# Inference strategies for solving semi-Markov decision processes

Matthew Hoffman  Nando de Freitas

Department of Computer Science
University of British Columbia, Canada

**Abstract**

Semi-Markov decision processes are used to formulate many control problems and also play a key role in hierarchical reinforcement learning. In this chapter we show how to translate the decision making problem into a form that can instead be solved by inference and learning techniques. In particular, we will establish a formal connection between planning in semi-Markov decision processes and inference in probabilistic graphical models, then build on this connection to develop an expectation maximization (EM) algorithm for policy optimization in these models.

## 1 Introduction

Researchers in machine learning have long attempted to join the fields of inference and learning with that of decision making. Influence diagrams, for example, explicitly cast the decision making process as inference in a graphical model [see e.g. 4, 20]. However, while these methods are a straight-forward application of inference techniques they only apply to finite-horizon problems and only learn non-stationary policies.

For goal-directed decision problems, more general techniques such as that of Attias [1] exist for finding the *maximum a posteriori* action sequence. (This technique was later extended in [27] to compute the *maximal probable explanation.*) It is crucial to note, however, that these approaches are not optimal in an expected reward sense. Instead, they can be interpreted as maximizing the probability of reaching the goal.

While it is well known in the optimal control literature that there exists a fundamental duality between inference and control for the special case of linear-quadratic Gaussian models [15], this result does not hold in general. Extending these ideas to more general models has been attempted by locally approximating the optimal solution [see e.g. 24, 23].

A key step in realizing general inference-based approaches while still maintaining optimality with respect to expected rewards was originally addressed by [6] for immediate reward decision problems. In particular this work proposes an expectation maximization (EM) approach to the problem which works by optimizing a lower bound on the expected rewards. This technique was then greatly formalized by Toussaint et al. [26] who extend it to the infinite-horizon case [see also 25]. This line of research has since enjoyed substantial success in the field of robotics [19, 16, 28], where empirical evidence has indicated that these methods can often outperform traditional stochastic planning and control methods as well as more recent policy gradient schemes.

The focus of this chapter is two-fold: to act as an introduction to the "planning as inference" methodology and to show how to extend these techniques to semi-Markov Decision Processes (SMDPs). SMDPs are an extension of the MDP formalism that generalize the notion of time—in particular, by allowing the time-intervals between state transitions to vary stochastically. This allows us to handle tradeoffs not only between actions based on their expected rewards, but also based on the amount of time that each action takes to perform.

SMDPs are interesting problems in their own right, with applications to call admission control and queueing systems [21, 5]. This formalism also serves as a natural platform in robotics for building complex motions from sequences of smaller motion "templates" [18]. Finally, SMDPs are a crucial building block for hierarchical reinforcement learning methods [10, 22, 9]. While this chapter serves as an introductory text to the paradigm of inference and learning, and its application to SMDPs, we hope that future work in this area will leverage advances in structured inference techniques for hierarchical tasks of this nature.

In Section 2 we will describe the basic mixture of MDPs model that we build on and in Section 3 will show how to extend this to the SMDP formalism. Section 4 describes an EM algorithm for solving these problems. Finally, in Section 5 we apply this approach to a small SMDP example.

## 2 A mixture of finite-time MDPs

Following the notation of [12] an MDP can be succinctly described via the following components:

- an initial state model $p(x_0)$,
- a state transition model $p(x_{n+1}|x_n, u_n)$,
- an immediate reward model $r(x_n, u_n)$,
- and finally a stochastic policy $\pi_\theta(u_n|x_n)$.

In this model, $n = 1, 2, \dots$ is a discrete-time index, $\{x_n\}$ is the state process, and $\{u_n\}$ is the action process. The model further assumes a randomized policy, but one can also easily adopt a deterministic policy $\pi_\theta(u|x) = \delta_{\phi_\theta(x)}(u)$, where $\delta$ denotes the Dirac function and $\phi$ is a deterministic mapping from states to actions. (By this same reasoning we can also encode knowledge of the initial state using a Dirac mass.) We will assume that the policy-parameters are real-valued, i.e. $\theta \in \mathbb{R}^d$.

Having defined the model, our objective is to maximize the expected future reward with respect to the parameters of the policy $\theta$:

$$J(\theta) = \mathbb{E}\Big[\sum_{n=0}^{\infty} \gamma^n \, r(x_n, u_n)\big|\theta\Big], \tag{1}$$

where $0 < \gamma < 1$ is a discount factor. In order to ease notation later we will also note that for a given $\theta$ this model induces a Markov chain over state/action pairs $z_n = (x_n, u_n)$. The transition probabilities for this "extended state space" can then be written as

$$\begin{aligned} p_\theta(z_0) &= p(x_0)\,\pi_\theta(u_0|x_0) \text{ and} \\ p_\theta(z_{n+1}|z_n) &= p(x_{n+1}|x_n, u_n)\,\pi_\theta(u_{n+1}|x_{n+1}), \end{aligned}$$

where the joint distribution over any finite $k$-length sequence of state/action pairs is defined as

$$p_\theta(z_{0:k}|k) = p_\theta(z_0)\prod_{n=1}^{k} p_\theta(z_n|z_{n-1}). \tag{2}$$

Finally, we will also write the rewards as $r(z) = r(x, u)$.

