On Solving General State-Space Sequential Decision Problems using Inference Algorithms

Univ. of British Columbia, Computer Science
Technical Report TR-2007-04

Matt Hoffman
Computer Science
UBC
Vancouver, Canada

Arnaud Doucet
Computer Science and Statistics
UBC
Vancouver, Canada

Nando de Freitas
Computer Science
UBC
Vancouver, Canada

Ajay Jasra
Mathematics
Imperial College
London, UK

Abstract

A recently proposed formulation of the stochastic planning and control problem as one of parameter estimation for suitable artificial statistical models has led to the adoption of inference algorithms for this notoriously hard problem. At the algorithmic level, the focus has been on developing Expectation-Maximization (EM) algorithms. For example, Toussaint *et al* (2006) uses EM with optimal smoothing in the E step to solve finite state-space Markov Decision Processes. In this paper, we extend this EM approach in two directions. First, we derive a non-trivial EM algorithm for linear Gaussian models where the reward function is represented by a mixture of Gaussians, as opposed to the less flexible classical single quadratic function. Second, in order to treat arbitrary continuous state-space models, we present an EM algorithm with particle smoothing. However, by making the crucial observation that the stochastic control problem can be reinterpreted as one of trans-dimensional inference, we are able to propose a novel reversible jump Markov chain Monte Carlo (MCMC) algorithm that is more efficient than its smoothing counterparts. Moreover, this observation also enables us to design an alternative full Bayesian approach for policy search, which can be implemented using a single MCMC run.

1 Introduction

Continuous state-space Markov Decision Processes (MDPs) are notoriously difficult to solve. Except for a few rare cases, including linear Gaussian models with quadratic cost, there is no closed-form solution and approximations are required [3]. A large number of methods have been proposed in the literature relying on value function approximation and policy search; including [2, 9, 12, 15, 16]. In this paper, we follow the policy learning approach because of its promise and remarkable success in complex domains; see for example [13, 14]. Our work is strongly motivated by a recent formulation of stochastic planning and control problems as inference problems. This line of work appears to have been initiated in [4], where the authors used EM as an alternative to standard stochastic gradient algorithms to maximize an expected cost. In [1], a planning problem under uncertainty was solved using a Viterbi algorithm. This was later extended in [19]. In these works, the number of time steps to reach the goal was fixed and the plans were not optimal in expected reward. An important step toward surmounting these limitations was taken in [17, 18]. In these works, the standard discounted reward control problem was expressed in terms of an infinite mixture of MDPs. To make the problem tractable, the authors proposed using the estimated posterior horizon time to truncate the mixture.

Here, we make the observation that, in the probabilistic approach to stochastic control, the objective function can be written as the expectation of a positive function with respect to a trans-dimensional probability distribution, *i.e.* a probability distribution defined on an union of subspaces of different dimensions. By reinterpreting this function as a (artificial) marginal likelihood, it is easy to see that it can also be maximized using an EM-type algorithm in the spirit of [4]. However, the observation that we are dealing with a trans-dimensional distribution enables us to go beyond EM. We believe it creates many opportunities for exploiting a large body of sophisticated inference algorithms in the decision-making context. In this paper, we make three contributions in this research direction.

First we extend the approach initiated in [17] and present a *closed-form derivation* of the EM algorithm for policy optimization in linear Gaussian models with a flexible model of the reward (a mixture of Gaussians). We believe this is an important step as it eliminates the need for quadratic cost functions in one of the most popular control algorithms in academia and industry: Linear Quadratic Gaussian (LQG) control.

Second, to attack general state-space models, we present two Monte Carlo implementations of the EM algorithm. The first method involves a Sequential Monte Carlo (SMC) implementation of optimal smoothing [8]. However, this method is computationally intensive and relies on a truncation of the time domain, as in [17]. Consequently, we propose an alternative trans-dimensional MCMC scheme to implement the E-step of the EM algorithm, which bypasses partially such problems.

Third, we propose a full Bayesian policy search alternative to the EM algorithm. In this approach, we set a prior distribution on the set of policy parameters and derive an artificial posterior distribution which is proportional to the prior times the expected return. In the simpler context of myopic Bayesian experimental design, a similar method was developed in [10] and applied successfully to high-dimensional problems [11]. Our method can be interpreted as a trans-dimensional extension of [10]. We sample from the resulting artificial posterior distribution using a single trans-dimensional MCMC algorithm, which only involves a simple modification of the MCMC algorithm developed to implement EM.

Although, the Bayesian policy search approach can benefit from gradient information, it does not require gradients. Moreover, since the target is proportional to the expected reward, the simulation is guided to areas of high reward automatically. This property results in an immediate reduction in variance in policy search.

2 Markov Decision Processes

2.1 Model

Let $n = 1, 2, \ldots$ be a discrete-time index. Consider the following model defined by $X_1 \sim \mu$,

$$\begin{aligned}
&X_n | (X_{n-1} = x, A_{n-1} = a) \sim f_a(\cdot | x), \\
&R_n | (X_n = x, A_n = a) \sim g_a(\cdot | x), \\
&A_n | (X_n = x, \theta) \sim \pi_\theta(\cdot | x),
\end{aligned}$$

where $\{X_n\}$ is the $\mathcal{X}-$valued state process, $\{A_n\}$ is the $\mathcal{A}-$valued action process, $\{R_n\}$ is a positive real-valued reward process, f_a the transition density, g_a the reward density and π_θ the randomized policy. If we have a deterministic policy then $\pi_\theta(a|x) = \delta_{\varphi_\theta(x)}(a)$ and $f_a(\cdot|x)$ (resp. $g_a(\cdot|x)$) becomes $f_\theta(\cdot|x)$ (resp. $g_\theta(\cdot|x)$).

