Analyzing and Escaping Local Optima in Planning as Inference for Partially Observable Domains

Pascal Poupart[1], Tobias Lang[2], and Marc Toussaint[2]

[1] David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada
ppoupart@cs.uwaterloo.ca
[2] Machine Learning and Robotics Lab
FU Berlin, Berlin, Germany
{tobias.lang,marc.toussaint}@fu-berlin.de

Abstract. Planning as inference recently emerged as a versatile approach to decision-theoretic planning and reinforcement learning for single and multi-agent systems in fully and partially observable domains with discrete and continuous variables. Since planning as inference essentially tackles a non-convex optimization problem when the states are partially observable, there is a need to develop techniques that can robustly escape local optima. We investigate the local optima of finite state controllers in single agent partially observable Markov decision processes (POMDPs) that are optimized by expectation maximization (EM). We show that EM converges to controllers that are optimal with respect to a one-step lookahead. To escape local optima, we propose two algorithms: the first one adds nodes to the controller to ensure optimality with respect to a multi-step lookahead, while the second one splits nodes in a greedy fashion to improve reward likelihood. The approaches are demonstrated empirically on benchmark problems.

1 Introduction

Toussaint et al. [20] recently showed that policy optimization in probabilistic domains is equivalent to maximizing the likelihood of normalized rewards. This connection between planning and inference has opened the door to the application of a wide range of machine learning and probabilistic inference techniques. However, policy optimization in partially observable domains is generally non-convex, including when reformulated as a likelihood maximization problem. As a result, policies often get stuck in local optima, which may be far from optimal.

In this paper, we analyze the local optima of finite state controllers in partially observable Markov decision processes (POMDPs) that are optimized by Expectation Maximization (EM). More precisely, we show that EM optimizes controllers by essentially performing a one-step lookahead where each parameter is adjusted in isolation (i.e., while keeping the other parameters fixed). We propose two techniques to help EM escape local optima. The first technique extends EM's one-step

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part II, LNAI 6912, pp. 613–628, 2011.

forward search to multiple steps. New nodes are added to the controller when the forward search detects a suboptimal action choice. The second approach splits controller nodes in two new nodes that are optimized by EM.

The paper is organized as follows. Sec. 2 reviews POMDPs, finite state controllers and planning as inference. Sec. 3 analyzes the properties of EM's local optima. Sec. 4 describes the two techniques to escape local optima. Sec. 5 evaluates the escape techniques on benchmark problems. Finally, Sec. 6 concludes the paper.

2 Background

2.1 Partially Observable Markov Decision Processes

Consider a partially observable Markov decision process (POMDP) described by a set $\mathcal{S}$ of states s, a set $\mathcal{A}$ of actions a, a set $\mathcal{O}$ of observations o, a stochastic transition function $\Pr(s'|s,a) = p_{s'|sa}$, a stochastic observation function $\Pr(o'|s,a) = p_{o'|sa}$ and a reward function $R(s,a) = r_{sa} \in \Re$. An important class of policies (denoted by π) are those representable by a stochastic finite state controller (FSC), which is a directed acyclic graph such that each node n chooses an action a stochastically according to an action distribution $\pi(a|n) = \pi_{a|n}$, each edge is labeled with an observation o' that chooses a successor node n' stochastically according to a successor node distribution $\Pr(n'|n,o') = \pi_{n'|no'}$ and the initial node is chosen stochastically according to $\Pr(n) = \pi_n$. The value $V(n,s) = V_{ns}$ of a FSC is the discounted sum of the rewards earned while executing the policy it encodes. We can compute this value by solving a linear system:

$$V_{ns} = \sum_a \pi_{a|n} r_{sa} + \gamma \sum_{s'o'n'} p_{s'|sa} p_{o'|sa} \pi_{n'|no'} V_{n's'} \ \forall ns$$

Hence, for a given initial state distribution (a.k.a. belief) $\Pr(s) = p_s$, the value of the policy is $\sum_{sn} p_s \pi_n V_{ns}$. Initial algorithms to optimize FSCs were based on policy iteration [6], however the size of the controller tends to grow exponentially with the number of iterations. Alternatively, since there are often good policies that can be represented by small controllers, one can search for the best controller with a fixed number of nodes. This search can be formulated as a non-convex optimization problem [1][1]:

$$\max_{\pi_n, \pi_{a|n}, \pi_{n'|no'}, V_{ns}} \quad \sum_{sn} p_s \pi_n V_{ns}$$
$$\text{s.t. } V_{ns} = \sum_{as'o'n'} \pi_{a|n} [r_{sa} + \gamma p_{s'|sa} p_{o'|sa} \pi_{n'|no'} V_{n's'}] \ \forall ns$$

[1] The optimization problem described in [1] assumes a fixed initial node and merges $\pi_{a|n}$ and $\pi_{n'|no'}$ into a single distribution $\pi_{an'|no'}$, but these differences are irrelevant for this paper.

$$\sum_a \pi_{a|n} = 1 \; \forall n, \; \pi_{a|n} \geq 0 \; \forall an$$
$$\sum_{n'} \pi_{n'|no'} = 1 \; \forall no', \; \pi_{n'|no'} \geq 0 \; \forall n'no'$$

Algorithms to find an optimal policy include gradient ascent [11], sequential quadratic programming [1], bounded policy iteration [14] and stochastic local search [3]. However, the non-convex nature of the problem generally prevents any of these algorithms from finding an optimal controller with certainty.