In order to transform the problem into one that is more amenable to inference methods we will first note that any maximum of $(1-\gamma)J(\theta)$ is also a maximum of $J(\theta)$, as this extra multiplicative term just rescales the expected reward. Now, by expanding (1) we can write the (rescaled) expected reward as

$$(1-\gamma)J(\theta) = (1-\gamma)\int \Big[p_\theta(z_0)\prod_{n=1}^{\infty} p_\theta(z_n|z_{n-1})\Big]\Big[\sum_{k=0}^{\infty}\gamma^k\, r(z_k)\Big]\, dz_{0:\infty}$$

and by exchanging the order of integration and summation we arrive at

$$\begin{aligned} &= \int (1-\gamma)\gamma^0 p_\theta(z_0|k=0)\, r(z_0)\, dz_0 \\ &\qquad + \int (1-\gamma)\gamma^1 p_\theta(z_{0:1}|k=1)\, r(z_1)\, dz_{0:1} + \dots \\ &= \sum_{k=0}^{\infty}\int \underbrace{(1-\gamma)\gamma^k}_{\text{time prior}}\,\underbrace{p_\theta(z_{0:k}|k)}_{\substack{\text{state/action}\\ \text{prior}}}\, r(z_k)\, dz_{0:k}. \end{aligned}$$

It is for this reason that the additional factor of $(1-\gamma)$ was introduced. Under this formulation the discounting terms can be seen as a geometric distribution $p(k) = (1-\gamma)\gamma^k$ and the expected reward under this random time-horizon can be written as

$$\begin{aligned} (1-\gamma)J(\theta) &= \mathbb{E}_{k,z_{0:k}}\big[r(z_k)|\theta\big], \\ p(k, z_{0:k}) &= p(k)\,p(z_{0:k}|k). \end{aligned} \tag{3}$$

As originally noted by Toussaint et al. [26], we can now view this problem as an infinite mixture of finite horizon MDPs where rewards only occur at the end of a chain whose length is given by the random variable $k$. A diagram of this interpretation is shown in Figure 1. We must emphasize, however, that

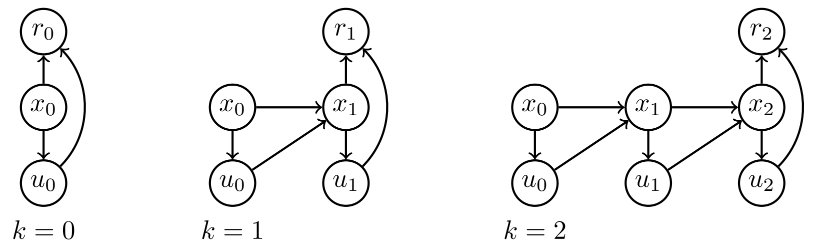

Figure 1: Illustration of the mixture of finite horizon MDPs described in Section 2. Expected rewards are computed by mixing over individual MDPs with probability $p(k)$ and taking reward $r_k = r(x_k, u_k)$.

we have not changed the model nor the form of the expected rewards, but are instead departing from the standard *interpretation* of these types of decision problems. The classical approach to these problems is to take the expectation of an infinite-length trajectory and sum over increasingly discounted rewards (i.e. the rewards are worth less as time passes due to the discount factor). Instead we are taking the expectation with respect to a *finite-length* trajectory whose length is stochastic with probability equal to the discounting associated with this length. We then evaluate the expected reward only at the end of this chain, but by taking the expectation with respect to $k$ we are essentially summing over the rewards at all such trajectory lengths. Note in particular that this is still an infinite-horizon problem!

This formulation is the basis for many inference and learning approaches to solving MDPs. In the next sections we will show how to extend this formulation in the context of SMDPs and will then describe an Expectation-Maximization (EM) algorithm for optimizing the parameters $\theta$. We will first note, however, that these techniques are quite general and can be applied in both discrete[1] and continuous state-spaces [26, 12]. As we will briefly see in Section 4.3 these methods can be applied to situations where the models themselves are unknown and can only be sampled from [16, 29]. Finally, it is also possible to derive Markov chain Monte Carlo (MCMC) methods to optimize via sampling in parameter space. While we will omit discussion of these methods entirely we can point the interested reader towards [13, 14] for an introduction. This further enables the solution of planning problems using generic inference algorithms [see 11].

## 3 An extension to semi-MDPs

Formally we can define an SMDP as a continuous-time controlled stochastic process $z(t) = (x(t), u(t))$ consisting, respectively, of states and actions at every point in time $t$. In particular we will assume that the system transitions at random *arrival times* $t_n$ and that the process is stationary in between jumps,

[1]For discrete models the integrals become sums (i.e. integration with respect to the counting measure).

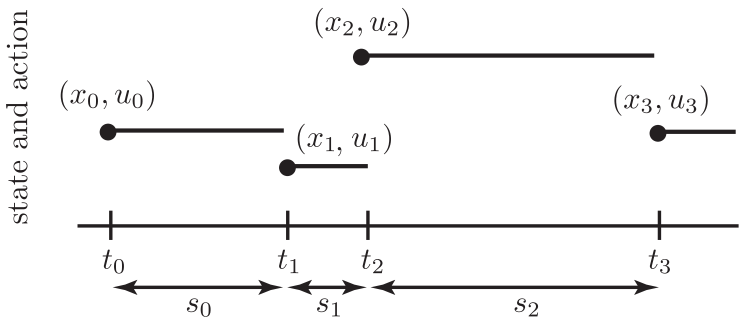

Figure 2: Relationship between arrival times $t_n$, sojourn times $s_n$, and the system state $(x_n, u_n)$.

i.e. $x(t) = x_n$ and $u(t) = u_n$ for all $t \in [t_n, t_{n+1})$. Here each of the state/action pairs evolves according to models defined in Section 2 for MDPs. What makes this model a *semi*-MDP is the use of random transition times. In order to handle this generalization we will introduce random *sojourn times* $s_n > 0$ which represent the amount of time spent in the $n$th state. We will then assume the following distribution:

- a time model $p(s_n|x_n, u_n)$,
- where $t_n = t_{n-1} + s_n$ and $t_0 = 0$.