We are here interested in maximizing with respect to the parameters of the policy θ the expected future return

$$V_\mu^\pi(\theta) = \mathbb{E}\left[\sum_{n=1}^{\infty} \gamma^n R_n\right]$$

where $0 < \gamma < 1$ is a discount factor.

Let us introduce the trans-dimensional probability distribution on $\biguplus \{k\} \times \mathcal{X}^k \times \mathcal{A}^k \times \mathbb{R}^+$ given by

$$\begin{aligned}
&p_\theta(k, x_{1:k}, a_{1:k}, r_k) = (1-\gamma)^{-1} \gamma^k \mu(x_1) \times \\
&g_{a_k}(r_k | x_k) \prod_{n=2}^{k} f_{a_{n-1}}(x_n | x_{n-1}) \prod_{n=1}^{k} \pi_\theta(a_n | x_n)
\end{aligned} \tag{1}$$

then we can rewrite $V_\mu^\pi(\theta)$ as

$$V_\mu^\pi(\theta) = (1-\gamma)\, \mathbb{E}_{p_\theta}[R_K] \tag{2}$$

for a randomized policy. For a deterministic policy, the representation (2) also holds for the trans-dimensional probability distribution defined on $\biguplus \{k\} \times \mathcal{X}^k \times \mathbb{R}^+$ given by

$$\begin{aligned}
&p_\theta(k, x_{1:k}, r_k) = (1-\gamma)^{-1} \gamma^k \mu(x_1) \times \\
&g_\theta(r_k | x_k) \prod_{n=2}^{k} f_\theta(x_n | x_{n-1}).
\end{aligned} \tag{3}$$

2.2 Policy Optimization via EM

The representation (2) was used in [5] to compute the value function through MCMC for a fixed θ. In [17], this representation is exploited to maximize $V_\mu^\pi(\theta)$ using the EM algorithm which, applied to this problem, proceeds as follows at iteration i

$$\theta_i = \underset{\theta \in \Theta}{\arg\max}\, Q(\theta_{i-1}, \theta)$$

where

$$Q(\theta_{i-1}, \theta) = \mathbb{E}_{\widetilde{p}_{\theta_{i-1}}}\left[\log\left(R_K . p_\theta(K, X_{1:K}, A_{1:K}, R_K)\right)\right],$$

$$\widetilde{p}_\theta(k, x_{1:k}, a_{1:k-1}, r_k) = \frac{r_k p_\theta(k, x_{1:k}, a_{1:k}, r_k)}{\mathbb{E}_{p_\theta}[R_K]}.$$

We can rewrite $Q(\theta_{i-1}, \theta)$ as

$$\begin{aligned}
&Q(\theta_{i-1}, \theta) \equiv \mathbb{E}_{\widetilde{p}_{\theta_{i-1}}}\left[\textstyle\sum_{n=1}^{K} \log \pi_\theta(A_n | X_n)\right] \\
&= \textstyle\sum_{n=1}^{\infty} \int \log \pi_\theta(a_n | x_n) . \widetilde{p}_{\theta_{i-1}}(k \geq n, x_n, a_n)\, dx_n
\end{aligned}$$

where '$\equiv$' means equal up to an additive constant independent of θ.

For a deterministic policy, we introduce

$$\widetilde{p}_\theta(k, x_{1:k}, r_k) = \frac{r_k p_\theta(k, x_{1:k}, r_k)}{\mathbb{E}_{p_\theta}[R_K]}$$

and $Q(\theta_{i-1}, \theta) := \mathbb{E}_{\widetilde{p}_{\theta_{i-1}}}[\log(R_K . p_\theta(K, X_{1:K}, R_K))]$ is given by

$$\begin{aligned} Q(\theta_{i-1}, \theta) &\equiv \mathbb{E}_{\widetilde{p}_{\theta_{i-1}}}\Big[\log g_\theta(R_K | X_k) + \\ &\qquad + \textstyle\sum_{n=1}^{K} \log f_\theta(X_n | X_{n-1})\Big] \\ &= \textstyle\sum_{n=1}^{\infty} \big(\int \log g_\theta(r_n | x_n) . \widetilde{p}_{\theta_{i-1}}(n, x_n, r_n)\, dx_n dr_n \\ &+ \int \log f_\theta(x_n | x_{n-1}) . \widetilde{p}_{\theta_{i-1}}(k \geq n, x_{n-1:n})\, dx_{n-1:n}\big). \end{aligned} \tag{4}$$

where $f_\theta(x_1 | x_0) = \mu(x_1)$.

For simplicity, we restrict the presentation to deterministic policies and reward functions $g_\theta(r_n | x_n) = \delta_{r(x_n)}(r_n)$; the extension of our algorithms to the randomized case is straightforward. With this restriction in mind, we need to be able to compute distributions of the form $\widetilde{p}_\theta(k, x_{n-1:n})$ to implement EM where

$$\widetilde{p}_\theta(k, x_{n-1:n}) = \widetilde{p}_\theta(x_{n-1:n} | k)\, \widetilde{p}_\theta(k)$$

with

$$\widetilde{p}_\theta(x_{n-1:n} | k) \propto \int p_\theta(x_{1:k})\, r(x_k)\, dx_{1:n-2} dx_{n+1:k},$$
$$\widetilde{p}_\theta(k) \propto \int p_\theta(x_{1:k})\, r(x_k)\, dx_{1:k}.$$

3 EM for Continuous State-Space Models

In this section we present three Monte-Carlo EM algorithms for stochastic control and planning. The first two algorithms are based on a smoothing procedure outlined in Section 3.1. The third one, presented in Section 3.4, is relies on direct sampling from a trans-dimensional distribution in the E-step. As we shall see, this algorithm is not only cheaper when dealing with Monte Carlo state spaces, but also avoids the need for truncation in the time domain.

