Journal of Artificial Intelligence Research 53 (2015) 223-270 Submitted 12/14; published 06/15

Probabilistic Inference Techniques for Scalable Multiagent Decision Making

Akshat Kumar AKSHATKUMAR@SMU.EDU.SG
School of Information Systems
Singapore Management University, Singapore

Shlomo Zilberstein SHLOMO@CS.UMASS.EDU
College of Information and Computer Sciences
University of Massachusetts, Amherst, USA

Marc Toussaint MARC.TOUSSAINT@INFORMATIK.UNI-STUTTGART.DE
Department of Computer Science
University of Stuttgart, Germany

Abstract

Decentralized POMDPs provide an expressive framework for multiagent sequential decision making. However, the complexity of these models—NEXP-Complete even for two agents—has limited their scalability. We present a promising new class of approximation algorithms by developing novel connections between multiagent planning and machine learning. We show how the multiagent planning problem can be reformulated as inference in a mixture of dynamic Bayesian networks (DBNs). This planning-as-inference approach paves the way for the application of efficient inference techniques in DBNs to multiagent decision making. To further improve scalability, we identify certain conditions that are sufficient to extend the approach to multiagent systems with dozens of agents. Specifically, we show that the necessary inference within the expectation-maximization framework can be decomposed into processes that often involve a small subset of agents, thereby facilitating scalability. We further show that a number of existing multiagent planning models satisfy these conditions. Experiments on large planning benchmarks confirm the benefits of our approach in terms of runtime and scalability with respect to existing techniques.

1. Introduction

Decentralized partially observable MDPs (Dec-POMDPs) have emerged in recent years as a prominent framework for modeling sequential decision making by a team of collaborating agents (Bernstein, Givan, Immerman, & Zilberstein, 2002). Their expressive power makes it possible to tackle coordination problems in which agents must act based on different partial information about the environment and about each other so as to maximize a global reward function. Applications of Dec-POMDPs include coordinating the operation of planetary exploration rovers (Becker, Zilberstein, Lesser, & Goldman, 2004), coordinating firefighting robots (Oliehoek, Spaan, & Vlassis, 2008), target tracking by a team of sensor agents (Nair, Varakantham, Tambe, & Yokoo, 2005) and improving throughput in wireless networks (Pajarinen, Hottinen, & Peltonen, 2014).

While there has been rapid progress with exact algorithms for Dec-POMDPs (Oliehoek, Spaan, Amato, & Whiteson, 2013), such optimal solutions can only be obtained for rela-

tively smaller problems. In terms of computational complexity, optimally solving a finite-horizon Dec-POMDP is NEXP-Complete (Bernstein et al., 2002). In contrast, finite-horizon POMDPs are PSPACE-complete (Mundhenk, Goldsmith, Lusena, & Allender, 2000), a strictly lower complexity class that highlights the difficulty of solving Dec-POMDPs.

1.1 Related Work

For the finite-horizon case, a substantial number of promising point-based approximate algorithms have been developed (Kumar & Zilberstein, 2010b; Wu, Zilberstein, & Chen, 2010; Dibangoye, Mouaddib, & Chaib-draa, 2009; Kumar & Zilberstein, 2009a; Seuken & Zilberstein, 2007). However, unlike their point-based counterparts for POMDPs (Pineau, Gordon, & Thrun, 2006; Smith & Simmons, 2004), they cannot be easily adopted for the infinite-horizon case due to a variety of reasons. For example, POMDP algorithms represent the policy as compact α-vectors, whereas most Dec-POMDP algorithms explicitly store the policy as a mapping from observation sequences to actions, making them unsuitable for the infinite-horizon case. This has motivated the development of alternative approaches, approximating factored finite-horizon Dec-POMDPs by a series of collaborative graphical Bayesian games (Oliehoek, Whiteson, & Spaan, 2013) or using genetic algorithms (Eker & Akin, 2013).

Recently, a number of approaches have been developed that transform a Dec-POMDP into a continuous-state MDP and then use techniques from the POMDP literature to solve the continuous-state MDP (Dibangoye, Amato, Doniec, & Charpillet, 2013a; Dibangoye, Amato, Buffet, & Charpillet, 2013b). The state in such a continuous MDP reformulation of a Dec-POMDP, also called occupancy state, is the probability distribution over the world state and the history of observations each agent has received. Despite the adoption of efficient POMDP techniques to such a reformulation, a drawback of the approach is that the size of observation histories increases exponentially with respect to the plan horizon. In contrast, our approach uses the notion of finite-state controllers (FSCs) that summarize relevant features of the planning problem. Often, compact FSCs can provide a good approximation for large planning problems. The occupancy state based formulation has also been applied to infinite-horizon problems by converting an infinite-horizon problem to an approximate finite-horizon version. This is done by using the future reward discount factor to derive some finite-horizon H after which the remaining rewards have a negligible contribution to the overall value function (Dibangoye, Buffet, & Charpillet, 2014). A drawback of the truncated horizon approach is that, for high discount factors, the required horizon H can be prohibitively large.

In terms of solution representation, most algorithms for infinite-horizon problems represent agent policies as finite-state controllers (Amato, Bernstein, & Zilberstein, 2010; Bernstein, Amato, Hansen, & Zilberstein, 2009), unlike algorithms for finite-horizon problems that often use policy trees (Hansen, Bernstein, & Zilberstein, 2004). The resulting solution is approximate because of the limited memory of the controllers and because optimizing the action selection and transition parameters is extremely hard. Previous approaches to optimize finite-state controller based policies include decentralized bounded policy iteration (DEC-BPI) (Bernstein et al., 2009) and a technique based on non-linear programming (NLP) (Amato et al., 2010). The DEC-BPI algorithm uses a linear programming formula-

tion to improve the parameters of one node of one finite-state controller at a time. That is, it fixes the parameters of all the nodes of all the controllers, except a single node of a particular agent. Then it uses a linear program to find better action selection and transition parameters for that particular node. This LP guarantees that the policy value is increased for every belief state. A major drawback of this scheme is that it is not designed to optimize the value of a particular belief state. Therefore, to produce a good policy, DEC-BPI may need a large number of controller nodes, which reduces the effectiveness of the LP formulation.

In contrast to the DEC-BPI approach, the NLP formulation of Amato et al. (2010) can optimize the controllers for a given initial belief state. This formulation has a linear objective function. However, the Bellman constraints, which involve additional variables representing the value of each node, are nonlinear and non-convex in all the variables. This can cause the NLP solver to get stuck in a local optimum. Furthermore, empirically we observed that the performance of the state-of-the-art NLP solvers such as SNOPT (Gill, Murray, & Saunders, 2002) degrades quickly even with a moderate increase in the number of nonlinear constraints. This highlights the challenges presented by scaling the NLP approach for larger ($\gg 2$ agents) problems. A complementary research direction has been to investigate the kind of structure in a controller that can enable better quality solutions. For example, layered controllers can be developed for POMDPs and Dec-POMDPs, that are then optimized by point based approaches (Pajarinen & Peltonen, 2011b). The EM based planning algorithms we develop in our work can also take advantage of such controller structures. There are other approaches to compute policies for infinite-horizon Dec-POMDPs that are not based on a controller representation of the joint-policy (MacDermed & Isbell, 2013). However, a key advantage of policies based on finite-state controllers is their ease of execution in resource constrained environments (Grzes, Poupart, & Hoey, 2013; Grześ, Poupart, Yang, & Hoey, 2015), without any expensive belief update operations required in other approaches. Furthermore, policies represented as finite-state controllers can carry more semantic information, where each controller node summarizes some relevant aspects of the observation history.

Generalizing Dec-POMDP algorithms to more than two agents has been a persistent challenge from both problem representation and algorithmic viewpoints. Many recent attempts to increase the scalability of planners with respect to the number of agents impose restrictions on the form of interaction among the agents. Examples of such restrictions include transition independence (Becker et al., 2004; Nair et al., 2005), weak-coupling among agents that is limited to certain states (Varakantham, Kwak, Taylor, Marecki, Scerri, & Tambe, 2009), and directional transition dependence (Witwicki & Durfee, 2010). One of the earliest models that demonstrated scalability by limiting interactions among agents was transition-independent Dec-MDP (TI-Dec-MDP) (Becker, Zilberstein, Lesser, & Goldman, 2003). Agents in these models can fully affect and observe their local state, but they cannot affect the local states or observations of other agents. The dependence among agents is limited to the joint reward function. This restricted model has been shown to be NP-Complete by Goldman and Zilberstein (2004), who also conducted a detailed complexity analysis of different subclasses of Dec-POMDPs with various limitations on agent interactions. An extension of TI-Dec-MDP was proposed by Becker, Zilberstein, and Lesser (2004), introducing structured *transition dependencies*.

The TI-Dec-MDP model was extended to handle partial observability in a problem instance by Nair et al. (2005), resulting in a model called network-distributed POMDP (ND-POMDP). It was shown that the value function of ND-POMDPs factorizes among smaller sub-groups of agents based on immediate reward decomposability. This property was used to develop a number of algorithms that scale relatively well with respect to the number of agents (Nair et al., 2005; Varakantham, Marecki, Yabu, Tambe, & Yokoo, 2007; Marecki, Gupta, Varakantham, Tambe, & Yokoo, 2008). Another restricted class of models is the transition-decoupled POMDP (TD-POMDP) (Witwicki & Durfee, 2010). This model explicitly differentiates between an agent's private state that can only be affected by the agent's local action and shared states that can be affected by other agents. Structured interactions have also been used for deciding how to communicate among agents (Mostafa & Lesser, 2009, 2011). Within the framework of multiagent MDPs (MMDPs), the submodularity property of the value function for a class of sensor network planning problems has been exploited by Kumar and Zilberstein (2009b).

By and large, the above models try to identify restrictions on agent interactions that facilitate scalable planning. With the exception of the work of Witwicki and Durfee (2011), there has not been much work towards a general characterization of conditions under which multiagent planning can be made scalable. Witwicki and Durfee provide a characterization of weak-coupling among agents similar to our work. However, a significant difference is that we propose a concrete algorithm that can efficiently exploit such weak-coupling among agents to enable scalability to large multiagent systems. The algorithmic outline presented by Witwicki and Durfee to exploit weak interactions among agents by mapping the policy optimization problem to constraint optimization is not scalable as it involves enumerating the policy space of agents.

This article extends two conference papers published at UAI'10 (Kumar & Zilberstein, 2010a) and IJCAI'11 (Kumar, Zilberstein, & Toussaint, 2011) with more detailed background, new theorems and proofs, simplified EM derivations, and detailed derivation of the EM updates for the ND-POMDP model. The EM based policy optimization approach that we developed is the foundation of several other efforts that explore different aspects of multiagent planning. For example, Wu, Zilberstein, and Jennings (2013) developed sampling-based E-step inference techniques. This technique is particularly useful in the model-free setting in which the underlying model is not known. Pajarinen and Peltonen (2011a) extend our EM approach to factored Dec-POMDPs. They present an approximation of the E-step using a factored representation of the forward and backward messages as defined in Section 4.4. Their approach increases the scalability of the EM algorithm w.r.t. the number of agents and other problem parameters at the expense of making the EM algorithm approximate. That is, the E and M steps are approximate and may lead to non-monotonic improvement in the policy value. Pajarinen and Peltonen (2013) further extend the EM approach to average reward Dec-POMDPs.

The techniques we present in this article differ from such previous approaches in that rather than approximating the computationally challenging inference for larger multiagent systems, we explore some natural ways to *decompose* the inference process and produce an exact, yet scalable EM algorithm. We further show that the necessary conditions for such value factorization based decomposition are satisfied in several existing planning models, confirming the generality of the developed framework.

1.2 Summary of Contributions

We present a promising new class of algorithms that combines planning with probabilistic inference and opens the door to the application of rich inference techniques to solving infinite-horizon Dec-POMDPs. Our approach is based on Toussaint *et. al.*'s approach of transforming a single-agent planning problem into its equivalent mixture of dynamic Bayes nets (DBNs) and using likelihood maximization in this framework to optimize the policy value (Toussaint & Storkey, 2006; Toussaint, Harmeling, & Storkey, 2006). Such approaches have been successful in solving MDPs and POMDPs (Toussaint et al., 2006) and they easily extend to factored or hierarchical structures (Toussaint, Charlin, & Poupart, 2008). Furthermore, they can handle continuous action and state spaces thanks to advanced probabilistic inference techniques (Hoffman, Kueck, de Freitas, & Doucet, 2009b). We show how Dec-POMDPs, which are much harder to solve than MDPs or POMDPs, can also be reformulated as a mixture of DBNs. We then present an Expectation Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977) to maximize the reward likelihood and the policy value in this framework. This approach offers an attractive anytime algorithm because EM improves the likelihood—and hence the policy value—with each iteration. Our experiments on benchmark domains show that EM compares favorably against previous approaches to optimize FSCs, DEC-BPI and NLP-based optimization.

To address scalability with respect to the number of agents, we identify conditions based on *value function factorization* that are sufficient to make planning amenable to a scalable approximation. This is achieved by constructing a graphical model that exploits the locality of interactions among the agents. We show how to efficiently extend the likelihood maximization paradigm to this generalized graphical model. Furthermore, we show that the necessary inference can be decomposed into processes that involve a small subset of agents—according to the interaction graph—thereby facilitating scalability. We derive a global update rule that combines these local inferences to monotonically increase the overall solution quality. Several existing multiagent planning models are shown to satisfy these conditions, thereby validating the generality of the value factorization property. A further benefit of our approach is that it is amenable to a massively parallel implementation, since it relies on local computations and message-passing among neighboring agents.

Experiments on a large multiagent planning benchmark, with joint state and joint action spaces up to 5^{22}, 3^{20} respectively, confirm that our approach is scalable with respect to the number of agents and can provide good quality solutions for planning problems for up to 20 agents, which cannot be handled by the best existing approaches. For smaller multiagent systems, our approach provides better solution quality and is about an order-of-magnitude faster than the previous best nonlinear programming approach for optimizing FSCs.

2. Decentralized POMDP Model

The Dec-POMDP model is a natural extension of MDPs and POMDPs. It can be defined by a tuple $\langle I, S, \{A^i\}, P, R, \{Y^i\}, O, \gamma \rangle$, where:

- I denotes a finite set of n agents
- S denotes a finite set of states with designated initial state distribution η_0

- A^i denotes a finite set of actions for each agent i
- P denotes state transition probabilities: $P(s'|s,\vec{a})$, the probability of transitioning from state s to s' when the joint-action $\vec{a}$ is taken by the agents
- R denotes the reward function: $R(s,\vec{a})$ is the immediate reward for being in state s and joint-action taken as $\vec{a}$
- Y^i denotes a finite set of observations for each agent i
- O denotes the observation probabilities: $O(\vec{y}|s',\vec{a})$ is the probability of receiving the joint-observation $\vec{y}$ when the last joint-action taken was $\vec{a}$ that resulted in the environment state being s'
- γ denotes the reward discounting factor

An agent i's policy, $\boldsymbol{\theta}^i : \bar{Y}^i \rightarrow A^i$, maps the set of all possible observation histories $\bar{Y}^i$ to actions. Solving a Dec-POMDP entails finding the joint-policy $\boldsymbol{\theta} = \langle \boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^n \rangle$ that maximizes the total expected reward.

$$\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R\big(s_t, \vec{a}_t; \boldsymbol{\theta}\big)\Big] \tag{1}$$

where $\boldsymbol{\theta}$ denotes the joint-policy and subscript t denotes the dependence on time. In a Dec-POMDP, agents are acting under uncertainty not only about the underlying environment state but also about each other. Although the joint-observation received by agents may be correlated, each agent only observes its own component of the joint-observation. This makes the coordination problem particularly challenging. In Section 2.1, we show a compact representation of the policy as finite-state controllers, rather than long sequences of observations.

When there are two agents in a given multiagent system, we adopt a simplified notation as follows. The action set of agent 1 is denoted by $a \in A$ and of agent 2 by $b \in B$. The state transition probability $P(s'|s,a,b)$ depends upon the actions of both the agents. Upon taking the joint-action $\langle a,b \rangle$ in state s, agents receive the joint-reward $R(s,a,b)$. Y is the finite set of observations for agent 1 and Z for agent 2. $O(yz \mid s,a,b)$ denotes the probability $P(y,z|s,a,b)$ of agent 1 observing $y \in Y$ and agent 2 observing $z \in Z$ when the joint-action $\langle a,b \rangle$ was taken and resulted in state s. For infinite-horizon Dec-POMDPs, a reward discounting factor $\gamma < 1$ is used.

2.1 Finite State Controllers

In the case of infinite-horizon Dec-POMDPs, agents operate continuously with the immediate reward R being discounted by a factor $\gamma < 1$. Representing agents' local policies using an explicit tree structured representation is not feasible in this case. Therefore, agents' policies are represented as cyclic finite-state controllers (FSC).

