t}$. We have

$$\begin{aligned} Q(\pi,\pi^*) &= \langle \text{terms indep. of } \pi^* \rangle + \sum_{T=0}^{\infty}\sum_{X,A} P(\hat{r}=1,X,A,T;\boldsymbol{\pi}) \Big[\sum_{t=0}^{T} \log \pi^*_{a_t x_t}\Big] \\ &= \cdots + \sum_{T=0}^{\infty}\sum_{t=0}^{T}\sum_{ax} P(T)\; P(\hat{r}=1,a_t=a,x_t=x \,|\, T,\boldsymbol{\pi})\; \log \pi^*_{ax} \\ &= \cdots + \sum_{ax} \big(\log \pi^*_{ax}\big) \sum_{T=0}^{\infty}\sum_{t=0}^{T} P(T)\; q_{T-t}(a,x)\; \pi_{ax}\; a_t(x) \end{aligned} \tag{20}$$

The time summations can be resolved as follows

$$\begin{aligned} &\sum_{T=0}^{\infty}\sum_{t=0}^{T} P(T)\; q_{T-t}(a,x)\; \alpha_t(x) = \sum_{t=0}^{\infty}\sum_{T=t}^{\infty} (1-\gamma)\; \gamma^T\; q_{T-t}(a,x)\; \alpha_t(x) \\ &= (1-\gamma) \sum_{t=0}^{\infty}\sum_{\tau=0}^{\infty} \gamma^\tau\; \gamma^t\; q_\tau(a,x)\; \alpha_t(x) = \frac{1}{1-\gamma}\Big[\sum_{\tau=0}^{\infty} P(\tau)\; q_\tau(a,x)\Big]\Big[\sum_{t=0}^{\infty} P(t)\; \alpha_t(x)\Big] \\ &=: \frac{1}{1-\gamma}\; \widehat{q}(a,x)\; \widehat{\alpha}(x)\;, \end{aligned} \tag{21}$$

which gives us

$$Q(\theta,\theta^*) = \cdots + \frac{1}{1-\gamma} \sum_{ax} \big(\log \pi^*_{ax}\big)\; \pi_{ax}\; \widehat{q}(a,x)\; \widehat{\alpha}(x)\;. \tag{22}$$

Since π^*_{ax} must be a distribution over a (constrained to normalize over a), this is maximized by

$$\pi^*_{ax} = \frac{\pi_{ax}}{C}\; \widehat{q}(a,x) \tag{23}$$

for some normalization constant C. Note the independence on $\widehat{\alpha}(x)$ because one can maximize for each x separately. □

3.3 Greedy M-step

In the case of a plain unstructured MDP we can also derive a greedy and usually faster version of the M-step. Note however that this does not generalize to arbitrarily structured DBNs. In fact, in the POMDP case we will investigate later we are not aware of such a greedy version of the M-step.

By integrating over the 0-th time slice we can write the likelihood as

$$\begin{aligned} P(\hat{r}=1;\boldsymbol{\pi}) &= \sum_{ax} P(\hat{r}=1 \,|\, a_0=a,x_0=x;\boldsymbol{\pi})\; \pi_{ax}\; P(x_0=x;\boldsymbol{\pi}) \\ &= \sum_{ax} \widehat{q}(a,x)\; \pi_{ax}\; P_x\;. \end{aligned} \tag{24}$$

Following the EM idea, we can approximate $P(\hat{r}=1;\boldsymbol{\pi}^*)$ by estimating $\widehat{q}(a,x)$ with the old parameter $\boldsymbol{\pi}$ and maximize $\sum_{ax} \widehat{q}(a,x)\; \pi^*_{ax}\; P_x$ w.r.t. the new parameter $\boldsymbol{\pi}^*$. Again, maximization separates for each x and is thereby independent of P_x. We have

$$\pi^*_{ax} = \delta(a,a^*(x))\;, \quad a^*(x) = \operatorname*{argmax}_a \widehat{q}(a,x)\;. \tag{25}$$

Note that this update corresponds to a greedy version of the previous M-step. For instance, if we would iterate (18) without recomputing $\widehat{q}(a,x)$ each time (skipping intermediate E-steps) we would converge to this greedy M-step. Further, as we know from Reinforcement Learning, the greedy M-step can be thought of exploiting our knowledge that the optimal policy must be a deterministic one.

3.4 Relation to Policy Iteration

So far we have developed our approach for a plain unstructured MDP. It turns out that in this case the E- and M-step are very closely related to the policy evaluation and update steps in standard Policy Iteration.

We introduced the mixture of MDPs in a way such that the likelihood is proportional to the expected future return. In standard forward-backward algorithms the backward propagated messages are likelihoods. For the unstructured MDP these are $\beta_\tau(x) := P(\hat{r}=1 \,|\, x_{T-\tau}=x; \boldsymbol{\pi})$ which is proportional to the expected reward in exactly τ time steps in the future (from "now") conditioned on the current state (the value function of the finite-time MDP of length τ). Hence, the mixture of β's,

$$\widehat{\beta}(x) = \sum_\tau P(T=\tau)\, \beta_\tau(x) = (1-\gamma)\, V^{\boldsymbol{\pi}}(x)\,, \tag{26}$$

is proportional to the value function $V^{\boldsymbol{\pi}}(x)$ of the original MDP. Analogously we defined $q_\tau(a,x) = P(\hat{r}=1 \,|\, a_{T-\tau}=a, x_{T-\tau}=x; \boldsymbol{\pi})$ and we have

$$\widehat{q}(a,x) = \sum_\tau P(T=\tau)\, q_\tau(a,x) = (1-\gamma)\, Q^{\boldsymbol{\pi}}(a,x)\,, \tag{27}$$

where $Q^{\boldsymbol{\pi}}(a,x)$ is the Q-function in the original MDP. Note that this is also the action-conditioned likelihood (13) and, interestingly, the action posterior (14) is proportional to $\pi_{ax}\, Q^{\boldsymbol{\pi}}(a,x)$.

We conclude that the E-step in an unstructured MDP is a form of Policy Evaluation since it also yields the value function (at least quantities from which we can retrieve the value function). However, quite different to traditional Policy Evaluation, the E-step also computes α's, i.e., probabilities to visit states given the current policy, which may be compared to previous approaches like Diverse Densities (in the context of subgoal analysis (McGovern & Barto 2001)) or Policy Search by Density Estimation (Ng, Parr, & Koller 1999). The full E-step provides us with posteriors over actions, states and the total time. In practice we can use the α's in an efficient heuristic for pruning computations during inference (see section 4.1 and appendix B). Further, the E-step generalizes to arbitrarily structured DBNs and thereby goes beyond standard Policy Evaluation, particularly when using approximate inference techniques like message passing or approximate belief representations.

Concerning the M-step, in the unstructured MDP the greedy M-step is *identical* to the policy update in Policy Iteration. That means that one iteration of the EM will yield exactly the same policy update as one iteration of Policy Iteration (provided one does exact inference and exact value function computation without time horizon cutoffs). Again, the M-step goes beyond a standard policy update in the generalized case. This becomes particularly apparent when in structured DBNs (e.g. the POMDP case in section 5) the full posteriors computed via inference (including forward propagated messages analogous to α's) are necessary for the M-step. Only computing backward propagated messages (like value functions or β's) is not sufficient for the M-step.

In summary,

Lemma 3.2. *The EM-algorithm on an unstructured MDP using exact inference and the greedy M-step is equivalent to Policy Iteration in terms of the policy updates performed.*

Interestingly, this also means they are equivalent w.r.t. convergence. (Recall that Policy Iteration is guaranteed to converge to the global optimum whereas EM-algorithms are only guaranteed to converge to local optima.) The computational costs of both methods may differ depending on the implementation (see experiment 4.1 and appendix B). On structured DBNs or using approximate inference techniques or approximate belief representations the EM-algorithm goes beyond Policy Iteration.

