Efficient Planning for Factored Infinite-Horizon DEC-POMDPs

Joni Pajarinen
Aalto University, Department of Information and Computer Science,
P.O. Box 15400, FI-00076 Aalto, Finland
Joni.Pajarinen@tkk.fi

Jaakko Peltonen
Aalto University, Department of Information and Computer Science, Helsinki Institute for Information Technology HIIT,
P.O. Box 15400, FI-00076 Aalto, Finland
Jaakko.Peltonen@tkk.fi

Abstract

Decentralized partially observable Markov decision processes (DEC-POMDPs) are used to plan policies for multiple agents that must maximize a joint reward function but do not communicate with each other. The agents act under uncertainty about each other and the environment. This planning task arises in optimization of wireless networks, and other scenarios where communication between agents is restricted by costs or physical limits. DEC-POMDPs are a promising solution, but optimizing policies quickly becomes computationally intractable when problem size grows. Factored DEC-POMDPs allow large problems to be described in compact form, but have the same worst case complexity as non-factored DEC-POMDPs. We propose an efficient optimization algorithm for large factored infinite-horizon DEC-POMDPs. We formulate expectation-maximization based optimization into a new form, where complexity can be kept tractable by factored approximations. Our method performs well, and it can solve problems with more agents and larger state spaces than state of the art DEC-POMDP methods. We give results for factored infinite-horizon DEC-POMDP problems with up to 10 agents.

1 Introduction

Planning for multiple agents arises in several domains such as robotics (where agents are robots) or wireless networking (where agents are devices transmitting over the network). The task is especially difficult when the agents have a *joint goal*, but they have only *uncertain knowledge* of each other and the environment, and they *cannot directly communicate* to share knowledge. The lack of communication can be due to physical limits of the agents, or it may be necessary to minimize costs like power drain. The agents must then act separately, under uncertainty about each other and the environment.

For example, in wireless networking devices may have a joint goal of minimizing transmission delays in the network. Each device senses only local transmissions and does not know upcoming data streams of other devices. Simultaneous transmissions from several devices can collide, causing delays. The task is to optimize an overall action policy, where each device can follow its own part of the policy without inter-device communication, and the overall policy minimizes delays.

Problems such as optimizing the policy of several wireless radio devices can be framed as DEC-POMDP problems [Oliehoek, 2010] and solving the DEC-POMDP yields the optimal policy for each agent. However in DEC-POMDPs, the computational complexity of optimization grows rapidly in terms of the number of agents and the number of variables describing the environment. Current algorithms can only handle very few agents.

We will show how to optimize policies efficiently for factored infinite-horizon DEC-POMDPs with large state spaces and many agents. *Factored* means we exploit independence relationships in transitions of state variables, observations, and rewards. Although many DEC-POMDP problems are factored, such problems are hard for current methods, since any momentary factorization of the state does not survive into the future. We will show we can exploit factorization better than previously by reformulating the learning and using approximations that maintain the factorization. To our knowledge this is the first investigation into factored infinite-horizon DEC-POMDPs. Our approach builds on the work of using Expectation Maximization (EM) for optimizing finite state controllers (FSCs) for POMDPs [Toussaint *et al.*, 2006] and for DEC-POMDPs [Kumar and Zilberstein, 2010a]. Our proposed methods have *polynomial complexity with respect to the number of state variables and number of agents*.

2 Background

DEC-POMDPs are a general and hence computationally complicated class of models for multi-agent sequential decision making. At simplest, Markov decision processes (MDPs) can optimize policies for one agent that knows the full state of the environment; partially observable Markov decision processes (POMDPs) can optimize policies for one agent under uncertainty; and DEC-POMDPs treat the even harder case of several agents. The harder problems need much computation: the worst case computational complexity of finding solutions is PSPACE-hard for finite-horizon POMDPs and NEXP-complete for DEC-POMDPs [Allen and Zilberstein, 2009]. DEC-POMDPs are harder to solve than POMDPs since in DEC-POMDPs each agent must consider more effects of its actions: not only effects on the world state, but also effects on the observations and thus the actions of other agents. In POMDPs the agent can maintain a probability distribution ("belief") over the world state, but in DEC-POMDPs all the observation history is needed for optimal decisions.

Efficient point based algorithms exist for solving large POMDPs [Pajarinen *et al.*, 2010] and for finite-horizon DEC-POMDPs [Kumar and Zilberstein, 2010b], but not for infinite-horizon DEC-POMDPs. To keep the policy size bounded, state of the art infinite-horizon DEC-POMDP methods [Amato *et al.*, 2007; Kumar and Zilberstein, 2010a] keep the policy of an agent as a stochastic finite state controller (FSC). Amato *et al.* [2007] formulate FSC optimization as a non-linear constraint satisfaction (NLP) problem and find FSC parameters by an NLP solver. Kumar and Zilberstein [2010a] optimize FSC parameters using expectation maximization (EM). The EM approach scales well and is flexible. It can for example be used with continuous probability distributions [Toussaint *et al.*, 2006]. Bernstein *et al.* [2005] use linear programming to improve the value of each FSC in turn with other FSCs fixed. Szer *et al.* [2005] find deterministic finite state controllers of fixed size by a best-first search. Best accuracy and scaling have been demonstrated for the NLP and EM methods.