2.2 Planning as Inference

In another line of research, Toussaint et al. [19] proposed to reformulate policy optimization as a likelihood maximization problem. The idea is to treat rewards as random variables by normalizing them. Let $\bar{R}$ be a binary variable such that

$$\Pr(\bar{R}{=}true|s,a) = p_{\bar{r}_{true}|sa} = (r_{sa} - \min_{sa} r_{sa})/(\max_{sa} r_{sa} - \min_{sa} r_{sa}))$$

Similarly, we treat the decision variables A and N as random variables with conditional distributions corresponding to $\pi_{a|n}$, π_n and $\pi_{n'|no'}$. This gives rise to a graphical model where the value of a policy can be recovered by estimating the probability that each reward variable is true. However, rewards are discounted and added together, so Toussaint et al. propose to use a mixture of dynamic Bayesian networks (DBNs) where each DBN is t time steps long with a single reward variable at the end and is weighted by a term proportional to γ^t. Hence, the value of a policy is proportional to $\Pr(\bar{R}{=}true)$ in this mixture of DBNs. To optimize the policy, it suffices to search for the distributions π_n, $\pi_{n'|no'}$ and $\pi_{a|n}$ that maximize $\Pr(\bar{R}{=}true)$. This is essentially a maximum a posteriori estimation problem that can be tackled with algorithms such as Expectation Maximization (EM) [5]. More specifically, EM alternates between computing the expectations

$$E(n|\bar{R}{=}true, \pi^i) = E_{n|\bar{r}_{true}\pi^i}$$
$$E(a,n|\bar{R}{=}true, \pi^i) = E_{an|\bar{r}_{true}\pi^i}$$
$$E(n',o',n|\bar{R}{=}true, \pi^i) = E_{n'o'n|\bar{r}_{true}\pi^i}$$

and updating the distributions

$$\pi_n^{i+1} = E_{n|\bar{r}_{true}\pi^i} / \sum_n E_{n|\bar{r}_{true}\pi^i}$$
$$\pi_{a|n}^{i+1} = E_{an|\bar{r}_{true}\pi^i} / \sum_a E_{an|\bar{r}_{true}\pi^i}$$
$$\pi_{n'|o'n}^{i+1} = E_{n'o'n|\bar{r}_{true}\pi^i} / \sum_{n'} E_{n'o'n|\bar{r}_{true}\pi^i}$$

The expectations are obtained as follows

$$E_{n|\bar{r}_{true}\pi^i} = \sum_s p_s \pi^i_n \beta_{ns} \tag{1}$$

$$E_{an|\bar{r}_{true}\pi^i} = \sum_{ss'o'n'} \alpha_{sn} \pi^i_{a|n} [p_{\bar{r}_{true}|sa} + \gamma p_{s'|sa} p_{o'|s'a} \pi^i_{n'|o'n} \beta_{n's'}] \tag{2}$$

$$E_{n'o'n|\bar{r}_{true}\pi^i} = \sum_{ss'a} \alpha_{sn} \pi^i_{a|n} p_{s'|sa} p_{o'|s'a} \pi^i_{n'|o'n} \beta_{n's'} \tag{3}$$

where $\alpha = \lim_{t\to\infty} \alpha^t$ and $\beta = \lim_{t\to\infty} \beta^t$ are the forward and backward terms obtained in the limit according to the following recursions[2]:

$$\alpha^0_{sn} = p_s \pi_n$$

$$\alpha^t_{s'n'} = b_{s'} \pi_{n'} + \gamma \sum_{asno'} \alpha^{t-1}_{sn} \pi_{a|n} p_{s'|sa} p_{o'|as'} \pi_{n'|no'} \ \forall t > 0 \tag{4}$$

$$\beta^0_{sn} = \sum_a \pi_{a|n} p_{\bar{r}_{true}|sa}$$

$$\beta^t_{sn} = \sum_{as'n'o'} \pi_{a|n} [p_{\bar{r}_{true}|sa} + \gamma p_{s'|sa} p_{o'|as'} \pi_{n'|no'} \beta^{t-1}_{s'n'}] \ \forall t > 0 \tag{5}$$

An implication of the reformulation of policy optimization as an inference problem is that it opens the door to a variety of inference techniques and allows continuous [7], hierarchical [18], reinforcement learning [21] and multi-agent [8] variants to be tackled with the same machinery. Nevertheless, an important problem remains: policy optimization is inherently non-convex and therefore the DBN mixture reformulation does not get rid of local optima issues.

2.3 State Splitting

Siddiqi et al. [15] recently proposed an approach to discover the number of hidden states in HMMs by state splitting. Since there is no restriction on the number of hidden states, this approach can be viewed as a technique to escape local optima. In Section 4.2, we adapt this approach to POMDP controllers where internal nodes are split to escape local optima.

State splitting in HMMs works as follows. Run EM to learn the parameters of an HMM and Viterbi to find the most likely state paths. Then, for each state s, investigate the possibility of splitting s in two new states s_1 and s_2. Let T_s be the set of time steps where s is the most likely state. Replace the parameters that involve s by new parameters that involve s_1 or s_2. Optimize the new parameters with respect to the time steps in T_s. In other words, clamp the states outside of T_s to their most likely value and re-run EM to learn the parameters that involve s_1 and s_2 while keeping all other parameters fixed. At each iteration, greedily select the split that improves the model the most, then run full EM to converge to a new local optimum.

[2] In practice, $\alpha \approx \alpha^t$ and $\beta \approx \beta^t$ for large enough t depending on the discount γ.

3 Local Optima Analysis

When EM gets trapped in a local optimum, a simple strategy to escape consists of adding new nodes to the controller. However, unless the new nodes are carefully initialized, they will not help EM escape. For instance, as discussed in the experiments, adding random nodes is ineffective. Hence, there is a need to understand the conditions under which EM gets trapped so that the new nodes can be initialized to break those conditions. In this section, we show that EM stops making progress when the parameters of the nodes are optimal with respect to a one-step look ahead from a special set of beliefs. Based on this insight, in the next section, we propose an escape technique that adds nodes to the controller according to a multi-step lookahead.