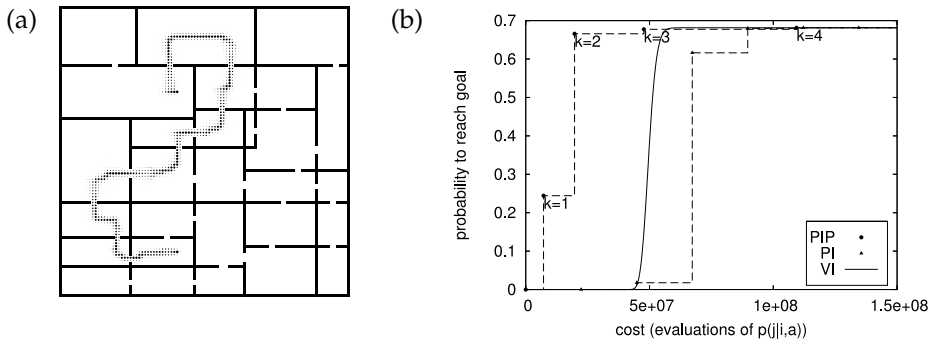

Figure 4: (a) State visiting probability calculated by EM for some start and goal state. The radii of the dots are proportional to (16). (b) The probability of reaching the goal (for EM) and the value calculated for the start state (PS) against the cost of the planning algorithms (measured by evaluations of $p(j|i,a)$) for both start/goal configurations.

4 MDP experiments

4.1 Discrete maze examples

Efficiency. We tested the EM algorithm with greedy M-step on a discrete maze of size 100×100 and compared it to standard Value Iteration (VI) and Policy Iteration (PI). Walls of the maze are considered to be trap states (leading to unsuccessful trials) and actions (north, south, east, west, stay) are highly noisy in that with a probability of 0.2 they lead to random transitions. In the experiment we chose a uniform time prior (discount factor $\gamma = 1$), initialized $\boldsymbol{\pi}$ uniformly, and iterated the policy update $k = 5$ times. To increase computational efficiency we exploited that the algorithm explicitly calculates posteriors which can be used to prune unnecessary computations during inference as explained in appendix B. For policy evaluation in PI we performed 100 iterations of standard value function updates.

Figure 4(a) displays the posterior state visiting probabilities (Equation (16)) of the optimal policy computed the EM for a problem where a reward of 1 is given when the goal state g is reached and the agent is initialized at a start state s (i.e., $P_x = \delta(x, s)$). Computational costs are measured by the number of evaluations of the environment $P_{x'ax}$ needed during the planning procedure. Figure 4(b) displays the probability of reaching the goal $P(\hat{r} = 1; \boldsymbol{\pi})$ against these costs. Note that for EM (and PI) we can give this information only after a complete E- and M-step cycle (policy evaluation and update) which are the discrete dots (triangles) in the graph. The graph also displays the curve for VI, where the currently calculated value V_A of the start state (which converges to $P(\hat{r} = 1)$ for the optimal policy) is plotted against how often VI evaluated $P_{x'ax}$.

In contrast to VI and PI, the EM algorithm takes considerable advantage of knowing the start state in this planning scenario: the forward propagation allows for the pruning and the early decision on cutoff times in the E-step as described in appendix B. It should thus not surprise and not be overstated that the EM is more efficient in this specific scenario. Certainly, a similar kind forward propagations could also be introduced for VI or PI to achieve equal efficiency. Nonetheless, our approach provides a principled way of pruning by exploiting the computation of proper posteriors. The policies computed by all three methods are equal for states which have significantly non-zero state visiting probabilities.

Multi-modal time posteriors. The total time T plays a special role as a random variable in our mixture model. We use another simple experiment to illustrate this special role by considering the total time posteriors. Modelling walls as trap states leads to interesting trade-offs between staying away from walls in favor of security and choosing short paths. Figure 5 displays a 15x20 maze with three possible pathways from the start (bottom left) to the goal (bottom right) state.

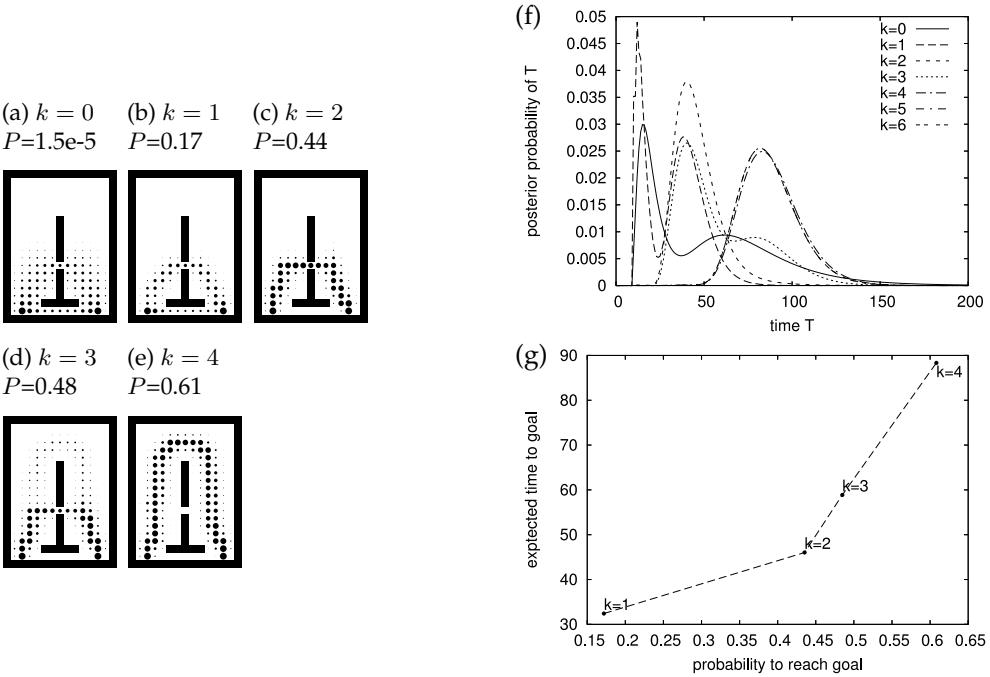

Figure 5: (a-e) State visiting probabilities for various EM-iterations k. The start and goal states are to the bottom left and right, respectively. The radii of the dots are proportional to $P(x \in X \,|\, \hat{r}=1)$. The indicated P is $P(\hat{r}=1)$. (f) The different possible pathways lead to a multi-modal time posterior $P(T \,|\, \hat{r}=1)$. (g) The trade-off between the expected time to goal (mean of $P(T \,|\, \hat{r}=1)$) and the probability to reach the goal. The dots corresponds to $k=1,..,4$ (from left to right).

The direct pathway is a narrow aisle clinched between two walls and thus highly risky. The next one up requires a step through a narrow doorway. The most top one is secure but longest. The five Figures illustrate the state visiting probability $P(x \in X \,|\, \hat{r}=1)$ for random walks ($k=0$) and the policies calculated by EM for $k=1,..,4$ iterations. Also the success probability $P(\hat{r}=1)$ is indicated. Figure 5(f) displays the corresponding time posteriors $P(T \,|\, \hat{r}=1)$ for the different k's. Interesting is the multi-modality of these time posteriors in that specific environment. The multi-modality in some way reflects the topological properties of the environment: that there exists multiple possible pathways from the start to the goal with different typical lengths (maxima of the time posterior) and different success probabilities (area (integral) of a mode of the time posterior). Already for $k=0$ one can see that, besides the direct pathway (of typical length ≈ 15), there exist alternative, longer routes which comprise significant success probability. One way to exploit this insight could be to choose a new time prior for the next inference iteration that explicitly favors these longer routes. Figure 5(g) nicely exhibits the trade-off between the expected time to goal and the probability to reach the goal.

4.2 Stochastic optimal control

Gaussian belief state propagation. Next we want to show that the framework naturally allows to transfer other inference techniques to the problem of solving MDPs. We address the problem of stochastic optimal control in the case of a continuous state and control space. A standard inference technique in continuous state spaces is to assume Gaussian belief states as representations for α's and β's and propagate forward-backward and using the unscented transform to handle also non-linear transition dynamics (see (Murphy 2002) for an overview on inference techniques in DBNs). Note that using Gaussian belief states implies that the effective value function (section 3.4) becomes a mixture of Gaussians.