In many real world problems the world state, observations, and rewards can be divided into many variables. For example, in a wireless network the size of each packet buffer (see Section 4) can be described as a separate state variable, instead of a single world state describing all buffers. For POMDPs, efficient algorithms exist for solving large factored infinite-horizon problems [Pajarinen *et al.*, 2010], but for DEC-POMDPs, similar research exists only for factored *finite-horizon* problems [Oliehoek *et al.*, 2008; Oliehoek, 2010]; we are not aware of research on efficiently solving *factored infinite-horizon DEC-POMDPs*. Although the specification of a general factored DEC-POMDP problem can be simpler than of a non-factored problem, the worst case computational complexity is the same [Allen and Zilberstein, 2009]: a special case where agents act independently but with the same rewards has lower computational complexity, but for most interesting real world DEC-POMDP problems the worst case complexity is NEXP-complete [Allen and Zilberstein, 2009].

We next present definitions for infinite-horizon DEC-POMDPs and factored DEC-POMDPs, and the recent EM approach of Kumar and Zilberstein [2010a].

2.1 Infinite-Horizon DEC-POMDP: Definition

An infinite-horizon DEC-POMDP is specified by a tuple $\langle\{\alpha_i\}, S, \{A_i\}, P, \{\Omega_i\}, O, R, b_0, \gamma\rangle$. Each α_i is an agent, S is the set of world states, and A_i and Ω_i are the sets of actions and observations available to agent α_i. The Markovian transition function $P(s'|s,\vec{a})$ specifies the probability for moving from state s to state s', given the actions of all agents which are denoted as the joint action $\vec{a} = \langle a_1, \ldots, a_N\rangle$, where N is the number of agents. The observation function $O(\vec{o}|s',\vec{a})$ is the probability for the agents to observe $\vec{o} = \langle o_1, \ldots, o_N\rangle$ (where each o_i is the observation of agent i), when actions $\vec{a}$ caused a transition to state s'. $R(s,\vec{a})$ is the real-valued global reward for executing actions $\vec{a}$ in state s. Lastly, $b_0(s)$ is the initial state distribution and γ is the discount factor.

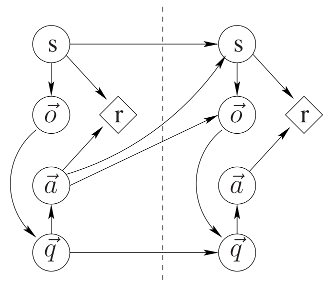

Figure 1: Influence diagram for a DEC-POMDP with finite state controllers $\vec{q}$, states s, joint observations $\vec{o}$, joint actions $\vec{a}$ and reward r. A dotted line separates two time slices.

For brevity we denote transition probabilities with $P_{s's\vec{a}}$, observation probabilities with $P_{\vec{o}s'\vec{a}}$, reward functions with R_{sa}, and other agents than i with $\bar{i}$. At each time step each agent i performs action a_i, the world transitions from state s to state s' with probability $P_{s's\vec{a}}$, and the agents receive observations $\vec{o}$ with probability $P_{\vec{o}s'\vec{a}}$. The goal is to compute a joint policy π for the agents that maximizes the expected discounted infinite-horizon reward $E\left[\sum_{t=0}^{\infty} \gamma^t R(s,\vec{a})|\pi\right]$.

The joint policy to be optimized is a set of stochastic finite state controllers (FSCs), one for each agent. A FSC resembles a Markov model, but transitions depend on observations from the environment, and the FSC emits actions affecting the environment. The FSC for agent i is specified by the tuple $\langle Q_i, \nu_{q_i}, \pi_{a_i q_i}, \lambda_{q_i' q_i o_i}\rangle$, where Q_i is the set of FSC nodes q_i, ν_{q_i} is the initial distribution $P(q_i)$ over nodes, $\pi_{a_i q_i}$ is the probability $P(a_i|q_i)$ to execute action a_i when in node q_i, and $\lambda_{q_i' q_i o_i}$ is the probability $P(q_i'|q_i, o_i)$ to move from node q_i to node q_i' when observing o_i. Figure 1 illustrates the setup.

2.2 Factored DEC-POMDPs: Definition

In a factored DEC-POMDP [Allen and Zilberstein, 2009; Oliehoek, 2010] states and observations are described as a combination of several variables, and the reward is a sum of reward sub-functions. The factored DEC-POMDP specification can be compact since states and observation variables and reward sub-functions depend only on few variables. The state s is a combination of variables $s = s_1 \times \cdots \times s_M$ and observation o_i for agent i is a combination of variables $o_i = o_{i,1} \times \cdots \times o_{i,K}$. Transition probabilities are written as $P(s'|s,\vec{a}) = \prod_i P(s_i'|\text{Pa}(s_i'))$, where $\text{Pa}(s_i)$ denotes the set of state and action variables that s_i' depends on; observation probabilities are written as $O(\vec{o}|s',\vec{a}) = \prod_i \prod_j P(o_{i,j}|\text{Pa}(o_{i,j}))$, where $\text{Pa}(o_{i,j})$ is the set of state and action variables that $o_{i,j}$ depends on. Reward functions are defined as functions over subsets S_k of state and action variables: $R(s,\vec{a}) = \sum_{k=0}^{K} R_k(S_k)$, where R_k is a reward function operating on subset S_k.