Importantly, the conditional probability for sojourn times does not include the duration of the previous interval. See Fig. 2 for an illustration of this process. More generally, we could also allow the sojourn times $s_n$ to depend on the next state $x_{n+1}$. While the methods we will describe are fully capable of handling this situation, we will ignore this dependency for notational simplicity. Finally, we can write the joint probability over sequences of states/action pairs and sojourn times as

$$p_\theta(z_{0:k}, s_{0:k}) = p_\theta(z_0)\, p(s_0|z_0) \prod_{n=1}^{k} p(s_n|z_n)\, p_\theta(z_n|z_{n-1}) \tag{4}$$

for any choice of horizon $k$.

Just as in the standard MDP formulation we must also specify a reward function $r(z) = r(x, u)$ over states and actions. However, unlike in an MDP our discounting behaves differently in order to take into account the variable time in each state. In particular, we will discount the reward continuously over our entire trajectory, which because of the jump-Markov nature of our transitions will simplify to

$$\begin{aligned} J(\theta) &= \mathbb{E}\Big[\textstyle\int_0^\infty \beta e^{-\beta t}\, r(z(t))\, dt \Big| \theta\Big] \\ &= \mathbb{E}\Big[\textstyle\sum_{n=0}^\infty e^{-\beta t_n} (1 - e^{-\beta s_n})\, r(z_n) \Big| \theta\Big]. \end{aligned} \tag{5}$$

Here $\beta > 0$ is a discount rate which is analogous to the discount factor in an MDP. In fact, when the sojourn times are deterministically given by $s_n = 1$ we recover a standard infinite-horizon, discounted MDP with a discount factor of $\gamma = e^{-\beta}$. Those readers more familiar with continuous-time control problems may note that in the first line of the expected reward definition we have introduced an additional multiplicative term of $\beta$. We do this so that later we can give the discount factors a more intuitive interpretation as a probability distribution, and we point out that since this is a multiplicative constant it will not change the optimal policy parameters of the system.

Based on the intuition developed in Section 2 for MDPs we can interpret the discount terms in (5) as a distribution over random time horizons $k$ and write the following joint distribution over paths and path lengths:

$$p_\theta(k, z_{0:k}, s_{0:k}) = e^{-\beta(s_0+\cdots+s_{k-1})}(1 - e^{-\beta s_k})\, p_\theta(z_{0:k}, s_{0:k}). \tag{6}$$

This distribution is not, however, nearly as straightforward as that in the MDP case. Nor is there the simple division between path-lengths and paths.

**Proposition 1.** *The joint distribution introduced in (6) is properly defined and normalized, i.e. it integrates to 1.*

*Proof.* In theory we can integrate out each of the infinitely many trajectory and time variables see that for any $k$ the marginal over path lengths is given by

$$p_\theta(k) = \mathbb{E}[e^{-\beta s_0}] \cdots \mathbb{E}[e^{-\beta s_{k-1}}]\,(1 - \mathbb{E}[e^{-\beta s_k}]).$$

Given this marginal we can consider an infinite sequence of non-identical Bernoulli trials, where $p_n = \mathbb{E}[e^{-\beta s_n}]$ is the probability that the $n$th trial fails. Note, we need not compute this quantity in practice.

Due to the restriction that sojourn times are greater than zero we know that each such quantity lies in the range $0 < p_n \leq 1$. With this in mind, we can then see that $p_\theta(k)$ is the probability that this sequence of trials has its first success after $k-1$ failures. As a result the marginal distribution over $K$ can be properly defined as a conjunction of Bernoulli random variables and thus the full joint (i.e. not integrating over the trajectory and time variables) must be similarly well defined and integrate to 1. $\square$

Finally, given the joint distribution $p_\theta(k, z_{0:k}, s_{0:k})$ we can rewrite our objective from (5) as the *expected final reward* under this distribution, i.e.

$$\begin{aligned}
J(\theta) &= \int p_\theta(z_{0:\infty}, s_{0:\infty}) \Big[\textstyle\sum_{k=0}^{\infty} e^{-\beta t_k}(1 - e^{-\beta s_k})\, r(z_k)\Big]\, ds_{0:\infty}\, dz_{0:\infty} \\
&= \sum_{k=0}^{\infty} \int p_\theta(z_{0:k}, s_{0:k})\, e^{-\beta t_k}(1 - e^{-\beta s_k})\, r(z_k)\, ds_{0:k}\, dz_{0:k} \\
&= \sum_{k=0}^{\infty} \int p_\theta(k, z_{0:k}, s_{0:k})\, r(z_k)\, ds_{0:k}\, dz_{0:k} \\
&= \mathbb{E}_{k, z_{0:k}}\big[r(z_k)\big|\theta\big]. \qquad (7)
\end{aligned}$$

Similar to the MDP case we have obtained this result by exchanging the order of integration and summation and pulling the discount factor into the previously introduced distribution from (6). In the next section we will take this one step further and treat the reward terms $r(z_k)$ as the likelihood of some "imaginary event". We can then use this to develop an EM algorithm for finding the most likely policy parameters under this "data".

