Hierarchical POMDP Controller Optimization by Likelihood Maximization

Marc Toussaint
Computer Science
TU Berlin
D-10587 Berlin, Germany
mtoussai@cs.tu-berlin.de

Laurent Charlin
Computer Science
University of Toronto
Toronto, Ontario, Canada
lcharlin@cs.toronto.edu

Pascal Poupart
Computer Science
University of Waterloo
Waterloo, Ontario, Canada
ppoupart@cs.uwaterloo.ca

Abstract

Planning can often be simplified by decomposing the task into smaller tasks arranged hierarchically. Charlin et al. (2006) recently showed that the hierarchy discovery problem can be framed as a non-convex optimization problem. However, the inherent computational difficulty of solving such an optimization problem makes it hard to scale to real-world problems. In another line of research, Toussaint et al. (2006) developed a method to solve planning problems by maximum-likelihood estimation. In this paper, we show how the hierarchy discovery problem in partially observable domains can be tackled using a similar maximum likelihood approach. Our technique first transforms the problem into a dynamic Bayesian network through which a hierarchical structure can naturally be discovered while optimizing the policy. Experimental results demonstrate that this approach scales better than previous techniques based on non-convex optimization.

1 Introduction

Planning in partially observable domains is notoriously difficult. However, many planning tasks naturally decompose into subtasks that may be arranged hierarchically (e.g., prompting systems that assist older adults with activities of daily living (Hoey *et al.* 2007) can be naturally decomposed into subtasks for each step of an activity). When a decomposition or hierarchy is known a priori, several approaches have demonstrated that planning can be performed faster (Pineau, Gordon, & Thrun 2003; Hansen & Zhou 2003). However, the hierarchy is not always known or easy to specify, and the optimal policy may only decompose *approximately*. To that effect, Charlin et al. (2006) showed how a hierarchy can be discovered automatically by formulating the planning problem as a non-convex quartically constrained optimization problem with variables corresponding to the parameters of the policy, including its hierarchical structure. Unfortunately, the inherent computational difficulty of solving this optimization problem prevents the approach from scaling to real-world problems. Furthermore, it is not clear that automated hierarchy discovery simplifies planning since the space of policies remains the same.

We propose an alternative approach that demonstrates that hierarchy discovery (i) can be done efficiently and (ii) performs a policy search with a different bias than non-hierarchical approaches that is advantageous when there exists good hierarchical policies. The approach combines Murphy and Paskin's (2001) factored encoding of hierarchical structures into a dynamic Bayesian network (DBN) with Toussaint et al.'s (2006) maximum-likelihood estimation technique for policy optimization. More precisely, we encode POMDPs with hierarchical controllers into a DBN in such a way that the policy and hierarchy parameters are entries of some conditional probability tables. We also consider factored policies that are more general than hierarchical controllers. The policy and hierarchy parameters are optimized with the expectation-maximization (EM) algorithm (Dempster, Laird, & Rubin 1977). Since each iteration of EM essentially consists of inference queries, the approach scales easily.

Sect. 2 briefly introduces partially observable Markov decision processes, controllers, policy optimization by maximum likelihood estimation and hierarchical modeling. Sect. 3 describes our proposed approach, which combines a dynamic Bayesian network encoding with maximum likelihood estimation to simultaneously optimize a hierarchy and the controller. Sect. 4 demonstrates the scalability of the proposed approach on benchmark problems. Finally, Sect. 5 summarizes the paper and discusses future work.

2 Background

2.1 POMDPs

Partially observable Markov decision processes (POMDPs) provide a natural and principled framework for planning. They are formally defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, p_s, p_{s'|as}, p_{o'|s'a}, r_{as} \rangle$ where $\mathcal{S}$ is the set of states s, $\mathcal{A}$ is the set of actions a, $\mathcal{O}$ is the set of observations o, $p_s = \Pr(S_0 = s)$ is the initial state distribution (a.k.a. initial belief), $p_{s'|as} = \Pr(S_{t+1} = s' \mid A_t = a, S_t = s)$ is the transition distribution, $p_{o'|s'a} = \Pr(O_{t+1} = o' \mid S_{t+1} = s', A_t = a)$ is the observation distribution and $r_{as} = R(A_t = a, S_t = s)$ is the reward function. Since states are not directly observable, let $b_s = \Pr(S = s)$ be a state distribution (conditioned on the history of past actions and observations) known as the *belief*. A belief b can be updated at each time step, based on the action a taken and the observation o' made according to Bayes' theorem: $b^{ao'}_{s'} = k \sum_s b_s p_{s'|as} p_{o'|s'a}$ (k is a normalization constant). A policy is a mapping from beliefs to actions (e.g., $\pi(b) = a$). The value $V^{\pi}(b)$ of a policy π starting in belief b is measured by the discounted sum of expected rewards: $V^{\pi}(b) = \sum_t \gamma^t E_{b_t|\pi}[r_{\pi(b_t)b_t}]$ where $r_{ab} = \sum_s b_s r_{as}$. The goal is to find an optimal policy π^* with the highest value V^* for all beliefs: $V^*(b) \geq V^{\pi}\ \forall \pi, b$. The optimal value function also satisfies Bellman's equation: $V^*(b) = \max_a r_{ab} + \sum_{o'} p_{o'|ab} V^*(b^{ao'})$ where $p_{o'|ab} = \sum_{ss'} b_s p_{s'|as} p_{o'|s'a}$.