A factored DEC-POMDP can be represented as a dynamic Bayesian network (DBN) [Murphy, 2002]. The right hand side of Figure 2 shows the DBN of a wireless network problem with two agents for the first two time slices. For clarity

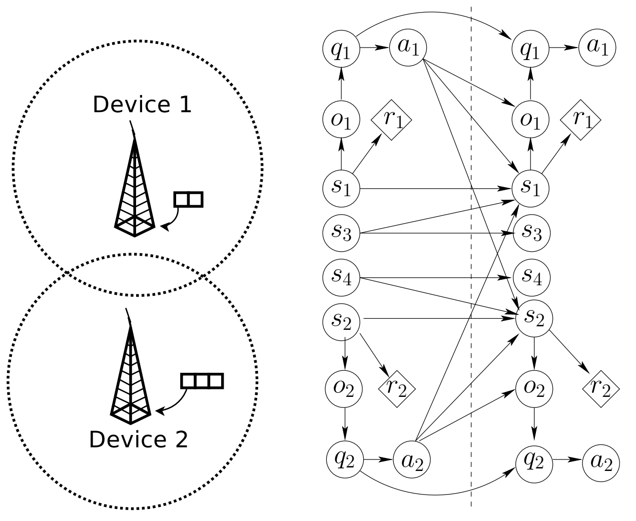

Figure 2: Wireless network DEC-POMDP example. Left: Illustration. Right: Influence diagram. Two wireless devices cause interference to each other, when transmitting simultaneously. A packet is successfully transmitted only if only one device is transmitting. Each device has a packet queue (state variables s_1 and s_2). A Markov process (state variables s_3 and s_4) inserts packets into the queue. The action (a_1, a_2) is chosen using the FSC state (q_1, q_2). The reward (r_1, r_2) is the negative queue length. A dotted line separates two consecutive time steps. For clarity this problem does not have all the dependencies possible in a factored DEC-POMDP.

this problem does not have all the possible dependencies a factored DEC-POMDP problem could have.

2.3 Expectation Maximization for DEC-POMDPs

Our approach builds on the expectation maximization (EM) method for DEC-POMDPs [Kumar and Zilberstein, 2010a]. The optimization is written as an inference problem and EM iteration is used to improve stochastic finite state controllors (FSCs). We outline the method; for details, see Toussaint *et al.* [2006] and Kumar and Zilberstein [2010a]. The reward function is scaled into a probability

$$\hat{R}(r=1|s,\vec{a}) = (R(s,\vec{a}) - R_{min})/(R_{max} - R_{min}) \ , \quad (1)$$

where R_{min} and R_{max} are the minimum and maximum rewards possible and $\hat{R}(r=1|s,\vec{a})$ is the conditional probability for the binary reward r to be 1. The FSC parameters θ are then optimized by maximizing the reward likelihood $\sum_{T=0}^{\infty} P(T)P(r=1|T,\theta)$ with respect to θ, where the horizon is infinite and $P(T) = (1-\gamma)\gamma^T$. This is equivalent to maximizing expected discounted reward in the DEC-POMDP. We next describe the E-step and M-step formulas used in the EM approach.

In the E-step, alpha messages $\hat{\alpha}(\vec{q},s)$ and beta messages $\hat{\beta}(\vec{q},s)$ are computed. Intuitively, $\hat{\alpha}(\vec{q},s)$ corresponds to the discounted weighted average probability that the world is in state s and the FSCs are in nodes $\vec{q}$, when following the policy defined by the current FSCs. $\hat{\alpha}(\vec{q},s)$ can be computed by starting from the initial nodes-and-state distribution and projecting it T time steps forward: if $\alpha_t(\vec{q},s)$ denotes the probability for being in state s and nodes $\vec{q}$ at time t, we have

$$\hat{\alpha}(\vec{q},s) = \sum_{t=0}^{T} \gamma^t (1-\gamma)\alpha_t(\vec{q},s) \ . \quad (2)$$

$\hat{\beta}(\vec{q},s)$ is intuitively the expected discounted total scaled reward, starting from state s and FSC nodes $\vec{q}$. First project the reward probability t time steps backwards; denote it $\beta_t(\vec{q},s)$. We then have

$$\hat{\beta}(\vec{q},s) = \sum_{t=0}^{T} \gamma^t (1-\gamma)\beta_t(\vec{q},s) \ . \quad (3)$$

The M-step FSC parameter update rules can be written as:

$$\nu^*_{q_i} = \frac{\nu_{q_i}}{C_i} \sum_{s,q_{\bar{i}}} \hat{\beta}(\vec{q},s)\nu_{q_{\bar{i}}} b_0(s) \quad (4)$$

$$\pi^*_{a_i q_i} = \frac{\pi_{a_i q_i}}{C_{q_i}} \sum_{s,s',q_{\bar{i}},\vec{q}',\vec{o},a_{\bar{i}}} \hat{\alpha}(\vec{q},s)\pi_{a_{\bar{i}} q_{\bar{i}}} \left[R_{s\vec{a}} + \frac{\gamma}{1-\gamma} P_{s' s\vec{a}} P_{\vec{o}s'\vec{a}} \lambda_{q'_{\bar{i}} q_{\bar{i}} o_{\bar{i}}} \lambda_{q_i' q_i o_i} \hat{\beta}(\vec{q}',s')\right] \quad (5)$$

$$\lambda^*_{q'_i q_i o_i} = \frac{\lambda_{q'_i q_i o_i}}{C_{q_i o_i}} \sum_{s,s',q_{\bar{i}},q'_{\bar{i}},o_{\bar{i}},\vec{a}} \hat{\alpha}(\vec{q},s) P_{s' s\vec{a}} P_{\vec{o}s'\vec{a}} \pi_{a_{\bar{i}} q_{\bar{i}}} \pi_{a_i q_i} \lambda_{q'_{\bar{i}} q_{\bar{i}} o_{\bar{i}}} \hat{\beta}(\vec{q}',s') \ , \quad (6)$$

where C_i, $C_{q_i o_i}$, and C_{q_i} are normalizing constants and $\bar{i}$ denotes all agents except i.