All the equations we derived remain valid when reinterpreted for the continuous case (summations become integrations, etc) and the exact propagation equations (7) and (9) are replaced by propagations of Gaussian belief states using the unscented transform. In more detail, let $\mathcal{N}(x, a, A)$ be the normal distribution over x with mean a and covariance A and let $\overline{\mathcal{N}}(x, a, A)$ be the respective *non-normalized* Gaussian function with $\overline{\mathcal{N}}(a, a, A) = 1$. As a transition model we assume

$$P(x'|u, x) = \mathcal{N}(x', \phi(u, x), Q(u)) \ , \quad Q(u) = C + (|u|/\mu)^2 \ I \tag{28}$$

where $\phi(u, x)$ is an non-linear function, C is a constant noise covariance, and we introduced a parameter μ for an additional noise term that is squared in the control signal. With the parameterization $\alpha_t(x) = \mathcal{N}(x, a_t, A_t)$ and $\beta_\tau(x) = \overline{\mathcal{N}}(x, b_\tau, B_\tau)$ (note that β's always remain non-normalized Gaussian likelihoods during propagation), forward and backward propagation read

$$\begin{aligned}(a_t, A_t) &= UT_\phi(a_{t-1}, A_{T-1}) \\ (b_\tau, B_\tau) &= UT_{\phi^{-1}}(b_{\tau-1}, B_{\tau-1}) \ ,\end{aligned}$$

where $UT_\phi(a, A)$ denotes the unscented transform of a mean and covariance under a non-linear function. In brief, this transform deterministically considers $2n + 1$ points (say with standard deviation distance to the mean) representing the Gaussian. In the forward case (the backward case) it maps each point forward using ϕ (backward using ϕ^{-1}), associates a covariance $Q(u)$ (a covariance $\phi'^{-1} \ Q(u) \ \phi'^{-1T}$, where ϕ'^{-1} is the local inverse linearization of ϕ at each point) with each point, and returns the Gaussian that approximates this mixture of Gaussians. Further, for any t and τ we have

$$\begin{aligned}&L^{\boldsymbol{\pi}}_{t+\tau} = \mathcal{N}(a_t, b_\tau, A_t + B_\tau) \\ &\gamma_{t\tau}(x) = \mathcal{N}(x, c_{t\tau}, C_{t\tau}) \ , \quad C^{-1}_{t\tau} = A^{-1}_t + B^{-1}_\tau \ , \quad c_{t\tau} = C_{t\tau} \ (A^{-1}_t \ a_t + B^{-1}_\tau \ b_\tau)\end{aligned}$$

The policy and the M-step. In general, the policy is given as an arbitrary non-linear function $\pi : \ x \mapsto u$. Clearly, we cannot store such a function in memory. However, via the M-step the policy can always be implicitly expressed in terms of the β-quantities of the previous E-step and numerically evaluated at specific states x. This is particularly feasible in our case because the unscented transform used in the belief propagation (of the next E-step) only needs to evaluate the transition function (and thereby π) at some states; and we have the advantage of not needing to approximate the function π in any way. For the M-step we need equation (12),

$$q_\tau(u, x) \overset{\tau \geq 1}{=} \int_{x'} P(x'|u, x) \ \overline{\mathcal{N}}(x', b_{\tau-1}, B_{\tau-1}) = |2\pi B_{\tau-1}|^{1/2} \ \mathcal{N}(b_{\tau-1}, \phi(u, x), B_{\tau-1} + Q(u)) \ , \tag{29}$$

and we maximize the mixture of Gaussians $\widehat{q}(u, x)$ with a gradient ascent using

$$\begin{aligned}\partial_u q_\tau(u, x) &= -q_\tau(u, x) \Big[\ h^T \Big(\partial_u \phi(u, x) \Big) - u \frac{1}{\mu^2} \Big(\mathrm{tr}(A^{-1}) - h^T h \Big) \Big] \\ A &:= B_{\tau-1} + Q(u) \ , \quad h := A^{-1} \ (\phi(u, x) - b)\end{aligned} \tag{30}$$

Examples. Consider a simple 2-dimensional problem where the start state is distributed around zero via $\alpha_0(x) = \mathcal{N}(x, \ (0, 0), \ .01I)$ and the goal region is determined by $P(\hat{r} = 1 \,|\, x) = \overline{\mathcal{N}}(x, \ (1, 1), \ \mathrm{diag}(.0001, .1))$. Note that this goal region around $(1, 1)$ is heavily skewed in that rewards depend more on the precision in the $x-$dimension than the $y-$dimension. The control law is simply $\phi(u, x) = x + .1 \, u$ and the discount factor $\gamma = 1$. When choosing $\mu = 0$ (no control-dependent noise), the optimal control policy will try to jump directly to the goal $(1, 1)$. Hence we first consider the solution when manually constraining the norm of $|u|$ to be small (effectively following the gradient of $P(r = 1 \,|\, u_t = u, x_t = x; \boldsymbol{\pi})$). Figure 6(a,b) shows the learned control policy π

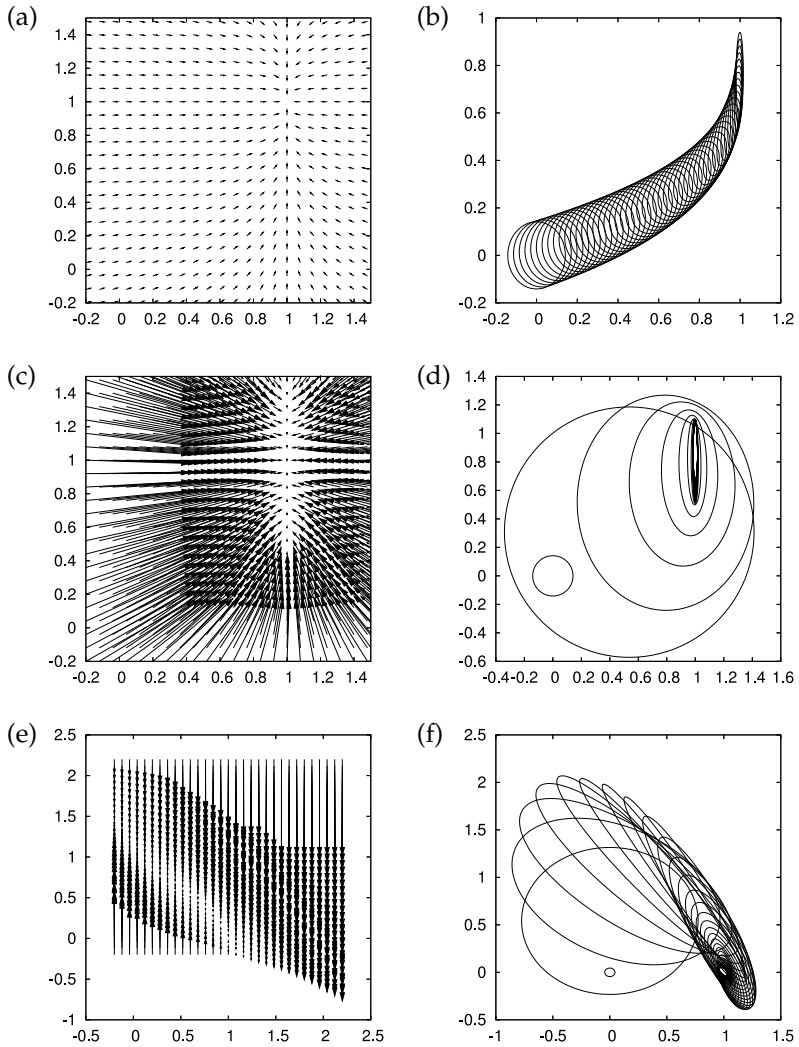

Figure 6: Learned policies (left) and forward simulation (α's) of these policies (right) for aspheric Gaussian-shaped targets. (a,b) are for the case of restricted action amplitude (the walker model). (c,d) are for unconstrained amplitude (the golfer model). And (e,f) are for the approach to a new position under phase space dynamics.

and the forward simulation given this policy by displaying the covariance ellipses for $\alpha_{0:T}(x)$ after $k = 3$ iterations. What we find is a control policy that reduces errors in $x-$dimension more strongly than in $y-$direction, leading to the tangential approach to the goal region. This is related to studies on redundant control or the so-called uncontrolled manifold.