Finally, as an aside, we should note how the formulations of this section simplify in the MDP case, and more importantly why these simplifications do not hold for SMDPs. In particular, for an MDP it is not necessary to reason about times $s_n$ (since these are deterministically set to 1) and the joint distribution (6) can be factorized as

$$p_\theta(k, z_{0:k}) = p(k)\, p_\theta(z_{0:k}|k).$$

Here the conditional distribution is given by (4) and the "time prior" $p(k)$ is given by a geometric distribution with success probability $\gamma = e^{-\beta}$, i.e. the constant discount factor. This factorization is what allowed [26] to reformulate the infinite-horizon MDP problem as a mixture of finite-time MDPs, where the random variable $k$ acts as an indicator variable. Unfortunately this interpretation does not hold in the case of more general SMDPs. By looking at the discount factors in (6) we can see that the probability of a specific trajectory length $k$ is a function of all sojourn times $s_{0:k}$, and as a result the distribution over the random variable $k$ depends on an infinite number of sojourn times. However, while the SMDP formalism does not have as clean of a probabilistic interpretation as MDPs, we can still apply this model by working directly with the joint distribution.

## 4 An EM algorithm for SMDPs

Expectation-Maximization (EM) is an algorithm formulation that is used to compute maximum likelihood estimates in the presence of unobserved or hidden variables. In our case, the unobserved variables consist of the trajectory length $k$ along with the state/action pairs and their corresponding sojourn times, $z_{0:k}$

and $s_{0:k}$ respectively. The observed variables for this model are then implicitly given by the reward function—i.e. we are assuming some *imaginary* random variable was observed where the likelihood of this observation conditioned on our hidden variables is given by $r(z_k)$. Note, treating the rewards as likelihoods does place some restrictions on their allowable forms. While they need not sum to 1, they must be positive. However, for finite models or models which are bounded below this is easily obtainable by adding some constant term.

Given this interpretation we can introduce the following quantities which are necessary for deriving the EM algorithm:

- The *complete data likelihood* is the likelihood of both our observed and unobserved data; here given by $r(z_k)\, p_\theta(k, z_{0:k}, s_{0:k})$.
- The *incomplete data likelihood* is the integral of the complete data likelihood with respect to the hidden variables; here given by $\mathbb{E}[r(z_k)|\theta]$.
- Finally, the *predictive distribution* over the hidden variables is given by the "posterior" that takes into account both the prior and likelihood, and is thus the ratio of complete and incomplete likelihoods.

In particular, we will write the predictive distribution using the following notation:

$$\widetilde{p}_\theta(k, z_{0:k}, s_{0:k}) = \frac{r(z_k)\, p_\theta(k, z_{0:k}, s_{0:k})}{\mathbb{E}[r(z_k)|\theta]}. \tag{8}$$

Given these quantities the EM algorithm is an iterative procedure which at iteration $i$ computes the *expected* complete data log-likelihood, parameterized by $\theta$, under the previous iteration's policy parameters $\theta^{(i-1)}$. This quantity, which we will denote with $Q(\theta, \theta^{(i-1)})$, is then optimized with respect to the policy parameters in order to obtain $\theta^{(i)}$. We can summarize this procedure as:

$$Q(\theta, \theta^{(i-1)}) = \mathbb{E}\big[\log\{p_\theta(k, z_{0:k}, s_{0:k})\, r(z_k)\}\big|\theta^{(i-1)}\big], \tag{E-step}$$

$$\theta_i = \arg\max_\theta Q(\theta, \theta^{(i-1)}). \tag{M-step}$$

This EM procedure is known to locally maximize the incomplete data likelihood [8, 17] and hence will maximize our original objective $J(\theta)$.

Before deriving the E-step in more detail we will first take a brief look at the quantities that are actually needed in order to perform the maximization in the M-step. We can first let $\theta'$ denote the previous iteration's parameter estimate and rewrite the $Q$-function as

$$\begin{aligned} Q(\theta, \theta') = \mathbb{E}\Big[&\log\big\{\textstyle\prod_{n=0}^k \pi_\theta(z_n)\big\}\Big|\theta'\Big] + \\ &\mathbb{E}\Big[\log\big\{r(z_k)\, p(x_0) \textstyle\prod_{n=1}^k p(x_n|z_{n-1})\big\}\Big|\theta'\Big] + \\ &\mathbb{E}\Big[\log\big\{\textstyle\prod_{n=0}^k p(s_n|z_n)\big\}\Big|\theta'\Big] \end{aligned}$$

where we have expanded the complete data likelihood and separated those terms which do and do not depend on $\theta$. Since only the first of these three quantities depends on $\theta$ we can drop the others and expand this expectation:

$$
\begin{aligned}
&= \sum_{k=0}^{\infty} \int \Big[ \sum_{n=0}^{k} \log \pi_\theta(z_n) \Big] \widetilde{p}_{\theta'}(k, z_{0:k}) \, dz_{0:k} + \text{const.} \\
&= \sum_{k=0}^{\infty} \sum_{n=0}^{k} \int \log \pi_\theta(z_n) \, \widetilde{p}_{\theta'}(k, z_n) \, dz_n + \text{const.} \qquad (9)
\end{aligned}
$$

Finally, in order to optimize the $Q$-function with respect to the policy parameters $\theta$ we will also need to evaluate the gradient

$$
\nabla_\theta Q(\theta, \theta') = \sum_{k=0}^{\infty} \sum_{n=0}^{k} \int \widetilde{p}_{\theta'}(k, z_n) \nabla_\theta \log \pi_\theta(z_n) \, dz_n. \qquad (10)
$$

Ultimately, this expansion informs how we will derive the steps required for the EM algorithm. In the E-step we need to construct the distribution $\widetilde{p}_{\theta'}(k, z_n)$ and in the M-step we will compute the expectation of the gradient $\nabla \log \pi_\theta(z_n)$ under this distribution. We should note that this is the same form of the marginal distribution that would be computed in the E-step for a standard MDP. We will see shortly, however, that the computations necessary to compute this distribution are different due the integration over sojourn-times. In the next two subsections we will discuss these steps in more detail.