We represent each agent's policy as a bounded, *finite state controller* (FSC). This approach has been used successfully for both POMDPs (Poupart & Boutilier, 2003; Amato, Bernstein, & Zilberstein, 2007) and Dec-POMDPs (Amato et al., 2010). In this case, each

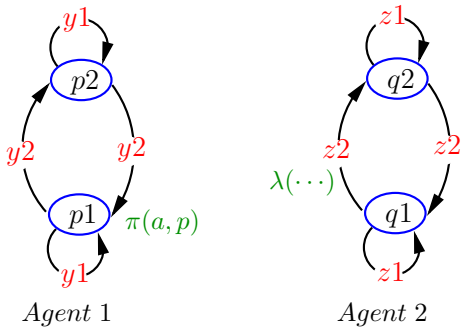


Figure 1: A two-agent infinite-horizon joint-policy represented using finite-state controllers. Each node is a memory state. Edges represent the node transition function. Agent 1 has two observations, $y1$ and $y2$. Agent 2 also has two observations, $z1$ and $z2$.

agent i has a finite internal memory state, $q^i \in Q^i$, which summarizes the *crucial* information obtained from past observations to support efficient action selection. The size of the set Q^i determines the expressiveness of the FSC based policy. For POMDPs, FSCs are beneficial due to their compactness and relative ease of policy execution compared to the full belief over world states. In Dec-POMDPs, belief over world states cannot be maintained during the execution time due to lack of availability of joint-observations. Therefore, FSCs are particularly useful as executing FSC-based policies does not require maintaining belief over world states.

The FSC of the ith agent is parameterized by $\theta^i = (\pi^i, \lambda^i, \nu^i)$ as explained below.

- An agent chooses actions depending on its internal state q: $P(a|q;\pi) = \pi_{a,q}$.
- The internal state is updated with each new observation, by the node transition function: $P(q'|q, y; \lambda) = \lambda_{q',q,y}$.
- Finally, ν_{q_0} is the initial node distribution $P(q_0)$ for each agent.

Figure 1 shows the structure of such controllers for two agents. Both the action selection parameter π and the node transition parameter λ could be deterministic or stochastic. We optimize stochastic controllers in this work because they can generally produce higher values (Poupart & Boutilier, 2003).

Figure 2 shows the complete DDN representation of a two-agent Dec-POMDP depicting how the environment and agents' policies interact with each other. We use the convention that subscripts denote time and superscripts, if any, identify agents.

2.2 Objective Function

Considering again the two-agent case, we denote the controller nodes for agent 1 by p and agent 2 by q. The value for starting the controllers in nodes $\langle p, q\rangle$ at state s is given by:

$$V(p,q,s) = \sum_{a,b} \pi_{a,p}\pi_{b,q} \left[R(s,a,b) + \gamma \sum_{s'} P(s' \mid s,a,b) \sum_{y,z} O(yz \mid s',a,b) \sum_{p'q'} \lambda_{p',p,y}\lambda_{q',q,z} V(p',q',s')\right]$$

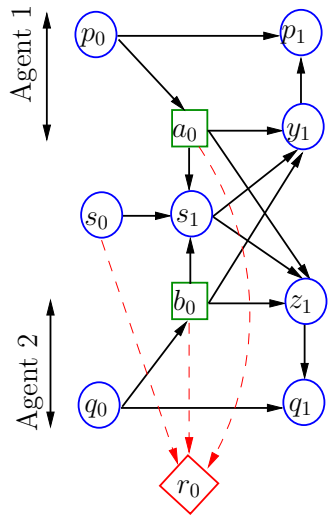


Figure 2: A two-time slice dynamic decision network (DDN) representation of a two-agent Dec-POMDP. All nodes are random variables. Square nodes represent decisions; diamond nodes represent the reward; p and q represent the states of policy for agent 0 and 1 respectively; subscripts denote time.

The goal is to set the parameters $\langle \pi, \lambda, \nu \rangle$ of the agents' controllers (of some given size) that maximize the expected discounted reward for the initial belief η_0:

$$V(\eta_0) = \sum_{p,q,s} \nu_p \nu_q \eta_0(s) V(p, q, s)$$

3. Dec-POMDPs as Mixture of DBNs

In this section, we describe how Dec-POMDPs can be reformulated as a mixture of DBNs, such that maximizing the reward likelihood defined below is equivalent to optimizing the joint-policy. Our approach is based on the framework proposed by Toussaint et al. (2006) and Toussaint and Storkey (2006) to solve Markovian planning problems using probabilistic inference. In this section, we develop the *planning-as-inference* strategy for two-agent Dec-POMDPs and later extend it to multiple ($\gg 2$) agents. The previous approach of Toussaint et al. (2006) and Toussaint and Storkey (2006) focused on single agent MDPs and POMDPs. In our work, we develop new mixture models that allow us to extend the planning-as-inference paradigm to multiple agents, a significant generalization of the single agent case. First, we briefly describe the intuition behind this reformulation and then we describe in detail the necessary modifications required for Dec-POMDPs.

A Dec-POMDP can be described using a single DBN where the reward is emitted at *each time step*, such as the unrolled DBN corresponding to the one shown in Figure 2. However, in our approach, it is described by an infinite mixture of a special type of DBNs where the reward is emitted *only at the end*. There is one mixture component for each time period from $T = 0$ to ∞. Furthermore, to simulate the discounting of rewards, the probability of T, which acts as mixture weight, is set as $P(T = t) = \gamma^t(1 - \gamma)$. This also

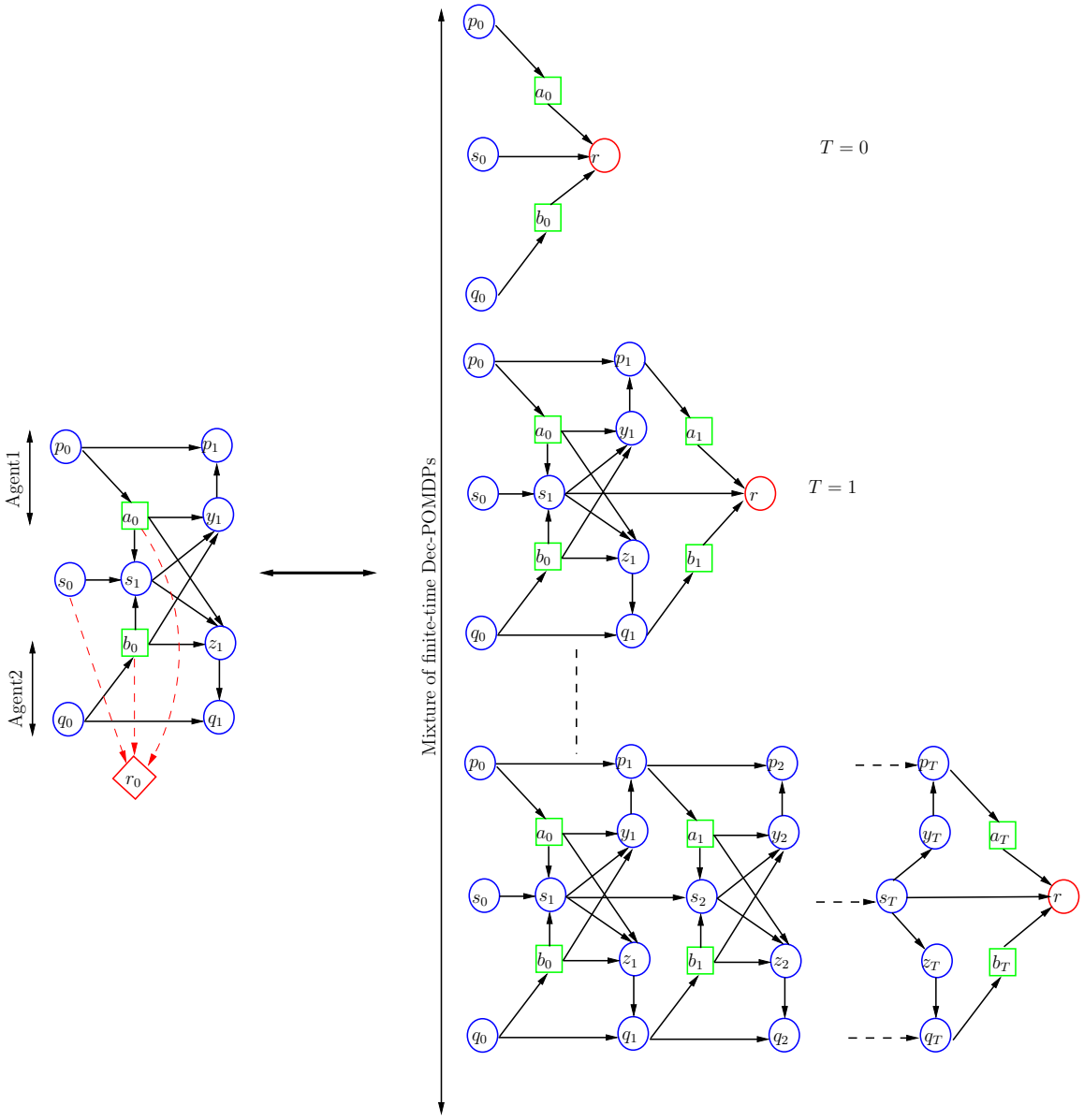


Figure 3: The time-dependent DBN mixture (right) corresponding to the two-agent Dec-POMDP (left). The first DBN component in the mixture corresponds to the reward for time step 1, second DBN corresponds to the reward for time step 2. The last DBN in the mixture shows the general structure of a T-step DBN.

ensures that the mixing weights are normalized or $\sum_{t=0}^{\infty} P(T=t)=1$. We now describe the structure of each mixture component of a single T-step DBN.

The first DBN in the DBN mixture model shown in Figure 3 describes the DBN for time $T = 0$. The key intuition is that for the reward emitted at any time step T, we have a separate DBN with the general structure as shown in the last T-step DBN shown in Figure 3. Such a DBN shows how the reward obtained at time step T depends on policy parameters and the underlying Dec-POMDP model.

The random variable r shown in the DBN mixture of Figure 3 is a binary variable with its conditional distribution (for any time T) described using the normalized immediate reward as:

$$\hat{R}_{sab} = P(r = 1|s_T = s, a_T = a, b_T = b) = (R_{sab} - R_{min})/(R_{max} - R_{min}).$$

The parameter R_{max} is the maximum reward for any state action pair for the given Dec-POMDP instance and R_{min} denotes the minimum reward. This scaling of the reward is the key to transforming the optimization problem from the realm of planning to likelihood maximization as stated below. Let $\boldsymbol{\theta}$ denote the joint parameters $\langle \pi, \lambda, \nu \rangle$ for each agent's controller.

Theorem 1. *By choosing the conditional probability of binary rewards r such that $\hat{R}_{sab} \propto R_{sab}$ and introducing the discounting time prior $P(T)=\gamma^T(1-\gamma)$, the joint-policy value $V^{\boldsymbol{\theta}}$ relates linearly to the likelihood $L^{\boldsymbol{\theta}}$ of observing the reward variable as 1:*

$$V^{\boldsymbol{\theta}} = \frac{(R_{max} - R_{min})L^{\boldsymbol{\theta}}}{(1-\gamma)} + \frac{R_{min}}{1-\gamma}$$

Proof. We have the value function defined as:

$$V^{\boldsymbol{\theta}} = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R\big(s_t, \vec{a}_t; \boldsymbol{\theta}\big)\Big] \tag{2}$$

Consider the T-step DBN in the mixture of Figure 3. We define the likelihood for this time step T DBN as follows:

$$L_T^{\theta} = P(r=1|T;\theta) \tag{3}$$

For the full mixture corresponding to a Dec-POMDP, we have:

$$L^{\theta} = \sum_T P(T) L_T^{\theta} = (1-\gamma) \sum_T \gamma^T P(r=1|T;\theta) \tag{4}$$

We already chose $P(r=1|s_T=s, a_T=a, b_T=b)=(R_{sab} - R_{min})/(R_{max} - R_{min})$. Therefore, using the construction of the T-step DBN, we have:

$$P(r=1|T;\theta) = \frac{\mathbb{E}\big[R(s_T, \vec{a}_T)\big] - R_{min}}{R_{max} - R_{min}} \tag{5}$$

Substituting back the above result in the expression for L^θ, we get:

$$L^\theta = (1-\gamma)\sum_T \gamma^T \frac{\mathbb{E}\big[R(s_T,\vec{a}_T\big] - R_{min}}{R_{max}-R_{min}} \tag{6}$$

$$= \frac{(1-\gamma)V^\theta - R_{min}}{R_{max}-R_{min}} \tag{7}$$

where we have used linearity of expectation such that

$$\sum_T \gamma^T \mathbb{E}\big[R(s_T,\vec{a}_T\big] = \mathbb{E}\big[\sum_T \gamma^T R(s_T,\vec{a}_T\big]$$

Combining the above result with the definition of the value function in Eq. (2), the theorem is proved. □

Using the above result, we establish the following result.

Lemma 1. *Maximizing the likelihood $L^{\boldsymbol{\theta}} = P(r=1;\boldsymbol{\theta})$ in the mixture of DBNs is equivalent to optimizing the Dec-POMDP policy.*

Theorem 1 and Lemma 1 show that the policy optimization problem can be reformulated as a parameter learning problem in a suitable DBN mixture. This immediately suggests using machine learning approaches to maximize the likelihood. Furthermore, this reformulation is lossless in the sense that optimizing the likelihood would exactly optimize the joint-policy value. We note that we never explicitly create the mixture of DBNs. All the computations on this DBN mixture can be implemented as message-passing over the original Dec-POMDP DDN of Figure 2. The only additional computational requirement is that of scaling the rewards using R_{max} and R_{min}, which can be done easily in linear time.

We next present the Expectation-Maximization (EM), a well known technique to maximize the likelihood.

4. Policy Optimization via Expectation Maximization

This section describes our application of the EM algorithm (Dempster et al., 1977) for maximizing the reward likelihood in the mixture of DBNs representing a Dec-POMDP. EM also possesses the desirable anytime characteristic as the likelihood (and the policy value which is proportional to the likelihood) is guaranteed to increase per iteration until convergence. We note that EM is not guaranteed to converge to a global optimum. However, in the experiments we show that EM almost always achieves similar values as the NLP based solver to optimize FSCs (Amato et al., 2010) and much better than DEC-BPI (Bernstein et al., 2009). Key potential advantages of using EM lie in its ability to easily generalize to much richer representations than currently possible for Dec-POMDPs such as hierarchical controllers (Toussaint et al., 2008), and continuous state and action spaces (Hoffman et al., 2009b). Another important advantage is the ability to generalize the solver to larger multiagent systems with more than 2 agents by exploiting the relative independence among agents, as we will show in later sections.

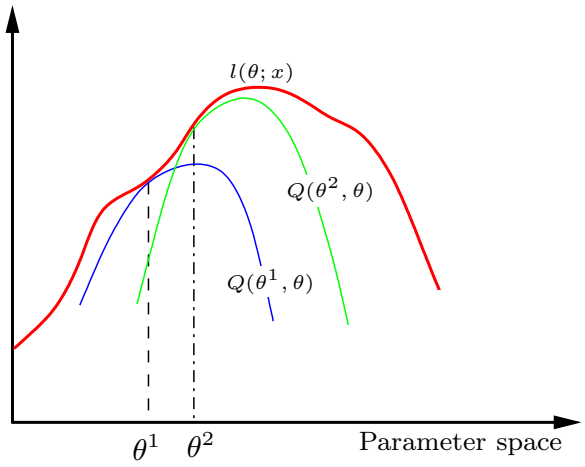


Figure 4: The coordinate ascent strategy of the EM algorithm

4.1 Overview of the EM Algorithm

We first provide a high level overview of the EM algorithm followed by detailing its adaptation to planning in Dec-POMDPs. The EM algorithm is a general approach to the problem of maximum likelihood (ML) parameter estimation in models with latent variables. In the given latent variable model, let X denote the observable variables and Z denote the hidden variables. Let θ denote the model parameters. The ML problem is to solve the following optimization problem:

$$\max_{\theta} l(\theta; x) = \max_{\theta} \log \sum_{z} p(x, z; \theta) \tag{8}$$

It is hard to optimize the above problem as the summation is inside the log. Furthermore, maximizing the log-likelihood $l(\theta; x)$ is generally a non-convex optimization problem as shown in Figure 4. Therefore, the EM algorithm iteratively performs coordinate ascent in the parameter space. First, the EM algorithm computes a lower bound for the function $l(\theta; x)$ such that this lower bound touches $l(\theta; x)$, say at point θ^1. This lower bound is denoted as $Q(\theta^1, \theta)$, and is defined as:

$$Q(\theta^1, \theta) = \sum_{z} p(z|x; \theta^1) \log p(x, z; \theta) - \sum_{z} p(z|x; \theta^1) \log p(z|x; \theta^1) \tag{9}$$

The last term in the above expression is the entropy of the variable $Z|x$. Figure 4 shows the lower bound $Q(\theta^1, \theta)$ by a blue curve. The function $Q(\theta^1, \theta)$ is also called *expected complete log-likelihood*. Importantly, the lower bound Q is *concave* in parameters θ and thus, can be optimized globally to provide a better parameter estimate θ^2. This process is also shown in Figure 4. Such coordinate ascent continues iteratively by defining a new lower bound $Q(\theta^2, \theta)$ at the point θ^2 as also shown in Figure 4, and then optimizing it to yield the next better parameter estimate until convergence.