Let's have a closer look at the policy updates performed by EM. In Eq. 1, 2 and 3 the policy terms π^i_n, $\pi^i_{a|n}$ and $\pi^i_{n'|o'n}$ can be factored out of the sum. This means that EM performs a multiplicative update:

$$\pi^{i+1}_n \propto \pi^i_n f_n \text{ where } f_n = \sum_s p_s \beta_{sn} \tag{6}$$

$$\pi^{i+1}_{a|n} \propto \pi^i_{a|n} g_{an} \text{ where } g_{an} = \sum_{ss'o'n} \alpha_{sn}[p_{\bar{r}_{true}|sa} + \gamma p_{s'|sa} p_{o'|s'a} \pi^i_{n'|o'n} \beta_{n's'}] \tag{7}$$

$$\pi^{i+1}_{n'|o'n} \propto \pi^i_{n'|o'n} h_{n'o'n} \text{ where } h_{n'o'n} = \sum_{ss'a} \alpha_{sn} \pi^i_{a|n} p_{s'|s,a} p_{o'|s'a} \beta_{n's'} \tag{8}$$

The multiplicative nature of the updates tells us that EM converges to a policy that chooses an initial node with maximal f_n, actions with maximal g_{an}, and successor nodes n' with maximal $h_{n'o'n}$. This is formalized by the following theorem.

Theorem 1. *If a policy π is a stable fixed point of EM (i.e., the fixed point is an attractor within an ϵ-hyperball), then*

$$\forall n \text{ if } \pi_n \neq 0 \text{ then } f_n = \max_n f_n \tag{9}$$

$$\forall an \text{ if } \pi_{a|n} \neq 0 \text{ then } g_{an} = \max_a g_{an} \tag{10}$$

$$\forall n'o'n \text{ if } \pi_{n'|o'n} \neq 0 \text{ then } g_{n'o'n} = \max_{n'} g_{n'o'n} \tag{11}$$

Proof. Suppose that π is a stable fixed point of EM. From (7) it follows that $\pi_{a|n} = c_n \pi_{a|n} g_{an}$ with a normalization constant c_n. For all non-zero $\pi_{a|n} \neq 0$, all actions in n have the same g-value $g_{an} = 1/c_n$. It remains to show that $1/c_n$ is also the maximum g_{an}, which we prove by contradiction. Let a^* be an action with $\pi_{a^*|n} = 0$ at the fixed point, but $g_{a^*n} > 1/c_n$. Consider an infinitesimal perturbation of the policy such that $\pi_{a^*|n} = \epsilon$. Independent of how small ϵ is, the probability of a^* will increase in the next iteration and $\pi_{a^*|n}$ will diverge from 0. Therefore, this fixed point is instable. The same reasoning holds for π_n with f_n and $\pi_{n'|o'n}$ with $h_{n'o'n}$.

Note that in practice, ensuring that parameters never become identically zero (e.g., by adding the smallest amount of noise in this case) ensures that EM always converges to such a stable fixed point.

The above theorem also gives a theoretical justification for the "smoothed greedy" update heuristics often used in practice [18]. Since EM will converge to a policy where $\pi_{a|n} = 0$ for all a, n where $g_{an} < \max_a g_{an}$, then it is reasonable to update $\pi_{a|n}$ by moving it towards a greedy policy that assigns non-zero probability only to the a, n pairs that are maximal in g (and similarly for π_n and $\pi_{n'|no'}$). Note however that there is no guarantee that such greedy updates will converge (they may lead to cycling), but they often speed up convergence in practice.

Let's interpret the conditions under which EM converges (i.e., conditions formalized in Theorem 1) since this will help us derive an escape technique in the next section. First, we point out that the backward terms β_{sn} in Eq. 5 can be interpreted as the total expected discounted rewards that will be earned when executing π from s, n. In other words, it is the value function V_{sn} of π at s, n. Similarly, the forward terms α_{sn} in Eq. 4 can be interpreted as the discounted occupancy frequency of π. In other words, α_{sn} indicates the expected number of times (discounted by γ^t for t time steps) that state s is reached in node n when executing π. We also define $b_{s|n} = \alpha_{ns} / \sum_s \alpha_{ns}$ to be the belief (e.g., state distribution) with respect to node n proportional to α_{ns}. Since $\alpha_{n\cdot}$ is the discounted sum of all reachable beliefs in node n, $b_{s|n}$ can be interpreted as a weighted average of the reachable beliefs in node n. Based on this, we can show that f_n is the expected value of π when starting its execution in node n. Similarly, g_{an} is proportional to the Q-function for belief $b_{s|n}$ and $h_{n'o'n}$ is proportional to the expected value of each successor node for the belief reachable from $b_{s|n}$ when executing π and receiving observation o'. This becomes evident when we replace α_{sn} by $b_{s|n}$ and β_{sn} by V_{sn} in Eq. 6, 7 and 8.

$$f_n \propto \sum_s p_s V_{ns} \tag{12}$$

$$g_{an} \propto \sum_s b_{s|n} [p_{\bar{r}_{true}|sn} + \sum_{s'o'n'} p_{s'|sa} p_{o'|s'a} \pi_{n'|o'n} V_{n's'}] \tag{13}$$

$$h_{n'o'n} \propto \sum_{sao'} b_{s|n} \pi_{a|n} p_{s'|sa} p_{o'|s'a} V_{n's'} \tag{14}$$