2.2 Finite State Controllers

A convenient representation for an important class of policies consists of finite state controllers (Hansen 1998). A controller with a finite set $\mathcal{N}$ of nodes n can encode a stochastic policy π with three distributions: $\Pr(N_0 = n) = p_n$ (initial node distribution), $\Pr(A_t = a \,|\, N_t = n) = p_{a|n}$ (action selection distribution) and $\Pr(N_{t+1} = n' \,|\, N_t = n, O_{t+1} = o') = p_{n'|no'}$ (successor node distribution). Such a policy can be executed by starting in a node n sampled from p_n, executing an action a sampled from $p_{a|n}$, receiving observation o', transitioning to node n' sampled from $p_{n'|no'}$ and so on. The value of a controller can be computed by solving a linear system: $V_{ns} = \sum_a p_{a|n}[r_{as} + \gamma \sum_{s'o'n'} p_{s'|as} p_{o'|s'a} p_{n'|no'} V_{n's'}]\ \forall n, s$. The value at a given belief b is then $V^{\pi}(b) = \sum_n \sum_s b_s p_n V_{ns}$. Several techniques have been proposed to optimize controllers of a given size, including gradient ascent (Meuleau *et al.* 1999), stochastic local search (Braziunas & Boutilier 2004), bounded policy iteration (Poupart & Boutilier 2003), non-convex quadratically constrained optimization (Amato, Bernstein, & Zilberstein 2007) and likelihood maximization (Toussaint, Harmeling, & Storkey 2006). We briefly describe the last technique since we will use it in Sect. 3.

Toussaint et al. (2006) recently proposed to convert POMDPs into equivalent dynamic Bayesian networks (DBNs) by normalizing the rewards and to optimize a policy by maximizing the likelihood of the normalized rewards. Let $\tilde{R}$ be a binary variable corresponding to normalized rewards. The reward function r_{as} is then replaced by a reward distribution $p_{\tilde{r}|sat} = \Pr(\tilde{R} = \tilde{r} \,|\, A_t = a, S_t = s, T = t)$ that assigns probability $r_{as}/(r_{max} - r_{min})$ to $\tilde{R} = 1$ and $1 - r_{as}/(r_{max} - r_{min})$ to $\tilde{R} = 0$ ($r_{min} = \min_{as} r_{as}$ and $r_{max} = \max_{as} r_{as}$). An additional time variable T is introduced to simulate the discount factor and the summation of rewards. Since a reward is normally discounted by a factor γ^t when earned t time steps in the future, the prior $p_t = \Pr(T = t)$ is set to $\gamma^t(1-\gamma)$ where the factor $(1-\gamma)$ ensures that $\sum_{t=0}^{\infty} p_t = 1$. The resulting dynamic Bayesian network is illustrated in Fig. 1. It can be thought of as a mixture of finite processes of length t with a 0-1 reward $\tilde{R}$ earned at the end of the process. The nodes N_t encode the internal memory of the controller. Given the controller distributions p_n, $p_{a|n}$ and $p_{n'|no'}$, it is possible to evaluate the controller by computing the likelihood of $\tilde{R} = 1$. More precisely, $V^{\pi}(p_s) = (\Pr(\tilde{R}=1) - r_{min})/[(r_{max} - r_{min})(1-\gamma)]$.

Optimizing the policy can be framed as maximizing the likelihood of $\tilde{R} = 1$ by varying the distributions p_n, $p_{a|n}$ and $p_{n'|no'}$ encoding the policy. Toussaint et al. use the expectation-maximization (EM) algorithm. Since EM is guaranteed to increase the likelihood at each iteration, the controller's value increases monotonically. However, EM is not guaranteed to converge to a global optimum. An important advantage of the EM algorithm is its simplicity both conceptually and computationally. In particular, the computation consists of inference queries that can be computed using a variety of exact and approximate algorithms.

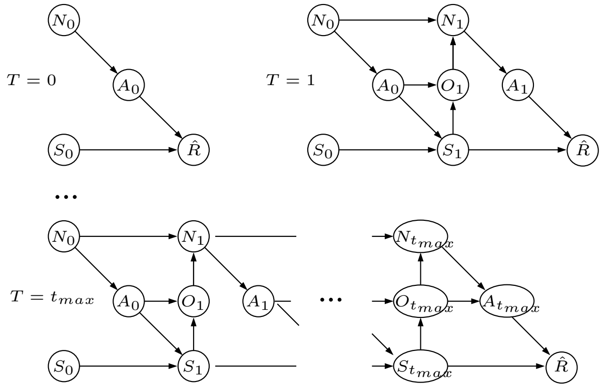


Figure 1: POMDP represented as a mixture of finite DBNs. For an infinite horizon, a large enough t_{max} can be selected at runtime to ensure that the approximation error is small.