Kumar and Zilberstein [2010a] give formulas for the complexity of the DEC-POMDP EM algorithm for two agents. For N agents the complexities are $\mathcal{O}(|Q|^{2N}|A|^N|O|^N|S|^2)$ for computing Markov transitions and $\mathcal{O}(T_{max}|Q|^{2N}S^2)$ for propagating alpha and beta messages. The computational complexity for Equation 5 is $\mathcal{O}(|Q|^{2N}|S|^2|A|^{N-1}|O|^N)$ and for Equation 6 $\mathcal{O}(|Q|^{2N}|S|^2|O|^N + |Q|^N|S|^2|A|^N|O|^N)$. The complexity is exponential in the number of agents, polynomial in the number of states, and for factored DEC-POMDPs, exponential in the number of state variables. This is much better than the complexity class of NEXP-complete, but to solve DEC-POMDPs with many agents and large state spaces a reduction in complexity is required.

3 Proposed Approach

We now propose our new method, based on the EM approach, to solve factored DEC-POMDPs with large state spaces and many agents. In order to apply the EM approach for such large-scale factored (DEC)-POMDPs, two main problems must be solved. **Problem 1:** In general factored (DEC)-POMDP problems, the influence between state variables grows when a belief is projected forward or backward in time; even if the problem initially has several independent factors, after some time steps, all variables depend on each other (this is usual in dynamic Bayesian networks [Murphy, 2002]). **Problem 2:** In the EM approach, scaling rewards into probabilities makes the state variables in the factored reward sub-functions depend on each other. Problem

1 causes the size of the probability distribution over states and FSC nodes to grow exponentially with the number of state variables and agents, and Problem 2 makes it grow exponentially with the number of state variables. In many problems the number of state variables depends on the number of agents, so both problems 1 and 2 make solving problems with many agents intractable.

We propose a new EM based approach with polynomial complexity in the number of state variables and agents. We make three main changes. **Change 1:** We transform Equations 4, 5, and 6 so that probabilities are projected *only forward in time*. This lets us apply efficient approximations during forward projection. **Change 2:** We use approximations that keep the probability distribution over state variables and FSC nodes in factored form during forward projections; this addresses Problem 1. **Change 3:** We keep the reward subfunctions separated, when computing probabilities; this addresses Problem 2. We next discuss the changes in detail.

3.1 Projecting Probabilities Forward

A major part in our E and M steps will be forward projection of a (factorized) probability distribution over FSC nodes and state variables, $\tilde{\alpha}_t(\vec{q}, s)$, using the dynamic Bayesian network (DBN) of the factored DEC-POMDP (Figure 2 shows an example DBN), and keeping the result factorized. We discuss this before detailing the E and M steps.

The junction tree algorithm or variable elimination can be used on the DBN between two successive time points to compute $\tilde{\alpha}_{t+1}(\vec{q}, s)$ from $\tilde{\alpha}_t(\vec{q}, s)$ [Murphy, 2002]. The complexity of the algorithm depends on the size of cliques in the junction tree; we simply computed cliques greedily for the DBN. In factored DEC-POMDPs dependencies are often initially limited to a few variables, but more dependencies quickly appear in future steps. We use approximations (details soon) to ensure that in each projection step, the found cliques are small (below a fixed size) even as time goes on or as more agents are added; the complexity then remains *polynomial in the number of state and FSC node variables* (for details see Section 3.5). After each forward projection step, we approximate the distribution $\tilde{\alpha}_{t+1}(\vec{q}, s)$ over variables $(q_1, \ldots, q_N, s_1, \ldots, s_M)$ with a factorized form: the approximation is factorized over *variable clusters* $C_1, \ldots, C_L$, where C_i is a set of FSC node and state variables. The factor for each C_i is obtained by marginalizing out variables not in C_i, which is fast to do. With fixed variable clusters, the next forward projection step is no harder than the previous one. Variables in the same cluster are in the same junction tree clique; if all variables are in one cluster, the projection is identical to the original EM method. We use two clusterings that yield fast computation.

Although we perform an approximation at each time step, the approximation error is bounded for disjoint variable clusters: the error contracts in each time step [Boyen and Koller, 1998]. The lower bound of Boyen and Koller [1998] for the contraction rate depends on the mixing rate of the stochastic process. We use stochastic FSCs; stochasticity in their actions, and in state transitions and observations, help the DEC-POMDP to have faster mixing and error contraction for factors containing FSC nodes and for factors affected by actions.

We investigate two different clusterings. **1.** The simplest approximation is to keep a cluster for each variable, called a fully factored approximation in DBNs; it decouples the FSC nodes from the world state after each update. The finite-horizon DEC-POMDP approach based on the Factored Frontier algorithm in [Oliehoek, 2010] keeps random variables also in a fully factored form. In the finite-horizon approach histories are used instead of FSC nodes. Also, in [Oliehoek, 2010] each variable is projected forward approximately, which is not efficient for FSCs, because of how transition probabilities bind variables (see Figure 2). Here one step is projected exactly and then the fully factorized approximation is made. **2.** More complex clusterings can be made by studying the structure of the DBN for two time steps. Intuitively, variables should cluster together if they influence the same variables for the future, as this reduces the number of "incoming" and "outgoing" influences across clusters from one time step to the next. Variables affecting a reward sub-function should also cluster together, because the goal is to maximize reward. In detail, we form a cluster for the variables involved in each reward function, transition probability, and observation probability; as the clustering is for state and node variables only, any action and observation variables involved in the rewards/transitions/observations are substituted with FSC node variables of the same agent. For example, rewards $R(s_2, a_2)$, transitions $P(s'_1|a_1)$, $P(s'_2|s_1)$, and observations $P(o_1|s'_1, a_1)$ would yield clusters $C_1 = s_2, q_2$, $C_2 = s_1, q_1$, $C_3 = s_1, s_2$, after pruning identical and subset clusters. For the problem in Figure 2 the clustering yields $P(q1, q2, s1, s2, s3, s4) = P(s1, s3|q1, q2)P(q1, q2, s2, s4)$.