3.1 Forward-Backward Recursion

We restrict k to the finite interval $\{1, ..., k_{\max}\}$. To compute $\widetilde{p}_\theta(x_{n-1:n} | k)$ and $\widetilde{p}_\theta(k)$, we can use the following forward-backward algorithm. The derivation is standard if we think of $\widetilde{p}_\theta(x_{1:k} | k)$ as a posterior distribution arising from the prior distribution $p_\theta(x_{1:k})$ and the likelihood $r(x_k)$ whereas $\widetilde{p}_\theta(k)$ can be thought of as a marginal likelihood.

<u>Forward recursion</u>

- Set $\alpha_1(x_1) = \mu(x_1)$.
- For $k = 2, ..., k_{\max}$
  - $\alpha_k(x_k) = \int \alpha_{k-1}(x_{k-1})\, f_\theta(x_k | x_{k-1})\, dx_{k-1}$.

<u>Backward recursion</u>

- Set $\beta_{k_{\max}}(x_{k_{\max}}) = r(x_{k_{\max}})$.
- For $k = k_{\max} - 1, ..., 1$
  - $\beta_k(x_k) = \int \beta_{k+1}(x_{k+1})\, f_\theta(x_{k+1} | x_k)\, dx_k$.

We have $\widetilde{p}_\theta(k) = \int \alpha_k(x_k)\, r(x_k)\, dx_k$ and we can compute $\widetilde{p}_\theta(x_{n-1:n} | k)$ for any $n \leq k$ and $k \in \{1, ..., k_{\max}\}$ using

$$\begin{aligned} \widetilde{p}_\theta(x_{n-1:n} | k) \propto\ & \alpha_{n-1}(x_{n-1})\, f_\theta(x_n | x_{n-1}) \\ & \times \beta_{k_{\max}+n-k}(x_n). \end{aligned} \tag{5}$$

3.2 Forward-Backward for Linear Gaussian Models with Mixture of Gaussians Reward

There are very few cases for which it is possible to compute the forward and backward recursion in closed-form. While improving upon the work initiated in [18], we address in this section the important class of linear Gaussian models where $\mu(x_1) = \mathcal{N}(x_1; \mu_1, \Sigma_1)$, $f_\theta(x_n | x_{n-1}) = \mathcal{N}(x_n; F_\theta x_{n-1} + m_\theta, \Sigma_\theta)$ when the reward is modelled by a sum of P Gaussian distributions, *i.e.*

$$r(x) = \sum_{i=1}^{P} \pi_i \mathcal{N}(y_i; M_i x, L_i),$$

where $\pi_i > 0, y_i, M_i$ and L_i are fixed parameters. Note that the model for $r(x)$ is quite flexible and allows essentially to model any bounded reward function on $\mathcal{X}$. In this case, we show how it is possible to perform the forward-backward calculations in closed-form. The backward recursion relies on the following parametrization of the backward message

$$\beta_k(x_k) = \sum_{i=1}^{P} \pi_i \widetilde{p}^i_\theta(y_i | k, x_n) \tag{6}$$

where $\widetilde{p}^i_\theta(y_i | k, x_k) = \mathcal{N}(y_i; M_i x_k, L_i)$,

$$\widetilde{p}^i_\theta(y_i | k, x_n) = \int \widetilde{p}^i_\theta(y_i | k, x_k) \prod_{j=n}^{k} f_\theta(x_n | x_{n-1})\, dx_{n+1:k},$$

which is parametrized as follows

$$-2 \log \widetilde{p}^i_\theta(y_i | k, x_n) \equiv c^i_{k,n} + x_n^{\mathrm{T}} \Omega^i_{k,n} x_n - 2 x_n^{\mathrm{T}} \mu^i_{k,n}. \tag{7}$$

Note that $\widetilde{p}^i_\theta(y_i | k, x_n)$ is *not* a Gaussian distribution for the argument x_n as it can satisfy $\int \widetilde{p}^i_\theta(y_i | k, x_n)\, dx_n = \infty$. The forward recursion is straightforward whereas the backward recursion relies on tedious calculations, which are omitted here in the interest of space. The algorithm proceeds as follows.

Forward recursion

- For $k = 2, ..., k_{\max}$
  - $m_k = F_\theta m_{k-1} + m_\theta$.
  - $\Sigma_k = F_\theta \Sigma_{k-1} F_\theta^{\mathrm{T}} + \Sigma_\theta$.