In the limit, since EM chooses an initial node with maximal f value (as well as actions with maximal g value and successor nodes with maximal h value), EM implicitly maximizes the initial node value for a fixed policy (as well as the Q-function for fixed successor node distributions and the successor nodes value for fixed action distributions). In the following, let us discuss how EM's convergence conditions are related to Bellman's global optimality conditions for controllers. More specifically, a controller is (globally) optimal when it satisfies the following condition for all beliefs b reachable in each node n:

$$V_{nb} = \max_a r_{ba} + \gamma \sum_{o'} p_{o'|ba} max_{n'} V_{n'b^{ao'}} \tag{15}$$

where $V_{nb} = \sum_s b_s V_{ns}$, $p_{o'|ba} = \sum_{ss'} b_s p_{s'|sa} p_{o'|s'a}$, $r_{ba} = \sum_s b_s r_{sa}$ and $b^{ao'}_{s'} \propto \sum_s b_s p_{s'|sa} p_{o'|s'a}$. The above equation can be rewritten in three optimality conditions for the choice of successor nodes, actions and initial node:

$$n^{bao'} = \operatorname*{argmax}_{n'} \sum_{ss'} b_s p_{s'|sa} p_{o'|s'a} V_{n's'} \ \forall bao' \tag{16}$$

$$a^n = \operatorname*{argmax}_{a} \sum_{s} b_s [r_{sa} + \gamma \sum_{s'o'} p_{s'|sa} p_{o'|s'a} V_{n^{bao's'}}] \ \forall n \tag{17}$$

$$n^0 = \operatorname*{argmax}_{n} \sum_{s} p_s V_{ns} \tag{18}$$

Note the similarity between these equations and the definitions of f, g and h in Eq. 12, 13 and 14. One important difference is that n^0, a^n and $n^{bao'}$ must be optimal for all beliefs b reachable in each node n to ensure global optimality, where as EM only ensures that the initial node, action and successor node choices are optimal for the single belief $b_{s|n}$ associated with each node n. Another important difference is that n^0, a^n and $n^{bao'}$ must be optimized simultaneously to ensure global optimality, where as EM adjusts the initial node, action and successor node distributions separately while keeping the other distributions fixed.

Below, in Theorem 2, we show that the convergence conditions described by Eq. 9, 10 and 11) are necessary, but not sufficient to ensure global optimality. Note that EM is already known to converge to local optima and therefore there must exist conditions that are necessary, but not sufficient. So the point of the theorem is to show that the particular conditions described by Eq. 9, 10 and 11) are indeed the ones for controller optimization. We encourage the reader to pay special attention to the proof since it shows that optimizing the parameters of the controller by a one-step lookahead from the beliefs $b_{s|n}$ associated with each node is a sensible thing to do even when these beliefs are not reachable. In the next section, we use this idea to develop a multi-step forward search from each $b_{s|n}$ instead of the initial belief to reduce the complexity of the search.

Theorem 2. *The conditions described by Eq. 9, 10 and 11 are necessary, but not sufficient, to ensure global optimality of FSCs.*

Proof. If $b_{s|n}$ is a reachable belief for each node n, then it is clear that the conditions described by Eq. 9, 10 and 11 are a subset of the global optimality conditions and therefore they are necessary, but not sufficient. However, the beliefs $b_{s|n}$ associated with each node n are not always reachable (even though they are weighted averages of the reachable beliefs). Nevertheless, we can still show that the conditions are necessary for optimality by observing that the beliefs $b_{s|n}$ are convex combinations of the reachable beliefs and therefore have the same optimal action and successor nodes as the reachable beliefs. Suppose that a controller is globally optimal, then $\pi_{a|n}$ and $\pi_{n'|o'n}$ must be optimal for all reachable beliefs of node n. We also know that $\pi_{a|n}$ and $\pi_{n'|o'n}$ are optimal for the convex hull of the reachable beliefs in node n since the optimal value function

Algorithm 1. Forward Search

Function: forwardSearch
Inputs: α, β, π and $timeLimit$
Output: $new\mathcal{N}$ (set of new nodes)
Let $b_{s|n} \propto \alpha_{ns}$ for each n
$d \leftarrow 0$
while $time < timeLimit$ **do**
  $d \leftarrow d + 1$
  for $n \in \mathcal{N}$ **do**
    $[new\mathcal{N}, gain] \leftarrow$
     $recursiveSearch(b_{s|n}, \beta, \pi, d)$
    if $gain > 0$ **then**
      return;
    end if
  end for
end while

Function: recursiveSearch
Inputs: b, β, π and d (depth)
Output: $new\mathcal{N}$ (new nodes) and $bestGain$
$bestGain \leftarrow 0$
$new\mathcal{N} \leftarrow \emptyset$
if $d = 1$ **then**
  Current value: $v \leftarrow \max_n \sum_s b_s \beta_{sn}$
  Compute v^*, $n^{ao'}$, a^* (Eq. 15-17)
  if $v^* - v > 0$ **then**
    $bestGain \leftarrow v^* - v$
    $new\mathcal{N} \leftarrow \{n\}$
      where $\pi_{a^*|n} = 1$, $\pi_{n^{a^*o'}|o'n} = 1$
  end if
else
  for $a \in A, o' \in O$ **do**
    $[\mathcal{N}, gain] \leftarrow$
     $recursiveSearch(b^{ao'}, \beta, \pi, d-1)$
    if $gain > bestGain$ **then**
      $bestGain \leftarrow gain$
      $new\mathcal{N} \leftarrow \mathcal{N} \cup \{n\}$
        where $\pi_{a|n} = 1$, $\pi_{last(\mathcal{N})|no'} = 1$
      end if
    end for
end if

is piece-wise linear and convex [17], which implies that the value function of node n is optimal for a polytope of the belief space that includes all reachable beliefs of node n. We can verify that $b_{s|n}$ is a convex combination of the reachable beliefs at node n since it is the normalized version of α_{sn}, which is a discounted sum of reachable beliefs. Hence, whether the beliefs $b_{s|n}$ are reachable or not for each n, Eq. 9, 10 and 11 hold for (globally) optimal controllers.

4 Escaping Local Optima

We describe two algorithms to escape local optima. The first approach does a multi-step forward search to find nodes that can be added to the controller in such a way that EM can resume its progress. The second approach searches for a node to be split in two such that optimizing the parameters of the two new nodes by EM yields the largest improvement.