2.3 Hierarchical Modeling

While optimizing a bounded controller allows an effective search in the space of bounded policies, such an approach is clearly suboptimal since the optimal controller of many problems grows doubly exponentially with the planning horizon and may be infinite for infinite horizons. Alternatively, hierarchical representations permit the representation of structured policies with exponentially fewer parameters. Several approaches were recently explored to model and learn hierarchical structures in POMDPs. Pineau et al. (2003) sped up planning by exploiting a user specified action hierarchy. Hansen et al. (2003) proposed hierarchical controllers and an alternative planning technique that also exploits a user specified hierarchy. Charlin et al. (2006) proposed recursive controllers (which subsume hierarchical controllers) and an approach that discovers the hierarchy while optimizing a controller. In another line of research, dynamic Bayesian networks are used to model hierarchical hidden Markov models (HMMs) (Murphy & Paskin 2001) and hierarchical POMDPs (Theocharous, Murphy, & Kaelbling 2004). We briefly review this DBN encoding since we will use it in our approach to model factored controllers.

Murphy and Paskin (2001) proposed to model hierarchical hidden Markov models (HMMs) as dynamic Bayesian networks (DBNs). The idea is to convert a hierarchical HMM of L levels into a dynamic Bayesian network of L state variables, where each variable encodes abstract states at the corresponding level. Here, abstract states can only call sub-HMMs at the previous level. Fig. 2 illustrates a two-level hierarchical HMMs encoded as a DBN. The state

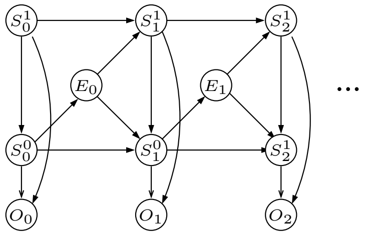


Figure 2: DBN encoding of a 2-level hierarchical HMM.

variables S_t^l are indexed by the time step t and the level l. The E_t variables indicate when a base-level sub-HMM has ended, returning its control to the top level HMM. The top-level abstract state transitions according to the top HMM, but only when the exit variable E_t indicates that the base-level concrete state is an exit state. The base-level concrete state transitions according to the base-level HMM. When an exit state is reached, the next base-level state is determined by the next top-level abstract state. Factored HMMs subsume hierarchical HMMs in the sense that there exists an equivalent factored HMM for every hierarchical HMM. In Sect. 3.1, we will use a similar technique to convert hierarchical controllers into factored controllers.

3 Factored Controllers

We propose to combine the DBN encoding techniques of Murphy et al. (2001) and Toussaint et al. (2006) to convert a POMDP with a hierarchical controller into a mixture of DBNs. The hierarchy and the controller are simultaneously optimized by maximizing the reward likelihood of the DBN. We also consider factored controllers which subsume hierarchical controllers.

3.1 DBN Encoding

Fig. 3a illustrates two consecutive slices of one DBN in the mixture (rewards are omitted) for a three-level hierarchical controller. Consider a POMDP defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, p_s, p_{s'|as}, p_{o'|s'a}, r_{as} \rangle$ and a three-level hierarchical controller defined by the tuple $\langle p_{a|n^l}, p_{n^{l-1}|n^l}, p_{n'^l|n^l o'} \rangle$ for each level l. The conditional probability distributions of the mixture of DBNs (denoted by $\hat{p}$) are:

- transition distribution: $\hat{p}_{s'|as} = p_{s'|as}$
- observation distribution: $\hat{p}_{o'|s'a} = p_{o'|s'a}$
- reward distribution: $\hat{p}_{\tilde{r}|as} = (r_{as} - r_{min})/(r_{max} - r_{min})$
- mixture distribution: $\hat{p}_t = (1-\gamma)\gamma^t$
- action distribution: $\hat{p}_{a|n^0} = p_{a|n^0}$
- base level node distribution:
$$\hat{p}_{n'^0|n^0 n'^1 o' e^0} = \begin{cases} p_{n'^0|n'^1} & \text{if } e^0 = exit \\ p_{n'^0|o'n^0} & \text{otherwise} \end{cases}$$
- middle level node distribution:
$$\hat{p}_{n'^1|n^1 n'^2 o' e^0 e^1} = \begin{cases} p_{n'^1|n'^2} & \text{if } e^1 = exit \\ p_{n'^1|o'n^1} & \text{if } e^0 = exit \text{ and } e^1 \neq exit \\ \delta_{n'^1 n^1} & \text{otherwise} \end{cases}$$
- top level node distribution:
$$\hat{p}_{n'^2|o'n^2 e^1} = \begin{cases} p_{n'^2|o'n^2} & \text{if } e^1 = exit \\ \delta_{n'^2 n^2} & \text{otherwise} \end{cases}$$