Backward recursion

- For $i = 1, ..., P$
  - Set $c^i_{k_{\max}} = \log(|L_i|) + y_i^{\mathrm{T}} L_i^{-1} y_i$, $\Omega^i_{k_{\max}} = M_i^{\mathrm{T}} L_i^{-1} M_i$, $\mu^i_{k_{\max}} = M_i L_i^{-1} y_i$.
  - For $k = k_{\max} - 1, ..., 1$
    - $\widetilde{\Sigma}_k^{-1} = \Omega^i_{k+1} + \Sigma_\theta^{-1}$.
    - $c^i_k = c^i_{k+1} + \log(|\Sigma_\theta|) + \log\left(\left|\widetilde{\Sigma}_k^{-1}\right|\right) - \left(\mu^i_{k+1} + \Sigma_\theta^{-1} m_\theta\right)^{\mathrm{T}} \Sigma_k^{-1} \left(\mu^i_{k+1} + \Sigma_\theta^{-1} m_\theta\right) + m_\theta^{\mathrm{T}} \Sigma_\theta^{-1} m_\theta$.
    - $\mu^i_k = F_\theta^{\mathrm{T}} \Sigma_\theta^{-1} \left(\widetilde{\Sigma}_k \mu^i_{k+1} + m_\theta\right)$.
    - $\Omega^i_k = F_\theta^{\mathrm{T}} \left(\Sigma_\theta^{-1} - \Sigma_\theta^{-1} \widetilde{\Sigma}_k \Sigma_\theta^{-1} \right) F_\theta$.

Based on $\alpha_k(x_k) = \mathcal{N}(x_k; m_k, \Sigma_k)$ and $\beta_k(x_k)$ given by (6), it is straightforward to compute the joint distributions $\widetilde{p}_\theta(x_{n-1:n}|k)$ using (5) for any $n \leq k$ and $k \in \{1, ..., k_{\max}\}$. These distributions are a mixture of P Gaussians. It is also straightforward to compute $\widetilde{p}_\theta(k)$. Note that this procedure can be extended to the case where we have a partially observed linear Gaussian model. In this case, the forward recursion is a Kalman filter and the backward recursion is computed using P generalized backward information filters; the generalization is coming from the fact that we need here to compute not only $\Omega^i_{k,n}$ and $\mu^i_{k,n}$ but also $c^i_{k,n}$.

3.3 Forward-Backward Recursion using SMC

For a general non-linear non-Gaussian model, it is impossible to compute in closed-form the forward and backward messages $\{\alpha_k(x_k), \beta_k(x_k)\}_{k=1}^{k_{\max}}$. A flexible approximation consists of relying on Monte Carlo methods. The first approximation consists of using the following algorithm which approximates the sequences of distributions $\{p_\theta(x_{1:k})\}$ and $\{\widetilde{p}_\theta(x_{1:k}|k)\}$ for $k = 1, ..., k_{\max}$.

- Sample $X_1^{(i)} \sim \mu_1$ for $i = 1, ..., N$, $\widehat{p}_\theta(x_1) = \frac{1}{N}\sum_{i=1}^N \delta_{X_1^{(i)}}(x_1)$, $W_1^{(i)} \propto r\left(x_1^{(i)}\right)$ and $\widehat{\widetilde{p}}_\theta(x_1|1) = \sum_{i=1}^N W_1^{(i)} \delta_{x_1^{(i)}}(x_1)$.
- For $k = 2, ..., k_{\max}$
  - Sample $X_k^{(i)}\Big| x_{k-1}^{(i)} \sim f_\theta\left(\cdot| x_{k-1}^{(i)}\right)$, $\widehat{p}_\theta(x_{1:k}) = \frac{1}{N}\sum_{i=1}^N \delta_{x_{1:k}^{(i)}}(x_{1:k})$, $W_k^{(i)} \propto r\left(x_k^{(i)}\right)$, $\widehat{\widetilde{p}}_\theta(x_{1:k}|k) = \sum_{i=1}^N W_k^{(i)} \delta_{x_{1:k}^{(i)}}(x_{1:k})$.

We also obtained an estimate of $\widetilde{p}_\theta(k)$ by computing the sample average of $r\left(x_k^{(i)}\right)$. Although this algorithm is straightforward to implement, it can only work well if $k_{\max}$ is moderate, e.g. if the discount factor is close to zero. Indeed the discrepancy between the distributions $p_\theta(x_{1:k})$ and $\widetilde{p}_\theta(x_{1:k}|k)$ can be quite large if the reward $r(x_k)$ is concentrated in regions of low probability masses of $p_\theta(x_k)$ and in this case the importance sampling estimate of $\widetilde{p}_\theta(x_{1:k}|k)$ is poor as the unnormalized importance weights have a large variance.

We propose here an SMC implementation of the forward backward recursion to limit this problem. The forward recursion is essentially similar to the one described above except that we only limit ourselves to the approximation of the marginals $p_\theta(x_k) = \alpha_k(x_k)$. To approximate the backward messages $\{\beta_k(x_k)\}$, we underline that it is not possible to use standard SMC algorithms (or even standard methods such as the Extended or the Unscented Kalman filter) as $\beta_k(x_k)$ is *not* a probability distribution in argument x_k. Moreover, although it can seem intuitive to define a reverse dynamics to compute such messages using the inverse of the dynamic equation (i.e. if $X_n = \varphi(X_{n-1}, V_n)$ then use $X_{n-1} = \varphi^{-1}(X_n, V_n)$), this common choice in the literature is flawed and leads to erroneous results; see [8] for a discussion of these problems. To implement a valid backward recursion, we can however introduce a sequence of artificial prior distributions $\{\eta_n(x_n)\}$ and approximate the distributions $\varrho_k(x_n) \propto \eta_n(x_n)\beta_k(x_n)$ using the following algorithm; see [8] for details.