4.1 Forward Search

Since global optimality is ensured when optimal action and successor node distributions are used for all reachable beliefs, we could perform a forward search from the initial belief to add new nodes each time suboptimal actions or successor

nodes are chosen for some reachable beliefs. However, such a search grows exponentially with the planning horizon. In contrast, EM finds a controller where each action and successor node distribution is optimal according to a forward search of just one step starting from the belief $b_{s|n}$ associated with each n. We propose a technique that gradually extends EM's myopic search by searching increasingly deeper from each $b_{s|n}$. The optimality conditions become stronger with the depth of the search and sufficient in the limit. An infinitely deep search verifies that the controller is optimal for all reachable beliefs while cutting off the search at a finite depth d ensures that the controller is at most $(r_{max} - r_{min})\gamma^d/(1-\gamma)$ away from optimal.

Algorithm 1 describes an incremental forward search technique to escape local optima. The approach verifies whether the action and successor node distributions are optimal with respect to the value function of the controller for all beliefs reachable from some $b_{s|n}$ at increasing depth. When a non-optimal action or successor node choice is detected, a new node is created with optimal action and successor node distributions. We also create nodes for each belief traversed on the path since their action and successor node distributions may change too. These new nodes are added to the controller and the successor node distributions of the existing nodes are perturbed slightly to include an ϵ probability of reaching the new nodes. This is necessary to allow the existing nodes to link to the new nodes since zero probabilities are fixed points in EM.

Note that starting the multi-step search from each $b_{s|n}$ is fine even when $b_{s|n}$ is not a reachable belief as described in the proof of Theorem 2. The benefit of starting the search from each $b_{s|n}$ instead of the initial belief is that a shallower search is often sufficient to find a suboptimal action choice or successor node. Recall that each $b_{s|n}$ is a weighted combination of reachable beliefs that may be arbitrarily deep, hence starting the search from those weighted combinations may significantly reduce the time of the search.

4.2 Node Splitting

Alternatively, we can escape local optima by adapting the HMM state splitting approach to POMDP controllers as described in Algorithm 2. For each node of the controller, consider the possibility of splitting that node in two new nodes n_1 and n_2. To initialize the split, we replace the parameters that involve the node n_{split} being split by parameters that involve n_1 or n_2 in a way that does not change likelihood. More precisely, the parameters $\pi_{a|n_1}$, $\pi_{n'|o'n_1}$, $\pi_{a|n_2}$ and $\pi_{n'|o'n_2}$ are set equal to $\pi_{a|n_{split}}$ and $\pi_{n'|o'n_{split}}$ respectively. As for π_{n_1}, π_{n_2}, $\pi_{n'_1|o'n}$ and $\pi_{n'_2|o'n}$, they are initialized randomly while ensuring that $\pi_{n_1} + \pi_{n_2} = \pi_{n_{split}}$ and $\pi_{n'_1|o'n} + \pi_{n'_2|o'n} = \pi_{n'_{split}|o'n}$. After this neutral initialization of the split, we optimize the parameters by running EM again. To speed up computation, we initialize α and β with those of the unsplit controller. This is similar to keeping the most likely states clamped outside of T_s since α and β are the forward and backward terms that summarize the computation before and after a node split. The M-step adapts the new parameters based on the

Algorithm 2. Node Splitting

Inputs: α, β, π, $iters$
Output: α, β, π (for split controller)
for $n_{split} \in N$ **do**
  split n_{split} into n_1 and n_2
  initialize the splitted controller such that $\pi_{a|n_1} = \pi_{a|n_2} = \pi_{a|n_{split}}$, $\pi_{n'|o'n_1} = \pi_{n'|o'n_2} = \pi_{n'|o'n_{split}}$, $\pi_{n_1} + \pi_{n_2} = \pi_{n_{split}}$ and $\pi_{n'_1|o'n} + \pi_{n'_2|o'n} = \pi_{n'_{split}|o'n}$.
  also split α and β such that $\alpha_{n_1} + \alpha_{n_2} = \alpha_{n_{split}}$, $\beta_{n_1} = \beta_{n_2} = \beta_{n_{split}}$.
  for $i = 1$ to $iters$ **do**
    propagate α and β following Eq. 4 and 5
    perform M-step based on α and β
  end for
  $gain(n)$ = value after M-step
end for
$n^* = \text{argmax}_n \, gain(n)$
assign π, α, β to results of splitting n^*

expectations as usual. After retraining each potential split like this, we select the split which brought the largest increase in likelihood.

4.3 Computational Complexity

We report the computational complexity of EM, node splitting and forward search in Table 1. The complexity of EM is equal to the number of iterations I times the complexity of computing the forward and backward terms in Eq. 4-5, and the expectations in Eq. 1-3. The forward and backward terms are computed recursively by executing t_{max} steps (equal to the planning horizon), where each step can be performed efficiently by variable elimination with a complexity corresponding to the size of the largest factors. Similarly the complexity of the expectations correspond to the largest factors obtained by variable elimination.

The node splitting technique (Alg. 2) evaluates each split by executing EM. Since there are $|\mathcal{N}|$ possible nodes that can be split and the algorithm incrementally grows the controller by splitting one node at a time until there are $|\mathcal{N}|$ nodes, the complexity is $|\mathcal{N}|^2$ times that of EM. As a result, there is a quartic dependency on the size of the controller and this is the dominating factor for large controllers.

The forward search technique (Alg. 1) alternates between EM and adding new nodes based on a recursive search up to some depth d. This search is exponential in d with base $|\mathcal{A}||\mathcal{O}|$, however one can often reduce this complexity by branch-and-bound or sub-sampling. Also, the search is performed by increasing the depth gradually until a non-optimal setting of the parameters is found, at which point the search is terminated. Since we perform a search from each of the $\mathcal{N}$ nodes and the controller is grown incrementally up to $\mathcal{N}$ nodes, the overall complexity of forward search is $|\mathcal{N}|^2$ times that of the recursive search plus $|\mathcal{N}|$ times that of EM. In comparison to node splitting, forward search has a cubic dependency on the size of the controller and therefore tends to scale better for large controllers, but it also has an exponential dependency on the search depth.