Clustering 2 is based on the factored DBN structure of the DEC-POMDP problem. It yields overlapping clusters. Boyen and Koller [1999] investigate overlapping clusters of variables formally; they conclude that if variables have weak interaction, their dependency can be ignored without incurring a large approximation error. A set of overlapping clusters corresponds to a conditionally independent approximation of the full probability distribution. In the wireless network example of Figure 2, the variable clusters resulting from clustering 2 are (s_1, s_3, q_1, q_2) and (s_2, s_4, q_1, q_2): the packet buffer sizes and source model states are thus assumed independent between agents given their FSCs, a reasonable approximation.

The modified E- and M-steps are discussed next.

3.2 E-step

In standard EM, messages $\hat{\alpha}(\vec{q}, s)$ and $\hat{\beta}(\vec{q}, s)$ are both computed in the E-step. However, $\hat{\beta}(\vec{q}, s)$ would require additional approximation beyond what was discussed in Section 3.1, to sum up rewards efficiently. We avoid that by not computing $\hat{\beta}(\vec{q}, s)$ explicitly in the E-step; instead, we compute it implicitly as part of the M-step.

In the E-step it remains to compute $\tilde{\alpha}(\vec{q}, s)$, the approximate alpha message: we project the distribution over states and FSC nodes $\tilde{\alpha}_t(\vec{q}, s)$ forward for T time steps, keeping it factorized for each t as described in Section 3.1; we then compute the final summed discount weighted $\tilde{\alpha}(\vec{q}, s)$. T is estimated essentially as in [Kumar and Zilberstein, 2010a], but we project beliefs only forward for $2T$ time steps. Because of discounting, reward probability mass accumulation decreases exponentially with time, therefore T is small. We approximate Equation 2 to compute it in an efficient factorized way for each variable cluster separately:

$$\tilde{\alpha}^i(C_i) = \frac{1}{\sum_t \gamma^t} \sum_t \gamma^t \tilde{\alpha}^i_t(C_i) \; . \tag{7}$$

This approximation minimizes the Kullback-Leibler (KL) divergence between the approximation $\tilde{\alpha}(\vec{q}, s)$ and the original weighted sum $\hat{\alpha}(\vec{q}, s)$. Proof: the KL divergence is $KL(\tilde{\alpha}(\vec{q}, s))||\hat{\alpha}(\vec{q}, s)) = \sum_{\vec{q},s} \tilde{\alpha}(\vec{q}, s) \log \frac{\tilde{\alpha}(\vec{q},s)}{\hat{\alpha}(\vec{q},s)} = const - \sum_{\vec{q},s} \hat{\alpha}(\vec{q}, s) \log \tilde{\alpha}(\vec{q}, s)$. The log makes the product of variable cluster probabilities a sum, and the approximation for each cluster can be computed separately as in Equation 7.

3.3 M-step

In the M-step we substitute discount weighted beta messages $\hat{\beta}(\vec{q}, s)$ with forward projection and computing reward probabilities at each of the T time steps. Compared to the original EM method this multiplies the time complexity of the M-step by T, but rewards can be kept in factored form and the same factored forward projection approximations as in the E-step used. First we show the forward projection equations (equal to original EM equations) and then discuss how we use them.

Project the distribution $\nu_{q_{\bar{i}}} b_0(s)$ forward t steps, with q_i fixed for time zero. Call the projected distribution $\phi_t(\vec{q}, s|q_i^0)$, where the superscript denotes time. The M-step parameter update Equation 4 is then equal to:

$$\nu^*_{q_i} = \frac{\nu_{q_i}}{C_i} \sum_{t=0}^{T} \gamma^t \sum_{\vec{q},s,\vec{a}} \phi_t(\vec{q}, s|q_i^0 = q_i) \hat{R}_{s\vec{a}} \pi_{\vec{a}\vec{q}} \, . \tag{8}$$

Project $\hat{\alpha}(\vec{q}, s)$ forward t steps, with q_i and a_i fixed for time zero. Call the projected distribution $\phi_t(\vec{q}, s|a_i^0, q_i^0)$. The M-step parameter update Equation 5 is then equal to:

$$\pi^*_{a_i q_i} = \frac{\pi_{a_i q_i}}{C_{q_i}} \sum_{t=0}^{T} \gamma^t \sum_{\vec{q},s,\vec{a}} \phi_t(\vec{q}, s|a_i^0 = a_i, q_i^0 = q_i) \hat{R}_{s\vec{a}} \pi_{\vec{a}\vec{q}} \, . \tag{9}$$

Project $\hat{\alpha}(\vec{q}, s)$ forward t steps, with q_i', q_i, and q_i fixed for time step zero. Call the projected distribution $\phi_t(\vec{q}, s|q_i^1, q_i^0, o_i^0)$. The M-step parameter update Equation 6 is then equal to:

$$\lambda^*_{q_i' q_i o_i} = \frac{\lambda_{q_i' q_i o_i}}{C_{q_i o_i}} \sum_{t=0}^{T} \gamma^t \sum_{\vec{q},s,\vec{a}} \phi_{t+1}(\vec{q}, s|q_i^1 = q_i', q_i^0 = q_i, o_i^0 = o_i) \hat{R}_{s\vec{a}} \pi_{\vec{a}\vec{q}} \, . \tag{10}$$

Forward projection of $\phi_t(\vec{q}, s)$ is done similarly to forward projection of $\hat{\alpha}_t(\vec{q}, s)$ in the E-step, i.e. using the junction tree algorithm on the DBN of two successive time points.