- For $k = 1, ..., k_{\max}$
  - Sample $X_{k,k}^{(i)} \sim \eta_k$ for $i = 1, ..., N$ and set $\widehat{\eta}_k(x_k) = \frac{1}{N}\sum_{i=1}^N \delta_{x_{k,k}^{(i)}}(x_k)$.
  - Resample N times from $\sum_{i=1}^N W_{k,k}^{(i)} \delta_{x_{k,k}^{(i)}}(x_k)$ where $W_{k,k}^{(i)} \propto r\left(x_{k,k}^{(i)}\right)$ to obtain N samples also denoted $\left\{x_{k,k}^{(i)}\right\}$
  - For $n = k-1, ..., 2$
    - Sample $X_{n,k}^{(i)}\Big| x_{n+1,k}^{(i)} \sim q_\theta\left(\cdot| x_{n+1,k}^{(i)}\right)$.
    - Resample N times from $\sum_{i=1}^N W_{n,k}^{(i)} \delta_{x_{n,k}^{(i)}}(x_n)$

where $W_{n,k}^{(i)} \propto \frac{\eta_n\left(x_{n,k}^{(i)}\right) f_\theta\left(x_{n+1,k}^{(i)} \middle| x_{n,k}^{(i)}\right)}{\eta_{n+1}\left(x_{n+1,k}^{(i)}\right) q_\theta\left(x_{n,k}^{(i)} \middle| x_{n+1,k}^{(i)}\right)}$ to obtain N samples also denoted $\left\{x_{n,k}^{(i)}\right\}$.

The empirical measure of the samples $\left\{x_{n,k}^{(i)}\right\}$ is an approximation of $\varrho_k(x_n)$. Using the following identity which is a generalization of (5)

$$\widetilde{p}_\theta(x_{n-1:n}|\, k) \propto \frac{\alpha_{n-1}(x_{n-1}) f_\theta(x_n|\, x_{n-1}) \varrho_k(x_n)}{\eta_n(x_n)},$$

it is possible to combine the SMC approximations of $\varrho_k(x_n)$ and $\eta_n(x_n)$ to compute $\widetilde{p}_\theta(x_{n-1:n}|\, k)$.

This method performs significantly better than the naive method discussed at the beginning of this subsection but is computationally very intensive as it requires computing $k_{\max}$ backward recursions instead of one. That is, it is equivalent to running the backward recursion for $k_{\max}(k_{\max}+1)/2$ time steps instead of $k_{\max}$ time steps. It would be possible to use $\eta_n = \eta$ to bypass this problem but this choice is typically extremely inefficient as we need to select $\eta_n(x_n)$ as close as possible to the optimal choice $\alpha_n(x_n)$. Moreover, this approach is computationally inefficient in the sense that it does not focus the computational efforts at the time instants k which have an high probability mass $\widetilde{p}_\theta(k)$. Assume for example that most of the mass of this distribution is located in the interval $\{45, 55\}$ and $k_{\max} = 100$. Then the previous algorithm spends approximately 90% of the computational resources computing terms which have a very small contribution to $Q(\theta_{i-1}, \theta)$. This problem suggests the need for developing an alternative approach.

3.4 Trans-Dimensional Markov chain Monte Carlo

A simpler Monte Carlo approach to approximate numerically $Q(\theta_{i-1}, \theta)$ given by (4) would consist of obtaining directly samples from the trans-dimensional distribution $\widetilde{p}_\theta(k, x_{1:k})$. The main advantage of this approach compared to the forward backward is that we would obtain samples which concentrate themselves automatically in regions where $\widetilde{p}_\theta(k)$ has high probability masses. Moreover, contrary to previous approaches, it is no longer necessary to truncate the time domain. It is impossible to sample directly from this distribution but we can use the class of trans-dimensional MCMC algorithms pioneered by the Reversible Jump MCMC (RJMCMC) method of [6, 7] to achieve this.

We present here a simple RJMCMC methods composed of two reversible moves (birth and death) and several update moves. Assume the current state of the Markov chain targeting $\widetilde{p}_\theta(k, x_{1:k})$ is $(k, x_{1:k})$. With probability b_k, we propose a birth move; that is we sample a location uniformly in the interval $\{1, ..., k+1\}$, i.e. $J \sim \mathcal{U}\{1, ..., k+1\}$, and propose the candidate $(k+1, x_{1:j-1}, x^*, x_{j+1:k})$ where $X^* \sim q_\theta(\cdot|\, x_{j-1:j+1})$. This candidate is accepted with probability $A_{birth} = \min\{1, \alpha_{birth}\}$ where we have for $j \in \{2, ..., k-1\}$

$$\begin{aligned}\alpha_{birth} &= \frac{\widetilde{p}_\theta(k+1, x_{1:j-1}, x^*, x_{j+1:k})\, d_{k+1}}{\widetilde{p}_\theta(k, x_{1:k})\, b_k q_\theta(x^*|\, x_{j-1:j+1})} \\ &= \frac{\gamma f_\theta(x^*|\, x_{j-1}) f_\theta(x_{j+1}|\, x^*)\, d_{k+1}}{f_\theta(x_j|\, x_{j-1})\, b_k q_\theta(x^*|\, x_{j-1:j+1})},\end{aligned}$$

for $j = 1$

$$\alpha_{birth} = \frac{\gamma\mu(x^*) f_\theta(x_1|\, x^*)\, d_{k+1}}{\mu(x_1)\, b_k q_\theta(x^*|\, x_{1:2})}$$

and $j = k+1$

$$\alpha_{birth} = \frac{\gamma r(x^*) f_\theta(x^*|\, x_k)\, d_{k+1}}{r(x_k)\, b_k q_\theta(x^*|\, x_{k-1:k})}.$$