## 4.1 The E-step

As noted in the previous section, we need to construct the marginals $\widetilde{p}_\theta(k, z_n)$ in order to compute the gradient of the expected complete log-likelihood. In this section, we will derive an efficient method for recursively constructing this distribution. We start by writing the marginal distribution as the integral of the predictive distribution with respect to all those terms other than $k$ and $z_n$,

$$
\widetilde{p}_\theta(k, z_n) = \int \widetilde{p}_\theta(k, z_{0:k}, s_{0:k}) \, dz_{0:n-1} \, dz_{n+1:k} \, ds_{0:k}.
$$

This integral can then be broken into those components that come before and after $n$ respectively, and we can then see that the marginal distribution is proportional to

$$
\begin{aligned}
\widetilde{p}_\theta(k, z_n) \propto &\int e^{-\beta(s_0 + \cdots + s_{n-1})} \, p_\theta(z_{0:n}, s_{0:n-1}) \, dz_{0:n-1} \, ds_{0:n-1} \quad \times \qquad (11) \\
&\int e^{-\beta(s_n + \cdots + s_{k-1})} (1 - e^{-\beta s_k}) \, r(z_k) \, p_\theta(z_{n+1:k}, s_{n:k} | z_n) \, dz_{n+1:k} \, ds_{n:k}.
\end{aligned}
$$

We should emphasize the fact that we are *not* integrating over $z_n$, which enables us to break the original integral into two independent integrals. Here we have also omitted the constant of proportionality, given by the expected reward $\mathbb{E}[r(z_k)]$.

In an analogy to the forward-backward algorithm for hidden Markov models we will call the two sets of integrals from (11) "forward" and "backward" messages, denoting them as $\alpha_\theta(z_n)$ and $\beta_\theta(z_n|\tau = k-n)$ respectively[2]. Using these messages we can then write the marginal distribution as

$$\widetilde{p}_\theta(k, z_n) = \frac{1}{\mathbb{E}[r(z_k)]}\,\alpha_\theta(z_n)\,\beta_\theta(z_n|k-n). \tag{12}$$

Intuitively the forward messages are integrating information forward in time from the initial-state distribution where the $n$th such message is the state/action distribution at step $n$ weighted by the expected discounting up to step $n-1$. Meanwhile, the backward messages are integrating information backward from the rewards and can be seen as the expected reward at step $k$ given the state/action pair at step $n$ (weighted by the expected discounting between $n$ and $k$). It is crucial to note, however, that these messages are not probability distributions due to the way the discount factors have been split between the forward and backward components, namely:

$$\underbrace{e^{-\beta(s_0+\cdots+s_{n-1})}}_{\text{forward}}\,\underbrace{e^{-\beta(s_n+\cdots+s_{k-1})}(1-e^{-\beta s_k})}_{\text{backward}}.$$

This causes no technical (or conceptual) difficulties, though, because when combined in (12) these messages form the desired probability distribution. This is similar in spirit to techniques used to maintain numerical stability when working with hidden Markov models [3].

At this point, we can also more fully describe the use of the $\tau$ term in defining the backward messages. The reason behind this notation stems from the fact that naively computing these messages for each choice of $k$ and $n$ turns out to involve a great deal of redundant computation. Under the predictive distribution defined in (8) the reward factors only interact with the end of a finite-length trajectory. As a result the backward messages depend only on the difference $\tau = k - n$, i.e. how far in the future the rewards are obtained. Because of this, we can instead define the backward messages purely in terms of the "time-to-go". This notation was originally presented by [26], but here the messages are generalized to account for the fact that in the semi-Markov setting the $k$th reward introduces a factor over the state and action $z_n$ and duration $s_n$ at every epoch $n$ prior to $k$.

Finally, by integrating the components of the first integral in (11) succes-

[2]The notation here is slightly confusing in that we have a term $\beta$ denoting the continuous discount factor and $\beta_\theta(\cdot|\tau)$ denoting the backward messages. This confusion, however, seems unavoidable as both of these terms are unanimously used in their respective literatures. To somewhat alleviate this confusion we note that the backward messages are *always* subscripted.

sively we can recursively define the forward messages as

$$\begin{aligned}
&\alpha_\theta(z_0) = \mu(x_0)\,\pi_\theta(u_0|x_0),\\
&\alpha_\theta(z_n) = \int \alpha_\theta(z_{n-1})\,p_\theta(z_n|z_{n-1})\,dz_{n-1} \times \int e^{-\beta s_{n-1}}\,p_\theta(s_{n-1}|z_{n-1})\,ds_{n-1}.
\end{aligned} \tag{13}$$

Here we can see that we have the standard MDP forward message recursion multiplied by an additional integral due to the sojourn times. Similarly, we can recursively define the backwards messages as

$$\begin{aligned}
&\beta_\theta(z_n|0) = r(z_n) \int (1 - e^{-\beta s_n})\,p_\theta(s_n|z_n)\,ds_n,\\
&\beta_\theta(z_n|\tau) = \int \beta_\theta(z_{n+1}|\tau-1)\,p_\theta(z_{n+1}|z_n)\,dz_{n+1} \times \int e^{-\beta s_n}\,p_\theta(s_n|z_n)\,ds_n.
\end{aligned} \tag{14}$$

Again we can see that these messages can be seen as two integrals, one corresponding to the standard MDP message and a sojourn time message. Given the format of these two messages we can further introduce what we call an "expected discount factor"

$$\gamma(z) = \int e^{-\beta s}\,T(s|z)\,ds \tag{15}$$

which corresponds to the integral over sojourn times noted above. We can consider this term as a generalization of the MDP formalism wherein discount factors are no longer constant and instead depend on the current state and the action taken from that state. Further, we can see that for any exponential-family distribution this integral will exist in closed form.