To connect such an iterative maximization strategy with planning in Dec-POMDPs, we note that the likelihood function is directly proportional to the joint-policy value for policy

parameters θ (see Theorem 1). Therefore, for adapting the EM approach to Dec-POMDPs, we need to perform the following steps:

1. (E-step) Formulate the expected complete log-likelihood $Q(\theta^i, \theta)$ for the DBN mixture model in Figure 3 for iteration i
2. (M-step) Maximize $Q(\theta^i, \theta)$ w.r.t. θ to yield a better policy parameters θ^{i+1}
3. Repeat steps 1 and 2 until convergence, i.e., until $\theta^i \approx \theta^{i+1}$

We next explain how steps 1 and 2 can be implemented specifically for Dec-POMDPs.

4.2 Step 1: Formulating the Expected Log-Likelihood $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star)$

In the DBN mixture of Figure 3, every variable is hidden and the only observed data is the binary reward variable $r = 1$. For a particular DBN for time step T, such as the last DBN in the mixture of Figure 3, let $\tilde{L} = (\mathcal{P}, \mathcal{Q}, \mathcal{A}, \mathcal{B}, \mathcal{S}, \mathcal{Y}, \mathcal{Z})$ denote the latent variables for a T-step DBN, where each variable denotes a sequence of length T. That is, $\mathcal{P} = p_{0:T}$, denotes the T-step long sequence of controller state p_t of agent 1. EM maximizes the following expected complete log-likelihood for the Dec-POMDP DBN mixture. Let $\boldsymbol{\theta}$ denote the previous iteration's parameters and $\boldsymbol{\theta}^\star$ denotes new parameters. The expected log-likelihood (ignoring the entropy term independent of $\theta^\star$) is given as:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{T=0}^{\infty} \sum_{\tilde{L}} P(r{=}1, \tilde{L}, T; \boldsymbol{\theta}) \log P(r{=}1, \tilde{L}, T; \boldsymbol{\theta}^\star) \tag{10}$$

In the rest of the section, all the derivations refer to the general T-step DBN structure as shown in Figure 3. We also adopt the convention that for any random variable v, v' refers to the next time slice and $\bar{v}$ refers to the previous time slice. For any group of variables $\mathbf{v}$, $P_t(\mathbf{v}, \mathbf{v}')$ refers to $P(\mathbf{v}_t = \mathbf{v}, \mathbf{v}_{t+1} = \mathbf{v}')$. The joint probability of all the variables is:

$$\begin{aligned} P(r = 1, \tilde{L}, T; \theta) = P(T)\big[\hat{R}_{sab}\big]_{t=T} \prod_{t=1}^{T} \big[\pi_{ap}\pi_{bq} P_{s\bar{s}\bar{a}\bar{b}} O_{yzs\bar{a}\bar{b}} \lambda_{p\bar{p}y} \lambda_{q\bar{q}z}\big]_t \\ \cdot \big[\pi_{ap}\pi_{bq}\nu_p\nu_q\eta_0(s)\big]_{t=0} \end{aligned} \tag{11}$$

where brackets indicate the time slices, i.e., $\big[\hat{R}_{sab}\big]_{t=T} = \hat{R}(s_T, a_T, b_T)$. We also used the shorthand $P_{s\bar{s}\bar{a}\bar{b}} = P(s|\bar{s}, \bar{a}, \bar{b})$; similarly for $O_{yzs\bar{a}\bar{b}}$. Taking the log, we get:

$$\begin{aligned} \log P(r = 1, \tilde{L}, T) = \ldots + \sum_{t=0}^{T} \log \pi_{a_t p_t} + \sum_{t=0}^{T} \log \pi_{b_t q_t} \\ + \sum_{t=1}^{T} \log \lambda_{p_t p_{t-1} y_t} + \sum_{t=1}^{T} \log \lambda_{q_t q_{t-1} z_t} + \log \nu_{p_0} + \log \nu_{q_0} \end{aligned} \tag{12}$$

where the missing terms represent the quantities independent of $\boldsymbol{\theta}$, and they do not affect the maximization of $\boldsymbol{\theta}$. As all the policy parameters $\langle \pi, \lambda, \nu \rangle$ get separated out for each agent in the log above, the expression for expected log-likelihood can also be formulated

separately for action parameters, controller node transition parameters and initial node distribution. For action parameters of an agent, we have the simplified expression as:

$$Q_{\boldsymbol{a},\boldsymbol{p}}(\boldsymbol{\theta},\boldsymbol{\theta}^\star) = \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} \sum_{a,p} P(r=1, a_t=a, p_t=p|T;\theta) \log \pi^\star_{ap} \tag{13}$$

The above expression was derived by substituting $\sum_{t=0}^{T} \log \pi_{a_t p_t}$ for $\log P(r=1, \tilde{L}, T; \boldsymbol{\theta}^\star)$ in Eq. (10) and marginalizing out the remaining variables (See derivation in Appendix A). Analogous expected log-likelihood expressions can be written for node transition λ and initial node distribution ν parameters.

4.3 Step 2: Maximizing the Expected Log-Likelihood $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star)$

As highlighted in section 4.1, once we have formulated the expected log-likelihood, the next step is to maximize it to get a better policy $\theta^\star$. We show such a maximization first for action updates. The expected log-likelihood for action updates is given as:

$$Q_{\boldsymbol{a},\boldsymbol{p}}(\boldsymbol{\theta},\boldsymbol{\theta}^\star) = \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} \sum_{a,p} P(r=1, a_t=a, p_t=p|T;\theta) \log \pi^\star_{ap} \tag{14}$$

The maximization step (M-step) involves solving the following convex optimization problem:

$$\max_{\{\pi^\star_{ap}\}} Q_{\boldsymbol{a},\boldsymbol{p}}(\boldsymbol{\theta},\boldsymbol{\theta}^\star) \tag{15}$$

$$\text{subject to: } \sum_{a} \pi^\star_{ap} = 1 \ \forall p \tag{16}$$

The above optimization problem can be easily solved analytically by solving for the Karush-Kuhn-Tucker (KKT) conditions (Boyd & Vandenberghe, 2004). The resulting update for the controller action parameters is given as:

$$\pi^\star_{ap} = \frac{\sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} P(r=1, a_t=a, p_t=p|T;\theta)}{C_p} \tag{17}$$

$$= \frac{\mathbb{E}_\theta[r=1, a, p]}{C_p} \tag{18}$$

where C_p is a normalization constant.

All parameter updates: Analogous to the action updates above, we can write the controller transition λ as well as initial node distribution updates ν as follows:

$$\begin{aligned} \pi^\star_{ap} &= \frac{\mathbb{E}_\theta[r=1, a, p]}{C_p} \\ \lambda^\star_{p\bar{p}y} &= \frac{\mathbb{E}_\theta[r=1, p, \bar{p}, y]}{C_{\bar{p}y}} \\ \nu^\star_p &= \frac{\mathbb{E}_\theta[r=1, p]}{C} \end{aligned} \tag{19}$$

where we have:

$$\mathbb{E}_\theta[r=1,p,\bar{p},y] = \sum_{T=1}^{\infty} P(T) \sum_{t=1}^{T} P(r=1,p_t=p,p_{t-1}=\bar{p},y_t=y|T;\theta) \tag{20}$$

$$\mathbb{E}_\theta[r=1,p] = \sum_{T=0}^{\infty} P(T) P(r=1,p_0=p|T;\theta) \tag{21}$$

Sections 4.2 and 4.3 summarize the two main steps that form the core of the EM algorithm. These two steps are applied iteratively until convergence and result in monotonic improvement of solution quality.

4.4 Inference for Parameter Updates

We now detail the inference procedure that computes different expectations $\mathbb{E}_\theta$ required for parameter updates shown in Eq. (19). Computing such expectations also forms the E-step of the EM algorithm. In this step, for the fixed parameter θ, forward messages α and backward messages β are propagated. Such forward-backward message passing is very similar to the message passing in the Baum-Welch algorithm for parameter estimation in hidden Markov models (Koller & Friedman, 2009). First, we define the following Markovian transitions on the (p,q,s) state in a T-step DBN of Figure 3. These transitions are independent of the time t due to the stationary joint-policy.

$$P(p',q',s'|p,q,s) = \sum_{aby'z'} \lambda_{p'py'} \lambda_{q'qz'} O_{y'z'abs'} \pi_{ap} \pi_{bq} P_{s'sab} \tag{22}$$

Definition 1. *The forward message α_t is defined to be the probability that the FSC of agent 1 is in state p, FSC of agent 2 is in state q and the world is in state s at time t*

$$\alpha_t(p,q,s) = P(p_t=p, q_t=q, s_t=s;\theta).$$

It might appear that we need to propagate α messages for each DBN in the DBN mixture separately, but as pointed out by Toussaint et al. (2006), only one sweep is required as the head of the DBN is shared among all the mixture components. That is, α_2 is the same for all the T-step DBNs with $T \geq 2$. We will omit using θ as long as it is unambiguous.

$$\alpha_0(p,q,s) = \nu_p \nu_q \eta_0(s) \tag{23}$$

$$\alpha_t(p',q',s') = \sum_{p,q,s} P(p',q',s'|p,q,s) \alpha_{t-1}(p,q,s) \tag{24}$$

The β messages are propagated backwards and are defined as $P_t(r=1|p,q,s;\theta)$. However, this particular definition would require separate inference for each DBN as for T and T' step DBN, β_t will be different due to difference in the time-to-go ($T-t$ and $T'-t$). To circumvent this problem, β messages are indexed backward in time and defined as follows.

Definition 2. *The backward message β_τ is defined to be the probability of variable $r=1$ given that the FSC of agent 1 is in state p, FSC of agent 2 is in state q and the world is in state s and there are τ steps-to-go in a T-step DBN:*

$$\beta_\tau(p,q,s) = P(r=1|p_{T-\tau}=p, q_{T-\tau}=q, s_{T-\tau}=s)$$

where $\tau = 0$ denotes the last time slice $t = T$. As the tails of all the DBNs in the mixture of Figure 3 is exactly the same, such backward indexing avoids performing separate inference in each DBN. For example, β_3 is same for all the DBNs with length $T \geq 3$ as β_3 computes the probability of reward variable being one given the state of the Markov chain is (p, q, s) when there are 3 more time steps to go. The recursive definition is as follows:

$$\beta_0(p, q, s) = \sum_{ab} \hat{R}_{sab} \pi_{ap} \pi_{bq} \tag{25}$$

$$\beta_\tau(p, q, s) = \sum_{p',q',s'} \beta_{\tau-1}(p', q', s') P(p', q', s'|p, q, s) \tag{26}$$

Based on the α and β messages we compute two more quantities:

$$\hat{\alpha}(p, q, s) = \sum_{t=0}^{\infty} P(T = t) \alpha_t(p, q, s) \tag{27}$$

$$\hat{\beta}(p, q, s) = \sum_{\tau=0}^{\infty} P(T = \tau) \beta_\tau(p, q, s) \tag{28}$$

These quantities are used in the M-step. The cut-off time for message propagation can either be fixed a priori or be more flexible based on the likelihood accumulation. If α messages are propagated for t-steps and β-messages for τ steps, then the likelihood for $T = t + \tau$ is given as:

$$L^\theta_{t+\tau} = P(r{=}1|T = t + \tau; \theta) = \sum_{p,q,s} P(r{=}1, p_t{=}p, q_t{=}q, s_t{=}s|T{=}t + \tau) \tag{29}$$

$$= \sum_{p,q,s} P(r{=}1|p_t{=}p, q_t{=}q, s_t{=}s, T{=}t + \tau) P(p_t{=}p, q_t{=}q, s_t{=}s|T{=}t + \tau) \tag{30}$$

$$= \sum_{p,q,s} \alpha_t(p, q, s) \beta_\tau(p, q, s) \tag{31}$$

If both α and β messages are propagated for k steps and $L^\theta_{2k} \ll \sum_{T=0}^{2k-1} \gamma^T L^\theta_T$, then the message propagation can be stopped. This criterion simply means that when the expected value (or likelihood) for time step $T = 2k$ is too small when compared to the already accumulated value until time step $2k - 1$, then the message propagation can be stopped. Once we have computed such α and β messages, we can compute the expectations required for different updates in Section 4.3. The derivations of such updates are provided in the Appendix A.

Definition 3. *Based on computing quantities $\hat{\alpha}$ and $\hat{\beta}$ in Eq.* (27)- (28)*, the parameters of an agent's controller are updated after each EM's iteration as follows:*

$$\pi^\star_{ap} = \frac{\pi_{ap}}{C_p} \sum_{qs} \hat{\alpha}(p, q, s) \left[\sum_b \hat{R}_{sab} \pi_{bq} + \frac{\gamma}{1 - \gamma} \sum_{p'q's'y'z'} \hat{\beta}(p', q', s') \lambda_{p'py'} \lambda_{q'qz'} \sum_b O_{y'z's'ab} \pi_{bq} P_{s'sab} \right]$$

$$\lambda^\star_{p\bar{p}y} = \frac{\lambda_{p\bar{p}y}}{C_{\bar{p}y}} \sum_{sq\bar{s}\bar{q}z} \hat{\alpha}(\bar{p}, \bar{q}, \bar{s}) \hat{\beta}(p, q, s) \lambda_{q\bar{q}z} \sum_{ab} O_{yzsab} P_{s\bar{s}ab} \pi_{a\bar{p}} \pi_{b\bar{q}}$$

$$\nu^\star_p = \frac{\nu_p}{C} \sum_{qs} \hat{\beta}(p, q, s) \nu_q P_s \eta_0(s)$$

4.5 Complexity

Calculating the Markov transitions on the (p, q, s) chain in Section 4.4 has the complexity $O(|Q|^4|S|^2|A|^2|Y|^2)$, where $|Q|$ is the maximum number of nodes for a controller. The message propagation has complexity $O(T_{max}|Q|^4|S|^2)$. Techniques to effectively reduce this complexity without sacrificing accuracy will be discussed in the experiments section.

The complexity of updating all action parameters is $O(|Q|^4|S|^2|A||Y|^2)$. Updating node transitions requires $O(|Q|^4|S|^2|Y|^2 + |Q|^2|S|^2|Y|^2|A|^2)$. This is relatively high when compared to a single agent POMDP updates requiring $O(|Q|^2|S|^2|A||Y|)$ mainly due to the scale of the interactions present in Dec-POMDPs.

In our experimental settings, we observed that having a relatively small sized controller ($|Q| \leq 5$) suffices to yield good quality solutions. The main contributor to the complexity is the factor S^2 as we experimented with large domains having nearly 250 states. The good news is that the structure of the E and M-step equations provides a way to effectively reduce this complexity by a significant factor without sacrificing accuracy. For a given state s, joint-action $\langle a, b\rangle$ and joint-observation $\langle y, z\rangle$, the possible next states can be calculated as follows: $succ(s, a, b, y, z) = \{s'|P(s'|s, a, b)O(y, z|s', a, b) > 0\}$. For most of the problems, the size of this set is typically a constant $k < 10$. Such simple reachability analysis and other techniques speed up the EM algorithm by more than an order of magnitude for large problems. The effective complexity reduces to $O(|Q|^4|S||A||Y|^2k)$ for the action updates and $O(|Q|^4|S||Y|^2k + |Q|^2|S||Y|^2|A|^2k)$ for node transitions.

4.6 The Intuition Behind the EM Update Strategy

In this section, we provide some insights about the update strategies of EM. The action parameter update according to Section 4.3 is given as:

$$\pi^{\star}_{ap} = \frac{\mathbb{E}_{\boldsymbol{\theta}}\big[r = 1, a, p\big]}{C_p} \tag{32}$$

where C_p is a normalization constant. To understand Eq. (32) more clearly, consider the following iterative algorithm for optimizing the controller. First, we fix every controller node's parameters for every agent except for the parameters for a single controller node for a particular agent. Now, we deterministically try every action setting for the particular node and greedily set the action parameter for the node to be the action that results in maximum joint value. Clearly, this strategy will monotonically improve the policy value until it reaches a local optima. Such a strategy has been used in other decision making models such as influence diagrams and is referred to as *Single Policy Update* (SPU) algorithm (Lauritzen & Nilsson, 2001).

The updates of the EM algorithm are essentially a *soft* version of the above greedy and deterministic rule. To understand this, let $a^{\star}$ denote the action with maximum expectation:

$$a^{\star} = \arg\max_{a \in A} \mathbb{E}_{\boldsymbol{\theta}}\big[r = 1, a, p\big] \tag{33}$$

Now consider applying the update rule of Eq. (32) infinitely many times without recomputing the E-step. Clearly in the limit, we will have $\pi^{\star}_{a^{\star}p} = 1$ and the rest of the action parameters will converge to zero. This is essentially the SPU algorithm.

The above connection also provides insight as to why the soft-max approach of EM may be a better strategy than the greedy deterministic update rule. First, the greedy update rule computes deterministic controllers for both the agents. It has been already shown that stochastic controllers can achieve better solution quality than deterministic controllers (Poupart & Boutilier, 2003). EM updates can provide stochastic controllers, which is an advantage. Second, it has been already known in the graphical models community that the greedy update rule, also referred to as *Hard-Assignment EM* (Koller & Friedman, 2009, ch. 19) and the EM algorithm optimize different objectives. They can in general lead to different solutions, for example, in situations when stochastic controllers are preferred to deterministic ones. The hard-assignment EM traverses a combinatorial path and needs to fix all the parameters except one. The soft-assignment EM, on the other hand, can simultaneously change the parameters of multiple nodes. Thus, the moves in the parameter space of the soft-assignment EM are more sophisticated and in general, infeasible for the hard-assignment EM, which cannot simultaneously change multiple parameters. Thus, soft-assignment EM can converge to a better policy. Therefore, using the soft-max based update strategy of the EM algorithm can be more advantageous than the greedy deterministic rule.