Next we can investigate what the effect of control-dependent noise is without a constraint on the amplitude of u. Figure 6(c,d) displays results (after $k = 3$ iterations) for $\mu = 1$ and no additional constraints on u. The process actually resembles a golf player: the stronger the hit, the more noise. The optimal strategy is to hit fairly hard in the beginning, hopefully coming closer to the goal, such that later a number of smaller and more precise hits can be made. The reason for the small control signals around the goal region is that small steps have much more accuracy and reward expectation is already fairly large for just the $x-$coordinate being close to 1.

Finally we think of x being a phase space and consider the dynamics $\phi(u, x) = (x_1 + .1x_2, x_2 + .1u)$ where u is the 1-dimensional acceleration of the velocity x_2, and x_1 is a position. This time we

set the start and goal to $(0,0)$ and $(1,0)$ respectively, both with variance .001 and choose $\mu = 10$. Figure 6(e,f) display the result and show nicely how the learned control policy approaches the new position on the $x-$axis by first gaining and then reducing velocity.

5 POMDPs and the policy model

A stationary, partially observable Markov decision process (POMDP, see e.g. (Kaelbling, Littman, & Cassandra 1998)) is given by four time-independent probability functions,

the initial world state distribution	$P(x_0 = x) =: P_x$
the world state transitions	$P(x_{t+1} = x' \mid a_t = a, x_t = x) =: P_{x'ax}$
the observation probabilities	$P(y_t = y \mid x_t = x) =: P_{yx}$
the reward probabilities	$P(r_t = r \mid a_t = a, x_t = x) =: P_{rax}$.

These functions are considered known. We assume the world states, actions, and observations (x_t, y_t, a_t) are discrete random variables while the reward r_t is a real number

The POMDP only describes one "half" of the process to be described as a DBN — the other half is the agent interacting with the environment. Our point of view is that the agent could use an arbitrary "internal machinery" to decide on actions. FSCs are a simple example. However, a general DBN formulation of the agent's internal machinery allows us to consider much more structured ways of behavior organization, including factorized and hierarchical internal representations (see, e.g., Theocharous et al. 2004). In the remainder of this section we investigate a policy model that is slightly different to finite state controllers but still rather simple. However, the approach is generally applicable to any DBN formulation of the POMDP and the agent.

To solve a given POMDP challenge an agent needs to maintain some internal memory variable (if not the full belief state) that represents information gained from previous observations and actions. We assume that this variable is updated depending on the current observation and used to *gate* reactive policies rather than to directly emit actions. More precisely, the dynamic Bayesian network in Figure 7(a) captures the POMDP and the agent model which is defined by the

initial internal memory distribution	$P(b_0 = b) =: \nu_b$
internal memory transition	$P(b_{t+1} = b' \mid b_t = b, y_t = y) =: \lambda_{b'by}$
reactive policies	$P(a_t = a \mid b_t = b, y_t = y) =: \pi_{aby}$.

Here we introduced b_t as the agent's internal memory variable. It is comparable to the "node state" in finite state controllers (Figure 7(b)), but differs in that it does not directly emit actions but rather gates reactive policies: for each internal memory state b the agent uses a different "mapping" π_{aby} (i.e. a different reactive policy) from observations to actions.

As for the MDP case, solving the POMDP in this approach means to find parameters $\theta = (\boldsymbol{\nu}, \boldsymbol{\lambda}, \boldsymbol{\pi})$ of the DBN in Figure 7 that maximize the expected future return $V^\theta = \mathrm{E}\left\{\sum_{t=0}^{\infty} \gamma^t \, r_t; \theta\right\}$ for a discount factor $\gamma \in [0,1]$.

6 EM-algorithm for the POMDP model

Let us consider briefly but explicitly how the previous EM-algorithm generalizes to the POMDP case. Each finite-time model (Figure 8) of the infinite mixture is limited in time by T and emits only a single reward $\hat{r}$ at the final time step. Let $z_t = (x_t, y_t, b_t, a_t)$ subsume all latent variables in a time slices and $Z = z_{0:T}$ be its trajectory. Given $\widehat{R}_{ax} \propto R_{ax}$ the joint for a finite-time MDP reads

$$P(\hat{r}, Z \mid T; \theta) = \Big[\widehat{R}_{ax}\, \pi_{aby}\, P_{yx}\Big]_{t=T} \Big[\prod_{t=0}^{T-1} \lambda_{b'by}\, P_{x'ax}\, \pi_{aby}\, P_{yx}\Big] \Big[\nu_b\, P_x\Big]_{t=0} , \tag{31}$$

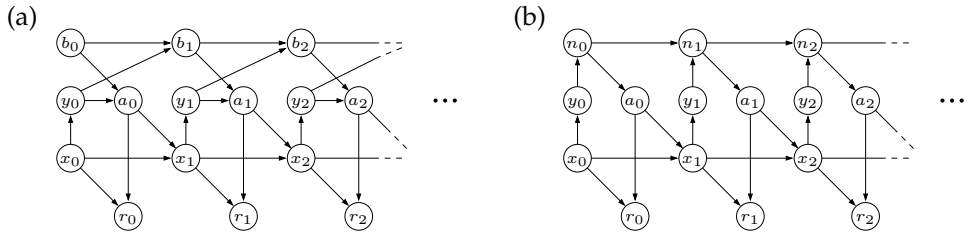

Figure 7: (a) DBN of the POMDP and policy with internal memory b_t; the time is unbounded and rewards are emitted at every time step. (b) For comparison: the DBN of a POMDP with a standard FSC, with "node state" n_t.

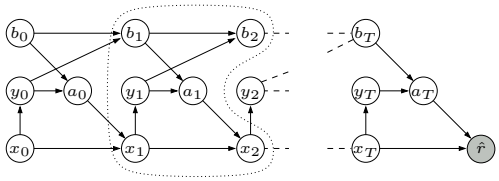

Figure 8: The finite-time model, where the reward is emitted only at the final state. The dotted line captures a clique of the graphical model; the junctions are (b_t, x_t) for $t = 1..T$, on which the E-steps are performed.

where again the brackets clarify the time indices ($[P_{x'ax}]_t \equiv P_{x_{t+1}a_tx_t}$). The joint for the full mixture of finite-time models is $P(\hat{r}, Z, T; \theta) = P(\hat{r}, Z \,|\, T; \theta)\, P(T)$. Each finite-time model defines a likelihood $L_T^\theta = P(\hat{r}{=}1 \,|\, T; \theta)$. Consequently, the mixture model defines the likelihood

$$L^\theta = P(\hat{r}{=}1; \theta) = \sum_T P(T)\; P(\hat{r}{=}1 \,|\, T; \theta) = (1-\gamma) \sum_T \gamma^T \,\mathrm{E}\,\{\hat{r} \,|\, T; \theta\} \propto V^\theta \,. \tag{32}$$

Concerning the E-step we perform exact inference by grouping random variables into cliques. Figure 8 indicates a clique in the graphical model with junctions (b, x). Thus, forward-backward propagation can most efficiently be realized on the reduced Markovian process on the (b, x)-state with transitions

$$P(b', x' \,|\, b, x) = \sum_{a,y} P_{x'ax}\; \lambda_{b'by}\; \pi_{aby}\; P_{yx} \,. \tag{33}$$