With probability d_k, we propose a death move; that is $J \sim \mathcal{U}\{1, ..., k\}$ and we propose the candidate $(k-1, x_{1:j-1}, x_{j+1:k})$ which is accepted with probability $A_{death} = \min\{1, \alpha_{death}\}$ where for $j \in \{2, ..., k-1\}$

$$\begin{aligned}\alpha_{death} &= \frac{\widetilde{p}_\theta(k-1, x_{1:j-1}, x_{j+1:k})\, b_{k+1} q_\theta(x_j|\, x_{j-1:j+1})}{\widetilde{p}_\theta(k, x_{1:k})\, d_k} \\ &= \frac{f_\theta(x_{j+1}|\, x_{j-1})\, b_{k+1} q_\theta(x_j|\, x_{j-1:j+1})}{\gamma f_\theta(x_{j+1}|\, x_j) f_\theta(x_j|\, x_{j-1})\, d_k},\end{aligned}$$

for $j = 1$

$$\alpha_{birth} = \frac{\mu(x_2)\, q_\theta(x_1|\, x_{1:2})\, b_{k+1}}{\gamma\mu(x_1) f_\theta(x_2|\, x_1)\, d_k}$$

and for $j = k$

$$\alpha_{birth} = \frac{r(x_{k-1})\, q_\theta(x_k|\, x_{k-2:k-1})\, b_{k+1}}{\gamma r(x_k) f_\theta(x_k|\, x_{k-1})\, d_k}.$$

Finally with probability $u_k = 1 - b_k - d_k$, we propose a standard (fixed dimensional) move where we update all or a subset of the components $x_{1:k}$ using say Metropolis-Hastings or Gibbs moves. There are many design possibilities for these moves. In general, one should block some of the variables so as to improve the mixing time of the Markov chain. If one adopts a simple one-at-a time Metropolis-Hastings scheme with proposals $q_\theta(x^*|\, x_{j-1:j+1})$ to update the j-th term, then the candidate is accepted with probability $A_{upd} = \min\{1, \alpha_{upd}\}$ where for $j \in \{2, ..., k-1\}$

$$\begin{aligned}\alpha_{upd} &= \frac{\widetilde{p}_\theta(k, x_{1:j-1}, x^*, x_{j+1:k})\, q_\theta(x_j|\, x_{j-1}, x^*, x_{j+1})}{\widetilde{p}_\theta(k, x_{1:k})\, q_\theta(x^*|\, x_{j-1:j+1})} \\ &= \frac{f_\theta(x^*|\, x_{j-1}) f_\theta(x_{j+1}|\, x^*)\, q_\theta(x_j|\, x_{j-1}, x^*, x_{j+1})}{f_\theta(x_j|\, x_{j-1}) f_\theta(x_{j+1}|\, x_j)\, q_\theta(x^*|\, x_{j-1:j+1})},\end{aligned}$$

for $j = 1$

$$\alpha_{upd} = \frac{\mu(x^*) f_\theta(x_2 | x^*) q_\theta(x_1 | x^*, x_2)}{\mu(x_1) f_\theta(x_2 | x_1) q_\theta(x^* | x_{1:2})}$$

and for $j = k$

$$\alpha_{upd} = \frac{r(x^*) f_\theta(x^* | x_{k-1}) q_\theta(x_k | x^*, x_{k-1})}{r(x_k) f_\theta(x_k | x_{k-1}) q_\theta(x^* | x_{k-1:k})}.$$

Under weak assumptions on the model, the Markov chain $\left\{K^{(i)}, X_{1:K}^{(i)}\right\}$ generated by this transition kernel will be irreducible and aperiodic and hence will generate asymptotically samples from the target distribution $\widetilde{p}_\theta(k, x_{1:k})$.

We emphasize that the structure of the distributions $\widetilde{p}_\theta(x_{1:k} | k)$ will not in many applications vary significantly with k and we will often have $\widetilde{p}_\theta(x_{1:k} | k) \approx \widetilde{p}_\theta(x_{1:k} | k+1)$. Hence the probability of having the reversible moves accepted will be reasonable. Standard Bayesian applications of RJMCMC usually do not enjoy this property and it makes it more difficult to design fast mixing algorithms. In this respect, our problem is easier.

4 Bayesian Policy Exploration

The previous algorithm provides a point estimate of θ. As the algorithm is based on EM and its Monte Carlo variants, we are not guaranteed to find the global optimum of the expected return. The algorithm like any other EM algorithm is sensitive to initialization and might get trapped in a severe local maximum.

We propose an alternative full Bayesian approach. In the simpler context of experimental design, this approach was successfully developed in [10], [11]. The idea consists of introducing a vague prior distribution $p(\theta)$ on the parameters of the policy θ. We then define the new artificial probability distribution defined on $\Theta \times \biguplus \{k\} \times \mathcal{X}^k \times \mathbb{R}^+\mathcal{X}$ by

$$\overline{p}(\theta, k, x_{1:k}) \propto r(x_k) p_\theta(k, x_{1:k}) p(\theta).$$

By construction, this target distribution admits the following marginal in θ

$$\overline{p}(\theta) \propto V_\mu^\pi(\theta) p(\theta)$$

and we can select an improper prior distribution $p(\theta) \propto 1$ if $\int_\Theta V_\mu^\pi(\theta) d\theta < \infty$.

If we could sample from $\overline{p}(\theta)$, then the generated samples $\left\{\theta^{(i)}\right\}$ would concentrate themselves in regions where $V_\mu^\pi(\theta)$ is large. We cannot sample from $\overline{p}(\theta)$ directly but we can developed a trans-dimensional MCMC algorithm which will generate asymptotically samples from $\overline{p}(\theta, k, x_{1:k})$, hence samples from $\overline{p}(\theta, k, x_{1:k})$.