4.7 Discussion

We presented a new approach to solve Dec-POMDPs using inference in a mixture of DBNs. The main benefit of the EM approach is that it opens up the possibility of using powerful probabilistic inference techniques to solve decentralized planning problems. Using a graphical DBN structure, EM can easily extend to larger multi-agent systems with more than 2 agents, as will be shown in the following sections. Furthermore, the planning-as-inference viewpoint can help to generalize to richer representations such as factored or hierarchical controllers (Toussaint et al., 2008), or continuous state and action spaces (Hoffman, de Freitas, Doucet, & Peters, 2009a).

Incidentally, there is an increasing interest in applying variational inference techniques from the machine learning and graphical models literature, such as marginal MAP inference and belief propagation (Liu & Ihler, 2013, 2012), to planning under uncertainty. Particularly related to our work is the use of such marginal MAP (MMAP) based inference to single agent POMDP planning (Kiselev & Poupart, 2014a, 2014b). In such MMAP based planning, a *single* dynamic Bayesian network is developed by introducing additional binary variables similar to the binary reward variable r in our case. Once a single graphical model is developed for the POMDP model, it is shown that maximizing the likelihood of observing one such special binary variable is equal to optimizing the joint policy.

The key difference between our approach and that of Kiselev and Poupart is the use of different graphical models to represent the underlying planning problem. The approach of Kiselev and Poupart uses two additional binary variables per time step that represent the reward discounting and the value function, which our approach does not need. Furthermore, in our approach, the time indexing of backward message propagation does not need to be fixed a priori as shown in Section 4.4, whereas a fixed horizon of the DBN needs to prespecified in Kiselev and Poupart's (2014b) method. Nonetheless, using MMAP inference for planning represents another variational inference-based approach that can be applied to

planning under uncertainty problems. Similar to the development of the EM in our case, one can also develop an EM algorithm for policy optimization in the single DBN model of Kiselev and Poupart. It remains to be seen, however, how the approach of Kiselev and Poupart (2014a, 2014b) gets translated to the multiagent setting w.r.t. to scalability and solution quality.

5. Achieving Scalability Under Restricted Models

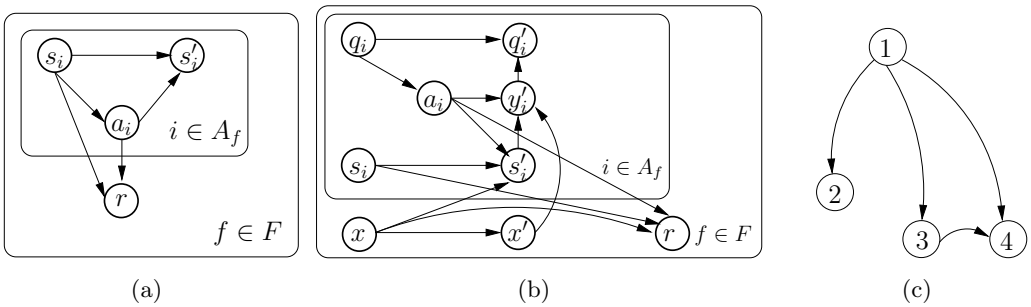
In previous sections, we developed probabilistic inference based approximate algorithms that can efficiently solve 2-agent Dec-POMDPs. However, scaling even such approximate algorithms for more than 2-agents is a non-trivial task. In fact, a naive extension leads to exponential increase in complexity with the number of agents. Therefore, in the following sections, we present a general characterization of interactions among agents that when present in a multiagent planning model leads to relatively scalable approximate algorithms.

In this section, we identify conditions that are sufficient to make multiagent planning amenable to a scalable approximation w.r.t. the number of agents. This is achieved by constructing a graphical model that exploits such restricted interactions among agents. We again illustrate a close relationship with machine learning by showing that likelihood maximization in such a graphical model is equivalent to policy optimization. Using the Expectation-Maximization framework for likelihood maximization, we show that the necessary inference can be decomposed into processes that often involve a small subset of agents, thereby facilitating scalability. We derive a global update rule that combines these local inferences to monotonically increase the overall solution quality. Furthermore, our approach is easily parallelizable and takes the form of message-passing among agents, ideally suited for large multiagent systems.

Experiments on a large multiagent planning benchmark, with joint state and action spaces up to 5^{22}, 3^{20} respectively, confirm that our approach is scalable w.r.t. the number of agents and can provide good quality solutions for planning problems for up to 20 agents, which cannot be handled by previous best approaches. For smaller multiagent systems, our approach provides better solution quality and is about an order-of-magnitude faster than the previous best nonlinear programming approach for optimizing FSCs.

5.1 The Value Factorization Framework

The value factorization framework leads to efficient inference for large multiagent systems and is general enough to subsume several existing planning models. As before, we represent each agent's policy as a bounded, *finite state controller* (FSC). We assume that the state space S is factored s.t. $S = (S^1 \times ... \times S^M)$—a common assumption in multiagent planning models such as ND-POMDPs (Nair et al., 2005) and TD-POMDPs (Witwicki & Durfee, 2010). Without making further (conditional independence) assumptions on the problem structure, a general Dec-POMDP requires exact inference in the full corresponding (finite-time) DBNs, which would be exponential in the number of state variables and agents. Our approach relies on a general simplifying property of agent interaction, which we later show to be consistent with many of the existing multiagent planning models.



(a) (b) (c)

Figure 5: Value factorization property in different models: (a) Plate notation for transition independent Dec-MDPs; (b) Plate notation for ND-POMDPs; (c) Agent interaction digraph for TD-POMDPs

Definition 4. *A **value factor** f defines a subset $A_f \subseteq \{1, .., N\}$ of agents and a subset $S_f \subseteq \{1, .., M\}$ of state variables.*

Definition 5. *A multiagent planning problem satisfies **value factorization** if the joint-policy value function can be decomposed into a sum over value factors:*

$$V(\boldsymbol{\theta}, s) = \sum_{f \in F} V_f(\theta^f, s^f) \ , \tag{34}$$

where F is a set of value factors, $\theta^f \equiv \theta^{A_f}$ is the collection of parameters of the agents of factor f, and $s^f \equiv s^{S_f}$ is the collection of state variables of this factor.

Even when the value factorization property holds, planning in such models is still highly coupled because factors may overlap. That is, an agent can appear in multiple factors as can state variables. Therefore, a value factor cannot be optimized *independently*. But, as we show later, it leads to an efficient Expectation Maximization algorithm. Such additive value functions have also been used to solve large factored MDPs (Koller & Parr, 1999). Witwicki and Durfee (2011) use such factored value functions to analyze the complexity of solving multiagent planning problems. Our work, in contrast, uses such value factorization property in conjunction with graphical models and probabilistic inference to develop a scalable likelihood maximization based algorithm. We require that each value factor V_f can be evaluated using the DBN mixture based approach of Section 3. However, this does not limit generality, as the DBN-based approach can model arbitrary Markovian planning problems.

Theorem 2. *The worst case complexity of optimally solving a multiagent planning problem satisfying the value factorization property is NEXP-Hard.*

The proof of the above theorem is straightforward—any two agent finite-horizon Dec-POMDP is NEXP-Complete and also satisfies the value factorization property as there is only a single factor involving two agents. In fact, in previous sections we precisely addressed this issue using the EM framework (see Section 4). Next, we investigate when this property holds and when it is computationally advantageous, and establish the following result.

Theorem 3. *The value factorization property holds in Transition-Independent Dec-MDPs (Becker et al., 2004), Network-Distributed POMDPs (Nair et al., 2005) and Transition-Decoupled POMDPs (Witwicki & Durfee, 2010).*

Proof. The joint value is shown to be factorized based on the immediate-reward factorization in transition independent Dec-MDPs (Becker et al., 2004) and ND-POMDPs (Nair et al., 2005). Figure 5 shows the plate notation for our value factor representation for both of these models. The outer plate shows a factor f and the inner plate depicts the interaction among agent parameters which include state, action and observation variables. In both these models, a key assumption that leads to scalability is that only a few agents are involved in a single reward function. Thus, each value factor is small and leads to efficient inference.

Our approach can also model Transition-Decoupled POMDPs (TD-POMDPs) (Witwicki & Durfee, 2010). In this case, agents have local parameters (factored local state and rewards). However, certain features of the local state can depend on other agents' actions. Such features are called *nonlocal* features for an agent i. This dependency among agents is described using an *agent interaction digraph* (Witwicki, 2011, Section 3.5.1.2). There is a node for each agent i. There is a directed edge from node i to node j if agent i affects a nonlocal feature of agent j. Let $\Lambda(i)$ denote all the ancestors of a node i in the agent interaction digraph for a TD-POMDP. As shown by Witwicki (2011, Thm. 3.33), the joint value function for a TD-POMDP defined over N agents can be factored as:

$$V(\boldsymbol{\theta}) = \sum_{i=1}^{N} V_i\big(\theta^i, \langle \theta^j \mid j \in \Lambda(i) \rangle\big) \tag{35}$$

The state variables involved in each factor V_i are the local states for each agent in $\{i\} \cup \Lambda(i)$. Furthermore, each value factor V_i can be evaluated by constructing a DBN mixture involving only the agents $\{i\} \cup \Lambda(i)$. Therefore, the TD-POMDP model satisfies the value factorization property. Consider for example the agent interaction digraph in Figure 5(c). The joint value factorizes as $V(\boldsymbol{\theta}) = V_1(\theta^1) + V_2(\theta^2, \theta^1) + V_3(\theta^3, \theta^1) + V_4(\theta^4, \theta^1, \theta^3)$.

Furthermore, TD-POMDPs are mainly useful for weakly-coupled planning problems (Witwicki, 2011). This implies that the number of agents involved in a single value factor should be small compared to the total number of agents, potentially leading to computational savings in approaches that can exploit the structure of such smaller value factors. □

We note that the value factorization property of Eq. (34) is trivially satisfied when all the agent and state variables are included in a single factor. Obviously, the computational advantages of our approach are limited to settings where each factor is *sparse*, involving much fewer agents than the entire team. This allows for efficient inference in the respective DBNs (inference can still be efficient for special cases such as TD-POMDPs that have larger factors). In the general case, the additive value function may include components depending on all states and agent parameters. This is analogous to the factored HMMs (Ghahramani & Jordan, 1995) where, conditioned on the observations, all Markov chains become coupled and the exact E-step of EM becomes infeasible. While this is beyond the scope of this paper, a promising approach for the general case is using variational methods to approximate the posterior $P(s_{1:T}^{1:M} \mid r = 1)$ (minimizing the KL-divergence between the factored representation and the true posterior) (Ghahramani & Jordan, 1995). Given such an approximate

posterior, the M-step updates can be used to realize an approximate EM scheme, as also shown by Pajarinen and Peltonen (2011a).

In the next section, we describe a new DBN mixture model for the value factorization framework. Before that, we highlight the key result behind the scalability of the EM algorithm

Theorem 4. *In models satisfying the value factorization property, the inferences required for the E-step of the EM algorithm can be performed independently for each value factor f. For e.g., for action updates for an agent j, we have*

$$\mathbb{E}_\theta[r=1, a^j, q^j] = \sum_{f \in F(j)} \mathbb{E}^f_\theta[r=1, a^j, q^j]$$

where j denotes a particular agent, $F(j)$ denotes the set of value factors the agent j is involved in, (a^j, q^j) denote the action and controller state of agent j, $\mathbb{E}^f_\theta[\cdot]$ (to be defined precisely in Section 5.4.1) denotes inference only in the DBN mixture corresponding to the valued factor f. A constructive proof of the above result is provided below. This result highlights the generality and scalability of our approach, which—unlike previous works—does not require any further independence assumptions.

In the next section, we define a latent variable model, again based on a mixture of DBNs, such that likelihood maximization (LM) in such a mixture model is equivalent to joint-policy optimization. Once we have established such a relationship between LM and policy optimization, we show how to perform different steps of EM in this mixture model as outlined in section 4.1.

5.2 DBN Mixture for Value Factors

Figure 6 shows a new problem-independent DBN mixture, also called *value factor* mixture, which models Eq. (34). It consists of two mixture variables: $\mathcal{F}$ and T. $\mathcal{F}$ ranges from 1 to $|F|$, the total number of value factors. Intuitively, $\mathcal{F} = i$ denotes the time dependent DBN mixture for value factor i. A zoomed-in view of this DBN mixture is provided in Figure 6(b). The mixture variable $\mathcal{F}$ has a fixed, uniform distribution ($= 1/|F|$). The distribution of the variable T is set as in Theorem 1.

This model relies on the fact that in our representation, each value factor can be represented and evaluated using a time dependent DBN mixture of Figure 6(b) and the binary reward variable r, as also shown in the Section 3. This DBN mixture for a particular value factor f will contain all the variables for agents involved in factor f: actions, controller nodes and observations, and the state variables S_f. The valuation $V_f(\theta^f)$ can be calculated by finding the likelihood $L_f(\theta^f; r=1) = P(r=1; \theta^f)$ of observing the binary reward variable as $r = 1$ in the DBN mixture for value factor f. Using Theorem 1, we have the following result:

$$V_f(\theta^f) = kL_f(\theta^f; r=1) + k_f \tag{36}$$

where k and k_f are constants, with k being the same for all value factors. This can be easily ensured by making all the original rewards positive by adding a suitable constant. Next we state one of our main results. We are also assuming that the policy is being optimized for

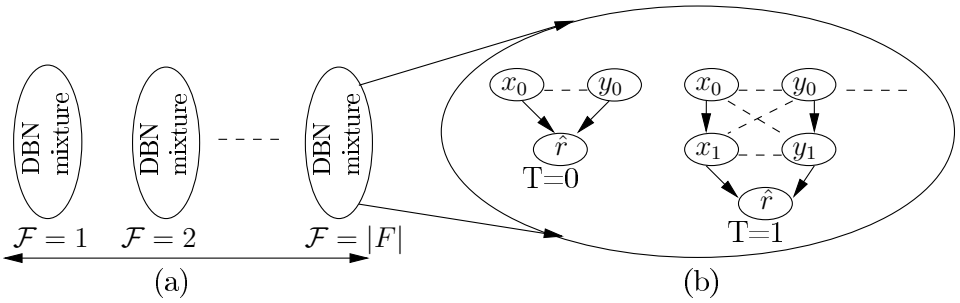


Figure 6: (a) Value factor mixture; (b) Zoomed-in view of each mixture component (x, y are generic placeholders for random variables).

an initial belief η_0. We will also use a notational convenience that $\sum_{\mathcal{F}=1}^{|F|}$ is equivalent to $\sum_{f \in F}$.

Theorem 5. *Maximizing the likelihood $L(\boldsymbol{\theta}; r=1)$ of observing $r=1$ in the value factor mixture (Figure 6(a)) is equivalent to optimizing the global policy $\boldsymbol{\theta}$.*

Proof. The overall likelihood is given by:

$$L(\boldsymbol{\theta}; r=1) = P(r=1; \boldsymbol{\theta}) = \sum_{f \in F} \frac{1}{|F|} L_f(\theta^f; r=1) \tag{37}$$

The theorem follows by substituting the value of each $L(\theta^f; r=1)$ in the previous equation from Eq. (36) and the joint-policy value decomposition assumption of Eq. (34). □

5.3 Step 1: Formulating the Expected Log-Likelihood $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star)$

We now formulate the expected log-likelihood $Q(\theta, \theta^\star)$ for the DBN mixture of Figure 6(a). Only $r=1$ is the observed data, the rest of the variables are latent. Note that our derivations differ markedly by Toussaint and Storkey (2006) as they focus on a single-agent problem or from the EM approach for 2-agent Dec-POMDPs (see Section 4). Our focus is on scalability w.r.t. the number of agents and generality.

An assignment to the mixture variables T and $\mathcal{F}=f$ denotes a T-step DBN for the value factor f. For example, assume that the time dependent DBN mixture in Figure 6(b) is for value factor f. Then $\mathcal{F}=f$ and $T=1$ denote the 1-step DBN (the second DBN) in Figure 6(b). Let Z^f denote a complete assignment to all the variables from slice 0 to T in the T-step DBN for factor f. We assume for simplicity that each value factor f involves k agents. The full joint for the mixture is:

$$P(r=1, Z^f, T, \mathcal{F}=f) = P(T)P(\mathcal{F}=f)\left[\hat{R}_{s^f a^f}\right]_{t=T} \prod_{i=1}^{k}\prod_{t=0}^{T}\left[\pi_{f_i}(a,q)\right]_t \prod_{i=1}^{k}\prod_{t=1}^{T}\left[\lambda_{f_i}(q,\bar{q},y)\right]_t$$
$$\prod_{i=1}^{k}[\nu_{f_i}(q)]_0 P(Z^f \backslash (A^f, Q^f) | T, \mathcal{F}=f) \tag{38}$$

where the index f_i denotes the respective parameters for agent i involved in factor f. The square brackets denote dependence upon time: $[\pi_{f_i}(a,q)]_t = \pi_{f_i}(a_t = a, q_t = q)$. We also use $[P(v,\bar{v})]_t$ to denote $P(v_t = v, v_{t-1} = \bar{v})$.