Forward and backward propagation reads

$$\alpha_t(b', x') = P(x_t{=}x', b_t{=}b' \,|\, T; \theta) = \sum_{b,x} P(b', x' \,|\, b, x)\; \alpha_{t-1}(b, x) \tag{34}$$

$$\beta_\tau(b, x) = P(\hat{r}{=}1 \,|\, x_{T-\tau}{=}x, b_{T-\tau}{=}b, T; \theta) = \sum_{b',x'} P(b', x' \,|\, b, x)\; \beta_{\tau-1}(b', x') \,, \tag{35}$$

where β's are again indexed backward in time (with the time-to-go τ). They are initialized with

$$\alpha_0(b, x) = \nu_b\; P_x \,, \quad \beta_0(b, x) = \sum_{a,y} \widehat{R}_{ax}\; \pi_{aby}\; P_y x \,. \tag{36}$$

and we can synchronously propagate forward and backward. The result of the E-step are the quantities

$$\widehat{\alpha}(b, x) := \sum_{t=0}^{\infty} P(T{=}t)\; \alpha_t(b, x) \,, \quad \widehat{\beta}(b, x) := \sum_{\tau=0}^{\infty} P(T = \tau)\; \beta_\tau(b, x) \,. \tag{37}$$

The M-step is derived from maximization of the expected complete log-likelihood

$$Q(\theta, \theta^*) = \sum_{T=0}^{\infty} \sum_{Z} P(\hat{r}=1, Z, T; \theta) \ \log P(\hat{r}=1, Z, T; \theta^*) \ , \tag{38}$$

where expectation is taken w.r.t. the old parameters θ and we need to maximize w.r.t. the new parameters θ^*. In Appendix C we derive the exact M-steps

$$\pi^*_{aby} = \frac{\pi_{aby}}{C_{by}} \ \sum_x \Big[\frac{\gamma}{1-\gamma} \ \sum_{b'x'} \widehat{\beta}(b', x') \ \lambda_{b'by} \ P_{x'ax} \ + \ \widehat{R}_{ax} \Big] \ P_{yx} \ \widehat{\alpha}(b, x) \ , \tag{39}$$

$$\lambda^*_{b'by} = \frac{\lambda_{b'by}}{C'_{by}} \ \sum_{x',a,x} \widehat{\beta}(b', x') \ P_{x'ax} \ \pi_{aby} \ P_{yx} \ \widehat{\alpha}(b, x) \ , \tag{40}$$

$$\nu^*_b = \frac{\nu_b}{C''_b} \sum_x \widehat{\beta}(b, x) \ P(x_0 = x) \ , \tag{41}$$

where C_{by}, C'_{by} and C''_b are normalization constants. Note that in these updates the α's (related to state visiting probabilities) play a crucial role. Also we are not aware of a greedy version of these updates that proved efficient (i.e. without immediate convergence to a local minimum).

Complexity

Let X, B, A and Y momentarily denote the cardinalities of random variables x, b, a, y, respectively. The main computational cost accumulates during α- and β-propagation (34) and (35), both of which have complexity $O(T_{\max} \, B^2 \, X^2)$. Here and below, X^2 scales with the number of non-zero elements in the transition matrix $P(x'|x)$ (assuming non-zero action probabilities). We always use sparse matrix representations for transition matrices. The number of propagations $T_{\max}$ scales with the expected time of reward (for a simple start-goal scenario this is the expected time to goal). Sparse vector representations of α's and β's ruther reduce the complexity depending on the topological dimensionality of $P(x'|x)$. The computational complexity of the M-step scales with $O(AYB^2X^2)$; in total this adds to $O((T_{\max} + AY)B^2X^2)$ for one EM-iteration.

For comparison, let N denote the number of nodes in a FSC. The computation of a policy gradient w.r.t. a *single* parameter of a FSC scales with $O((T_{\max}+AY)N^2X^2)$ (taken from (Meuleau et al. 1999), top of page 7). A fully parameterized FSC has $NA + N^2Y$ parameters, bearing a total complexity of $O((T_{\max} + AY)N^4X^2Y)$ to compute a full policy gradient.

For EM-learning as well as gradient ascent, the complexity additionally multiplies with the number k of EM-iterations respectively gradient updates.

7 POMDP experiments

Scaling. The POMDP EM-algorithm has no free parameters except for the initializations of $\lambda_{b'by}$, π_{aby}, and ν_b. Roughly, we initialized ν_b and π_{aby} approximately uniformly, while $\lambda_{b'by}$ was initialized in a way that favors not to switch the internal memory state, i.e., the diagonal of the matrix $\lambda_{b'b\cdot}$ was initialized larger than the off-diagonal terms. More precisely, we first draw non-normalized numbers

$$\pi_{aby} \sim 1 + 0.1 \, \mathcal{U}([0,1]) \ , \quad \lambda_{b'by} \sim 1 + 5 \, \delta_{b'b} + 0.1 \, \mathcal{U}([0,1]) \ , \quad \nu_b = 1 \tag{42}$$

where $\mathcal{U}([0,1])$ is the uniform distribution over $[0,1]$, and then normalize these parameters.

To start with, we test the scaling behavior of our EM-algorithm and compare it with that of gradient ascent for a FSC (FSC-GA). We tried three options for coping with the problem that the simple policy gradient in (Meuleau et al. 1999) ignores the normalization constraints of the parameters: (1) projecting the gradient on the simplex, (2) using a step-size-adaptive gradient ascent (RPROP) with added soft-constraint gradients towards the simplex, (3) using MATLAB's

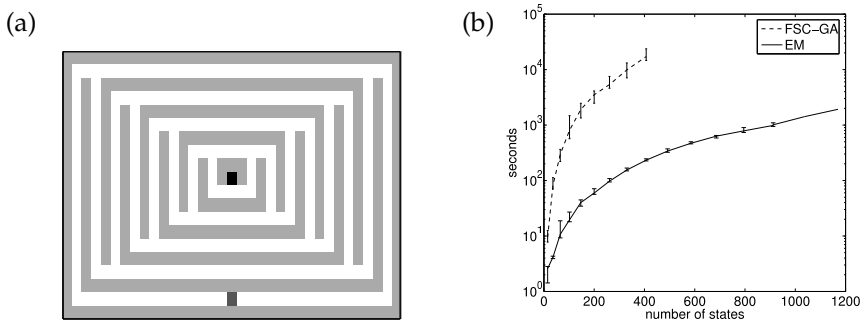

Figure 9: (a) A simple scalable maze from (Meuleau et al. 1999), here with 330 states. The start (goal) position is marked gray (black). The robot has five actions (north, south, eat, west, stay) and his observation is a 4 bit number encoding the presence of adjacent walls. (b) Running times of EM-learning and FSC gradient ascent that show how both methods scale with the maze size. The lines are medians, the errorbars min and max of 10 independent runs for each of the various maze sizes. (For maze sizes beyond 1000 we display only 1 run).

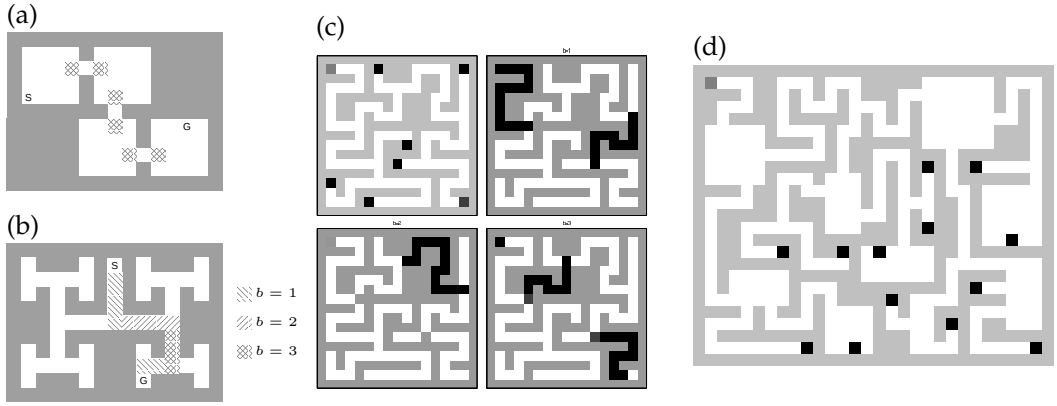

Figure 10: Further mazes considered in the experiments. (c): Top left is the maze with the start (light gray), goal (bottom right) and drain-states (dark). The other three illustrations display the internal memory state b (grey value $\propto \widehat{\alpha}(b, x)$ for $b = 1, 2, 3$) at different locations in the maze. (d): Large maze with 1516 turtle states.

gradient-based constraint optimization method 'fmincon'. The second option gave the best results and we refer to those in the following. Note that our algorithms does not have such problems: the M-step assigns correctly normalized parameters. Figure 9 displays the results for the simple maze considered in (Meuleau et al. 1999) for various maze sizes. Our policy model needs $B = 2$ internal memory states, the FSC $N = 5$ graph nodes to solve these problems. The discount factor was chosen $\gamma = .99$. The results confirm the differences we noticed in the complexity analysis.