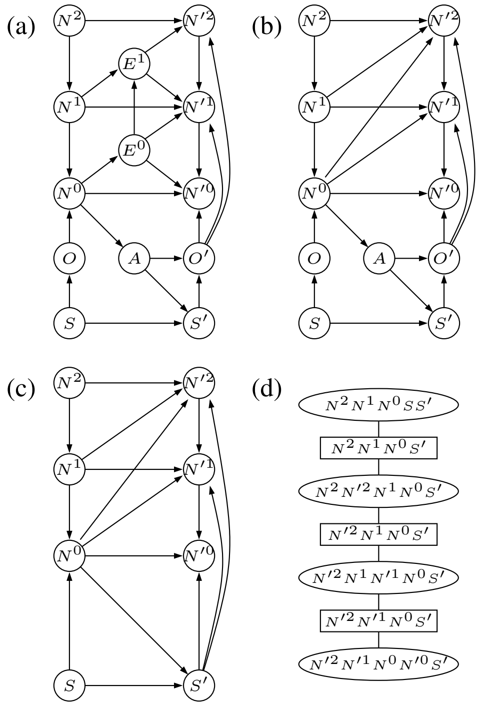


Figure 3: (a) Two slices of the DBN encoding the hierarchical POMDP controller. (b) Exit variables are eliminated. (c) Variables O and A are eliminated. (d) Corresponding junction tree.

- base-level exit distribution:
$$\hat{p}_{e^0|n^0} = \begin{cases} 1 & \text{if } n^0 \text{ is an end node} \\ 0 & \text{otherwise} \end{cases}$$
- middle-level exit distribution:
$$\hat{p}_{e^1|n^1 e^0} = \begin{cases} 1 & \text{if } e^0 = exit \text{ and } n^1 \text{ is an end node} \\ 0 & \text{otherwise} \end{cases}$$

While the E_t^l variables help clarify when the end of a sub-controller is reached, they are not necessary. Eliminating them yields a simpler DBN illustrated in Fig. 3b. The conditional probability distributions of each N_t^l become:

- base level node distribution:
$$\hat{p}_{n'^0|n^0 n'^1 o'} = \begin{cases} p_{n'^0|n'^1} & \text{if } n^0 \text{ is an end node} \\ p_{n'^0|o'n^0} & \text{otherwise} \end{cases}$$
- middle level node distribution:
$$\hat{p}_{n'^1|n^1 n'^2 o'} = \begin{cases} p_{n'^1|n'^2} & \text{if } n^1 \text{ and } n^0 \text{ are end nodes} \\ p_{n'^1|o'n^1} & \text{if } n^0 \text{ is an end node, but not } n^1 \\ \delta_{n'^1 n^1} & \text{otherwise} \end{cases}$$
- top level node distribution:
$$\hat{p}_{n'^2|n^2 o' e^1} = \begin{cases} p_{n'^2|n^2 o'} & \text{if } n^1 \text{ and } n^0 \text{ are end nodes} \\ \delta_{n'^2 n^2} & \text{otherwise} \end{cases}$$

Note that ignoring the above constraints in the conditional distributions yields a *factored controller* that is more flexible than a hierarchical controller since the conditional probability distributions of the N_t^l variables do not have to follow the structure imposed by a hierarchy.

3.2 Maximum Likelihood Estimation

Following Toussaint et al. (2006), we optimize a factored controller by maximizing the reward likelihood. Since the policy parameters are conditional probability distributions of the DBN, the EM algorithm can be used to optimize them. Computation alternates between the E and M steps below. We denote by n^{top} and n^{base} the top and base nodes in a given time slice. We also denote by $\phi(V)$ and $\phi(v)$ the parents of V and a configuration of the parents of V.

- E-step: expected frequency of the hidden variables
  $E_{n^{top}} = \Pr(N_0^{top} = n^{top} | \tilde{R} = 1)$
  $E_{an^{base}} = \sum_t \Pr(A_t = a, N_t^{base} = n^{base} | \tilde{R} = 1)$
  $E_{n'^l \phi(n'^l)} = \sum_t \Pr(N_{t+1}^l = n'^l, \phi(N_{t+1}^l) = \phi(n_{t+1}^l) | \tilde{R} = 1)$
- M-step: relative frequency computation
  $p_{n^{top}} = E_{n^{top}} / \sum_{n^{top}} E_{n^{top}}$
  $p_{a|n^{base}} = E_{an^{base}} / \sum_a E_{an^{base}}$
  $p_{n'^l|\phi(n'^l)} = E_{n'^l \phi(n'^l)} / \sum_{n'^l} E_{n'^l \phi(n'^l)} \; \forall l$