Table 1. Computational Complexity. Here I is the number of iterations in EM, t_{max} is the length of the planning horizon, d is the depth of the forward search and $\bar{O}$ is the computational complexity with respect to $|\mathcal{N}|$ and d only.

Forward-Backward step (FB) (Eq. 4 and 5): $O(t_{max}(\|\mathcal{N}\|\|\mathcal{S}\|^2 + \|\mathcal{N}\|^2\|\mathcal{S}\|)) = \bar{O}(\|\mathcal{N}\|^2)$ Expectation step (Exp) (Eq. 1-3): $O(\|\mathcal{N}\|\|\mathcal{A}\|\|\mathcal{S}\|^2 + \|\mathcal{N}\|\|\mathcal{A}\|\|\mathcal{S}\|\|\mathcal{O}\| + \|\mathcal{N}\|^2\|\mathcal{S}\|\|\mathcal{O}\|) = \bar{O}(\|\mathcal{N}\|^2)$ Expectation-Maximization (EM): $O(I(\text{FB} + \text{Exp})) = \bar{O}(\|\mathcal{N}\|^2)$
Node splitting (Alg. 2): $O(\|\mathcal{N}\|^2\text{EM}) = \bar{O}(\|\mathcal{N}\|^4)$
Recursive search (RS) (Alg. 1): $O(\|\mathcal{N}\|(\|\mathcal{A}\|\|\mathcal{O}\|)^d\|\mathcal{S}\|^2) = \bar{O}(\|\mathcal{N}\|(\|\mathcal{A}\|\|\mathcal{O}\|)^d)$ Forward search (Alg. 1): $O(\|\mathcal{N}\|^2\text{RS} + \|\mathcal{N}\|\text{EM}) = \bar{O}(\|\mathcal{N}\|^3(\|\mathcal{A}\|\|\mathcal{O}\|)^d)$

5 Experiments

We tested four different methods for escaping local optima. The first two are alternatives that add nodes based on a forward search. The third one is the node splitting heuristic. The fourth one is a baseline that uses random restarts for all controller sizes instead of incrementally adding nodes. The detailed settings for these methods are as follows:

- *Forward Search:* Starting from a randomly initialized FSC of $|\mathcal{N}| = |\mathcal{A}|$ nodes, EM is performed till convergence. Then a forward search of increasing depth (up to a time limit) is performed from the belief $b_{s|n}$ associated with each node according to Algorithm 1. As soon as a non-optimal action or successor node distribution is discovered, the search is halted and new nodes for each reachable belief on the current path are added to the controller.
- *Forward Search From Init:* This technique is the same as the previous one except that the forward search is performed only from the initial belief p_s (instead of each $b_{s|n}$. It serves as a baseline to show that forward search from each $b_{s|n}$ is advantageous.
- *Node Splitting:* Starting from a randomly initialized FSC of $|\mathcal{N}| = |\mathcal{A}|$ nodes, we iterate the node splitting procedure of Algorithm 2 until the controller reaches a desired size.
- *Random Restarts:* For each FSC size $|\mathcal{N}| \in \{|\mathcal{A}|, .., n\}$ we randomly initialize a FSC and train it with 500 EM iterations. Note that we also implemented a technique that adds random nodes to escape local optima, but it performed worse than random restarts, so we only report the performance of random restarts as the baseline.

The left column of Figure 1 shows the performance of each method as the number of nodes increases for 6 POMDP benchmarks. Each curve is the median of 21 runs from different initial random controllers with error bars corresponding to

the 25% and 75% quantiles. For the incremental methods, the curves show how the value increases as the number of nodes increases; for the random restarts the results for different $|\mathcal{N}|$ are mutually independent.

The Cheese-Taxi (variant from [12]) and Heaven-and-Hell (variant from [3]) problems are known to have difficult local optima in the sense that the optimal policies include a long sequence of actions such that any small deviation from that sequence is bad. Hence, policy search techniques have trouble finding this sequence by gradually refining a policy since that would involve trying nearby sequences with low values. Only the forward search techniques found good policies because of their ability to modify sequences of actions in one step by adding several nodes at once. Forward search from each $b_{s|n}$ finds good policies with fewer nodes than a forward search from the initial belief because it doesn't have to search as deeply. The random restart and node splitting techniques did not escape the trivial local optima. Since the node splitting technique only changes one node per step, it cannot change multiple actions at once.

The optimal policy of the Chain-of-Chains problem [18] also includes a long sequence of actions, however small deviations are not penalized (i.e., no negative reward), but it will simply take longer for the agent to reach a high rewarding state. As a result, all techniques eventually find the optimal policy, but the forward search techniques find optimal controllers with much fewer nodes, followed by node splitting and random restarts.

The Hallway [10], Hallway2 [10] and Machine [4] problems do not have optimal policies with long sequences of actions that must not be deviated from, but they can still be challenging for policy search techniques that do not explicitly try to escape local optima such as random restarts. For these problems, splitting a node or adding a node based on a short forward search tends to be sufficient to escape local optima. The node splitting technique is generally better because of its ability to evaluate more accurately alternative controllers. Alternative splits are evaluated by re-running EM, which gives a more accurate value than forward search, which adds a node to correct the policy at some reachable belief without re-running EM.