Let $Z^f \backslash (A^f, Q^f)$ denote all the variables in this DBN except the action and controller nodes of all the agents. Importantly, the structure of this previous equation is *model independent* as by the conditional independence of *policy parameters* ($\pi(a,q) = P(a|q)$), the first part of the equation (the product terms) can always be written this way. By model independent, we imply that the structure of previous equation remains the same for different models with value factorization property. Since EM maximizes the expected log-likelihood, we take the log of the above to get:

$$\log P(r=1, Z^f, T, \mathcal{F}=f) = \sum_{i=1}^{k}\sum_{t=0}^{T}\big[\log \pi_{f_i}(a,q)\big]_t + \sum_{i=1}^{k}\sum_{t=1}^{T}\big[\log \lambda_{f_i}(q,\bar{q},y)\big]_t + \sum_{i=1}^{k}\big[\log \nu_{f_i}(q)\big]_0 + \langle \text{other terms}\rangle \tag{39}$$

where $\langle$other terms$\rangle$ denote terms independent of policy parameters $\boldsymbol{\theta}$. EM maximizes the following expected log-likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{f\in\mathcal{F}}\sum_{T=0}^{\infty}\sum_{Z^f} P(r=1, Z^f, T, \mathcal{F}=f; \theta^f)\log P(r=1, Z^f, T, \mathcal{F}=f; \theta^{f\star}) \tag{40}$$

where $\boldsymbol{\theta}$ denotes the previous iteration's joint-policy and $\boldsymbol{\theta}^\star$ is the current iteration's policy to be determined by maximization. The structure of the log term (Eq. (39)) is particularly advantageous as it allows us to perform maximization for each parameter of each agent *independently*. This does not imply complete problem decoupling as all the parameters still depend on the *previous iteration's* parameters for all other agents.

5.4 Step 2: Maximizing the Expected Log-Likelihood $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star)$

We first derive the update for action parameters of an agent j. $P(\mathcal{F})$ can be ignored as it is a constant. The expected log-likelihood for action updates, $Q^j_{\boldsymbol{a},\boldsymbol{q}}(\boldsymbol{\theta}, \boldsymbol{\theta}^\star)$, is given by:

$$Q^j_{\boldsymbol{a},\boldsymbol{q}}(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{f\in F(j)}\sum_{T} P(T)\sum_{Z^f} P(r=1, Z^f|T, f; \theta^f)\Big[\sum_{t=0}^{T}\big[\log \pi^\star_j(a,q)\big]_t\Big] \tag{41}$$

$$= \sum_{f\in F(j)}\sum_{T} P(T)\sum_{t=0}^{T}\sum_{a,q} P(r=1, a_t=a, q_t=q, |T, f; \theta^f)\log \pi^\star_j(a,q) \tag{42}$$

where $F(j)$ is the set of value factors that involve agent j. The M-step involves solving the following convex optimization problem:

$$\max_{\{\pi^\star_j(a,q)\ \forall a,q\}} Q^j_{\boldsymbol{a},\boldsymbol{q}}(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) \tag{43}$$

$$\text{subject to:} \sum_a \pi^\star_j(a,q) = 1\ \forall q \tag{44}$$

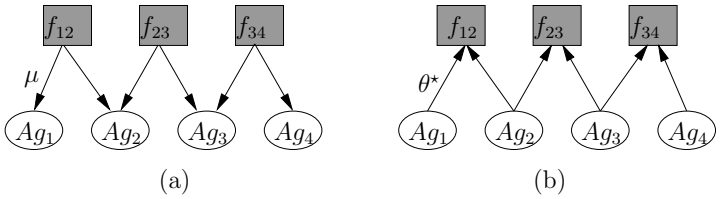


Figure 7: Message passing on the value factor graph: (a) shows the message direction for E-step; (b) shows the M-step.

The above optimization problem can be easily solved analytically by solving for the Karush-Kuhn-Tucker (KKT) conditions resulting in following action parameter updates:

$$\pi_j^\star(a,q) = \frac{\sum_{f\in F(j)} \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} P(r=1, a_t=a, q_t=q|T, f; \theta^f)}{C_q} \tag{45}$$

$$= \frac{\sum_{f\in F(j)} \mathbb{E}_\theta^f[r=1, a, q]}{C_q} \tag{46}$$

where C_q is the normalization constant.

5.4.1 All Parameter Updates

Analogous to the action updates in the previous section, we can write the controller transition λ as well as initial node distribution updates ν for the agent j as follows:

$$\pi_j^\star(a,q) = \frac{\sum_{f\in F(j)} \mathbb{E}_\theta^f[r=1, a, q]}{C_q} \tag{47}$$

$$\lambda_j^\star(q,\bar{q},y) = \frac{\sum_{f\in F(j)} \mathbb{E}_\theta^f[r=1, q, \bar{q}, y]}{C_{\bar{q}y}} \tag{48}$$

$$\nu_j^\star(q) = \frac{\sum_{f\in F(j)} \mathbb{E}_\theta^f[r=1, q]}{C} \tag{49}$$

We have omitted the superscript j above to denote the action and controller variable for agent j to avoid clutter. In the next two equations too, all action, controller variables belong to agent j.

$$\mathbb{E}_\theta^f[r=1, q, \bar{q}, y] = \sum_{T=1}^{\infty} P(T) \sum_{t=1}^{T} P(r=1, q_t=q, q_{t-1}=\bar{q}, y_t=y|T, f; \theta^f) \tag{50}$$

$$\mathbb{E}_\theta^f[r=1, q] = \sum_{T=0}^{\infty} P(T) P(r=1, q_0=q|T, f; \theta^f) \tag{51}$$

5.5 Scalability and Message Passing Implementation

The parameter updates in Section 5.4.1 highlight the high scalability of EM for Dec-POMDPs. Even though the planning problem is highly coupled with each agent and state

variables allowed to participate in multiple value factors (see Eq. (34)), updating policy parameters requires *separate* local inference, $\mathbb{E}^f_\theta[r=1,\cdot,\cdot]$, in each value factor.

The global update rules (Eq. (47)–(49)) combine such local inferences to provide a monotonic increase in the overall solution quality. Each local inference is model dependent and can be computed using standard probabilistic techniques. As our problem setting includes *sparse* factors, such local inference will be computationally much simpler than performing it on the complete planning model.

Furthermore, the E and M-steps can be implemented using a *parallel*, *distributed* message-passing on a bipartite *value-factor graph* $G=(U,V,E)$. The set U contains a node u_j for each agent j. The set V contains a node v_f for each factor $f\in F$. An edge $e=(u_j,v_f)$ is created if agent j is involved in factor f. Figure 7 shows such a graph with three value factors in the black squares (the set V) and 4 agents (the set U).

During the E-step, each factor node v_f computes and sends the message $\mu_{f\to j}=\mathbb{E}^f_\theta[r=1,\cdot,\cdot]$ to each node u_j that is connected to v_f. Figure 7(a) shows this step. An agent node u_j upon receiving all the μ messages from each factor node connected to it, updates its parameters as in Eq. (47)–(49) and sends the updated policy parameters $\theta^\star$ back to each factor node it is connected to (see Figure 7(b)). This procedure repeats until convergence. Based on this message-passing interpretation of EM, we can state the following result:

Theorem 6. *Each iteration of the EM algorithm has linear complexity in the number of edges in the value factor graph and exponential complexity with respect to the maximum number of agents involved in a single value factor.*

As stated earlier, when the value factors are sparse, EM algorithm will provide significant computational savings over an approach that is oblivious to the underlying interaction among agents. Another significant advantage of the EM algorithm is that all messages can be computed in *parallel* by each factor node. Our experiments, using an 8-core CPU resulted in near linear speedup over a sequential version. These characteristics of the EM algorithm significantly enhance its scalability for large, but sparse planning problems.

5.5.1 Nature of Local Optima

Variants of the Dec-POMDP model are often solved by fixing the policies of a group of agents and then finding the best response policy of the agent that interacts with the group (Nair, Tambe, Yokoo, Pynadath, Marsella, Nair, & Tambe, 2003; Witwicki & Durfee, 2010). EM offers significant advantages over such strategy. While both find local optima, EM is more stringent and satisfies the Karush-Kuhn-Tucker conditions (Bertsekas, 1999), which is the norm in nonlinear optimization. The local optima of the EM refers to the stationary point of the likelihood function (or the value function) $l(\theta;x)$ in Figure 4. The parameter space θ in this figure includes both the discrete parameters that can be found by local optimal algorithms such as any best response strategy and the continuous parameters that are found by algorithms such as EM.

The local optima of any best response strategy simply refers to the fact that the particular algorithm cannot improve the policy using the best-response strategy. Such algorithm specific local optima may or not be the stationary point of the value function in Figure 4. If such a solution is not the stationary point, then EM would be able to improve upon

this solution given that EM is guaranteed to converge to a stationary point of the value function. Thus, the local optima provided by EM satisfies the more stringent guarantee of being a stationary point of the value function. The best-response strategy provides no such guarantees.

5.6 Discussion

Despite rapid progress in multiagent planning, the scalability of the prevailing formal models has been limited. We developed a new approach to multiagent planning by identifying the general property of *value factorization* that facilitates the development of a scalable approximate algorithm using probabilistic inference. We show that several existing classes of Dec-POMDPs satisfy this property. In contrast to previous approaches, our framework does not impose any further restrictions on agent interaction beyond this property, thus providing a general solution for value factorization based multiagent planning.

The key result that supports the scalability of our approach is that, within the EM framework, the inference process can be decomposed into separate components that are much smaller than the complete model, thus avoiding an exponential complexity in the number of agents. Additionally, the EM algorithm allows for distributed planning using message-passing along the edges of the value-factor graph, and is amenable to parallelization. Results on large sensor network problems confirm the scalability of our approach. Empirically, our approach, which has linear complexity in the number of edges in the agent interaction graph could scale up to 20 agents, whereas the previous best approach based on nonlinear programming could only scale up to 5 agents due to increase in the number of nonlinear constraints.

We also highlight the key differences in our approach against the previous approach of Toussaint et al. (2006, 2008) for *single-agent* MDPs and POMDPs. Although the idea of decomposing the planning problem into a time-dependent DBN mixture remains the same in our approach, the key differences lie in the structure of the DBNs for two agent Dec-POMDPs and value factored Dec-POMDPs, the derivation of the lower bound function $Q(\theta, \theta^\star)$ and the maximization of this Q function for Dec-POMDPs as required within the EM framework (see Section 4.1), and computing the required inferences over the underlying Markov chain of DBNs as shown in Section 4.4.

The interactions between the FSC-based policy and the environment present in the DBN for MDPs and POMDPs are relatively simple when compared against the interactions present in a Dec-POMDPs. In Dec-POMDPs, agents not only interact with the environment, but also with each other. Such inter-agent interactions leads to a much different DBN structure, and planning challenges in Dec-POMDPs. For example, the DBN mixture for our value factorization model is *nested* as shown in Figure 6. This is a unique feature required in our formulation of the DBN approach, and not present in POMDPs. Due to such differences in the structure of a single DBN as well as the mixture model, our adaptation of EM requires different formulation of alpha and beta messages to account for inter-agent influences as shown in Section 4.4. Similarly, the structure of $Q(\cdot, \cdot)$ function is different owing to inter-agent influences (see Eq. (40)), and leads to different updates that provide scalability and inter-agent message-passing structure in multiagent systems as shown in Sections 5.4.1 and 5.5. Therefore, while adapting the EM approach to multiagent

Size	DEC-BPI	NLP	EM	DEC-BPI	EM
1	4.687	9.1	9.05	< 1s	< 1s
2	4.068	9.1	9.05	< 1s	< 1s
3	8.637	9.1	9.05	2s	1.7s
4	7.857	9.1	9.05	5s	4.62s

Table 1: Broadcast channel: Policy value, execution time

planning, one of our key contributions has been to deliberately investigate and exploit the independencies that are present in a multiagent system to translate into a scalable algorithmic structure. Furthermore, we have also shown general applicability of our approach to previous multiagent planning models in Theorem 3.

6. Empirical Evaluation

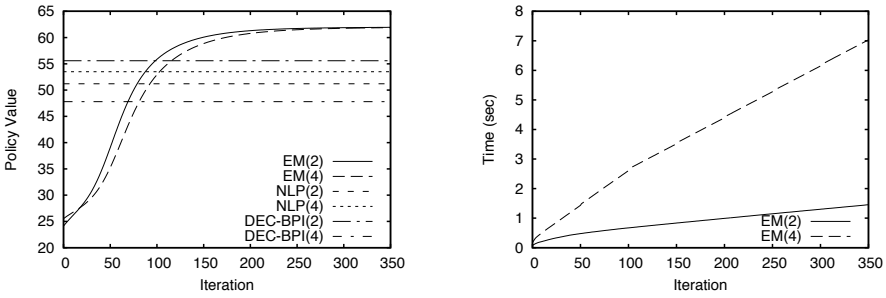
We begin the empirical evaluation with experiments conducted with 2-agent problems. This is followed by experiments with larger problems involving up to 20 agents.

6.1 Two Agents Dec-POMDPs

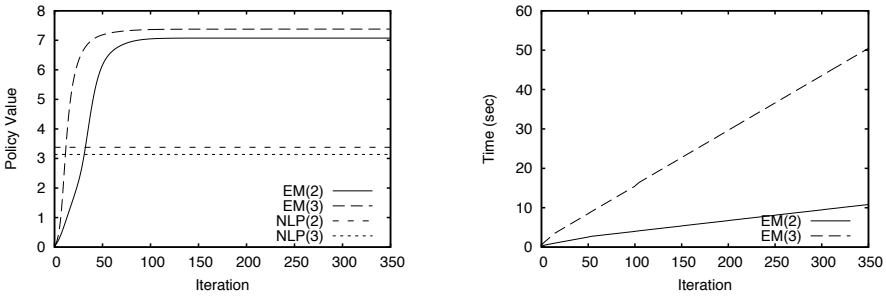
We experimented with several standard 2-agent Dec-POMDP benchmarks with discounting factor $\gamma = 0.9$. We compare our EM algorithm with the decentralized bounded policy iteration (DEC-BPI) algorithm (Bernstein et al., 2009) and a non-linear, non-convex optimization solver (NLP) (Amato et al., 2010). The DEC-BPI algorithm iteratively improves the parameters of a node using a linear program while keeping the other nodes' parameters fixed. The NLP approach, which has been the state-of-the-art for infinite-horizon Dec-POMDPs, recasts the policy optimization problem as a non-linear program and uses an off-the-shelf solver, Snopt (Gill et al., 2002), to obtain a solution. We implemented the EM algorithm in JAVA. All our experiments were on a Mac with 4GB RAM and 2.4GHz CPU. Each data point for every algorithm tested and parameter setting is an average of 10 runs with random initial controller parameters. In terms of solution quality, EM is always better than DEC-BPI and it achieves similar or higher solution quality than NLP. We note that the NLP solver (Gill et al., 2002) is an optimized package and therefore for larger two agent problems is currently faster than the EM approach. For the EM algorithm, we did not implement optimizations such as parallel execution using multithreading, that can decrease the runtime significantly.

Table 1 shows results for the *broadcast channel* problem, which has 4 states, 2 actions per agent and 5 observations. This is a networking problem where agents must decide whether or not to send a message on a shared channel and must avoid collision to get a reward. We tested with different controller sizes shown in the first column. On this problem, all the algorithms compare reasonably well, with EM being better than DEC-BPI and very close in value to NLP. The time for NLP is also $\approx 1s$.

Figure 8(a) compares the solution quality of the EM approach against DEC-BPI and NLP for varying controller sizes on the *recycling robots* problem. In this problem, two robots



(a) $|S| = 3, |A| = 3, |Y| = 2$ (b)

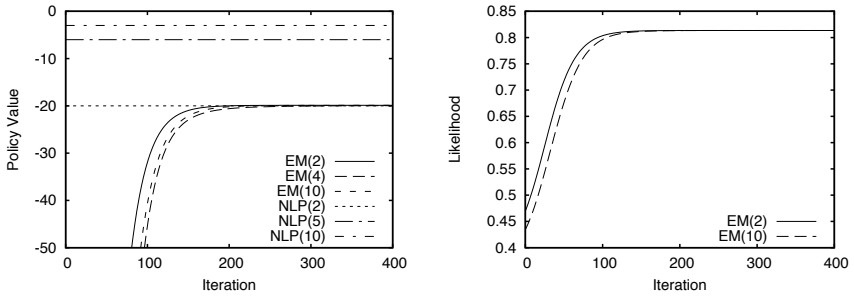


(c) $|S| = 16, |A| = 5, |Y| = 4$ (d)

Figure 8: Solution quality and runtime for 'recycling robots' (a) & (b) and 'meeting on a grid' (c) & (d)

have the task of picking up cans in an office building. They can search for a small can, a big can or recharge the battery. The large item is only retrievable by the joint action of the two robots. Their goal is to coordinate their actions to maximize the joint reward. EM(2) and NLP(2) show the results with controller size 2 for both agents in Figure 8(a). For this problem, EM works much better than both DEC-BPI and the NLP approach. EM achieves a value of ≈ 62 for all controller sizes, providing nearly 12% improvement over DEC-BPI ($= 55$) and 20% improvement over NLP ($= 51$). Figure 8(b) shows the time comparisons for EM with different controller sizes. Both the NLP and DEC-BPI take nearly $1s$ to converge. EM with controller size 2 has comparable performance, but as expected, EM with 4-node controllers takes longer as the complexity of EM is proportional to $O(|\mathcal{P}|^4)$, where $|\mathcal{P}|$ denotes the controller size.