## 4.2 The M-step

The M-step requires us to maximize the $Q$-function with respect to the policy parameters $\theta$. If possible we can analytically maximize this function by solving for the fixed point of $\nabla_\theta Q(\theta, \theta') = 0$. If this is not possible we can still evaluate the gradient at the current set of policy parameters $\nabla_\theta Q(\theta', \theta')$ and follow the resulting ascent direction, resulting in a generalized EM algorithm (GEM). When this procedure is iterated, both of these methods are known to locally maximize the incomplete data likelihood [8, 17].

While EM methods are, in general, only able to guarantee local convergence it can be shown via its relation to policy iteration that these methods exhibit global convergence for discrete models when using exact/analytic inference (see [25] for more details). In more general continuous settings no such guarantees can be made, however as noted by [12] a sufficiently exploratory initial policy does seem to have a tempering effect. This is especially true if the exact EM updates can be used, as additional exploratory noise does not cause the dramatic increase in variance associated with sample-based methods (such as policy gradients).

### 4.3 Monte Carlo EM

It is also possible to perform a Monte Carlo approximation during the E-step in order to optimize these problems when either the necessary distributions are unknown or the updates cannot be computed in closed form. As is derived for MDPs in [16, 29], we can sample from the initial-state and transition distributions in order to approximate the $Q$-function. Given $M$ trajectories $\{z_{0:k}^{(i)}, s_{0:k}^{(i)}\}_{i\leq M}$ sampled from $p_{\theta'}(z_{0:k}, s_{0:k})$ we can approximate the joint distribution for any $n < k$ with

$$\widetilde{p}_{\theta'}(k, z_n) \approx \frac{1}{MA}\sum_{i=1}^{M}\left[e^{-\beta t_k^{(i)}}\left(1 - e^{-\beta s_k^{(i)}}\right)\right] r(z_k^{(i)})\,\delta_{z_n^{(i)}}(z_n),$$

where $A$ is a proportionality constant, given by $\mathbb{E}[r(z_k)]$ as noted earlier. If we assume some maximum time-horizon $K_{\max}$ we can approximate the $Q$-function as

$$\begin{aligned} Q(\theta, \theta') &\approx \sum_{i=1}^{M}\sum_{k=0}^{K_{\max}}\sum_{n=0}^{k}\left[e^{-\beta t_k^{(i)}}\left(1 - e^{-\beta s_k^{(i)}}\right)\right] r(z_k^{(i)}) \log \pi_\theta(z_n^{(i)}) \\ &= \sum_{i=1}^{M}\sum_{n=0}^{K_{\max}} \log \pi_\theta(z_n^{(i)}) \sum_{k=n}^{K_{\max}}\left[e^{-\beta t_k^{(i)}}\left(1 - e^{-\beta s_k^{(i)}}\right)\right] r(z_k^{(i)}). \end{aligned}$$

This function can then be optimized using the same techniques as in the standard M-step.

## 5 Discrete models with Gamma-distributed time

As an illustrative experiment we will consider a simple model where the states and actions are discrete and sojourn times are given by a Gamma distribution. While simple, this model nonetheless presents an interesting scenario for planning and control domains because it can naturally be extended to cases when we want to reason about more complex distributions over the time to complete an action. In our experiments, we define the discrete transition, initial-state, and reward models according to

$$\begin{aligned} \mu_x &= p(x_0 = x), \\ P_{xux'} &= p(x_{n+1} = x' | x_n = x, u_n = u), \text{ and} \\ R_{xu} &= r(x, u) \end{aligned}$$

where the sojourn times are Gamma-distributed according to

$$p(s_n | x_n = x, u_n = u) = \Gamma(s_n; k_{xu}, \sigma_{xu}).$$

Finally, we will assume a discrete policy where $\theta_{xu} = \pi_\theta(u|x)$.

Under this formulation the forward and backward messages will be representable as matrices, and by dropping the $\theta$ index we will let $\alpha^n_{xu}$ and $\beta^\tau_{xu}$ denote the $n$-step forward message and $\tau$-step backward messages respectively. By plugging initial state, transition, and reward matrices into (13) and (14) we can explicitly write these messages as

$$\alpha^n_{xu} = \theta_{xu} \sum_{x',u'} \alpha^{n-1}_{x'u'}\, P_{x'u'x}\, \gamma_{x'u'}, \qquad \alpha^0_{xu} = \theta_{xu}\, \mu_x; \tag{16}$$

$$\beta^\tau_{xu} = \gamma_{xu} \sum_{x',u'} \beta^{\tau-1}_{x'u'}\, P_{xux'}\, \theta_{x'u'}, \qquad \beta^0_{xu} = R_{xu}(1-\gamma_{xu}). \tag{17}$$

To make notation easier we will also introduce a forward message defined only over states, $\overline{\alpha}_x = \sum_u \alpha_{xu}$. In this setting the expected discount factor noted earlier can be written as the matrix

$$\begin{aligned}\gamma_{xu} &= \int \Gamma(s; k_{xu}, \sigma_{xu})\, e^{-\beta s}\, ds \\ &= \int s^{k_{xu}-1} \frac{\exp\big(-(\beta+\sigma^{-1}_{xu})\, s\big)}{\Gamma(k_{xu})\, \sigma^{k_{xu}}_{xu}}\, ds = (1+\beta\sigma_{xu})^{-k_{xu}}.\end{aligned}$$

This particular form arises purely from the use of Gamma-distributed sojourn times, and in fact we can imagine extending this to continuous spaces using functions $k(x,u)$ and $\sigma(x,u)$.