In Equations 8, 9, and 10 the old parameters are multiplied with a discounted reward probability expectation in order to get the new parameters.

Note that M-step parameter update rules equal the ones in the original EM, when using only one variable cluster and the same reward scaling.

3.4 Factored Rewards

Rewards are kept in factored form. To do so, we compute the reward or reward probability for each reward sub-function $R_k(S_k)$ separately at each time step in the M-step equations 8, 9, and 10. The discounted reward sum is computed from these and scaled using minimum and maximum rewards into a discounted reward probability expectation. Computing minimum and maximum rewards is not tractable from the joint reward function for large problems, as the state space has exponential size in the number of state variables. We find a lower bound $\tilde{R}_{min}$ for the complete reward function by summing all minimum rewards of the $R_k(S_k)$ and an upper bound $\tilde{R}_{max}$ by summing all maximum rewards of the $R_k(S_k)$. The right side of Equation 8 is then

$$\sum_{t=0}^{T} \gamma^t \sum_{\vec{q},s,\vec{a}} \phi_t(\vec{q}, s|q_i^0 = q_i) \hat{R}_{s\vec{a}} \pi_{\vec{a}\vec{q}}$$
$$= \big(\sum_{t=0}^{T} \gamma^t \sum_{\vec{q},s,\vec{a}} \phi_t(\vec{q}, s|q_i^0 = q_i) \pi_{\vec{a}\vec{q}} \sum_{k=0}^{K} R_k(S_k) - \sum_{t=0}^{T} \gamma^t \tilde{R}_{min} \big) / \big(\tilde{R}_{max} - \tilde{R}_{min} \big) \; .$$

The same technique is used for Equations 9 and 10.

3.5 Properties of the approach

In general EM is guaranteed to increase or maintain the likelihood, which is proportional to the discounted reward, in each iteration. There is no such theoretical guarantee for the proposed approximations provided here. However, we expect the properties to hold better for good approximations than bad ones. In the experiments, with sufficient running time, the approximations converged to good stable solutions.

The overall computational complexity for one EM iteration for N agents is $\mathcal{O}(N(N+M)T(|Q|+|A||Q|+|Q||O||Q|) \cdot \text{maxCliquePotential})$ where maxCliquePotential is described below. Minimum memory requirements are very small and the algorithm is parallelizable, because we can do computations for parameters separately.

The E-step consists of propagating a belief for $2T$ time steps forward. The M-step consists of propagating a belief for each of the $N(|Q| + |A||Q| + |Q||O||Q|)$ FSC parameters for T time steps forward. Belief propagation complexity is proportional to the number of cliques in the junction tree of the two time slice dynamic Bayesian network, which is $\mathcal{O}(N + M)$ for $\mathcal{O}(N)$ reward sub-functions, and is also proportional to "maxCliquePotential" which denotes the size of the largest clique potential in that junction tree (size of the largest probability distribution over a group of variables that form a clique).

In theory a factored DEC-POMDP could be constructed, where "maxCliquePotential" is $\mathcal{O}(\text{constant}^{\sqrt{N+M}})$, but in all factored DEC-POMDP benchmarks we know of, the largest clique has bounded size (see Section 4 for examples). When dependencies are local, like in wireless networks, the size of cliques in the junction tree is limited.

4 Experiments

We compare our method with two variable clusterings to the expectation maximization (denoted "Flat EM") method of Kumar and Zilberstein [2010a] and to the non-linear programming (denoted "NLP") method of Amato *et al.* [2007]. We used uniform random behavior as a baseline. As infinite-horizon benchmarks we used (modified) factored fire fighting [Oliehoek *et al.*, 2008] and a wireless network problem.

The factored fire fighting problem was introduced by Oliehoek *et al.* [2008] for finite-horizon factored DEC-POMDPs. There are N robotic fire fighting agents and $N+1$ houses in a row. The goal is to extinguish fire in all houses. Agents are placed between houses; each agent chooses which of the two neighboring houses to extinguish. Each house has a fire level from 0 to 2. An agent observes the house it tries to extinguish is on fire, with probability 0.2 for fire level 0, 0.5 for level 1 and 0.8 otherwise. The reward is the single-step expected reward for negative house fire levels, so the reward for house 1 depends on fire level of houses 1 and 2 and action of agent 1. In the original finite-horizon problem all houses could be extinguished in few time steps [Oliehoek, 2010]. We used a small variation on fire catching probabilities so that fires take longer to put out, to emphasize the infinite-horizon nature of the problem. If no agent is extinguishing a house, the fire level increases by 1 with probability 0.8 if a neighbor house is burning, or probability 0.4 if the house is burning but neighbor houses are not. If one agent is extinguishing a house, the fire level increases by 1 with probability 0.32 if a neighbor house is burning, decreases with probability 0.12 if the house is burning, and if the house is burning but no neighbors are, the fire level decreases by 1. If two agents are extinguishing a house, its fire is completely extinguished. For N agents this problem has 3^{N+1} states, 2 actions and 2 observations for each agent; with $|Q|$ nodes per FSC, a joint probability distribution over states and FSC nodes has size $3^{N+1}|Q|^N$.