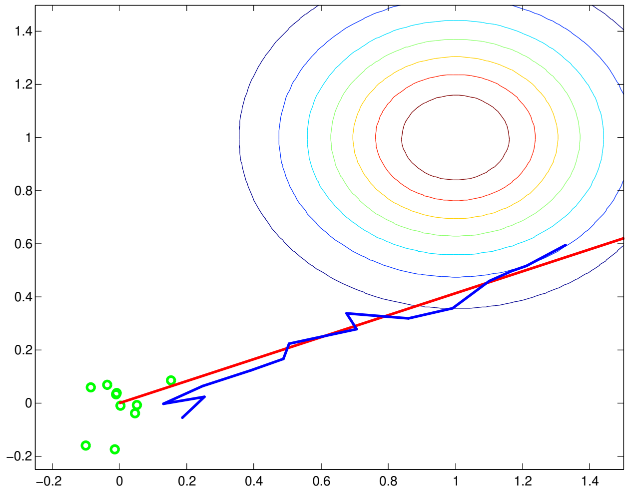

Figure 1: **Example state-space.** This figure shows an illustration of the 2d state-space described in section 5. Ten sample points are shown distributed according to μ, the initial distribution, and the contour plot corresponds to the reward function r. The red line denotes the policy parameterized by some angle θ, while a path is drawn in blue sampled from this policy.

To sample from $\overline{p}(\theta, k, x_{1:k})$, we can propose an algorithm which is very similar to the one described earlier to implement the E-step of EM. Assume the current state of the Markov chain targeting $\overline{p}(\theta, k, x_{1:k})$ is $(\theta, k, x_{1:k})$. We propose first to update the components $(k, x_{1:k})$ conditional upon θ using the combination of birth, death and update moves designed earlier. Then we propose to update θ conditional upon the current value of $(k, x_{1:k})$. This can be achieved using a simple Metropolis-Hastings algorithm or a more sophisticated dynamic Monte Carlo scheme.

5 Experiments

It should be noted from the outset that the results presented in this paper are preliminary, and serve mainly as an illustration of the Monte Carlo algorithms presented earlier. With that note aside, even these simple examples will give us some intuition about the algorithms' performance and behaviour. We are also very optimistic as to the possible applications of the analytic expressions for linear Gaussian models, but space has not allowed us to present simulations for this class of models here.

We will consider state- and action-spaces $\mathcal{X} = \mathcal{A} = \mathbb{R}^2$ such that each state $x \in \mathcal{X}$ is a 2d position and each action $a \in \mathcal{A}$ is a vector corresponding to a change in position. A new state at time n is given by

$X_n = X_{n-1} + A_{n-1} + \nu_{n-1}$ where ν_{n-1} denotes zero-mean Gaussian noise. Finally we will let μ be a normal distribution about the origin, and consider a reward (as in [17]) given by an unnormalized Gaussian about some point m, i.e. $r(x) = \exp(-\frac{1}{2}(x-m)^T \Sigma^{-1}(x-m))$. An illustration of this space can be seen in Figure 1 where $m = (1, 1)$.

For these experiments we chose a simple, stochastic policy parameterized by $\theta \in [0, \pi/2]$. Under this policy, an action A_n is normally distributed about $nw(\cos\theta, \sin\theta) - x_n$ for some (small) constant step-length w. Intuitively, this ensures that an agent following this policy will advance on a path along the angle θ. While unrealistic from a real-world perspective, this allows us to easily evaluate the optimal policy. For a state-space with initial distribution and reward function defined as in Figure 1 the optimal policy corresponds to $\theta = \pi/4$.

The plots in Figure 2 show the three previously described algorithms performing on this synthetic example—here the inferred value of θ is shown against cpu time, averaged over 5 runs. The first thing of note is the terrible performance of the SMC-based algorithm. This comes as no surprise considering the $\mathrm{O}(N^2 k_{\max}^2)$ time complexity necessary for computing the importance weights. While there do exist methods [8] for reducing this complexity to $\mathrm{O}(N \log N k_{\max}^2)$, the discrepancy between this and the reversible jump MCMC method suggests that the MCMC approach may be more adapted to this class of problems. In the finite/discrete case it is also possible, as shown by Toussaint *et al* (2006), to reduce the $k_{\max}^2$ term to $k_{\max}$ by calculating updates only using messages from the backwards recursion. The SMC method might further be improved by better choices for the artificial distribution $\eta_n(x_n)$—in this problem we used a vague Gaussian centered on the relevant state-space. It is however possible that any added benefit from a more informative η distribution is counterbalanced by the time required to calculate this η, for example by simulating particles forward in order to find the invariant distribution, etc.

The reversible jump Monte Carlo EM algorithm and the fully Bayesian approach performed comparably on this synthetic example. In this case the Bayesian approach performed slightly better, in general showing less in-run variance, as well as less variance between runs. The EM algorithm was also more sensitive, and we were forced to increase the number of samples N used by the E-step as the algorithm progressed, as well as controlling the learning rate with a smoothing parameter. For higher dimensional and/or larger models it is not inconceivable that this could have an adverse affect on the algorithms performance.