At every iteration, for a given set of parameters $\theta'$, the E-step consists of calculating the forward and backward messages given by Equations (16,17). By plugging these terms into the $Q$-function defined in (9) we can write

$$\begin{aligned}Q(\theta,\theta') &= \frac{1}{\mathbb{E}[r(z_k)]} \sum_{k=0}^{\infty} \sum_{n=0}^{k} \sum_{u,x} (\log\theta_{xu})\, \theta'_{xu}\, \overline{\alpha}^n_x\, \beta^{k-n}_{xu} \\ &= \frac{1}{\mathbb{E}[r(z_k)]} \sum_{u,x} (\log\theta_{xu})\, \theta'_{xu} \Big[\sum_{n=0}^{\infty} \overline{\alpha}^n_x\Big] \Big[\sum_{\tau=0}^{\infty} \beta^\tau_{xu}\Big],\end{aligned}$$

where the second equality can be obtained by rearranging the sums over $k$ and $n$. This alternate formulation is particularly useful in discrete models where the sum over forward and backward messages can expressed as finite quantities, i.e. a vector and a matrix respectively. Further, given this formulation we can optimize the $Q$-function for each state $x$ individually, which is possible because in discrete domains we can find the optimal action to take for each state regardless of the probability of visiting that state. By taking the gradient of the log-policy $\nabla \log \pi_\theta(u|x) = \theta^{-1}_{xu}$ and solving $\nabla Q(\theta,\theta') = 0$ for $\theta$, subject to the constraint that $\sum_u \theta_{xu} = 1$ for each $x$, we arrive at the following solution methods:

$$\theta_{xu} \propto \theta'_{xu} \sum_{\tau=0}^{\infty} \beta^\tau_{xu}, \tag{EM}$$

$$\theta_{xu} = \delta_{m(x)}(u) \quad \text{where } m(x) = \arg\max_{u'} \textstyle\sum_{\tau=0}^{\infty} \beta^\tau_{xu'}. \tag{greedy-EM}$$

Here the EM solution is performing exactly the optimization described above, while the greedy-EM solution, however, myopically chooses for each state $x$ the one action $u$ that maximizes the total future rewards when taken from that state. In particular, the greedy solution can be seen as iterations which correspond to repeated M-steps which skip intermediate E-steps as is shown by [25], and in this sense this method is equivalent (again only for discrete models) to policy iteration. In larger discrete models, however, the EM approach has the advantage over policy iteration in that it is possible to prune computation (by using the forward messages $\alpha$) in a principled manner; see [25] for more details.

We first test these algorithms on a small, 16-state, 5-action domain with randomly generated transition and sojourn parameters, as well as a randomly generated reward model. The principal challenge of this domain, over other discrete domains, is to take advantage of the structure in the sojourn time distribution. The top-left plot of Figure 3 displays convergence properties of the described algorithms as well as a comparison to a standard policy-gradient method; see e.g. [2]. In particular we should note that the resulting model was densely connected which allows for quick travel across the space, and explains the very good performance of the stochastic policy gradient algorithm. Also shown for policy gradients are error-bars corresponding to one standard deviation. The other methods don't need error bars because they are deterministic.

Building upon these results, the top-right plot shows the algorithms learning in a more structured environment. In particular the model used has grid-structured transitions on a small 4-by-4 grid. This model is especially interesting because we specify different Gamma-distributions for the inner nodes than the outer nodes such that the inner nodes move much more slowly. Also, we use a sparse reward model where most states have negligible reward and one state has high reward. The most important thing to note from this sub-figure is that the policy gradient method starts to break down under this sparse transition and reward model, even though the size of the state and action spaces are the same as in the previous example.

Lastly the bottom-left plot of this figure displays the progress of these algorithms on a much larger 20-by-20 grid, i.e. one in which there are 2000 state/action pairs. Similar to the previous example there is a single (relatively) large reward in the upper right corner of the grid and inner nodes with much slower sojourn times. Here we see that the EM algorithms vastly out-perform the policy gradient method and the learned policy successfully skirts the outside edge of the state-space in order to most quickly get to the high reward. Here the policy gradient method has relatively low-variance because it is not able to make any progress (i.e. it is stuck exploring a plateau with very little gradient information).

## 6 Conclusions

In this chapter, we have shown how it is possible to design effective planning algorithms in continuous-time domains by framing the policy optimization prob-