The right column of Figure 1 shows the performance as time increases. Depending on the problem, different techniques among node splitting, forward search and forward search from init may be best. As explained in Section 4.3, forward search tends to scale better than node splitting as the size of the controller increases due to a cubic dependency on the number of nodes (instead of a quartic dependency for node splitting). This is evident for the hallway and chain-of-chains problems. However, forward search may spend a lot of time in a search due to the exponential complexity and it does not necessarily add the best node since it corrects the first non-optimal parameter that it finds. We believe this explains the better performance of node splitting for the machine problem. In general, performing a forward search from each node is advantageous in comparison to a single search from the initial belief since a non-optimal parameter is often found with a shallower search. This explains why forward search performed better for heaven-hell, hallway, hallway2 and machine. However, when a single

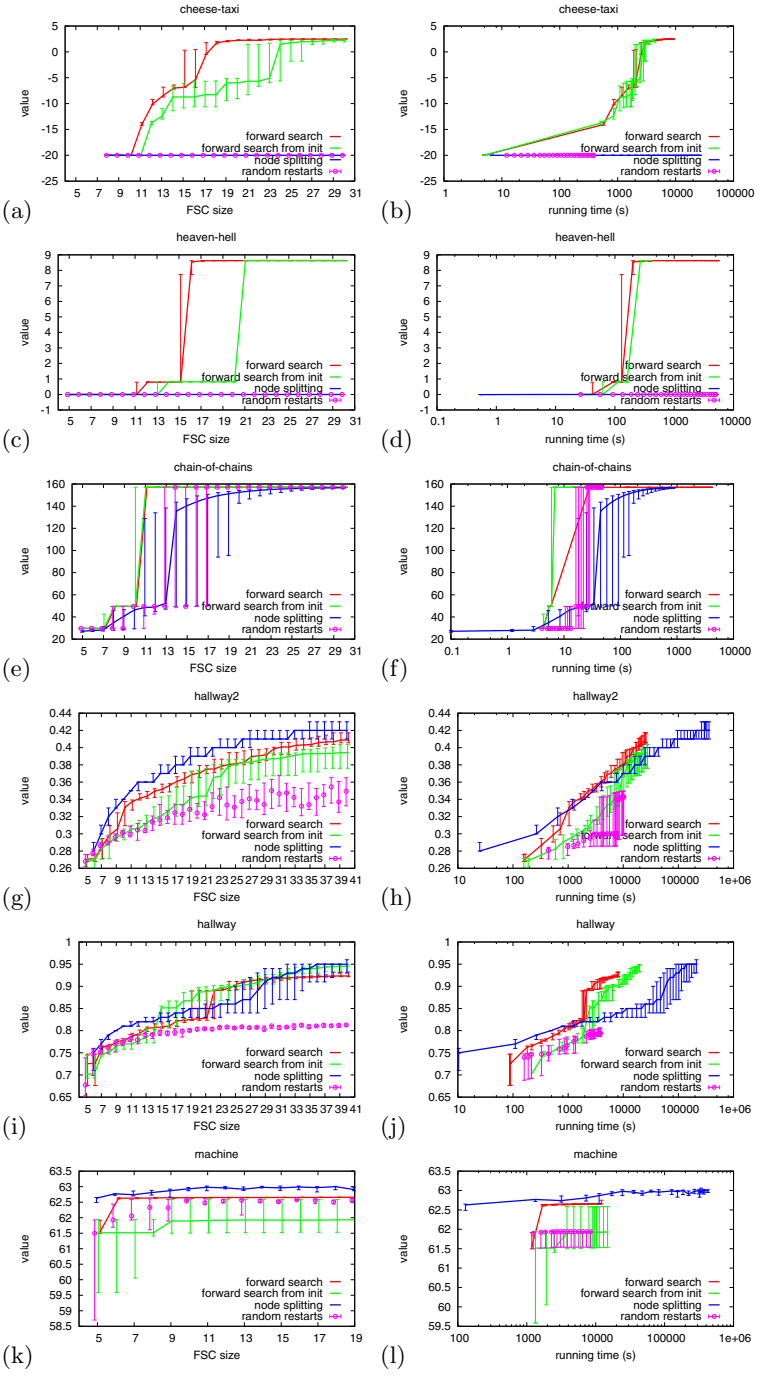

Fig. 1. Performance versus number of nodes (left column) and time (right column). NB: The graphs are best viewed in color.

Table 2. Comparison of the value (average/median over at least 10 runs) for controllers/value functions of different sizes (e.g., # of nodes/α-vectors) indicated in parentheses. Here n.a. indicates that the results are not available and ? indicates that the number of nodes is unknown.

Techniques	cheeseT	heavenH	chainOC	hallway2	hallway	machine
upper bound	2.48	8.64	157.1	0.88	1.18	66.1
SARSOP (10^5 sec)	2.48(168)	8.64(1720)	157.1(10)	0.44(3295)	1.01(4056)	63.2(1262)
SARSOP	-6.38(40)	0.45(55)	**157.1(10)**	0.11(50)	0.15(49)	35.7(42)
biased-BPI+escape	2.13(30)	3.50(30)	40.0(30)	0.41(40)	0.94(40)	63.0(30)
QCLP	n.a.	n.a.	n.a.	n.a.	0.72(8)	61.0(6)
BBSLS	n.a.	7.65(?)	n.a.	n.a.	0.80(10)	n.a.
Forward Search	**2.47(19)**	**8.64(16)**	157.1(11)	0.41(40)	0.92(40)	62.6(19)
Node Splitting	-20.0(30)	0.00(30)	157.1(23)	**0.43(40)**	**0.95(40)**	**63.0(16)**

search from the initial belief does not go deeper than multiple searches from each node, then a single search from the initial belief is faster, which explains the better performance for chain-of-chains.

In Table 2, we compare the forward search and node splitting techniques to a leading point-based value iteration technique (SARSOP [9]) and three policy search techniques for finite state controllers (biased BPI with escape [13], nonlinear optimization (QCLP) [1] and stochastic local search (BBSLS) [3]). The number of nodes was limited to 30 for cheese-taxi, heaven-hell and chain-of-chains, and 40 for hallway2, hallway and machine. Since each node leads to one α-vector in the value function representation, we report the number of alpha-vectors that is closest to 30 and 40 for SARSOP.