A wireless network problem, called inline Aloha, is introduced in [Oliehoek, 2010] for finite horizon DEC-POMDPs. Each network device has a limited size packet buffer; they send packets on the same channel, and only neighbouring devices interfere with each other. If two or more neighbouring devices send simultaneously a collision occurs, and packets must be resent. The reward is the negative sum of packet buffer sizes; this penalizes delay and awards throughput. We consider a more general problem: each device has a Markov process placing packets into the buffer, instead of a Poisson process. Figure 2 shows the setup for two agents. We used a two state Markov model as source, with parameters from simulated Web and voice over IP traffic: the probability to move from packet to idle state is 0.0741 and from idle to packet state 0.0470. N agents are arranged so that if two neighbor agents send at the same time, their transmissions fail and they must resend later. An agent can either send or listen. A packet is removed from an agents packet buffer if it sends and neighbours do not. Each agent's packet buffer has 0 to 3 packets. If the source Markov model is in the packet state, a packet is inserted into the buffer. The agent correctly observes the channel state with probability 0.9 and gets wrong observations with uniform probability. Possible channel state observations are: idle, one transmitter, collision. The agent also observes whether its packet buffer is empty. For N agents this problem has $4^N 2^N$ states, 2 actions and 6 (2×3) observations for each agent; a joint probability distribution over states and FSC nodes has size $4^N 2^N |Q|^N$.

We ran experiments for 2, 3, 4, 5, and 10 agents on both benchmarks problems. All methods were implemented with Matlab. Each method was trained for two hours, 10000 iterations or until convergence on an AMD Opteron core with 4GB memory. In the fire fighting problem with 4 agents 6GB was allocated for "Flat EM" to get more comparison results. Longer, three day experiments, denoted with "10*", were run for 10 agents. Finite state controllers were initialized randomly from a Dirichlet distribution with concentration parameter 2. Results are averages over 10 FSC initializations. For very large problems 500 evaluation runs were run for each FSC initialization.

Figure 3 shows the results. "Our1" and "Our2" denote the fully factored and the overlapping variable clusterings discussed in Section 3.1. Our method performs comparably on the small problems and solves the large problems where others fail. The overlapping approximation produces no noticable decrease in accuracy in the smaller problems compared to flat EM and scales well to larger problems. For many agents and large controllers fully factored clustering produces better results than overlapping clustering, because of limited training time.

As discussed in Section 3.5 the largest clique potential affects computation time. In factored fire fighting for "Our 1" and $|Q| = 2$ the largest found clique potential has size $|S_i|^4|A|^2$ for $N >= 2$, and for "Our 2" $|S_i|^6|A||Q|^3$ for $N >= 5$, where S_i is the fire level. In the wireless network problem for "Our 1" and $|Q| = 2$ the largest found clique potential has size $|S_1|^2|S_2||A|^3$ for $N >= 3$, and for "Our 2" $|S_1||S_2||A|^3|Q|^5$ for $N >= 4$, where S_1 and S_2 are "packet buffer" and "source" variables respectively.

5 Conclusions and discussion

We have introduced an efficient approximate expectation maximization method for factored infinite-horizon DEC-POMDPs. The method has polynomial complexity in numbers of state variables and agents; it works well in comparisons and solves large problems that current methods cannot. Optimality bounds for approximations are future work.

Acknowledgments

The authors belong to the Adaptive Informatics Research Centre (CoE of the Academy of Finland). The work was supported by Nokia, TEKES, Academy of Finland decision number 123983, and in part by the PASCAL2 EU NoE, ICT 216886. This publication only reflects the authors' views.

References

[Allen and Zilberstein, 2009] M. Allen and S. Zilberstein. Complexity of decentralized control: Special cases. In *Advances in Neural Information Processing Systems 22*, pages 19–27, 2009.

[Amato *et al.*, 2007] C. Amato, D. Bernstein, and S. Zilberstein. Optimizing Memory-Bounded Controllers for Decentralized POMDPs. In *Proc. of 23rd UAI*, pages 1–8. AUAI Press, 2007.