Training the memory to gate primitive reactive behaviors. To exemplify the approach's ability to learn an appropriate memory representation for a given task we investigate further maze problems. We consider a *turtle*, which can move forward, turn right or left, or wait. With probability $1-\epsilon$ this action is successful; with probability $\epsilon = .1$ the turtle does not respond. The state space is the cross product of positions and four possible orientations, and the observations are a 4 bit number encoding the presence of adjacent walls relative to the turtle's orientation. Further, whenever the agent reaches the goal (or a zero-reward drain state, see below) it is instantly reset to the start position.

Figures 10(a) and 10(b) display two small mazes with two specific difficulties: The interior

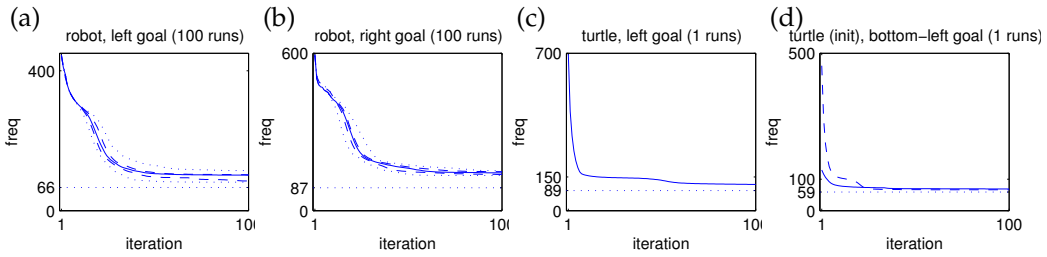

Figure 11: Learning curves for the maze in Figure 10(d). The expected reward interval (see footnote) is given over the number of EM-iterations. Solid: median over 100 independent trial runs (with noisy initializations (42)), dashed and dotted: 2.5, 25, 75 and 97.5 percentiles, dotted baseline: shortest path to the goal in the respective environment which *were* possible in the noise-free case. The optimal controller in our stochastic case is necessarily above this baseline.

states and also the entries and exits (cross-shaded) of halls in 10(a) all have the same observation 0000 and are highly susceptible for the agent to get lost. For $B = 2$, the turtle learns a wall-following strategy as a basic reactive behavior, while the internal memory is used only at the exists and entrances to halls: for observation $b = 1$ and $y = 0000$ the turtle turns left and switches to $b = 2$, while for $b = 2$ and $y = 0000$ the turtle goes straight and switches back to $b = 1$. The maze in Figure 10(b) is a binary-decision maze and poses the problem of remembering how many junctions have passed already: To reach the goal, the turtle has to follow aisles and at T-junctions make decisions [left, right, right, left]. For $B = 3$ the algorithm finds the obvious solution: Each internal memory state is associated with simple reactive behaviors that follows aisles and, depending on b, turn left or right at a T-junction. A finite state controller would certainly find a very similar solution. However, in our case this solution generalizes to situations when the corridors are not straight: Figure 10(c, top left) displays a maze with 30 locations (number of states is 480), where the start state is in the top left corner and the goal state in the bottom right. Again, the turtle has to make decisions [left, right, right, left] at T-junctions to reach the goal, but additionally has to follow complex aisles in between. Unlike with FSCs, our turtle needs again only $B = 3$ internal memory states to represent the current corridor. The shading in Figure 10(c) displays the probability of visiting a location on a trajectory while being in memory state $b = 1, 2$ or 3.

Finally, we investigate the maze in Figure 10(d) with 379 locations (1516 turtle states). The maze is a complex combination of corridors, rooms and junctions. On this maze we also tested a normal robot (north, south, west, east actions with noise $\epsilon = .1$), learning curves for $B = 3$ for the left and right most goals are given in Figure 11(a,b) and exhibit reliable convergence.[1] We also investigated single runs in the turtle case for the left most goal (Figure 11(c)) and the second left goal (Figure 11(d) dashed line). Again, the turtle utilizes that aisle following can be implemented with a simple reactive behavior; the internal memory is only used for decision making in halls and at junctions. Since the aisle following behavior can well be generalized to all goal settings we performed another experiment: we took the final policy π_{aby} learned for the left most goal as an initialization for the task of finding the second left goal. Note that the start-to-goal paths are largely disjoint. Still, the algorithm converges, particularly in the beginning, much faster (Figure 11(d) solid line), showing that such generalization is indeed possible.

To conclude these experiments, we can summarize that the agent learned internal memory

[1] As a performance measure we define the *expected reward interval* which is directly linked to $P(\hat{r} = 1)$. Consider a cyclic process that receives a reward of 1 every d time steps; the expected future reward of this process is

$$P(\hat{r}=1) = \sum_{T=1}^{\infty} P(dT) = (1-\gamma) \sum_{T=1}^{\infty} \gamma^{dT} = \frac{\gamma^d\ (1-\gamma)}{1-\gamma^d} \ .$$

Inverting this relation, we translate a given expected future reward into an expected reward interval via

$$d = \frac{\log P(\hat{r}=1) - \log(P(\hat{r}=1)+1-\gamma)}{\log \gamma} \ .$$

This measure is rather intuitive: The performance can directly be compared with the shortest path length to the goal. Note though that in stochastic environment even an optimal policy has an expected reward interval larger than the shortest path to goal.

representations to switch between reactive behaviors. In the experiments they mainly turned out to represent different corridors. Generalization of the reactive behaviors to new goal situations is possible. Further, memorization is not time bounded, e.g., independent of the length of an aisle the turtle agent can sustain the current internal memory while executing the reactive aisle following behavior.

8 Related work

Let us briefly draw links to related work that goes beyond what was mentioned in the introduction:

McGovern & Barto (2001) introduced Diverse Densities as a means to discover subgoals in RL. These densities are closely related the state occupancy distribution (16). In fact, if we imagined a typical office environment and average (16) over randomly chosen goals (for uniform P_x), this will peak at doorways (see also Figure 4).

Ng, Parr, & Koller (1999) introduced a method based on density estimation for a more efficient gradient-based policy search. The estimated density is basically the forward propagated α's in our inference approach, but used for estimating the policy gradient.

Wiegerinck, van den Broek, & Kappen (2006) proposed an approach for solving multi-agent problems using path integral methods based on a formulation of the objective function as a partition function. This seems related to our approach since we know that the EM-algorithm can be interpreted as a minimization of free energy. They do not address optimality w.r.t. discounted future return. Future research should investigate more into these relations.

Finally, maybe surprisingly, this work had its origin in considering how neural systems could implement planning. In (Toussaint 2006) we proposed a simple neural architecture to effectively represent an MDP that realizes anticipation and planning on the basis of neural activation dynamics. This approach relies on representing a neural activation field (a value function) on an explicit, non-distributed state representation. Extending this to distributed (factored) representations in the line of (Guestrin et al. 2003) seems biologically implausible. However, recent experiments with rats in a maze environment showed evidence for periodically traveling waves of neural activation in the hippocampus during decision making, which, given the place field mapping, correspond to a forward propagation through the spatial topology of the maze (Johnson & Redish 2006). These neural dynamics are stunningly similar to inference processes. In particular the special role of time in these processes (the fact that there are traveling waves) allows for coping with distributed representations in a similar ways as inference on factored DBNs does.