Figure 8(c) compares the solution quality of EM on the *meeting on a grid* problem. In this problem, agents start diagonally across in a 2×2 grid and their goal is to take actions such that they meet each other (i.e., share the same square) as much as possible. As the figure shows, EM provides much better solution quality than the NLP approach. EM achieves a value of ≈ 7, which nearly *doubles* the solution quality achieved by NLP ($= 3.3$). DEC-BPI results are not plotted as it performs much worse and achieves a solution quality of 0, essentially unable to improve the policy at all even for large controllers. Both DEC-BPI and NLP take around $1s$ to converge. Figure 8(d) shows the time comparison for EM versions. EM with 2-node controllers is very fast and takes $< 1s$ to converge (50 iterations).



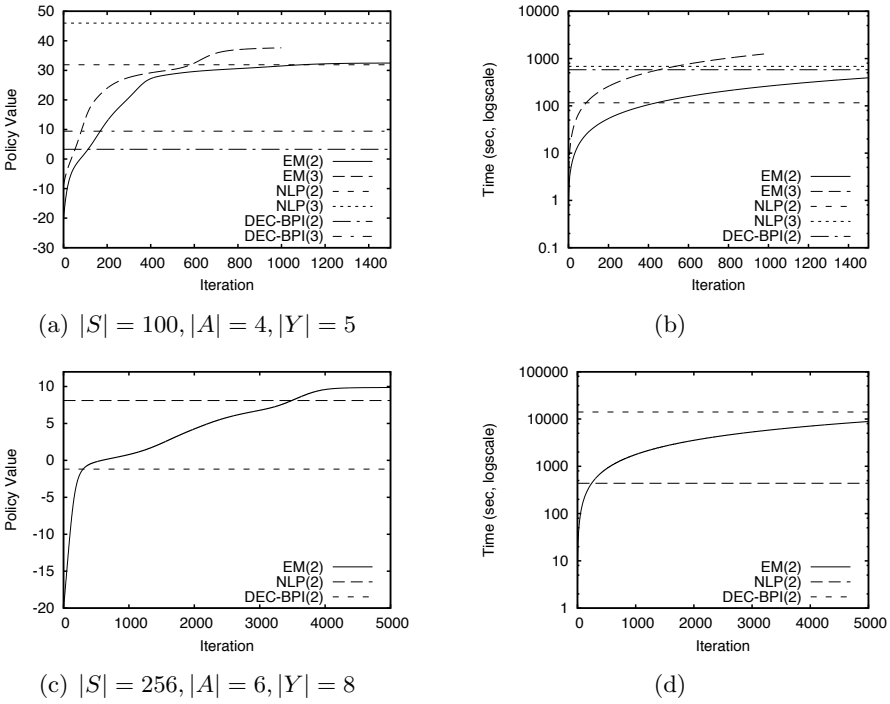
(a) $|S| = 2, |A| = 3, |Y| = 2$ (b)

Figure 9: Solution quality (a) and likelihood (b) for 'multiagent tiger'

Again, because of EM's quartic complexity in the controller size $|\mathcal{P}|$, the time required for larger controllers is higher. Also note that in both the cases, EM could run with much larger controller sizes ($\approx$10), but the increase in size did not provide tangible improvement in solution quality.

Figure 9 shows the results for the *multi-agent tiger* problem, involving two doors with a tiger behind one door and a treasure behind the other. Agents should coordinate to open the door leading to the treasure (Amato et al., 2010). Figure 9(a) shows the quality comparisons. EM does not perform well in this case; even after increasing the controller size, it achieves a value of -19. NLP works better with large controller sizes. However, this experiment presents an interesting insight into the workings of EM as related to the scaling of the rewards. Recalling the relation between the likelihood and the policy value from Theorem 1, the equation for this problem is: $V^\theta = 1210L^\theta - 1004.5$. For EM to achieve the same solution as the best NLP setting ($= -3$), the likelihood should be .827. Figure 9(b) shows that the likelihood EM converges to is .813. Therefore, from EM's perspective, it is finding a really good solution. Thus, the scaling of rewards has a significant impact (in this case, adverse) on the policy value. This is a potential drawback of the EM approach, which applies to other Markovian planning problems when using the technique of Toussaint et al. (2006). Incidently, DEC-BPI performs much worse and gets a quality of -77.

Figure 10 shows the results for the two largest Dec-POMDP domains—*Box pushing* and *Mars rovers*. In the box pushing domain, agents need to coordinate and push boxes into a goal area. In the Mars rovers domain, agents need to coordinate their actions to perform experiments at multiple sites. Figure 10(a) shows that EM performs much better than DEC-BPI for every controller size. For controller size 2, EM achieves better quality than NLP with comparable runtime (Figure 10(b), 500 iterations). However, for the larger controller size ($= 3$), it achieves slightly lower quality than NLP. For the largest Mars rovers domain (Figure 10(c)), EM achieves better solution quality ($= 9.9$) than NLP ($= 8.1$). However, EM also takes many more iterations to converge than for previous problems and hence, requires more time than NLP. EM is also much better than DEC-BPI, which achieves a quality of -1.18 and takes even longer to converge (Figure 10(d)). For this Mars rover domain, the NLP solver was not able to run for larger controller sizes due to the size of the nonlinear program.



(a) $|S| = 100, |A| = 4, |Y| = 5$ (b)

(c) $|S| = 256, |A| = 6, |Y| = 8$ (d)

Figure 10: Solution quality and runtime for 'box pushing' (a) & (b) and 'Mars rovers' (c) & (d)

To summarise, a simple implementation of the EM approach was competitive with an industrial strength off-the-shelf nonlinear programming solver. Our algorithm provided similar or better solution quality than the current best NLP approach. The main benefit of the EM approach lies in the fact that it opens up the possibility of using powerful probabilistic inference techniques to solve decentralised planning problems. As we shall see in the next section, EM scales significantly better than the NLP approach for larger ($\gg 2$) multiagent benchmarks where the NLP approach fails due to the large size of the resulting nonlinear programs.

6.2 Larger Multiagent Benchmarks

We experimented on a target tracking application in sensor networks modeled as an ND-POMDP (Nair et al., 2005). Figure 11 shows four sensor topologies: 5P, 11H and 15-3D by Marecki et al. (2008) and the largest 20D by Kumar and Zilberstein (2009b).

We describe and develop a significantly enriched variant of this application to better test our approach, originally introduced by Nair et al. (2005). Each node in these graphs is a sensor agent and edges are locations where targets can move. To track a target and gain reward ($=+80$), two sensors must scan the target location simultaneously, otherwise a penalty ($=-1$) is given. All targets have independent, stochastic trajectories and their all possible locations form the *external* state-space, implying target movement is not affected

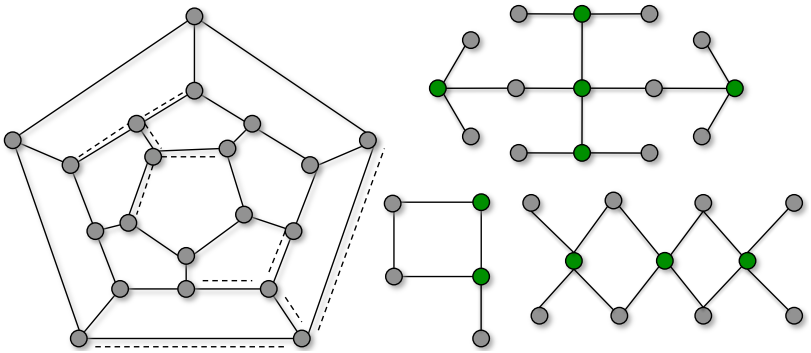


Figure 11: Benchmarks 20D (left), 15-3D, 5P and 11H (right)

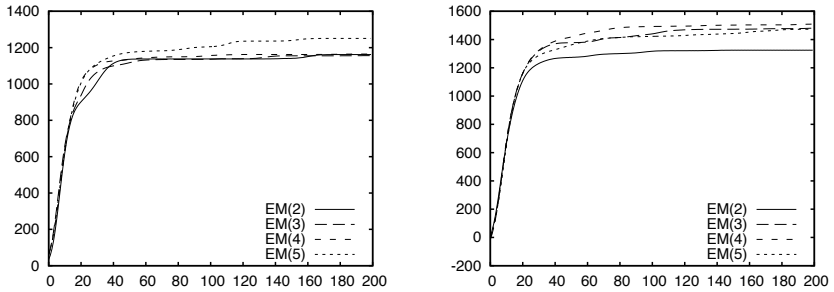
by the sensors' actions. Sensors have an *internal state*, an indication of the battery level of the sensor. Each scan action depletes the battery. In addition to scanning, sensors have two additional actions—*sensor off* and *recharge*. The *sensor off* action allows sensors to conserve energy by remaining idle. When the battery is completely depleted, a sensor must perform *recharge* action, which has a cost ($=-1$). Each sensor has three observation: *target present*, *target absent* and *idle*. False positives/negatives are allowed for the first two observations. At runtime, sensors operate in a *decentralized* manner without a central controller.

Note that our formulation of sensor network application is much richer and challenging than the previously used benchmarks (Marecki et al., 2008). Earlier benchmarks did not include any internal states and sensors were assumed to have an unbounded battery life. In the current formulation, planning is much more complex as sensors must reason not only about scanning, but also about conserving their energy, as they have a limited battery.

The EM algorithm was implemented in JAVA. All our experiments were done on an 8-core iMac with 2GB RAM. Our implementation used multithreading to parallelize EM as highlighted in Section 5.5 and utilized all 8-cores. Each datapoint is the average of 10 runs. To speed up EM's convergence, we used a greedy variant of the M-step presented by Toussaint et al. (2008). Such a step positively affects the rate of convergence of EM with relatively little affect on the solution quality. Such an M-step is essentially a softer version of the hard assignment EM (see Eq.(33)) and follows the same design (Toussaint et al., 2008). We next describe problem sizes for different instances.

The 5P domain has 2 targets, 11H has 3 targets, 15-3D has 5 targets, and 20D has 6 targets. All these problems have very large state-spaces: 6×5^5 for 5P, 64×5^{11} for 11H, 144×5^{15} for 15-3D and 2700×5^{20} for 20D. Evidently, EM efficiently exploits the factored representation of the state and action spaces and is highly scalable with *linear complexity* w.r.t. the number of edges in the graph. We also note that even solving the underlying factored MMDP optimally is not feasible due to large state and action spaces.

Figure 12 shows the solution quality EM achieves for each of these benchmarks with different controller sizes. A notable observation from these graphs is that EM converges quickly with the greedy M-step of Toussaint et al. (2008) , taking fewer than 200 iterations even for such large multiagent planning problems. The solution quality, as expected, increases monotonically with each iteration highlighting the anytime property of EM. In



(a) 5P (b) 11H

(c) 15-3D (d) 20D

Figure 12: Solution quality achieved by EM (y-axis denotes quality and x-axis denotes the iteration number).

Instance\Size	2-Node	3-Node	4-Node	5-Node
5P	.232	1.07	3.22	7.74
11H	1.29	6.07	18.90	45.23
15-3D	1.17	5.39	16.69	40.47
20D	5.03	22.01	67.85	171.26

Table 2: Time in seconds per iteration of EM

general, the solution quality increased with the number of controller nodes. For example, for 20D, a 2-node controller achieves a quality of 3585.06 and a 5 node controller achieves 4154.04. However, for 5P and 15-3D, we did not observe a significant increase in quality by increasing controller size, possibly due to the relative simplicity of these configurations.

Table 2 shows the runtime per iteration of EM for different instances and varying controller sizes. Encouragingly, the runtime is fairly small—particularly for smaller controller sizes—even for large problems such as 20D. To further decrease the runtime for larger controllers, we plan to use Monte-Carlo sampling techniques in the future.

Table 3 shows the solution quality comparison of EM with random controllers and a loose upper bound. The upper bound was computed by assuming that each target is detected at every time step including the battery recharge cost. Against random controllers, EM

Instance/Value	EM	U.B.	Random
5P	1250.89 (44.3%)	2820	61.23
11H	1509.27 (35.6%)	4230	8.41
15-3D	3094.05 (43.8%)	7050	104.2
20D	4154.04 (49.1%)	8460	−31.67

Table 3: Quality comparisons with a loose upper bound and random controllers for all instances

	Internal State = 2		Internal State = 3	
N	EM	NLP	EM	NLP
2	670.8/3.8	79/5.4	972.5/8.9	905.7/17.8
3	670.8/13.02	140.4/14.5	1053.16/35.8	887.2/139
4	710.4/35.8	140.4/139.4	1062.4/107.4	1024.8/1078.1

Table 4: Solution quality/time comparison of EM (100 iterations) with NLP for the 5P domain, N denotes controller size, Time in seconds

always achieves much better solution quality. When compared against the upper bound, we can see that EM performs quite well. Despite being a very loose bound, EM still achieves a quality within 49.1% of this bound for the largest 20D domain—a solid performance.

Previously, no algorithm could solve infinite-horizon ND-POMDPs (>2-agents). To further assess EM's performance, we developed a nonlinear programming (NLP) formulation of the problem and used a state-of-the-art NLP solver called Snopt (Gill et al., 2002). Snopt could only solve the smallest 5P domain and could not scale beyond controller size 4 and internal state 3 as it ran out of memory (=2GB). Table 4 shows the solution quality and time comparisons. For internal state size of 2, Snopt gets stuck in a poor local optimum compared to EM. It provides more competitive solutions for internal state 3, but EM is still better in solution quality. Furthermore, the runtime of Snopt degrades quickly with the increase in nonlinear constraints. This makes Snopt about an *order-of-magnitude* slower for controller size 4 and internal state 3. These results further highlight the scalability of EM, which could scale up to controller size 10 and internal state 5 within 2GB RAM and $\approx$ 4 hours for 100 iterations.

Table 5 shows a comparison of EM against handcrafted controllers designed to take into account the target trajectories and partial observation in the 11H benchmark (Figure 11). To simplify the problem for the handcrafted solution, all penalties were zero and the reward for detecting a target was 1. This allowed continuous scan by sensors without worrying about miscoordination penalty. The first row in this table shows this no penalty case. We see that EM is competitive with the handcrafted controller. The second row shows the results when there was a cost to charge batteries. In this case, sensors need to decide when to become *idle* to conserve power. The handcrafted controller cannot learn this behavior and hence EM produces much better quality in this case.

Version \ FSC Size	Handcrafted	2 (EM)	3 (EM)	4 (EM)
No penalty	13.92	13.95	15.48	15.7
Penalty (−.25)	-3.36	5.27	5.27	5.27

Table 5: Quality of handcrafted controllers vs. EM (11H)

Version \ FSC Size	2	3	4	5
Serial	41.05	177.54	543.52	1308.20
Parallel	5.03	22.01	67.85	171.26

Table 6: Serial vs. parallel execution times per EM iteration in 20D.

Finally, Table 6 highlights the significant opportunities that EM provides for parallel computation. We consistently obtained almost linear speedup when using multithreading on an 8-core CPU (total possible parallel threads in the largest domain 20D is 60). By using a massively parallel platform such a Google's MapReduce,we could easily scale to much larger team decision problems than currently possible.

7. Conclusion

Despite the rapid progress in multiagent planning, the scalability of the prevailing formal models and algorithms has been limited. We presented a new approach to multiagent planning by developing novel connections between multiagent planning and machine learning. We showed how the multiagent planning problem can be reformulated as inference in a mixture of dynamic Bayesian networks. By viewing multiagent planning through the lens of probabilistic inference, we open the door to the application of efficient inference techniques to multiagent decision making.

To further improve scalability to large multiagent systems, we identified a general condition called value factorization that facilitated the development of a scalable approximate algorithm using probabilistic inference. We showed that several existing classes of Dec-POMDPs satisfy this property. In contrast to previous approaches, our framework does not impose any further restrictions on agent interaction beyond this property, thus providing a general solution for value factorization based structured multiagent planning. The key result that supports the scalability of our approach is that, within the EM framework, the inference process can be decomposed into separate components that are much smaller than the complete model, thus avoiding an exponential complexity.

Empirically, we experimented with several standard and large multiagent planning benchmarks. Our inference-based approach was competitive with previous best approaches based on nonlinear and linear programming. Our approach, which has linear complexity in the number of edges in the agent interaction graph could scale up to 20 agents, whereas the previous best approach based on nonlinear programming could only scale up to 5 agents due to increase in the number of nonlinear constraints.