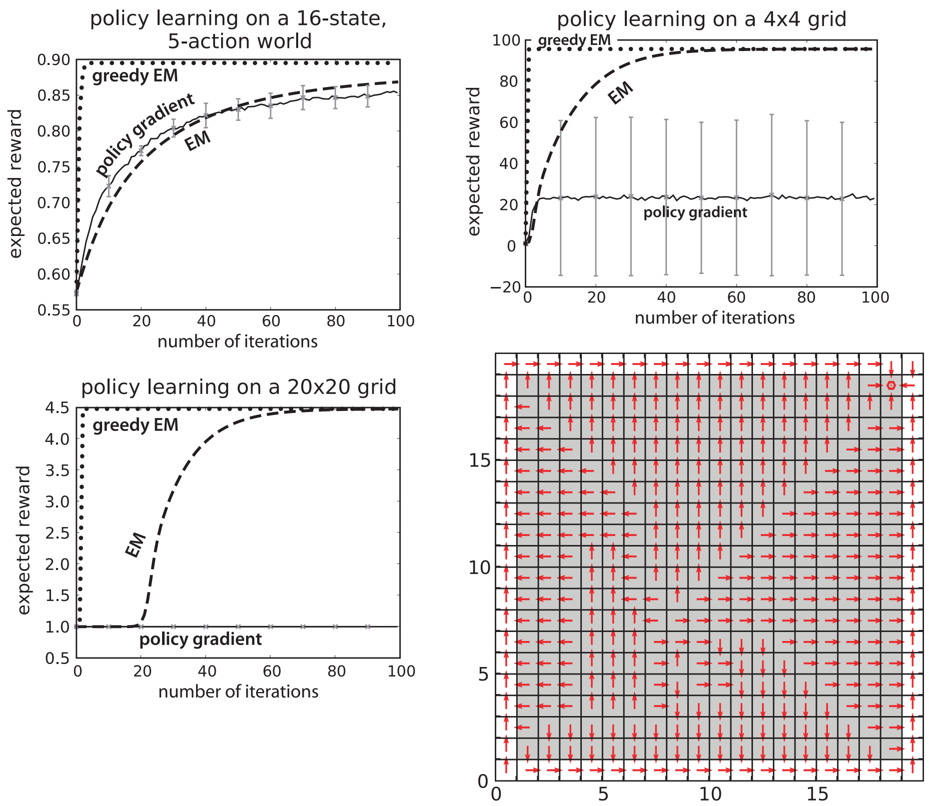

Figure 3: Results on various discrete models. The top-left plot shows convergence of the described algorithms on a randomly generated, dense model; the top-right plot shows performance on a model of the same size but with grid-like transitions and more structured transition times. The bottom-left plot shows performance on a larger grid domain. The bottom-right shows the learned policy in the larger domain. For both grid domains there is a single large reward (denoted with a dot) and transitions taken outside of the edge states (denoted with grey) have a larger expected sojourn time.

lem in an SMDP as one of inference in a related probabilistic graphical model. The connection between the reinforcement learning and probabilistic inference domains allows us to exploit existing developments in exact and approximate inference, and will perhaps provide us with leverage for tackling pressing problems in hierarchical reinforcement learning.

Of particular interest are inference approaches which attempt to exploit the structure of the underlying models. Preliminary work on applying such techniques to control problems includes the pruning steps of [25]. As evidenced by the small example in this chapter as well as larger continuous models in [12] we can see large performance gains by using exact inference methods. Another promising area of research would be the combination of these ideas with sample-based methods such as [16], perhaps via Rao-Blackwellization [7].

## References

[1] H. Attias. Planning by probabilistic inference. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2003.

[2] J. Baxter and P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15(4), 2001.

[3] C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[4] G. Cooper. A method for using belief networks as influence diagrams. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 1988.

[5] T. Das, A. Gosavi, S. Mahadevan, and N. Marchalleck. Solving semi-Markov decision problems using average reward reinforcement learning. *Management Science*, 45(4), 1999.

[6] P. Dayan and G. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2), 1997.

[7] N. de Freitas, R. Dearden, F. Hutter, R. Morales-Menendez, J. Mutch, and D. Poole. Diagnosis by a waiter and a Mars explorer. *Proceedings of the IEEE, special issue on sequential state estimation*, 92(3), 2004.

[8] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 39(1), 1977.

[9] T. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13(1), 2000.

[10] M. Ghavamzadeh and S. Mahadevan. Hierarchical average reward reinforcement learning. *The Journal of Machine Learning Research*, 8, 2007.

[11] N. Goodman, V. Mansinghka, D. Roy, K. Bonawitz, and J. Tenenbaum. Church: a language for generative models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2008.

[12] M. Hoffman, N. de Freitas, A. Doucet, and J. Peters. An expectation maximization algorithm for continuous Markov decision processes with arbitrary reward. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.

[13] M. Hoffman, A. Doucet, N. de Freitas, and A. Jasra. Bayesian policy learning with trans-dimensional MCMC. In *Advances in Neural Information Processing Systems*, 2007.

[14] M. Hoffman, H. Kück, N. de Freitas, and A. Doucet. New inference strategies for solving Markov decision processes using reversible jump MCMC. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2009.

[15] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1), 1960.

[16] J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems*, 2008.

[17] G. McLachlan and T. Krishnan. *The EM algorithm and extensions*. Wiley New York, 1997.

[18] G. Neumann, W. Maass, and J. Peters. Learning complex motions by sequencing simpler motion templates. In *Proceedings of the International Conference on Machine Learning*, 2009.

[19] J. Peters and S. Schaal. Reinforcement learning for operational space control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

[20] R. Shachter. Probabilistic inference and influence diagrams. *Operations Research*, 1988.

[21] S. Singh, V. Tadic, and A. Doucet. A policy gradient method for semi-Markov decision processes with application to call admission control. *European Journal of Operational Research*, 178(3):808–818, 2007.

[22] R. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1), 1998.

[23] E. Todorov and W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference, 2005. Proceedings of the 2005*, 2005.

[24] M. Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the International Conference on Machine Learning*, 2009.

[25] M. Toussaint, S. Harmeling, and A. Storkey. Probabilistic inference for solving (PO)MDPs. Technical Report EDI-INF-RR-0934, University of Edinburgh, School of Informatics Research, 2006.

[26] M. Toussaint and A. Storkey. Probabilistic inference for solving discrete and continuous state Markov decision processes. In *Proceedings of the International Conference on Machine Learning*, 2006.

[27] D. Verma and R. Rao. Planning and acting in uncertain environments using probabilistic inference. In *Intelligent Robots and Systems*, 2006.

[28] S. Vijayakumar, M. Toussaint, G. Petkos, and M. Howard. Planning and moving in dynamic environments: A statistical machine learning approach. In *Creating Brain Like Intelligence: From Principles to Complex Intelligent Systems*. Springer-Verlag, 2009.

[29] N. Vlassis and M. Toussaint. Model-free reinforcement learning as mixture learning. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.