Since the optimal policy is not known for several problems, we also report an upper bound on the optimal value (computed by SARSOP) as well as the best value found by SARSOP within 10^5 seconds when the number of α-vectors is not bounded. The results for SARSOP and BPI were obtained by running the APPL package and Poupart's BPI code. The results for QCLP and BBSLS are taken from [1] and [3]. When the number of nodes/α-vectors is limited, forward search achieves the best results for cheese-taxi and heaven-hell, node-splitting achieves the best results for hallway, hallway2 and machine, and SARSOP achieves the best results for chain-of-chains. Overall, forward search obtains the most reliable results by producing controllers that are close to the bests for all problems. SARSOP does not perform well when the number of α-vectors is limited, but can obtain slightly better results when the number of α-vectors is not limited. Note that SARSOP was specifically designed to use fewer α-vectors than other point-based techniques including HSVI2 [16]. Compact controllers/value functions become advantageous for embedded systems with limited memory and processing power. Furthermore, action selection with controllers is instantaneous since there is no computation, whereas a non-trivial search among all α-vectors must be performed to execute policies derived from a set of α-vectors.

6 Conclusion

The main contributions of this paper are a characterization of EM's local optima and the design of two techniques to help EM escape local optima. We showed that EM essentially performs a one-step forward search that optimizes the policy parameters in isolation. Based on this insight, we designed an escape technique that adds new nodes to the controller when a suboptimal action or successor node is detected according to a multi-step forward search that extends EM's implicit one-step search. We also designed a technique to split nodes in two and optimize the new parameters by EM. The forward search technique is the most reliable approach to effectively escape local optima for all 6 benchmark problems, while the node splitting technique finds slightly better controllers for the 3 benchmark problems that do not include a stringent sequence of actions in their optimal policies.

Although there already exist escape techniques for finite state controllers, none of them can be combined with EM (or planning as inference) since they rely on specific information computed by the optimization approaches they were designed for. Hence, assuming that planning as inference will become a leading POMDP solution technique in the near future, this work resolves an important issue by mitigating the effect of local optima and improving the reliability of EM. Our next step is to extend our implementation to factored domains since this is where planning as inference becomes really attractive. In particular, approximate inference techniques such as loopy belief propagation and variational techniques could be integrated within EM and our escape techniques to yield highly scalable algorithms for factored POMDPs.

This work could be extended in several directions, including hierarchical controllers [18], Mealy finite state machines [2] and multi-agent settings [8], which all face local optima, but of different nature than those investigated in this paper.

Acknowledgments. This work was supported by the German Research Foundation (DFG), Emmy Noether fellowship TO 409/1-3 as well as the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

1. Amato, C., Bernstein, D., Zilberstein, S.: Solving POMDPs using quadratically constrained linear programs. In: IJCAI, pp. 2418–2424 (2007)
2. Amato, C., Bonet, B., Zilberstein, S.: Finite-state controllers based on mealy machines for centralized and decentralized POMDPs. In: AAAI (2010)
3. Braziunas, D., Boutilier, C.: Stochastic local search for POMDP controllers. In: AAAI, pp. 690–696 (2004)
4. Cassandra, A.: Exact and approximate algorithms for partially observable Markov decision processes. PhD thesis, Brown University, Dept. of Computer Science (1998)
5. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. of the Royal Stat. Society, Series B 39(1), 1–38 (1977)

6. Hansen, E.: An improved policy iteration algorithm for partially observable MDPs. In: NIPS (1998)
7. Hoffman, M., Kueck, H., Doucet, A., de Freitas, N.: New inference strategies for solving Markov decision processes using reversible jump MCMC. In: UAI (2009)
8. Kumar, A., Zilberstein, S.: Anytime planning for decentralized POMDPs using expectation maximization. In: UAI (2010)
9. Kurniawati, H., Hsu, D., Lee, W.S.: SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: Proc. Robotics: Science and Systems (2008)
10. Littman, M., Cassandra, T., Kaelbling, L.: Learning policies for partially observable environments: scaling up. In: ICML, pp. 362–370 (1995)
11. Meuleau, N., Kim, K.-E., Kaelbling, L., Cassandra, A.: Solving POMDPs by searching the space of finite policies. In: UAI, pp. 417–426 (1999)
12. Pineau, J.: Tractable Planning Under Uncertainty: Exploiting Structure. PhD thesis, Robotics Institute, Carnegie Mellon University (2004)
13. Poupart, P.: Exploiting structure to efficiently solve large scale partially observable Markov decision processes. PhD thesis, University of Toronto (2005)
14. Poupart, P., Boutilier, C.: Bounded finite state controllers. In: NIPS (2003)
15. Siddiqi, S., Gordon, G., Moore, A.: Fast state discovery for HMM model selection and learning. In: AI-STATS (2007)
16. Smith, T., Simmons, R.: Point-based POMDP algorithms: improved analysis and implementation. In: UAI (2005)
17. Sondik, E.: The optimal control of partially observable decision processes over the infinite horizon: Discounted cost. Operations Research 26(2), 282–304 (1978)
18. Toussaint, M., Charlin, L., Poupart, P.: Hierarchical POMDP controller optimization by likelihood maximization. In: UAI (2008)
19. Toussaint, M., Harmeling, S., Storkey, A.: Probabilistic inference for solving (PO)MDPs. Technical Report EDI-INF-RR-0934, School of Informatics, University of Edinburgh (2006)
20. Toussaint, M., Storkey, A.J.: Probabilistic inference for solving discrete and continuous state Markov decision processes. In: ICML (2006)
21. Vlassis, N., Toussaint, M.: Model free reinforcement learning as mixture learning. In: ICML (2009)