Our theoretical and empirical results show that exploring methods that overlap machine learning and planning has a great potential to overcome the practical limitations of existing multiagent planning algorithms. In future work, we plan to explore several such

directions. We are interested in exploring the overlap of stochastic control theory and multiagent planning in continuous action and state space models similar to the work of Hoffman et al. (2009a, 2009b). We also plan to further explore ways to overcome the effect of local optima on the solution quality achieved by the EM algorithm. We specifically plan to investigate strategies to escape local optima (Poupart, Lang, & Toussaint, 2011) and to adapt them to the multiagent setting.

Another key issue with planning-as-inference strategy using the EM algorithm is the lack of any optimality guarantee or upper bounds on the controller based joint-policy. An interesting recent research direction in the graphical models and machine learning literature is the development of multiple inference strategies for marginal MAP (Liu & Ihler, 2013). It has been shown that the EM algorithm can be used as one such inference approach to solve the marginal MAP problem (Liu & Ihler, 2013). Planning under uncertainty problems can be categorized as an instance of the marginal MAP inference (Cheng, Liu, Chen, & Ihler, 2013). Therefore, developing graphical models that can exploit new inference approaches for the marginal MAP problem and provide quality guarantees is an interesting new direction to explore at the frontier of planning and machine learning.

Acknowledgments

This work was funded in part by the National Science Foundation under grants IIS-0812149 and IIS-1116917, the Air Force Office of Scientific Research under grant FA9550-08-1-0181, and the EU 3rdHand project.

Appendix A. Proof of EM Update Equations in Definition 3

We provide a constructive proof, showing the derivations of the relevant update equations.

A.1 Proof of Eq. (13)

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^\star) = \sum_{T=0}^{\infty} \sum_{\tilde{L}} P(r=1, \tilde{L}, T; \boldsymbol{\theta}) \big[\sum_{t=0}^{T} \log \pi^\star_{a_t p_t} \big] \tag{52}$$

$$= \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} \sum_{\tilde{L}} P(r=1, \tilde{L}|T; \boldsymbol{\theta}) \log \pi^\star_{a_t p_t} \tag{53}$$

In the above equation, the variable $\tilde{L}$ includes all the hidden variables in the DBN of length T. We can simplify the above summation using marginalization over variables other than $\{a_t, p_t\}$. We also use the fact that the policy is stationary to simplify as:

$$= \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} \sum_{a,p} \log \pi^\star_{ap} \sum_{\tilde{L} \setminus \{a_t, p_t\}} P(r=1, a_t = a, p_t = p, \tilde{L} \setminus \{a_t, p_t\}|T; \boldsymbol{\theta}) \tag{54}$$

$$= \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} \sum_{a,p} \log \pi^\star_{ap} P(r=1, a_t = a, p_t = p|T; \boldsymbol{\theta}) \tag{55}$$

A.2 Action Parameter Updates

The expectation required for action updates as given in Section 4.3 is given as:

$$\mathbb{E}_\theta[r=1,a,p] = \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T} \big[P(r=1,a,p|T;\theta)\big]_t \tag{56}$$

By breaking the above summation between $t=T$ and $t=0$ to $T-1$, we get

$$\mathbb{E}_\theta[r=1,a,p] = \sum_{T=0}^{\infty} P(T) \sum_{qbs} \hat{R}_{sab}\pi_{ap}\pi_{bq}\alpha_T(p,q,s) + \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T-1} \sum_{p'q's'} \beta_{T-t-1}(p',q',s') P_t(a,p,p',q',s') \tag{57}$$

In the above equation, we marginalized the last time slice over the variables (q,b,s). For the intermediate time slice t, we condition upon the variables (p',q',s') in the next time slice $t+1$. We now use the definition of $\hat{\alpha}$ and move the summation over time T inside for the last time slice and further marginalize over the remaining variables (q,s) in the intermediate slice t:

$$\mathbb{E}_\theta[r=1,a,p] = \sum_{q,b,s} \hat{R}_{sab}\pi_{ap}\pi_{bq}\hat{\alpha}(p,q,s) + \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T-1} \sum_{p'q's'sq} \beta_{T-t-1}(p',q',s')\pi_{ap} P(p',q',s'|a,p,q,s)\alpha_t(p,q,s) \tag{58}$$

Upon further marginalizing over the joint observations $y'z'$ and simplifying we get:

$$\mathbb{E}_\theta[r=1,a,p] = \pi_{ap} \sum_{qs} \Bigg[\sum_b \hat{R}_{sab}\pi_{bq}\hat{\alpha}(p,q,s) + \sum_{p'q's'y'z'} \sum_{T=0}^{\infty} P(T) \sum_{t=0}^{T-1} \beta_{T-t-1}(p',q',s') P(s'|a,q,s)\lambda_{p'py'}\lambda_{q'qz'} P(y'z'|a,q,s')\alpha_t(p,q,s) \Bigg] \tag{59}$$

We resolve the above time summation, as Toussaint et al. (2006), based on the fact that:

$$\sum_{T=0}^{\infty}\sum_{t=0}^{T-1} f(T-t-1)g(t) = \sum_{t=0}^{\infty}\sum_{T=t+1}^{\infty} f(T-t-1)g(t)$$

and then setting $\tau = T-t-1$ to get $\sum_{t=0}^{\infty} g(t) \sum_{\tau=0}^{\infty} f(\tau)$.
Finally we get:

$$\mathbb{E}_\theta[r=1,a,p] = \pi_{ap} \sum_{qs} \hat{\alpha}(p,q,s) \Bigg[\sum_b \hat{R}_{sab}\pi_{bq} + \frac{\gamma}{1-\gamma} \sum_{p'q's'y'z'} \hat{\beta}(p',q',s')\lambda_{p'py'}\lambda_{q'qz'} P(s'|a,q,s) P(y'z'|a,q,s') \Bigg] \tag{60}$$

The product $P(s'|a,q,s)P(y'z'|a,q,s')$ can be further simplified by marginalizing out over actions b of agent 2 as follows:

$$\mathbb{E}_\theta[r=1,a,p] = \pi_{ap} \sum_{qs} \hat{\alpha}(p,q,s) \Bigg[\sum_b \hat{R}_{sab}\pi_{bq} + \frac{\gamma}{1-\gamma} \sum_{p'q's'y'z'} \hat{\beta}(p',q',s')\lambda_{p'py'}\lambda_{q'qz'} \sum_b O_{y'z's'ab}\pi_{bq}P_{s'sab} \Bigg]$$

The expectation $\mathbb{E}_\theta[r=1,b,q]$ for the other agent can be found similarly by the analogue of the above equation.

A.3 Controller Node Transition Parameter Updates

The expectation required for the controller node transition parameters is as follows:

$$\mathbb{E}_\theta[r{=}1,p,\bar{p},y] = \sum_{T=1}^{\infty} P(T) \sum_{t=1}^{T} P(r{=}1,p_t{=}p,p_{t-1}{=}\bar{p},y_t{=}y|T;\theta) \tag{61}$$

By marginalizing over the variables (q,s) for the current time slice t, we get

$$\mathbb{E}_\theta[r{=}1,p,\bar{p},y] = \sum_{T=1}^{\infty} P(T) \sum_{t=1}^{T} \sum_{sq} \beta_{T-t}(p,q,s)P_t(p,\bar{p},y,s,q|T;\theta) \tag{62}$$

By further marginalizing over the variables $(\bar{s},\bar{q})$ for the previous time slice of t and over the observations z of the other agent, we get

$$\mathbb{E}_\theta[r{=}1,p,\bar{p},y] = \lambda_{p\bar{p}y} \sum_{T=1}^{\infty} P(T) \sum_{t=1}^{T} \sum_{sq\bar{s}\bar{q}z} \beta_{T-t}(p,q,s)\lambda_{q\bar{q}z} P(yz|\bar{p},\bar{q},s)P(s|\bar{p},\bar{q},\bar{s})\alpha_{t-1}(\bar{p},\bar{q},\bar{s}) \tag{63}$$

The above equation can be further simplified by marginalizing the product $P(yz|\bar{p},\bar{q},s)P(s|\bar{p},\bar{q},\bar{s})$ over actions a and b of both the agents as follows:

$$\mathbb{E}_\theta[r{=}1,p,\bar{p},y] = \lambda_{p\bar{p}y} \sum_{T=1}^{\infty} P(T) \sum_{t=1}^{T} \sum_{sq\bar{s}\bar{q}z} \beta_{T-t}(p,q,s)\lambda_{q\bar{q}z}\alpha_{t-1}(\bar{p},\bar{q},\bar{s}) \sum_{ab} O_{yzsab}P_{s\bar{s}ab}\pi_{a\bar{p}}\pi_{b\bar{q}} \tag{64}$$

Upon resolving the time summation as before, we get the final expression as:

$$\mathbb{E}_\theta[r{=}1,p,\bar{p},y] = \lambda_{p\bar{p}y} \sum_{sq\bar{s}\bar{q}z} \hat{\alpha}(\bar{p},\bar{q},\bar{s})\hat{\beta}(p,q,s)\lambda_{q\bar{q}z} \sum_{ab} O_{yzsab}P_{s\bar{s}ab}\pi_{a\bar{p}}\pi_{b\bar{q}} \tag{65}$$

The expectation $\mathbb{E}_\theta[r{=}1,q,\bar{q},z]$ for the other agent can be found in an analogous way.

A.4 Initial Node Distribution Update

The expectation for the initial node distribution update is given as:

$$\mathbb{E}_\theta[r=1,p] = \sum_{T=0}^{\infty} P(T)P(r{=}1, p_0{=}p|T;\theta) \tag{66}$$

The above expression can be computed as follows:

$$\begin{aligned}
\mathbb{E}_\theta[r=1,p] &= \sum_{T=0}^{\infty} P(T)[P(r=1,p|T;\theta)]_{t=0} && (67)\\
&= \sum_{T=0}^{\infty} P(T)\sum_{qs} P_0(r=1|p,q,s,T;\theta)P_0(p,q,s) && (68)\\
&= \sum_{T=0}^{\infty} P(T)\sum_{qs} \beta_T(p,s,q)\nu_p\nu_q\eta_0(s) && (69)\\
&= \nu_p \sum_{qs} \hat{\beta}(p,s,q)\nu_q\eta_0(s) && (70)
\end{aligned}$$

Appendix B. EM Derivation for the ND-POMDP Model

An n-agent ND-POMDP has the following parameters:

S $= \times_{1\le i\le n} S_i \times S_u$, where S_i is the local state of agent i; S_u is a set of uncontrollable or external states that are independent of the agents' actions. In the sensor network example, S_i is the battery level, while S_u corresponds to the set of locations where targets can be present.

A $= \times_{1\le i\le n} A_i$ where A_i is the set of actions for agent i. For the sensor network, $A_i = \{l_1,\ldots,l_k, \textit{Off}, \textit{Recharge}\}$, where $\{l_1,\ldots,l_k\}$ represents the edges in the graph which can be scanned by the given sensor agent.

Y $= \times_{1\le i\le n} Y_i$ is the joint observation set. For the sensor network case, $Y_i = \{$*target present, absent, sensor idle*$\}$. We assume that sensor i can observe its internal state S_i. This is a realistic assumption as sensors can normally monitor their own battery level. The noisy component of the observation set corresponds to target locations.

P $P(\mathbf{s}'|\mathbf{s},\mathbf{a}) = P_u(s'_u|s_u)\cdot\prod_{i=1}^{n} P_i(s'_i|s_i,s_u,a_i)$ is the transition model, where $\mathbf{a}=\langle a_1,\ldots,a_n\rangle$ is the joint action taken in joint state $\mathbf{s} = \langle s_u, s_1,\ldots,s_n\rangle$ resulting in joint state $\mathbf{s}' = \langle s'_u, s'_1,\ldots,s'_n\rangle$. The model relies on conditional (on external state) transition independence among the agents.

O $O(\mathbf{y}|\mathbf{s},\mathbf{a}) = \prod_{i=1}^{n} P_i(y_i|a_i,s_u,s_i,)$, where $\mathbf{y}$ is the joint observation after taking joint action $\mathbf{a}$ and transitioning to joint state $\mathbf{s}$. This relies on conditional observation independence.

R $R(\mathbf{s},\mathbf{a}) = \sum_l R_l(s_u,s_l,a_l)$ is the reward function, which is decomposable along subgroups of agents defined by a set of links l. If k agents $i_1,\ldots,i_k$ are members of a particular

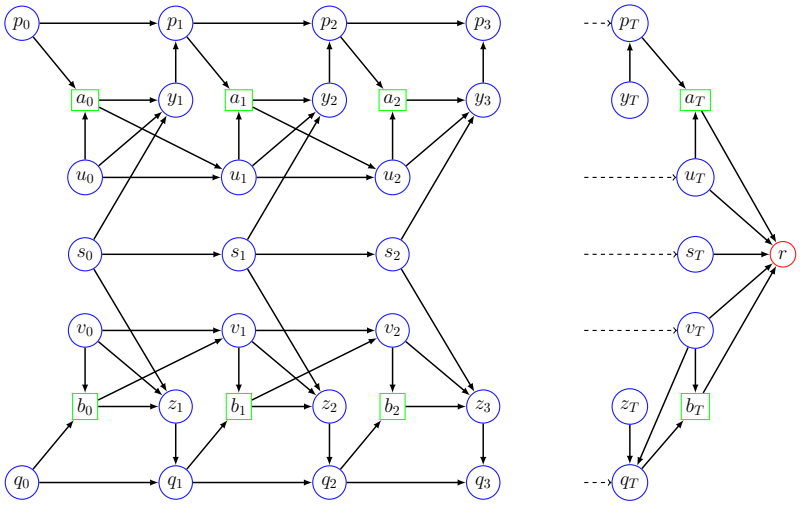


Figure 13: A T-step DBN for a link involving two agents. There is one such DBN for each link l and every time step T. In top three layers, p denotes first agent's (on link l) controller nodes, a denotes action and u denotes internal state. The layer s_i denotes the external state. In bottom three layers, q denotes second agent's controller nodes, b denotes action and v denotes the internal state

subgroup l, then $s_l = \langle s_{i_1}, \ldots, s_{i_k} \rangle$ denotes the internal states of these agents. Similarly, $a_l = \langle a_{i_1}, \ldots, a_{i_k} \rangle$. In the sensor network 5P in Figure 11, the reward is decomposed among five subgroups, one per each edge. The reward function thus induces an interaction hypergraph in which hyperlink l connects the subset of agents which form the reward component R_l.

b_o $b_o = (b_u, b_1, \ldots, b_n)$ is the initial belief for joint state $\mathbf{s} = \langle s_u, s_1, \ldots, s_n \rangle$ and $b(\mathbf{s}) = b_u(s_u) \cdot \prod_{i=1}^{n} b_i(s_i)$.

The joint-value function in ND-POMDPs satisfies the value factorization property as follows (Nair et al., 2005):

$$V(\boldsymbol{\theta}, \boldsymbol{s}) = \sum_l V_l(\theta^l, s_u, s_l).$$

There is one value factor V_l for each link l. We next present a T-step DBN for a factor l that involves two agents. This DBN is the basis for the time-dependent DBN mixture for value factor l. For ease of exposition, the nodes of the controller of one agent are denoted by p and those of the other agent by q. The internal states, actions, observations of the first agent are denoted by u, a, y respectively and v, b, z denote the same for the second agent. The external state is denoted by s. As shown in Section 5.4.1, the E step of the EM algorithm requires separate inference, one for each value factor V_l. Therefore, we only derive the inference required in the time dependent mixture corresponding to the DBN in Figure 13. Notice that it differs from the inference for a two-agent general Dec-POMDP due to the presence of conditional transition and observation independence. This property will be exploited during the E-step. The policy parameters to be optimized are defined for

agent 1 (analogously for agent 2 as well) as:

$$\pi_{apu} = P(a|p,u) \tag{71}$$
$$\lambda_{p'py} = P(p'|p,y) \tag{72}$$
$$\nu_p = P(p_{t=0} = p) \tag{73}$$

Notice that we also included the internal state u in the action parameter π_{apu}. This represents more expressive and accurate policy as agents have full observability of the internal state. Next, similar to Section 4.4, we define the following Markovian chain in the DBN of Figure 13:

$$P(p',q',s',u',v'|p,q,s,u,v) = P(p',u'|p,u,s)P(q',v'|q,v,s)P(s'|s) \tag{74}$$
$$P(p',u'|p,u,s) = \sum_{a,y} \lambda_{p'py} P_{u'au} P_{yasu} \pi_{apu} \tag{75}$$

$P(q',v'|q,v,s)$ can be expressed similarly. These transitions are independent of time as we use a stationary policy. Based on these transitions, we define the forward α messages as follows: $\alpha_t = P_t(p,q,s,u,v;\theta)$. Intuitively, α_t represents the probability that controllers of agents on the link are in state (p,q), their internal state is (u,v) and the external state is s at time t. These messages are defined as:

$$\alpha_0(p,q,s,u,v) = \nu_p \nu_q b_o(s,u,v) \tag{76}$$
$$\alpha_t(p',q',s',u',v') = \sum_{p,q,s,u,v} P(p',q',s',u',v'|p,q,s,u,v)\alpha_{t-1}(p,q,s,u,v) \tag{77}$$