9 Conclusion

We introduced a framework for solving (PO)MDPs by translating the problem of maximizing expected future return into a problem of likelihood maximization. One ingredient for this approach is the mixture of finite-time models we introduced in section 2. We have shown that this approach establishes equivalence for arbitrary reward functions, allows for an efficient inference procedure, propagating synchronously forward and backward without pre-fixing a finite time horizon T, and allows for the handling of discounting rewards. We also showed that in the case of an unstructured MDP the resulting EM-algorithm using exact inference and a greedy M-step is closely related to standard Policy Iteration.

However, unlike Policy Iteration, the EM-algorithm generalizes to arbitrary DBNs and the aim of this approach is to transfer the full variety of existing inference techniques to the problem of solving (PO)MDPs. This refers especially to structured problems, where DBNs allow us to consider structured representations of the environment (the world state, e.g. factorization or hierarchies) as well as the agent (e.g. hierarchical policies or multiple agents). Inference techniques like variational approaches, message-passing algorithms, or approximate belief representations in DBNs can now be used to exploit such structure in (PO)MDPs.

We exemplified the approach for exact inference on unstructured MDPs, using Gaussian belief state propagation on a non-linear stochastic optimal control problem, and on a more complex DBN formulation of a POMDP problem. An application to a robotics planning problems is presented in (Toussaint & Goerick 2007, submitted work) where we use loopy Gaussian BP in a factor graph representation. The 13D joint angle state of the robot, the 3D end-effector state, and the binary collision variable constitute the three random variables of the DBN description, while the robot's kinematics define their non-linear coupling. Conditioning the model to non-collision and a final target end-effector state, we can efficiently solve the typical redundant control problem under collision constraints. A natural extension of this approach to more complex domains is to use particle representations for multi-modal beliefs. Another interesting issue for future research is to consider max-product BP (a generalization of Viterbi) in the context of planning.

In the POMDP context there are even more aspects to consider: Can we use inference techniques also to estimate the number of internal states we need to solve a problem (cf. Infinite Hidden Markov models (Beal, Ghahramani, & Rasmussen 2002) as a method to learn the number of hidden states needed to model the data)? Or are there efficient heuristics to add hidden states in a DBN, e.g., analogous to how new nodes are added to FSCs in the Bounded FSCs approach (Poupart & Boutilier 2003)?

Acknowledgments

M.T. is grateful to the German Research Foundation (DFG) for the Emmy Noether fellowship TO 409/1-1. S.H. is grateful to the European Community for being supported by a Marie Curie Fellowship.

A Remarks

(i) The mixture of finite-time MDPs may be compared to a classical interpretation of reward discounting: Assume the agent has a probability $(1-\gamma)$ of dying after each time step. Then the distribution over his life span is the geometric distribution $P(T) = \gamma^T(1-\gamma)$. In our mixture of finite-time MDPs we treat each possible life-span T separately. From the agent's perspective, he knows that he has a finite life span T but he does not know what it is – he lives in a mixture of possible worlds. Each finite life span is terminated by a single binary reward (say, going to heaven or hell). The agent's behavior must reflect his uncertainty about his life span and act by accounting for the probability that he might die now or later on, i.e., he must "average" over the mixture of possible worlds he might live in.

(ii) In the original MDP, the rewards at two different time slices, say r_t and r_{t+1}, are strongly correlated. The mixture of finite-time MDPs does not include such correlations because the observations of reward at $T=t$ and $T=t+1$ are treated by separate finite-time MDPs. However, since the expected future return V^π is merely a *sum* of reward expectations at different time slices such correlations are irrelevant for solving the MDP and computing optimal policies.

(iii) When the original reward function R_{ax} has values between 0 and 1 we can set the binary reward probability $\widehat{R}_{ax} = P(\hat{r}\,|\,x,a)$ equal to R_{ax}. When we have rewards outside the range $[0,1]$ we can linearly rescale them by $\widehat{R}_{ax} = (R_{ax} - R_{\text{min}})/(R_{\text{max}} - R_{\text{min}})$.

(iv) The non-discounting case $\gamma = 1$ leads to a zero time prior $P(T) = 0$. We handle such cases simply by setting $P(T) = 1/C$ for $T < C$ and $P(T) = 0$ for $T \geq C$ for some very large constant C (larger than we could possibly propagate forward-backward).

B Pruning computations

Consider a finite state space and assume that we fixed the maximum allowed time T by some upper limit T_M (e.g., by deciding on a cutoff time based on the time posterior computed on the fly, see below). Then there are potentially large regions of the state space on which we may

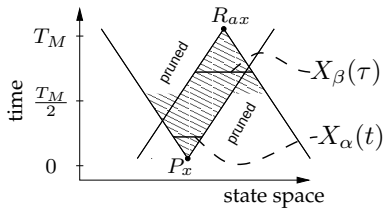

Figure 12: Only the envelopes emanating from the start distribution (P_x) and rewards (R_{ax}) contribute to the propagation. They are chopped where they do not overlap with the other envelope after $T_M/2$ iterations.

prune computations, i.e., states x for which the posterior $\gamma_{t\tau}(x) = 0$ for any t and τ with $t + \tau \leq T_M$. Figure 12 illustrates the idea. Let us consider the α-propagation first (all statements apply conversely for the β-propagation). For iteration time t we define a set of states

$$X_\alpha(t) = \{x \,|\, \alpha_t(x) \neq 0 \ \wedge \ (t < T_M/2 \ \vee \ \beta_{T_M - t}(x) \neq 0)\} \ .$$

Further, given $\beta_\tau(x) = 0 \Rightarrow \forall \tau' \leq \tau : \beta_{\tau'}(x) = 0$, it follows

$$\begin{aligned} i \in X_\alpha(t) &\Leftarrow \ \alpha_t(x) \neq 0 \ \wedge \ \beta_{T_M - t}(x) \neq 0 \\ &\Leftarrow \ \exists_{\tau \leq T_M - t} : \ \alpha_t(x) \neq 0 \ \wedge \ \beta_\tau(x) \neq 0 \\ &\iff \ \exists_{\tau \leq T_M - t} : \ \gamma_{t\tau}(x) \neq 0 \end{aligned} \tag{43}$$

Thus, every state that is potentially visited at time t (for which $\exists_{\tau : t+\tau \leq T_M} : \gamma_{t\tau}(x) \neq 0$) is included in $X_\alpha(t)$. We will exclude all states $x \notin X_\alpha(t)$ from the α-propagation procedure and not deliver their messages. The constraint $t < T_M/2$ concerning the β's was inserted in the definition of $X_\alpha(t)$ only because of the feasibility of computing $X_\alpha(t)$ at iteration time t. Initializing $X_\alpha(0) = \{x \,|\, P_x \neq 0\}$, we can compute $X_\alpha(t)$ recursively via

$$X_\alpha(t) = \begin{cases} X_\alpha(t-1) \cup \mathsf{OUT}(X_\alpha(t-1)) & \text{for } t < T_M/2 \\ \Big[X_\alpha(t-1) \cup \mathsf{OUT}(X_\alpha(t-1))\Big] \cap \{x \,|\, \beta_{T_M - t}(x) \neq 0\} & \text{for } t \geq T_M/2 \end{cases} ,$$

where $\mathsf{OUT}(X_\alpha(t-1))$ is the set of states which have non-zero probability transitions from states in $X_\alpha(t-1)$. Analogously, the book keeping for states that participate in the β-propagation is

$$X_\beta(0) = \{x \,|\, R_{ax} \neq 0\} \tag{44}$$

$$X_\beta(\tau) = \begin{cases} X_\beta(\tau-1) \cup \mathsf{IN}(X_\beta(\tau-1)) & \text{for } \tau < T_M/2 \\ \Big[X_\beta(\tau-1) \cup \mathsf{IN}(X_\beta(\tau-1))\Big] \cap \{x \,|\, \alpha_{T_M - \tau}(x) \neq 0\} & \text{for } \tau \geq T_M/2 \end{cases} . \tag{45}$$

For the discount prior, we can use a time cutoff T_M for which we expect further contributions to be insignificant. The choice of this cutoff involves a payoff between computational cost and accuracy of the E-step. Let T_0 be the minimum time for which $L^{\boldsymbol{\pi}}_{T_0} \neq 0$. It is clear that the cutoff needs to be greater than T_0. In the experiment in section 4.1 we used an increasing schedule for the cutoff time, $T_M = (1 + 0.2\,k)\,T_0$, depending on the iteration k of the EM-algorithm to ensure that with each iteratin we become more accurate.