Parameter initialization W.l.o.g. we initialize the start node N_0^{top} of the top layer to be the first node (i.e., $\Pr(N_0^{top} = 1) = 1$). The node conditional distributions $p_{n'^l|\phi(n'^l)}$ are initialized randomly as a mixture of three distributions: $p_{n'^l|\phi(n'^l)} \propto c_1 + c_2 \mathcal{U}_{n'^l \phi(n'^l)} + c_3 \delta_{n'^l n^l}$. The mixture components are a uniform distribution, a random distribution $\mathcal{U}_{\phi(n'^l)}$ (an array of uniform random numbers in $[0, 1]$), and a term enforcing n^l to stay unchanged. For the node distributions at the base level we choose $c_1 = 1, c_2 = 1, c_3 = 0$ and for all other levels we choose $c_1 = 1, c_2 = 1, c_3 = 10$. Similarly we initialize the action probabilities as $p_{a|n^{base}} \propto c_1 + c_2 \mathcal{U}_{an^{base}} + c_3 \delta_{a(n^{base} \% a)}$ with $c_1 = 1, c_2 = 1, c_3 = 100$, where the last term enforces each node $n^{base} = i$ to be associated with action $a = i \% a$.

E-step To speed up the computation of the inference queries in the E-step, we compute intermediate terms using a forward-backward procedure. Here, $\mathbf{N}$ and $\mathbf{n}$ denote all the nodes and their joint configuration in a given time slice.

- Forward term: $\alpha_{\mathbf{n}s}^t = \Pr(\mathbf{N}_t = \mathbf{n}, S_t = s)$
  $\alpha_{\mathbf{n}s}^0 = p_{\mathbf{n}} p_s$
  $\alpha_{\mathbf{n}'s'}^t = \sum_{\mathbf{n},s} \alpha_{\mathbf{n}s}^{t-1} p_{\mathbf{n}'s'|\mathbf{n}s}$
- Backward term: $\beta_{\mathbf{n}s}^\tau = \Pr(\tilde{R} = 1 | \mathbf{N}_{t-\tau} = \mathbf{n}, S_{t-\tau} = s, T = t)$
  $\beta_{\mathbf{n}s}^0 = \sum_a p_{a|\mathbf{n}} r_{as}$
  $\beta_{\mathbf{n}s}^\tau = \sum_{\mathbf{n}',s'} p_{\mathbf{n}'s'|\mathbf{n}s} \beta_{\mathbf{n}'s'}^{\tau-1}$

To fully take advantage of the structure of the DBN, we first marginalize the DBN w.r.t. the observations and actions to get the DBN in Fig. 3c. This 2-slice DBN corresponds to the joint transition distribution $p_{\mathbf{n}'s'|\mathbf{n}s}$ used in the above equations. Then we compile this 2-slice DBN into the junction tree (actually junction chain) given in Fig. 3d.

Let $\beta_{\mathbf{n}s} = \sum_\tau \Pr(T = \tau) \beta_{\mathbf{n}s}^\tau$ and $\alpha_{\mathbf{n}s} = \sum_t \Pr(T = t) \alpha_{\mathbf{n}s}^t$, then the last two expectations of the E-step can be computed as follows:

$$E_{an^{base}} \propto \sum_{s,\mathbf{n}-\{n^{base}\}} \alpha_{\mathbf{n}s} p_{a|n^{base}} \Big[r_{as} + \sum_{s',o',\mathbf{n}'} p_{s'|as} p_{o'|s'a} p_{\mathbf{n}'|o'\mathbf{n}} \beta_{\mathbf{n}'s'} \Big]$$

$$E_{n'^l \phi(n'^l)} \propto \sum_{s,a,\mathbf{n}-\phi(n'^l)} \alpha_{\mathbf{n}s} p_{a|n^{base}} \Big[r_{as} + \sum_{s',n'-l} p_{s'|as} p_{o'|s'a} p_{\mathbf{n}'|o'\mathbf{n}} \beta_{\mathbf{n}'s'} \Big] \; \forall l$$

M-step The standard M-step adjusts the parameters of the controller by normalizing the expectations computed in the E-step. Instead, to speed up convergence, we perform a softened *greedy M-step*. In the greedy M-step, each parameter $p_{v|\phi(v)}$ is greedily set to 1 when $v = argmax_{\bar{v}} f_{\bar{v}\phi(\bar{v})}$ and 0 otherwise, where $f_{v\phi(v)} = E_{v\phi(v)} / p_{v|\phi(v)}$. The greedy M-step can be thought of as the limit of an infinite sequence of alternating partial E-step and standard M-step where the partial E-step keeps f fixed. The combination of a standard M-step with this specific partial E-step updates $p_{v|\phi(v)}$ by a multiplicative factor proportional to $f_{v\phi(v)}$. In the limit, the largest $f_{v\phi(v)}$ ends up giving all the probability to the corresponding $p_{v|\phi(v)}$. EM variants with certain types of partial E-steps ensure monotonic improvement of the likelihood when the hidden variables are independent (Neal & Hinton 1998). This is not the case here, however by softening the greedy M-step we can still obtain monotonic improvement most of the time while speeding up convergence. We update $p_{v|\phi(v)}$ as follows:

$$p_{v|\phi(v)}^{new} \propto p_{v|\phi(v)}^{old} [\delta_{vv^*} + c + \epsilon] \text{ where } v^* = \text{argmax}_v \, f_{v\phi(v)}$$

For $c = 0$ and $\epsilon = 0$ this is the greedy M-step. We use $c = 3$ which softens (shortens) the step and ensures more robust convergence. Furthermore, adding small Gaussian noise $\epsilon \sim \mathcal{N}(0, 10^{-3})$ helps to escape local minima.