Similarly backward β messages are defined as follows: $\beta_\tau = P_{T-\tau}(r{=}1|p,q,s,u,v;\theta)$, with $\tau = 0$ representing the tail end of each DBN. As shown by Toussaint et al. (2006), thanks to this notation, all the DBNs share the same tail. Hence we only need one sweep to compute the β messages. We get:

$$\beta_0(p,q,s,u,v) = \sum_{a,b} \hat{R}_{suvab} \pi_{apu} \pi_{bqv} \tag{78}$$
$$\beta_\tau(p,q,s,u,v) = \sum_{p',q',s',u',v'} \beta_{\tau-1}(p',q',s',u',v') P(p',q',s',u',v'|p,q,s,u,v) \tag{79}$$

β_τ represents the normalized expected reward for the link when the controllers of the agents are in state (p,q), their internal state is (u,v) and the external state is s at time $T-\tau$, given the policy parameters θ. Based on α and β messages, we also calculate two more quantities:

$$\hat{\alpha}(p,q,s,u,v) = \sum_{t=0}^{\infty} P(T{=}t)\alpha_t(p,q,s,u,v), \tag{80}$$
$$\hat{\beta}(p,q,s,u,v) = \sum_{\tau=0}^{\infty} P(T{=}\tau)\beta_\tau(p,q,s,u,v). \tag{81}$$

The cutoff time for α and β message propagation can be fixed as shown in Section 4.4. As shown in Section 5.4.1, the expectation required for action updates is the following in the time dependent DBN mixture for the link l:

$$\mathbb{E}^l_\theta[r=1,a,p,u]=\sum_{T=0}^{\infty}P(T)\sum_{t=0}^{T}P(r=1,a_t=a,p_t=p,u_t=u|T;\theta^l) \tag{82}$$

Breaking the inner summation for the last time step T and the remainder, we get:

$$=\sum_{T=0}^{\infty}P(T)P_T(r=1,a,p,u)+\sum_{T=0}^{\infty}P(T)\sum_{t=0}^{T-1}P_t(r=1,a,p,u)$$
$$=\sum_{T=0}^{\infty}\sum_{svqb}\hat{R}_{suvab}\pi_{apu}\pi_{bqv}\alpha_T(p,q,s,u,v)+\sum_{T=0}^{\infty}P(T)\sum_{t=0}^{T-1}\sum_{p'q's'u'v'}\beta_{T-t-1}(p',q',s',u',v')P_t(a,p,u,p',q',s',u',v')$$

For the last equality, we marginalized over the variables in the intermediate time slice. By moving the summation over time T inside for the last time slice, and further marginalizing the intermediate time slice t over (q,s,v), we get:

$$=\pi_{apu}\sum_{svqb}\hat{R}_{suvab}\pi_{bqv}\hat{\alpha}(p,q,s,u,v)+\sum_{T=0}^{\infty}P(T)\sum_{t=0}^{T-1}\sum_{qvsp'q's'u'v'}\beta_{T-t-1}(p',q',s',u',v')P(p',u'|a,p,u,s)$$
$$P(q',v'|q,v,s)P_{s's}\pi_{apu}\alpha_t(p,q,s,u,v)$$

Upon resolving the time summation for the second part of the equation (Toussaint et al., 2006), we get the final expression:

$$\mathbb{E}^l_\theta[r=1,a,p,u]=\pi_{apu}\sum_{qsv}\hat{\alpha}(p,q,s,u,v)\Bigg[\sum_b\hat{R}_{suvab}\pi_{bqv}+\frac{\gamma}{1-\gamma}$$
$$\sum_{p'q's'u'v'}\hat{\beta}(p',q',s',u',v')P(p',u'|a,p,u,s)P(q',v'|q,v,s)P_{s's}\Bigg]$$

The expectation $\mathbb{E}^l_\theta[r=1,b,q,v]$ for the other agent can be calculated in an analogous manner.

We now derive the expectation for controller node transition update:

$$\mathbb{E}^l_\theta[r=1,p,\bar{p},y]=\sum_{T=1}^{\infty}P(T)\sum_{t=1}^{T}P(r=1,p_t=p,p_{t-1}=\bar{p},y_t=y|T;\theta^l) \tag{83}$$

By simplifying the above equation, we get:

$$=\sum_{T=1}^{\infty}P(T)\sum_{t=1}^{T}\sum_{qsuv}\beta_{T-t}(p,q,s,u,v)P_t(p,q,s,u,v,\bar{p},y)$$

Upon further marginalizing over the $(\bar{q}, \bar{s}, \bar{u}, \bar{v})$, we get:

$$\begin{aligned}
&= \sum_{T=1}^{\infty} P(T) \sum_{t=1}^{T} \sum_{\bar{q}\bar{s}\bar{u}\bar{v}qsuv} \Big[\beta_{T-t}(p,q,s,u,v) P_t(p,q,s,u,v|\bar{p},\bar{q},\bar{s},\bar{u},\bar{v},y) \cdot \\
&\qquad\qquad P(y|\bar{p},\bar{u},\bar{s}) \alpha_{t-1}(\bar{p},\bar{q},\bar{s},\bar{u},\bar{v}) \Big] \\
&= \sum_{T=1}^{\infty} P(T) \sum_{t=1}^{T} \lambda_{p\bar{p}y} \sum_{\bar{q}\bar{s}\bar{u}\bar{v}qsuv} \Big[\beta_{T-t}(p,q,s,u,v) \alpha_{t-1}(\bar{p},\bar{q},\bar{s},\bar{u},\bar{v}) P(y|\bar{p},\bar{u},\bar{s}) \cdot \\
&\qquad\qquad P(u|\bar{p},\bar{u},y) P_{s\bar{s}} P(q,v|\bar{q},\bar{v},\bar{s}) \Big] \\
&= \lambda_{p\bar{p}y} \sum_{\bar{q}\bar{s}\bar{u}\bar{v}qsuv} \hat{\beta}(p,q,s,u,v) \hat{\alpha}(\bar{p},\bar{q},\bar{s},\bar{u},\bar{v}) P(u,y|\bar{p},\bar{u},\bar{s}) P_{s\bar{s}} P(q,v|\bar{q},\bar{v},\bar{s})
\end{aligned}$$

where $P(u, y|\bar{p}, \bar{u}, \bar{s})$, $P(q, v|\bar{q}, \bar{v}, \bar{s})$ are defined as:

$$P(u,y|\bar{p},\bar{u},\bar{s}) = \sum_{\bar{a}} P_{u\bar{a}\bar{u}} P_{y\bar{a}\bar{s}\bar{u}} \pi_{\bar{a}\bar{p}\bar{u}} \tag{84}$$

$$P(q,v|\bar{q},\bar{v},\bar{s}) = \sum_{\bar{b}z} \lambda_{q\bar{q}z} P_{v\bar{b}\bar{v}} P_{z\bar{b}\bar{s}\bar{v}} \pi_{\bar{b}\bar{q}\bar{v}} \tag{85}$$

The final equation for $\mathbb{E}^l_\theta[r=1, p, \bar{p}, y]$ is given by:

$$\mathbb{E}^l_\theta[r=1,p,\bar{p},y] = \lambda_{p\bar{p}y} \sum_{\bar{q}\bar{s}\bar{u}\bar{v}qsuv} \hat{\beta}(p,q,s,u,v) \hat{\alpha}(\bar{p},\bar{q},\bar{s},\bar{u},\bar{v}) P(u,y|\bar{p},\bar{u},\bar{s}) P_{s\bar{s}} P(q,v|\bar{q},\bar{v},\bar{s}) \tag{86}$$

References

Amato, C., Bernstein, D. S., & Zilberstein, S. (2007). Solving POMDPs using quadratically constrained linear programs. In *International Joint Conference on Artificial Intelligence*, pp. 2418–2424.

Amato, C., Bernstein, D. S., & Zilberstein, S. (2010). Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*, *21*(3), 293–320.

Becker, R., Zilberstein, S., & Lesser, V. (2004). Decentralized Markov decision processes with event-driven interactions. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 302–309.

Becker, R., Zilberstein, S., Lesser, V., & Goldman, C. V. (2003). Transition-independent decentralized Markov decision processes. In *Proceedings of the 2nd International Conference on Autonomous Agents and Multi Agent Systems*, pp. 41–48.

Becker, R., Zilberstein, S., Lesser, V., & Goldman, C. V. (2004). Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, *22*, 423–455.

Bernstein, D. S., Amato, C., Hansen, E. A., & Zilberstein, S. (2009). Policy iteration for decentralized control of Markov decision processes. *Journal of Artificial Intelligence Research*, *34*, 89–132.

Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, *27*, 819–840.

Bertsekas, D. P. (1999). *Nonlinear Programming* (2nd edition). Athena Scientific.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA.

Cheng, Q., Liu, Q., Chen, F., & Ihler, A. (2013). Variational planning for graph-based MDPs. In *Advances in Neural Information Processing Systems*, pp. 2976–2984.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical society, Series B*, *39*(1), 1–38.

Dibangoye, J. S., Amato, C., Doniec, A., & Charpillet, F. (2013a). Producing efficient error-bounded solutions for transition independent decentralized MDPs. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*, pp. 539–546.

Dibangoye, J. S., Amato, C., Buffet, O., & Charpillet, F. (2013b). Optimally solving Dec-POMDPs as continuous-state MDPs. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 90–96.

Dibangoye, J. S., Buffet, O., & Charpillet, F. (2014). Error-bounded approximations for infinite-horizon discounted decentralized POMDPs. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 338–353.

Dibangoye, J. S., Mouaddib, A.-I., & Chaib-draa, B. (2009). Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 569–576.

Eker, B., & Akin, H. L. (2013). Solving decentralized POMDP problems using genetic algorithms. *Autonomous Agents and Multi-Agent Systems*, *27*(1), 161–196.

Ghahramani, Z., & Jordan, M. I. (1995). Factorial hidden Markov models. In *Advances in Neural Information Processing Systems*, Vol. 8, pp. 472–478.

Gill, P. E., Murray, W., & Saunders, M. A. (2002). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, *12*(4), 979–1006.

Goldman, C. V., & Zilberstein, S. (2004). Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, *22*, 143–174.

Grzes, M., Poupart, P., & Hoey, J. (2013). Controller compilation and compression for resource constrained applications. In *International Conference on Algorithmic Decision Theory*, pp. 193–207.

Grześ, M., Poupart, P., Yang, X., & Hoey, J. (2015). Energy efficient execution of POMDP policies. *IEEE Transactions on Cybernetics*, *preprint*.

Hansen, E. A., Bernstein, D. S., & Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th AAAI Conference on Artificial Intelligence*, pp. 709–715.

Hoffman, M., de Freitas, N., Doucet, A., & Peters, J. (2009a). An expectation maximization algorithm for continuous Markov decision processes with arbitrary rewards. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 232–239.

Hoffman, M., Kueck, H., de Freitas, N., & Doucet, A. (2009b). New inference strategies for solving Markov decision processes using reversible jump MCMC. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, pp. 223–231.

Kiselev, I., & Poupart, P. (2014a). Policy optimization by marginal-map probabilistic inference in generative models. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*, pp. 1611–1612.

Kiselev, I., & Poupart, P. (2014b). POMDP planning by marginal-MAP probabilistic inference in generative models. In *Proceedings of the 2014 AAMAS Workshop on Adaptive Learning Agents*.

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

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Kumar, A., & Zilberstein, S. (2009a). Constraint-based dynamic programming for decentralized POMDPs with structured interactions. In *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*, pp. 561–568.

Kumar, A., & Zilberstein, S. (2009b). Event-detecting multi-agent MDPs: Complexity and constant-factor approximation. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pp. 201–207.

Kumar, A., & Zilberstein, S. (2010a). Anytime planning for decentralized POMDPs using expectation maximization. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 294–301.

Kumar, A., & Zilberstein, S. (2010b). Point-based backup for decentralized POMDPs: Complexity and new algorithms. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1315–1322.

Kumar, A., Zilberstein, S., & Toussaint, M. (2011). Scalable multiagent planning using probabilistic inference. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 2140–2146.

Lauritzen, S. L., & Nilsson, D. (2001). Representing and solving decision problems with limited information. *Management Science*, *47*, 1235–1251.

Liu, Q., & Ihler, A. T. (2012). Belief propagation for structured decision making. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012*, pp. 523–532.

Liu, Q., & Ihler, A. T. (2013). Variational algorithms for marginal MAP. *Journal of Machine Learning Research*, *14*(1), 3165–3200.

MacDermed, L. C., & Isbell, C. (2013). Point based value iteration with optimal belief compression for Dec-POMDPs. In *Advances in Neural Information Processing Systems*, pp. 100–108.

Marecki, J., Gupta, T., Varakantham, P., Tambe, M., & Yokoo, M. (2008). Not all agents are equal: Scaling up distributed POMDPs for agent networks. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 485–492.

Mostafa, H., & Lesser, V. R. (2009). Offline planning for communication by exploiting structured interactions in decentralized MDPs. In *International Conference on Intelligent Agent Technology*, pp. 193–200.

Mostafa, H., & Lesser, V. R. (2011). Compact mathematical programs for DEC-MDPs with structured agent interactions. In *International Conference on Uncertainty in Artificial Intelligence*, pp. 523–530.

Mundhenk, M., Goldsmith, J., Lusena, C., & Allender, E. (2000). Complexity of finite-horizon Markov decision process problems. *J. ACM*, *47*(4), 681–720.

Nair, R., Varakantham, P., Tambe, M., & Yokoo, M. (2005). Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proceedings of the 20th AAAI Conference on Artificial Intelligence*, pp. 133–139.

Nair, R., Tambe, M., Yokoo, M., Pynadath, D., Marsella, S., Nair, R., & Tambe, M. (2003). Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pp. 705–711.

Oliehoek, F. A., Spaan, M. T. J., Amato, C., & Whiteson, S. (2013). Incremental clustering and expansion for faster optimal planning in Dec-POMDPs. *Journal of Artificial Intelligence Research*, *46*, 449–509.

Oliehoek, F. A., Spaan, M. T. J., & Vlassis, N. A. (2008). Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, *32*, 289–353.

Oliehoek, F. A., Whiteson, S., & Spaan, M. T. J. (2013). Approximate solutions for factored dec-pomdps with many agents. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pp. 563–570.

Pajarinen, J., Hottinen, A., & Peltonen, J. (2014). Optimizing spatial and temporal reuse in wireless networks by decentralized partially observable Markov decision processes. *IEEE Transactions on Mobile Computing*, *13*(4), 866–879.

Pajarinen, J., & Peltonen, J. (2011a). Efficient planning for factored infinite-horizon DEC-POMDPs. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 325–331.

Pajarinen, J., & Peltonen, J. (2011b). Periodic finite state controllers for efficient POMDP and DEC-POMDP planning. In *Advances in Neural Information Processing Systems*, pp. 2636–2644.

Pajarinen, J., & Peltonen, J. (2013). Expectation maximization for average reward decentralized POMDPs. In *Proceedings of the European Conference on Machine Learning*, pp. 129–144.

Pineau, J., Gordon, G., & Thrun, S. (2006). Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, *27*, 335–380.

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

Poupart, P., Lang, T., & Toussaint, M. (2011). Analyzing and escaping local optima in planning as inference for partially observable domains. In *European Conference on Machine Learning*, pp. 613–628.

Seuken, S., & Zilberstein, S. (2007). Memory-bounded dynamic programming for DEC-POMDPs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligences*, pp. 2009–2015.

Smith, T., & Simmons, R. (2004). Heuristic search value iteration for POMDPs. In *International Conference on Uncertainty in Artificial Intelligence*, pp. 520–527.

Toussaint, M., Harmeling, S., & Storkey, A. (2006). Probabilistic inference for solving (PO)MDPs. Tech. rep. EDIINF-RR-0934, University of Edinburgh, School of Informatics.

Toussaint, M., Charlin, L., & Poupart, P. (2008). Hierarchical POMDP controller optimization by likelihood maximization. In *International Conference on Uncertainty in Artificial Intelligence*, pp. 562–570.

Toussaint, M., & Storkey, A. J. (2006). Probabilistic inference for solving discrete and continuous state Markov decision processes. In *International Conference on Machine Learning*, pp. 945–952.

Varakantham, P., Marecki, J., Yabu, Y., Tambe, M., & Yokoo, M. (2007). Letting loose a SPIDER on a network of POMDPs: Generating quality guaranteed policies. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1–8.

Varakantham, P., Kwak, J., Taylor, M. E., Marecki, J., Scerri, P., & Tambe, M. (2009). Exploiting coordination locales in distributed POMDPs via social model shaping. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling*, pp. 313–320.

Witwicki, S. J. (2011). *Abstracting Influences for Efficient Multiagent Coordination Under Uncertainty*. Ph.D. thesis, Department of Computer Science, University of Michigan, Ann Arbor.

Witwicki, S. J., & Durfee, E. H. (2010). Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling*, pp. 185–192.

Witwicki, S. J., & Durfee, E. H. (2011). Towards a unifying characterization for quantifying weak coupling in Dec-POMDPs. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pp. 29–36.

Wu, F., Zilberstein, S., & Chen, X. (2010). Trial-based dynamic programming for multi-agent planning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pp. 908–914.

Wu, F., Zilberstein, S., & Jennings, N. R. (2013). Monte-Carlo expectation maximization for decentralized POMDPs. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pp. 397–403.