C Derivations of the POMDP M-steps

All M-steps are of the same form: maximize $Q(\theta, \theta^*) = \sum_z f(z, \theta) \log \theta^*_z$ subject to some normalization constraints, such that the M-step is $\theta^*_z \propto f(z, \theta)$. Given $Q(\theta, \theta^*)$ from (38) and in analogy

to the M-step in the MDP case, we derive the update (39) for π_{aby} as

$$Q(\theta,\theta^*) = \langle \text{terms indep. of } \pi^* \rangle + \sum_{T=0}^{\infty}\sum_{t=0}^{T}\sum_{aby}\big(\log \pi^*_{aby}\big)\; P(\hat{r}=1, a_t=a, b_t=b, y_t=y \,|\, T,\theta)\; P(T)$$

$$= \cdots + \sum_{aby}\big(\log \pi^*_{aby}\big)\; \sum_{T=0}^{\infty} P(T)\; \Big[\sum_{x} \widehat{R}_{ax}\; \pi_{aby}\; P_{yx}\; \alpha_T(b,x)$$

$$+ \sum_{t=0}^{T-1}\sum_{b'x'x} \beta_{T-t-1}(b',x')\; \lambda_{b'by}\; P_{x'ax}\; \pi_{aby}\; P_{yx}\; \alpha_t(b,x)\Big]$$

$$= \cdots + \sum_{aby}\big(\log \pi^*_{aby}\big)\; \pi_{aby}\; \sum_{x} P_{yx}\; \widehat{\alpha}(b,x)\Big[\widehat{R}_{ax} \;+\; \frac{\gamma}{1-\gamma}\; \sum_{b'x'} \widehat{\beta}(b',x')\; \lambda_{b'by}\; P_{x'ax}\Big] \tag{46}$$

The updates for λ_{bby} and ν_b follow similarly. The last equality simplified time summations as follows:

$$\sum_{T=0}^{\infty}\sum_{t=0}^{T-1} P(T)\; \beta_{T-t-1}(b',x')\; \alpha_t(b,x) = \sum_{t=0}^{\infty}\sum_{T=t+1}^{\infty} (1-\gamma)\; \gamma^T\; \beta_{T-t-1}(b',x')\; \alpha_t(b,x)$$

$$= (1-\gamma)\; \sum_{t=0}^{\infty}\sum_{\tau=0}^{\infty} \gamma^{\tau}\; \gamma^{t+1}\; \beta_{\tau}(b',x')\; \alpha_t(b,x) = \frac{\gamma}{1-\gamma}\Big[\sum_{\tau=0}^{\infty} P(\tau)\; \beta_{\tau}(b',x')\Big]\; \Big[\sum_{t=0}^{\infty} P(t)\; \alpha_t(b,x)\Big]$$

$$= \frac{\gamma}{1-\gamma}\; \widehat{\beta}(b',x')\; \widehat{\alpha}(b,x) \tag{47}$$

References

Atkeson, C. & J. Santamaría (1997). A comparison of direct and model-based reinforcement learning. In *Int. Conf. on Robotics and Automation*.

Attias, H. (2003). Planning by probabilistic inference. In C. M. Bishop & B. J. Frey (Eds.), *Proc. of the 9th Int. Workshop on Artificial Intelligence and Statistics*.

Beal, M. J., Z. Ghahramani, & C. E. Rasmussen (2002). The infinite hidden markov model. In T. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14*. MIT Press.

Boutilier, C., R. Dearden, & M. Goldszmidt (1995). Exploiting structure in policy construction. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI 1995)*, pp. 1104–1111.

Bui, H., S. Venkatesh, & G. West (2002). Policy recognition in the abstract hidden markov models. *Journal of Artificial Intelligence Research* **17**, 451–499.

Chavira, M., A. Darwiche, & M. Jaeger (2006). Compiling relational bayesian networks for exact inference. *International Journal of Approximate Reasoning* **42**, 4–20.

Guestrin, C., D. Koller, R. Parr, & S. Venkataraman (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research (JAIR)* **19**, 399–468.

Hauskrecht, M., N. Meuleau, L. P. Kaelbling, T. Dean, & C. Boutilier (1998). Hierarchical solution of Markov decision processes using macro-actions. In *Proc. of Uncertainty in Artificial Intelligence (UAI 1998)*, pp. 220–229.

Johnson, A. & A. Redish (2006). Neural ensembles in CA3 transiently encode paths forward of the animal at a decision point: a possible mechanisms for the consideration of alternatives. Program No. 574.2. 2006 Neuroscience Meeting Planner. Society for Neuroscience, 2006.

Kaelbling, L., M. Littman, & A. Moore (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* **4**, 237–285.

Kaelbling, L. P., M. L. Littman, & A. R. Cassandra (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence* **101**, 99–134.

Koller, D. & R. Parr (1999). Computing factored value functions for policies in structured MDPs. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI 1999)*, pp. 1332–1339.

Kveton, B. & M. Hauskrecht (2005). An MCMC approach to solving hybrid factored MDPs. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*.

Littman, M. L., S. M. Majercik, & T. Pitassi (2001). Stochastic boolean satisfiability. *Journal of Automated Reasoning 27*(3), 251–296.

McGovern, A. & A. G. Barto (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proc. of 18th Int. Conf. on Machine Learning (ICML 2001)*, pp. 361–368.

Meuleau, N., L. Peshkin, K.-E. Kim, & L. P. Kaelbling (1999). Learning finite-state controllers for partially observable environments. In *Proc. of Fifteenth Conf. on Uncertainity in Artificial Intelligence (UAI 1999)*, pp. 427–436.

Minka, T. (2001). A family of algorithms for approximate bayesian inference. PhD thesis, MIT.

Murphy, K. (2002). Dynamic bayesian networks: Representation, inference and learning. PhD Thesis, UC Berkeley, Computer Science Division. See particularly the chapter on DBN at http://www.cs.ubc.ca/ murphyk/Papers/dbnchapter.pdf.

Ng, A. Y., R. Parr, & D. Koller (1999). Policy search via density estimation. In *Advances in Neural Information Processing Systems 12*, pp. 1022–1028.

Poupart, P. & C. Boutilier (2003). Bounded finite state controllers. In *Advances in Neural Information Processing Systems 16 (NIPS 2003)*.

Theocharous, G., K. Murphy, & L. Kaelbling (2004). Representing hierarchical POMDPs as DBNs for multi-scale robot localization. In *Intl. Conf. on Robotics and Automation (ICRA 2004)*.

Toussaint, M. (2006). A sensorimotor map: Modulating lateral interactions for anticipation and planning. *Neural Computation* **18**, 1132–1155.

Toussaint, M. & C. Goerick (2007). Probabilistic inference for structured planning in robotics. Submitted to *Int. Conf. on Robotics and Automation (ICRA 2007)*.

Verma, D. & R. P. N. Rao (2006). Goal-based imitation as probabilistic inference over graphical models. In *Advances in Neural Information Processing Systems 18 (NIPS 2005)*.

Wiegerinck, W., B. van den Broek, & H. Kappen (2006). Stochastic optimal control in continuous space-time multi-agent systems. In *Proceedings UAI 2006*.

Zettlemoyer, L., H. Pasula, & L. P. Kaelbling (2005). Learning planning rules in noisy stochastic worlds. In *Proc. of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*.