Complexity For a flat controller, the number of parameters (neglecting normalization) is $|\mathcal{O}||\mathcal{N}|^2$ for $p_{n'|o'n}$ and $|\mathcal{A}||\mathcal{N}|$ for $p_{a|n}$. The complexity of the forward (backward) procedure is $O(t_{max}(|\mathcal{N}||\mathcal{S}|^2 + |\mathcal{N}|^2|\mathcal{S}|))$ where the two terms correspond to the size of the two cliques for inference in the 2-slice DBN after O and A are eliminated. The complexity of computing the expectations from α and β is $O(|\mathcal{N}||\mathcal{A}|(|\mathcal{S}|^2 + |\mathcal{S}||\mathcal{O}|) + |\mathcal{N}|^2|\mathcal{S}||\mathcal{O}|)$, which corresponds to the clique sizes of the 2-slice DBN including O and A.

In comparison, 2-level hierarchical and factored controllers with $|\mathcal{N}^{top}| = |\mathcal{N}^{base}| = |\mathcal{N}|^{0.5}$ nodes at each level have fewer parameters and a smaller complexity, but also a smaller policy space due to the structure imposed by the hierarchy/factorization. While there is a tradeoff between policy space and complexity, hierarchical and factored controllers are often advantageous in practice since they can find more quickly a good hierarchical/factored policy when there exists one.

A 2-level factored controller with $|\mathcal{N}|^{0.5}$ nodes at each level has $2|\mathcal{O}||\mathcal{N}|^{1.5}$ parameters for $p_{n'^{top}|o'n^{base}n^{top}}$ and $p_{n'^{base}|n'^{top}o'n^{base}}$, and $|\mathcal{A}||\mathcal{N}|^{0.5}$ parameters for $p_{a|n^{base}}$. The complexity of the forward (backward) procedure is $O(t_{max}(|\mathcal{N}||\mathcal{S}|^2 + |\mathcal{N}|^{1.5}|\mathcal{S}|))$ and the complexity of computing the expectations is $O(|\mathcal{N}||\mathcal{A}|(|\mathcal{S}|^2 + |\mathcal{S}||\mathcal{O}|) + |\mathcal{N}|^{1.5}|\mathcal{O}||\mathcal{S}| + |\mathcal{N}|^2|\mathcal{O}|)$. A 2-level hierarchical controller is further restricted and therefore has fewer parameters, but the same time complexity.

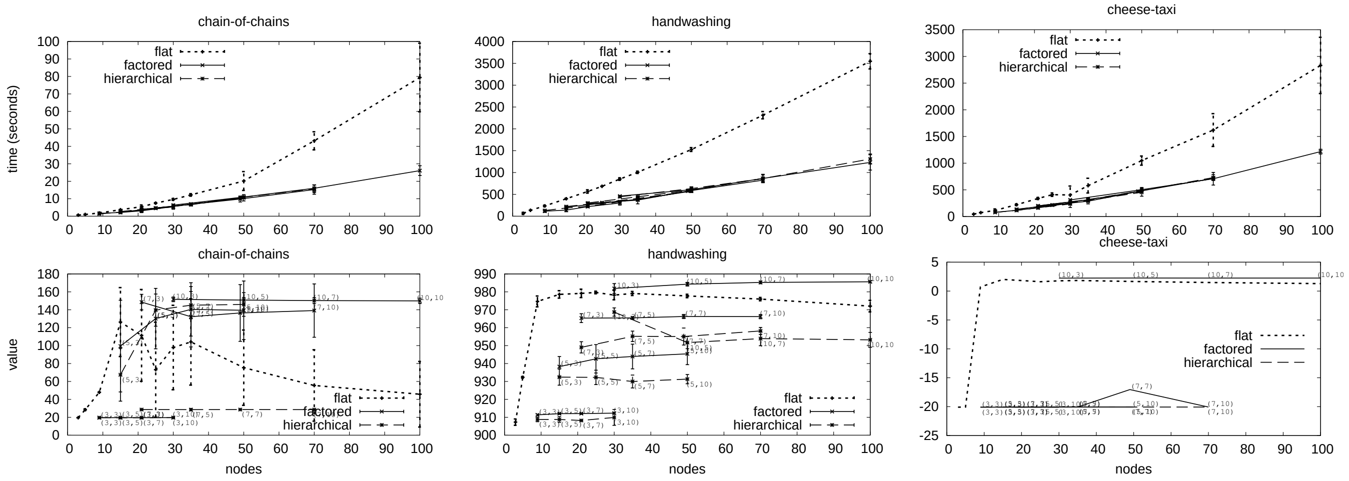
4 Experiments

We first compared the performance of the maximum likelihood (ML) approach to previous optimization-based approaches from (Charlin, Poupart, & Shioda 2006). Table 1 summarizes the results for 2-layer controllers with certain combinations of $|\mathcal{N}^{base}|$ and $|\mathcal{N}^{top}|$. The problems include paint, shuttle and 4x4 maze (previously used in (Charlin, Poupart, & Shioda 2006)) and three additional problems: chain-of-chains (described below), hand-washing (reduced version from (Hoey *et al.* 2007)) and cheese-