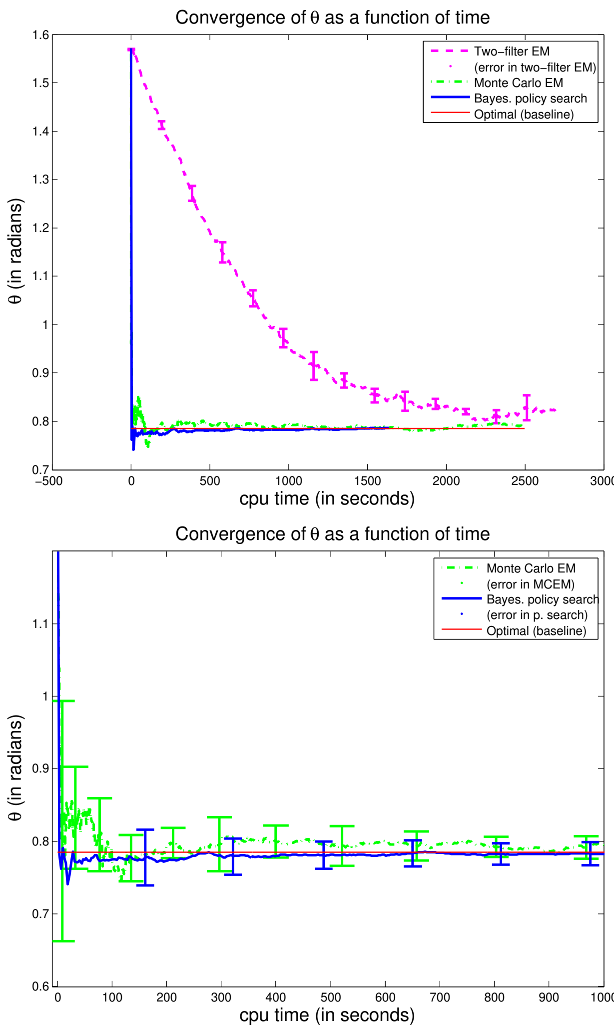

Figure 2: The top figure shows estimates for the policy parameter θ as a function of the cpu time used to calculate that value. This data is shown for the three discussed Monte Carlo algorithms as applied to a synthetic example and has been averaged over five runs; error bars are shown for the SMC-based EM algorithm. The bottom figure shows a "zoomed" version of this plot in order to see the reversible-jump EM algorithm and the fully Bayesian algorithm in more detail. In both plots the red line denotes the known optimal policy parameter of $\pi/4$.

6 Discussion

We believe that formulating stochastic control as a trans-dimensional inference problem is fruitful. It has enabled us to derive an algorithm that relaxes the quadratic cost constraint in LQR. This algorithm remains to be tested on high dimensional problems but preliminary results (not presented here) were very encouraging. This interpretation also led to the development of the first, to the best of our knowledge, trans-dimensional MCMC algorithm for policy search in general non-linear non-Gaussian control problems. Our results, on an illustrative example, showed that this trans-dimensional simulator is more effective that the simulators based on a particle smoothing implementation of the forward-backward. In the near future, we plan to apply our algorithms to several control and planning tasks of interest.

References

[1] H Attias, "Planning by probabilistic inference", *Proceedings of the 9th International Conference on Artificial Intelligence and Statistics*, 2003.

[2] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, vol. 15, pp. 319–350, 2001.

[3] D. P. Bertsekas, "Dynamic programming and optimal control "Athena Scientific, 1995.

[4] P. Dayan and G.E. Hinton, "Using EM for reinforcement learning", *Neural Computation*, vol. 9, pp. 271-278, 1997.

[5] A. Doucet and V.B. Tadić, "On solving integral equations using Markov chain Monte Carlo methods", TR-CUED-F-INFENG 444, Cambridge University, 2002.

[6] Green, P. J. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, *Biometrika* **82**: 711–732, 1995.

[7] P.J. Green, "Trans-dimensional Markov chain Monte Carlo", in *Highly Structured Stochastic Systems*, Oxford University Press, 2003.

[8] M. Klaas, M. Briers, N. de Freitas, A. Doucet, S. Maskell and D. Lang, "Fast particle smoothing: If I had a million particles", *ICML*, 2006.

[9] G. Lawrence, N. Cowan, and S. Russell, "Efficient gradient estimation for motor control learning," in *Uncertainty in Artificial Intelligence*, pp. 354-36, 2003.

[10] P. Müller, "Simulation based optimal design," in *Bayesian Statistics* 6, J.O. Berger, J.M. Bernardo, A.P. Dawid and A.F.M. Smith (eds.), pp. 459–474, Oxford University Press, 1999.

[11] P. Müller, B. Sansó and M. De Iorio, "Optimal Bayesian design by inhomogeneous Markov chain simulation", *J. American Stat. Assoc.*, vol. 99, pp. 788-798, 2004.

[12] A. Y. Ng and M. I. Jordan, "Pegasus: A policy search method for large MDPs and POMDPs," *UAI*, 2000.

[13] A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Inverted autonomous helicopter flight via reinforcement learning," *International Symposium on Experimental Robotics*, 2004.

[14] J. Peters and S. Schaal, "Policy gradient methods for robotics," *IEEE International Conference on Intelligent Robotics Systems*, 2006.

[15] J. M. Porta, N. Vlassis, M. T. J. Spaan and P. Poupart, "Point-based value iteration for continuous POMDPs", *Journal of Machine Learning Research*, vol. 7, pp. 2329-2367, 2006.

[16] S. Thrun, "Monte Carlo POMDPs", *NIPS* 12, pp. 1064-1070, 2000.

[17] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state Markov Decision Processes", *ICML*, 2006.

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

[19] D. Verma and R.P.N. Rao, "Planning and acting in uncertain environments using probabilistic inference", *Proceedings of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.