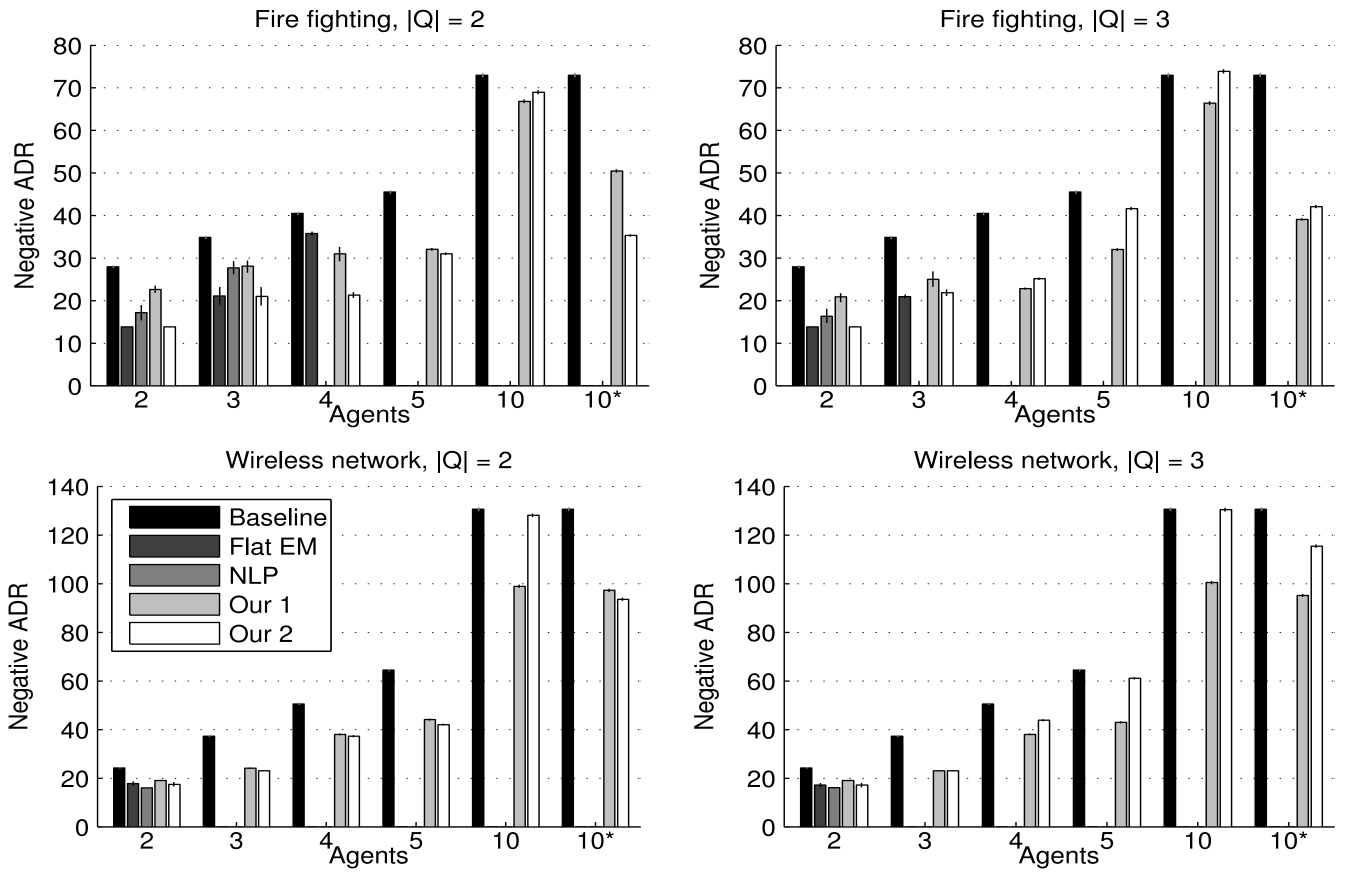

Figure 3: Negative average discounted reward (ADR) in two problems: factored fire fighting (top row) and wireless network (bottom row). Finite state controller sizes 2 (left) and 3 (right) are shown. Shorter bars (less negative reward) are better results. Missing bars are methods that ran out of memory. Longer, three day experiments run for 10 agents are denoted with "10*". 95% confidence intervals, computed using bootstrapping, are marked with error bars; in most cases they are very small.

[Bernstein *et al.*, 2005] D.S. Bernstein, E.A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proc. of 19th IJCAI*, pages 1287–1292. Morgan Kaufmann, 2005.

[Boyen and Koller, 1998] X. Boyen and D. Koller. Tractable Inference for Complex Stochastic Processes. In *Proc. of 14th UAI*, volume 98, pages 33–42. Morgan Kaufmann, 1998.

[Boyen and Koller, 1999] X. Boyen and D. Koller. Exploiting the architecture of dynamic systems. In *Proc. of 16th Nat. Conf. on AI*, pages 313–320. The AAAI Press, 1999.

[Kumar and Zilberstein, 2010a] A. Kumar and S. Zilberstein. Anytime Planning for Decentralized POMDPs using Expectation Maximization. In *Proc. of 26th UAI*, 2010.

[Kumar and Zilberstein, 2010b] A. Kumar and S. Zilberstein. Point-Based Backup for Decentralized POMDPs: Complexity and New Algorithms. In *Proc. of 9th AAMAS*, pages 1315–1322. IFAAMAS, 2010.

[Murphy, 2002] Kevin Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[Oliehoek *et al.*, 2008] F.A. Oliehoek, M.T.J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored DEC-POMDPs. In *Proc. of 7th AAMAS*, volume 1, pages 517–524. IFAAMAS, 2008.

[Oliehoek, 2010] Frans A. Oliehoek. *Value-Based Planning for Teams of Agents in Stochastic Partially Observable Environments*. PhD thesis, Informatics Institute, University of Amsterdam, February 2010.

[Pajarinen *et al.*, 2010] J. Pajarinen, J. Peltonen, A. Hottinen, and M. Uusitalo. Efficient Planning in Large POMDPs through Policy Graph Based Factorized Approximations. In *Proc. of ECMLPKDD 2010*, volume 6323, pages 1–16. Springer, 2010.

[Szer and Charpillet, 2005] D. Szer and F. Charpillet. An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. *Proc. of 16th ECML*, pages 389–399, 2005.

[Toussaint *et al.*, 2006] M. Toussaint, S. Harmeling, and A. Storkey. Probabilistic inference for solving (PO)MDPs. Technical report, University of Edinburgh, 2006.