Table 1: V^* denotes the optimal value, except for handwashing and cheese-taxi, where we show both the optimal value of the equivalent fully-observable problem as well as best values obtained by point-based value iteration (pbvi). The ML approach optimizes a factored controller for 200 EM iterations with a planning horizon of $t_{max} = 100$. (5,3) nodes means $|\mathcal{N}^{base}| = 5$ and $|\mathcal{N}^{top}| = 3$. For cheese-taxi, we get a maximum value of 2.25. N/A indicates that the solver did not complete successfully.

Problem	$\|\mathcal{S}\|$	$\|\mathcal{A}\|$	$\|\mathcal{O}\|$	V^*	Best results (Charlin et al., 2006)			ML approach (avg. over 10 runs)		
					nodes	time	V	nodes	time (secs)	V
paint	4	4	2	3.3	(1,3)	<1	3.29	(5,3)	0.96 ± 0.26	3.26 ± 0.004
shuttle	8	3	5	32.7	(1,3)	2	31.87	(5,3)	2.812 ± 0.2	31.6 ± 0.5
4x4 maze	16	4	2	3.7	(1,2)	30	3.73	(3,3)	2.8 ± 0.8	$3.72 \pm 8e{-}5$
chain-of-chains	10	4	1	157.07			N/A	(10,3)	6.4 ± 0.2	151.6 ± 2.6
handwashing	84	7	12	$\leqslant$1052 (pbvi:981)			N/A	(10,5)	655 ± 2	984 ± 1
cheese-taxi	33	7	10	$\leqslant$5.3 (pbvi:2.45)			N/A	(10,3)	311 ± 14	$-9 \pm 11 (2.25^*)$

Table 2: Optimization time and values depending on the number of nodes (200 EM iterations with $t_{max} = 100$ planning steps). The results are averaged over 10 runs with error bars of ±1 standard deviation. (5,3) nodes means $|\mathcal{N}^{base}| = 5$ and $|\mathcal{N}^{top}| = 3$.



taxi (variant from (Pineau 2004)). On the first three problems, ML reaches the same values as the optimization-based approaches, but with larger controllers. We attribute this to EM's weaker ability to avoid local optima than the optimization-based approaches. However, the optimization-based approaches run out of memory on the last three problems (memory exceeds 2 Gb of RAM), while ML scales gracefully (as analyzed in Sect. 3.2). The ML approach demonstrates that hierarchy discovery can be tractable.

The next experiment demonstrates that policy optimization while discovering a hierarchy can be done faster and/or yield higher value when there exists good hierarchical policies. Table 2 compares the performance when optimizing flat, hierarchical and factored controllers on chain-of-chains, hand-washing and cheese-taxi. The factored and hierarchical controllers have two levels and correspond respectively to the DBNs in Fig. 3a and 3b. The x-axis is the number of nodes for flat controllers and the product of the number of nodes at each level for hierarchical and factored controllers. Taking the product is justified by the fact that the equivalent flat controllers of some hierarchical/factored controllers require that many nodes. The graphs in the top row of Table 2 demonstrate that hierarchical and factored controllers can be optimized faster, confirming the analysis done in Sect. 3.2. There is no difference in computational complexity between the strictly hierarchical and unconstrained factored architectures. Recall however that the efficiency gains of the hierarchical and factored controllers are obtained at the cost of a restricted policy space. Nevertheless, the graphs in the bottom row of Table 2 suggest that hierarchical/factored controllers can still find equally good policies when there exist one. Factored controllers are generally the most robust. With a sufficient number of nodes, they find the best policies on all three problems. Note that factored and hierarchical controllers need at least a number of nodes equal to the number of actions in the base layer in order to represent a policy that uses all actions. This explains why hierarchical and factored controllers with less than 4 base nodes (for chain-of-chains) and 7 base nodes (for hand-washing and cheese-taxi) do poorly. The optimization of flat controllers tend to get stuck in local optima if too many nodes are used. Comparing the unconstrained factored architecture versus hierarchical, we find that the additional constraints in the hierarchical controller make the optimization problem harder although there are less parameters to optimize. As a result, EM gets stuck more often in local optima.

We also examine whether learnt hierarchies make intuitive sense. Good policies for the cheese-taxi and hand-

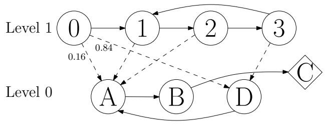


Figure 4: Hierarchical controller learnt for the chain-of-chains. The diamond indicates an exit node, for which $\hat{p}_{e^0|n^0} = 1$.

washing problems can often be represented hierarchically, however the hierarchical policies found didn't match hierarchies expected by the authors. Since these are non-trivial problems for which there may be many ways to represent good policies in a hierarchical fashion that is not intuitive, we designed the chain-of-chains problem, which is much simpler to analyze. The optimal policy of this problem consists of executing n times the same chain of n actions followed by a submit action to earn the only reward. The optimal policy requires n^2+1 nodes for flat controllers and $n+1$ nodes at each level for hierarchical controllers. For $n = 3$, ML found a hierarchical controller of 4 nodes at each level, illustrated in Fig. 4. The controller starts in node 0. Nodes at level 1 are abstract and descend into concrete nodes at level 0 by following the dashed edges. Control is returned to level 1 when an end node (denoted by a diamond) is reached. The optimal policy is to do A-B-C three times followed by D. Hence, a natural hierarchy would abstract A-B-C and D into separate subcontrollers. While the controller in Fig. 4 is not completely optimal (the vertical transition from abstract node 0 should have probability 1 of reaching A), it found an equivalent, but less intuitive abstraction by having subcontrollers that do A-B-C and D-A-B-C. This suggests that for real-world problems there will be many valid abstractions that are not easily interpretable and the odds that an automated procedure finds an intuitive hierarchy without any guidance are slim.

5 Conclusion

The key advantage of maximum likelihood is that it can exploit the factored structure in a controller architecture. This facilitates hierarchy discovery when the hierarchical structure of the controller is encoded into a corresponding dynamic Bayesian network (DBN). Our complexity analysis and the empirical run time analysis confirm the favorable scaling. In particular, we solved problems like handwashing and cheese-taxi that could not be solved with the previous approaches in (Charlin, Poupart, & Shioda 2006). Compared to flat controllers, factored controllers are faster to optimize and less sensitive to local optima when they have many nodes. Our current implementation does not exploit any factored structure in the state, action and observation space, however we envision that a factored implementation would naturally scale to large factored POMDPs.

For the chain-of-chains problem, maximum likelihood finds a reasonable hierarchy. For other problems like handwashing, there might be many hierarchies and the one found by our algorithm is usually hard to interpret. We cannot expect our method to find a hierarchy that is human readable. Interestingly, although the strictly hierarchical architectures have less parameters to optimize, they seem to be more susceptible to local optima as compared to a factored but otherwise unconstrained controller. Future work will need to investigate different heuristics to escape local optima during optimization.

In this paper, we made explicit assumptions about the structure – we prefixed the structure of the DBN to mimic a strict hierarchy or a level-wise factorization and we had to fix the number of nodes in each level. However, the DBN framework allows us to build on existing methods for structure learning of graphical models. A promising extension would be to use such structure learning techniques to optimize the factored structure of the controller. Since the computational complexity for evaluating (training) a single structure is reasonable, techniques like MCMC could sample and evaluate a variety of structures. This variety might also help to circumvent local optima, which currently define the most dominant limit of our approach.

References

Amato, C.; Bernstein, D.; and Zilberstein, S. 2007. Solving POMDPs using quadratically constrained linear programs. In *IJCAI*, 2418–2424.

Braziunas, D., and Boutilier, C. 2004. Stochastic local search for POMDP controllers. In *AAAI*, 690–696.

Charlin, L.; Poupart, P.; and Shioda, R. 2006. Automated hierarchy discovery for planning in partially observable environments. In *NIPS*, 225–232.

Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* 39(1):1–38.

Hansen, E., and Zhou, R. 2003. Synthesis of hierarchical finite-state controllers for POMDPs. In *ICAPS*, 113–122.

Hansen, E. 1998. An improved policy iteration algorithm for partially observable MDPs. In *NIPS*.

Hoey, J.; von Bertoldi, A.; Poupart, P.; and Mihailidis, A. 2007. Assisting persons with dementia during handwashing using a partially observable Markov decision process. In *ICVS*.

Meuleau, N.; Peshkin, L.; Kim, K.-E.; and Kaelbling, L. 1999. Learning finite-state controllers for partially observable environments. In *UAI*, 427–436.

Murphy, K., and Paskin, M. 2001. Linear time inference in hierarchical HMMs. In *NIPS*.

Neal, R., and Hinton, G. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M. I., ed., *Learning in Graphical Models*. Kluwer.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Policy-contingent abstraction for robust robot control. In *UAI*, 477–484.

Pineau, J. 2004. *Tractable Planning Under Uncertainty: Exploiting Structure*. Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University.

Poupart, P., and Boutilier, C. 2003. Bounded finite state controllers. In *NIPS*.

Theocharous, G.; Murphy, K.; and Kaelbling, L. P. 2004. Representing hierarchical POMDPs as DBNs for multi-scale robot localization. In *ICRA*, 1045–1051. IEEE.

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